TI Internal Data — Signed NDA Required for Distribution

LocostoC027 Multimedia Device Silicon Revision 2.0, 2.1

Texas Instruments OMAP™ Family of Products

Technical Reference Manual

Literature Number: SWPU092J
April 2006–Revised June 2007

WARNING: EXPORT NOTICE

Recipient agrees to not knowingly export or re-export, directly or indirectly, any product or technical data (as defined by the U.S., EU, and other Export Administration Regulations) including software, or any controlled product restricted by other applicable national regulations, received from Disclosing party under this Agreement, or any direct product of such technology, to any destination to which such export or re-export is restricted or prohibited by U.S. or other applicable laws, without obtaining prior authorisation from U.S. Department of Commerce and other competent Government authorities to the extent required by those laws. This provision shall survive termination or expiration of this Agreement.

According to our best knowledge of the state and end-use of this product or technology, and in compliance with the export control regulations of dual-use goods in force in the origin and exporting countries, this technology is classified as follows:

US ECCN: 5E002

EU ECCN: 5E002

And may require export or re-export license for shipping it in compliance with the applicable regulations of certain countries.

Contents

List of Figures

List of Tables

Read This First

About This Manual

This technical reference manual provides technical information on the LOCOSTO device.

Information About Cautions and Warnings

This book may contain cautions and warnings.

CAUTION

This is an example of a caution statement.

A caution statement describes a situation that could potentially damage your software or equipment.

WARNING

This is an example of a warning statement.

A warning statement describes a situation that could potentially cause harm to you.

The information in a caution or a warning is provided for your protection. Please read each caution and warning carefully.

FCC Warning

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

Trademarks

OMAP, TMS320C55x, and C55x are trademarks of Texas Instruments Incorporated.

ARM7TDMI is a registered trademark of ARM Limited.

PowerVR is a registered trademark of Imagination Technologies Limited.

All other trademarks are the property of their respective owners.

Survey

Help us meet your expectations:

We are always looking at ways to develop our service and improve our quality to fit your needs. So, please take a few minutes to complete our short questionnaire by:

- Providing general suggestions.
  or
- Rating a document and describing its critical points.

http://www.ti.com/csdocsurvey (password: 123survey)

Thank you.

History

Version	Literature Number	Date	Notes
A	SWPU092	April 2006	See Note (1)
B	SWPU092	June 2006	See Note (2)
C	SWPU092	July 2006	See Note (3)
D	SWPU092	August 2006	See Note (4)
E	SWPU092	September 2006	See Note (5)
F	SWPU092	October 2006	See Note (6)
G	SWPU092	November 2006	See Note (7)
H	SWPU092	December 2006	See Note (8)

(1) *LOCOSTO ES2.0 Technical Reference Manual* version A (SWPU092A)

(2) *LOCOSTO ES2.0 Technical Reference Manual* version B (SWPU092B). Chapters impacted by the changes between version A (SWPU092A) and version B (SWPU092B):
- Chapter 1: Introduction
- Chapter 8: Debug Unit
- Chapter 18: Configuration
- Chapter 23: USB Client

(3) *LOCOSTO ES2.0 Technical Reference Manual* version C. Chapters impacted by the changes between version B (SWPU092B) and version C (SWPU092C):
- Chapter 1: Introduction
- Chapter 6: Clocks, Reset, and Power Management
- Chapter 10: NAND Flash Controller
- Chapter 12: Interrupt Handlers
- Chapter 27: General-Purpose Interface

(4) *LOCOSTO ES2.0 Technical Reference Manual* version D (SWPU092D). Chapters impacted by the changes between version C (SWPU092C) and version D (SWPU092D):
- Chapter 1: Introduction
- Chapter 6: Clocks, Reset, and Power Management
- Chapter 10: NAND Flash Controller
- Chapter 27: General-Purpose Interface

(5) *LOCOSTO ES2.0 Technical Reference Manual* version E (SWPU092E). Chapters impacted by the changes between version D (SWPU092D) and version E (SWPU092E):
- Chapter 5: Interconnect
- Chapter 6: Clocks, Reset, and Power Management
- Chapter 19: UART/IrDA

(6) *LOCOSTO ES2.0 Technical Reference Manual* version F (SWPU092F). Chapters impacted by the changes between version E (SWPU092E) and version F (SWPU092F):
- Chapter 12: Interrupt Handlers
- Chapter 29: Light and Buzzer Control

(7) *LOCOSTO ES2.0 Technical Reference Manual*version G (SWPU092G). Chapters impacted by the changes between version F (SWPU092F) and version G (SWPU092G):
- Chapter 6: Clocks, Reset, and Power Management
- Chapter 30: Initialization

(8) *LOCOSTO ES2.0 Technical Reference Manual*version H (SWPU092H). Chapters impacted by the changes between version G (SWPU092G) and version H (SWPU092H):
- Chapter 6: Clocks, Reset, and Power Management
- Chapter 18: Configuration
- Chapter 30: Initialization

Version	Literature Number	Date	Notes
I	SWPU092	January 2007	See Note (9)
J	SWPU092	June 2007	See Note (10)

(9) *LOCOSTO ES2.0 Technical Reference Manual*version I (SWPU092I). Chapters impacted by the changes between version H (SWPU092H) and version I (SWPU092I):

- Chapter 5: Interconnect
- Chapter 9: Enhanced Memory Protection Unit
- Chapter 11: EMIF
- Chapter 18: Configuration
- Chapter 25: Multichannel Serial Interface

(10) *LOCOSTO ES2.0, ES2.1 Technical Reference Manual*version J (SWPU092J). Chapters impacted by the changes between version I (SWPU092I) and version J (SWPU092J):

- Chapter 1, Introduction
- Chapter 6: Clocks, Reset, and Power Management
- Chapter 19: UART/IrDA
- Chapter 28: C-port

Introduction

This chapter introduces the LOCOSTO device.

Topic **Page**

1.1 LOCOSTO Overview

The LOCOSTO device is an integrated solution that embeds a digital baseband (DBB) and a digital radio processor (DRP) on the same die. The LOCOSTO device targets solutions for GSM/GPRS (low-cost global system for mobile communications/general packet radio service).

The DRP is a digital radio-frequency (RF) transceiver that supports up to a GPRS class12. The DRP is designed for quad-band operation, supporting both the European and the US bands (E-GSM 900 and DCS 1800 bands, GSM 850 and PCS 1900 bands, respectively).

The DBB supports the processing of GSM radio signals in the switching circuit mode and the packet data mode (GPRS) for up to class 10, including evolutions such as the SAIC and localization system (A-GPS) in compliance with the European Telecommunications Standards Institute (ETSI) specification.

The LOCOSTO silicon process is a 90-nm digital CMOS technology.

The LOCOSTO device includes two versions: LOCOSTO and LOCOSTO Lite. The LOCOSTO Lite device does not include the camera interface and the GPRS.

Note: This document describes the features of both the LOCOSTO and the LOCOSTO Lite devices. Unless specified otherwise, the term LOCOSTO refers to both the LOCOSTO and the LOCOSTO Lite devices.

LOCOSTO offers the following features:

- Dual processors:
  - 104-MHz ARM7TDMI® microprocessor unit (MPU)
  - 104-MHz customized digital signal processor (cDSP) c54x
- Internal memory:
  - 154KW DSP read-only memory (ROM)
  - 30KW DSP static random-access memory (SRAM)
  - 2.5Mb MPU SRAM
  - 1.5Mb MPU ROM
- External memory support:
  - 1.8/3 V subscriber identity module (SIM) I/F
  - Direct memory access (DMA) to external memory
  - Burst mode, page-mode external memory: NAND and NOR flash and SRAM (frame buffer)
- Hardware security
  - Flash content
  - International mobile equipment identity (IMEI) protection
  - SIM lock
- Peripheral interfaces
  - Vibrator PWM control signal
  - Universal serial bus (USB) 2.0 full-speed client
  - Keypad
  - Universal asynchronous receiver/transmitter (UART)
  - Multichannel serial interface (MCSI)
  - Bluetooth
  - Camera
  - Primary liquid-crystal display (LCD): 8-bit parallel interface
    Up to QVGA (quarter video graphics array) 256K colors
  - Secondary LCD: Serial interface
- DRP2.0 RF integrated RF:
  - Digital RF 4 band GSM/GPRS up to Class 12 (On the LOCOSTO Lite device, GSM is dual-band

  - muxed and GPRS is not available.)
  - –110 dBm sensitivity
  - Digital PA driver output level +2 dBm
  - 0.7 degrees RMS phase error
  - Integrated digitally controlled crystal oscillator (DCXO)
- Software support:
  - GSM/GPRS layer 1,2,3 (GPRS is not available on LOCOSTO Lite.)
  - Adaptive multirate (AMR), full rate (FR), half rate (HR), enhanced full rate (EFR)
  - Teletypewriter (TTY)
  - SAIC over GSM
  - Man machine interface (MMI) for test
  - Wireless application protocol (WAP), enhanced message service (EMS), multimedia message service (MMS), JAVA
- Multimedia support:
  - Internal 300 KP camera support
  - MP3 player
  - Up to 32 polyphonies stereo midi player
  - Up to 32 polyphonies mono midi ringer
  - JPEG encode and decode

The LOCOSTO device communicates with the analog TWL3031 external subsystem, which provides the following capabilities:

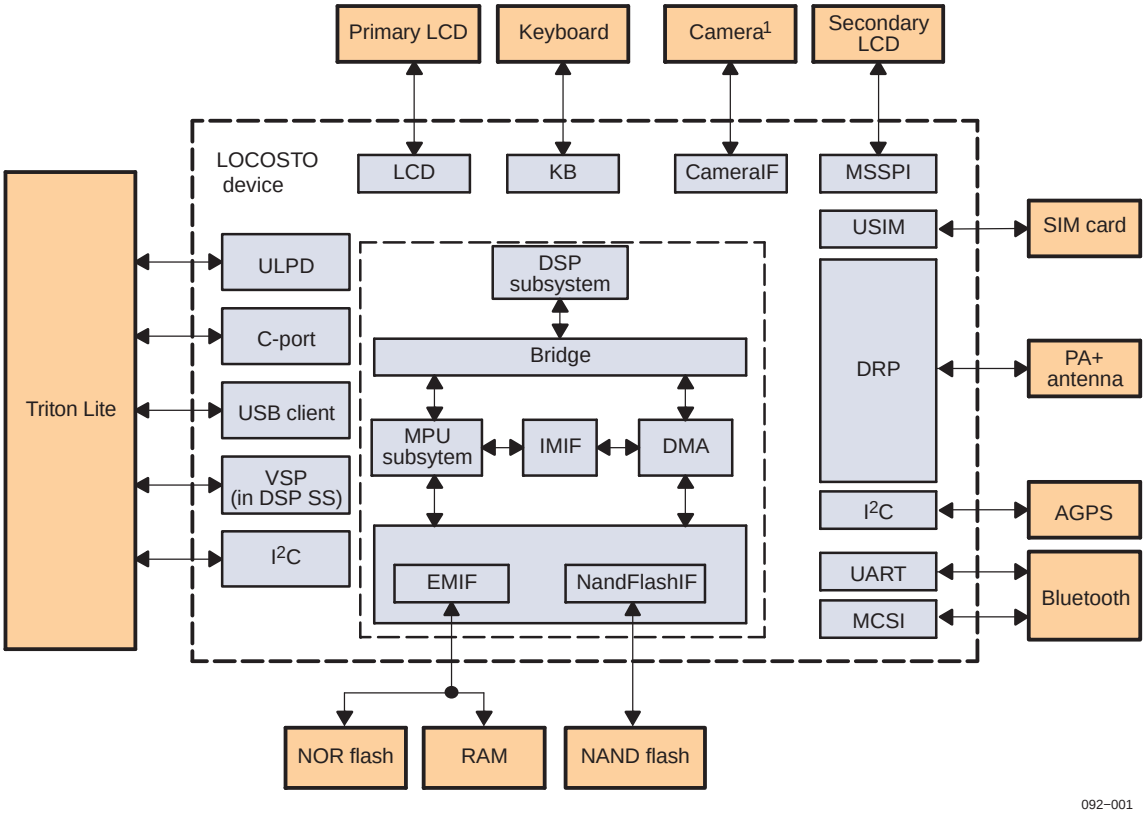
- Handset microphone and speaker
- Headset mono/stereo audio speakers and microphone connection
- Melody ringer (hands-free) and buzzer
- Battery pack (Nimh/Li-ion) and six 7-V regulated or 20-V nonregulated charger
- Vibrator motor control
- Real-time clock (RTC) 32-kHz crystal

The following external/extra subsystems are also supported:

- TI–Bluetooth (BRF6150) wireless short-distance connectivity
- Voice/audio and data
- TI-AGPS/TWL5002 localization system (I/F provision)
- Digital camera systems (not available on LOCOSTO Lite)
- VGA camera sensor/module, for example, AGILENT ADCM-2700 (not available on LOCOSTO Lite)
- OMAP™-DMxx(GoldenEye) camera companion chip (up to 3M-pixel) (not available on LOCOSTO Lite)
- UART cable or IrDA (infrared data association)
- Boot-manufacturing capability from the USB link
- FM radio receiver (for example, Philips TEA5767/68)
- Primary (Qcif) and secondary LCD
- NAND flash-based media storage (for example, on-board SmartMedia)
- USB carkit

Figure 1-1 shows a typical LOCOSTO application.

Figure 1-1. LOCOSTO Typical Application Overview



1.2 LOCOSTO Description

1.2.1 Physical Considerations

The LOCOSTO chip is offered in a 241-ball, 10x10-mm x 0.5-mm pitch, 1.2-mm height µStar-BGA package. LOCOSTO operating conditions are mainly fixed by the 1218C027 manufacturing process and the external system interconnection requirement.

The 1218C027 process (optimized for a 1.8 V LVCMOS I/O operation) uses five layers of second-generation, dual-damascene process (copper with low-K dielectric) combined with a 1.05-V, 1.2-V, and 1.3-V core power supply and low-peak power dissipation. This process provides the performance, highdensity integration, and power management capability required for powerbudgeted mobile equipment.

1.2.2 Architecture

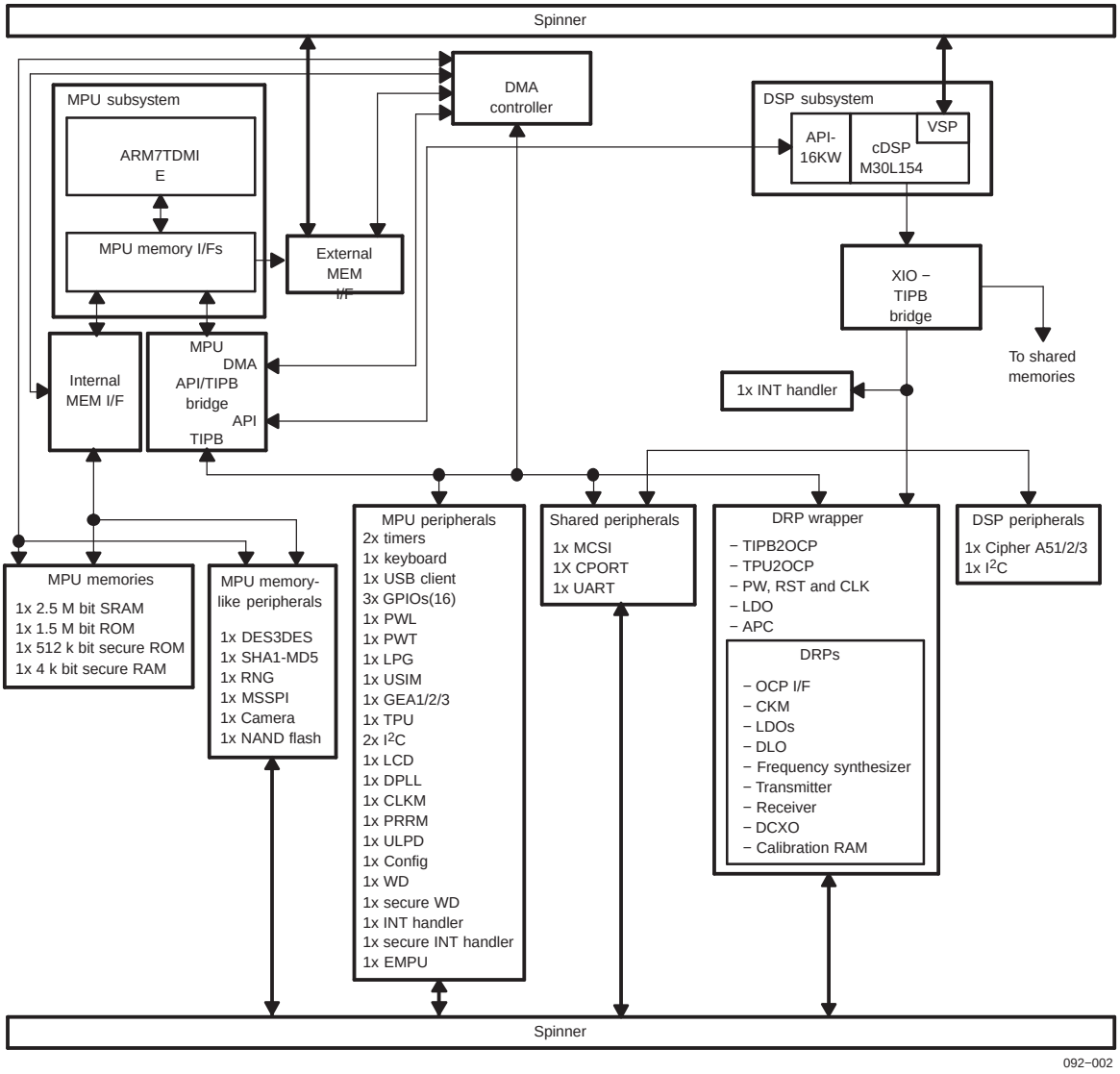
The LOCOSTO device is composed of the following blocks:

- MPU subsystem
- DSP subsystem
- DMA subsystem
- Memories
- Peripherals

The blocks are interconnected using bridges and memory interfaces.

Figure 1-2 shows a block diagram of the LOCOSTO device.

Figure 1-2. LOCOSTO Block Diagram



1.2.3 MPU Subsystem Description

The MPU subsystem is formed by the ARM7TDMIE processor and a memory interface.

The ARM7TDMIE can run at 104 MHz and offers the following performance with the surrounding environment:

- 2.5M-bit internal SRAM, single-cycle access at 104 MHz
- 1.5M-bit internal ROM single-cycle access at 104 MHz
- 16-Kw application programming interface (API), 2-cycle at 104 MHz
- Peripheral access: TI peripheral bus (TIPB) and memory-like peripherals; 2-cycle at 104 MHz, single cycle at 52 MHz
- Up to 124M-byte external memory accessed at 52 MHz max (~15 ns access time)
  - Muxed address-data burst memories: Flash and pseudostatic RAM (PSRAM)
  - 16-bit bus width: Supports byte, half-word (16 bit), and word (32-bit) access
  - 1/2 (default)/3/4 chip-selects (CSs) x 4/8 (default)/16/32M byte
  - Asynchronous/burst single-access read/write (default)
  - Continuous or 4 x 32-bit (8 x 16-bit) word burst read/write

  - 2-line 4 x 32-bit word (8 x 16-bit) prefetch buffer
  - Access mode and I/F timing configurable for each CS
  - Flexible timing control (wait/dummy cycles insertion) to support most common memories

The memory interface allows connections with the following modules:

- ARM port interface (API), permitting high-bandwidth parallel access to 32K-byte DSP resources by the MPU and system DMA
- Two levels of interrupt handler to configure priorities, masks, and level sensitivity of secure and nonsecure interrupts generated by the peripherals
- API/TIPB bridge, providing TIPB interfaces to access MPU private and shared peripherals
- Internal and external memories
- Memory-like peripherals
- Clock manager, providing the functional 104-MHz clock to the subsystem
- DRP external/internal memory

For more information about the MPU subsystem, see Chapter 3, *MPU Subsystem.*

1.2.4 DSP Subsystem Description

The DSP (M30L154) subsystem is a digital signal processor core that complies with the TMS320C54x™ family.

The DSP can run at 104 MHz and offers the following performance with the surrounding environment:

- 30-K 16-bit word RAM (including 16-Kw API) at 104 MHz
- 154-K 16-bit word ROM at 104 MHz
- 10-K 16-bit word RAM (external to the cDSP macro, intended for application; MP3, etc.) at 104 MHz
- Memory-mapped control and status register (XIO/TIPB at 52 MHz)

The DSP subsystem interacts with various modules, such as the following:

- API, permitting high-bandwidth parallel access to the 32K-byte DSP dualaccess RAM (DARAM) resources by the MPU and the system DMA
- XIO/TIPB bridge, providing TIPB interfaces to access DSP private and shared peripherals
- Interrupt handler to configure level-sensitivity of interrupts INT_0 to INT_11 generated by the peripherals
- Clock manager, providing the functional 104-MHz clock for the subsystem
- Bidirectional voice serial port (VSP), directly connected to the voice codec of the TWL3031

The DSP subsystem includes the following peripherals:

- DSP private peripherals:
  - Cipher A5
  - I^2C TWL3031
  - DRP external/ internal memory
  - XIO/TIPB bridge control register
  - API controller
  - Interrupt handler
- DSP shared peripherals:
  - DRP external memory
  - Automatic power control (APC). See the APC application note for details.
  - UART
  - MCSI
  - C-port

For more information about the DSP subsystem, see Chapter 4, *DSP Subsystem*.

1.2.5 DMA Controller Description

To sustain the increased data throughput required by multimedia peripherals, such as the LCD and camera I/F, the enhanced DMA controller has six physical channels, each capable of transferring data from/to any of the five supported ports. The DMA can access secure resources through a secure channel mode setting.

The following five ports and resources are supported:

- API
- Internal memory (memory section)
- Internal memory (peripheral section)
- MPU TIPB: The DMA cannot access peripherals mapped on the MPU TIPB strobe1 or the DSP TIPB.
- External memory

The DMA controller includes the following main features:

- Concurrent DMA transfers: The DMA subsystem can access all ports at the same time as soon as they are free.
- Two levels of priority on each channel: Each channel can be configured as a high- or low-priority channel. High-priority channels are served before low-priority channels.
- Both hardware and software request capability: Either a peripheral (hardware) or the software can initiate a DMA transfer.
- 8,16, or 32 bits data transfer
- Data pipelining capability
- Data packing/splitting: The DMA subsystem can either pack data when the destination port supports larger words than the source port or split data when the source port is aligned with larger words than the destination port. This feature increases the efficiency of the bandwidth use.
- Data bursting
- Three addressing modes: constant, post-increment, and frame-indexed
- Auto-initialization capability: Channels with this feature do not require software intervention to be reinitialized to serve the next DMA request.
- Generation of interrupts on various events

For more information, see Chapter 13, *Direct Memory Access*.

1.2.6 Interconnect Description

The LOCOSTO device is built around the MPU core and the DSP core (for more information, see Chapter 3, *MPU Subsystem*, and Chapter 4, *DSP Subsystem*). The device includes several peripherals allocated to the MPU, to the DSP, or to both processors.

The LOCOSTO device includes peripherals supporting one of the following three bus protocols:

- The TIPB provides communication between processors and peripherals.
- The API bus allows memory sharing between the MPU and the DSP.
- The open core protocol (OCP) bus uses a TIPB/OCP adapter to interface peripherals with the TIPB.

The LOCOSTO device interconnect is composed of four elements:

- The API/TIPB bridge converts MPU and DMA requests into API or TIPB requests.
- The XIO/TIPB bridge converts DSP requests into TIPB requests.
- The shared bridges allow multiple allocations of peripherals.
- The memory interface supports memory-like peripherals and internal memory.

Three types of switches allow the MPU and the DSP to access peripherals:

- The TIPB static switches share TIPB native peripherals.
- The TIPB/OCP static switches share OCP native peripherals.
- The dynamic switches access OCP native peripherals.

Each switch has a different arbitration scheme: hardware for dynamic switches and software for static switches. For more information, see Chapter 5, *Interconnect*.

1.2.7 Memory Interfaces

The LOCOSTO device includes an internal memory interface (IMIF) and an external memory interface (EMIF).

1.2.7.1 Internal Memory Interface

The MPU or the DMA can access memory-like peripherals or internal memories.

- Memory-like peripherals refer to peripherals that are addressed as memories.
  - DES3DES
  - SHA1-MD5
  - RNG
  - Master/slave serial port interface (MSSPI)
  - Camera
  - NAND flash
- The internal on-chip memories consist of the following elements:
  - Internal SRAM
  - Internal ROM
  - Boot ROM
  - Secure SRAM

For more information about the on-chip memories, see Section 1.3.2, *On-Chip Memory*.

1.2.7.2 External Memory Interface

The EMIF is an asynchronous/synchronous-burst EMIF that supports most common memory interface protocols through a flexible programming and timing signal control.

The EMIF controls up to four memory devices for a total address range up to 124M bytes (1 x 28M bytes + 3 x 32M bytes) without adding external logic.

The EMIF supports the following:

- 8-bit, 16-bit, and 32-bit read/write from the MPU or the DMA
- 16-bit word data bus; burst read/write from the DMA; supports instruction/ data prefetch from the MPU.
- Enhanced least recently used (ELRU) priority between the MPU/DMA
- Four CSs (CS0, CS1, CS2, and CS3) with 32M bytes for each CS (except CS0 with 28M bytes)
- Maximum of 124M bytes external memory; each CS is separately configurable for the following:
  - Asynchronous read/write (default)
    - Programmable wait-states
    - Supports dynamic access time extension through the external nRDY signal
  - Synchronous burst read/write
    - 4 x 32 bits or 8 x 16 bits word burst
    - 52-MHz clock for burst accesses
  - Supports dynamic access time extension through the external nRDY signal
- Wait-state insertions for read and write access
- Multiplexing of address/data only
- The EMIF supports the following common external memory signals:
  - Output enable
  - Write enable
  - Address valid
  - Byte enable
  - Ready
  - Device clock
  - Write protect

- External memory frequency configuration

For more information about the EMIF, see Chapter 11, *EMIF.*

1.2.8 Security

The MPU and protectable functions are distributed to the TIPB (16-bit width) and to the memory-like peripheral bus (32-bit width), both running at 52 MHz. The MPU security consists of cryptography and secure environment:

Note: The secure environment is available only on the high-security device.

- Memory-like peripheral bus (MPU)
  - 64K-byte boot ROM (high-security device only)
  - 512-byte secure RAM (high-security device only)
  - Protected mode control and status register (high-security device only)
  - Random number generator (RNG)
  - Hashing SHA-1/MD5 crypto-processor
  - Symmetrical encryptions DES/3-DES crypto-processor
- TIPB (MPU)
  - Secure interrupt handler (high-security device only)
  - Secure watch-dog/timer (high-security device only)
  - Protected resources reset manager
  - Enhanced memory protection unit (EMPU)
  - DMA secure-channel (high-security device only)
  - CLKM; system clock lock
  - Manufacturer public key (128-bit eFuses programmed at package) (high-security device only)

For more information about cryptography, see Chapter 22, *Security Feature*s.

1.3 LOCOSTO Peripherals

1.3.1 Peripherals Overview

The LOCOSTO device includes several kinds of interfaces for peripheral access. For shared peripherals accessible by both processors, arbitration is required to avoid concurrent accesses. Two kinds of switches are available:

- Static switches are programmable and allow software selection of the host.
- Dynamic switches perform hardware arbitration. Both the MPU and the DSP can access the shared peripheral without any software setting.

For more information about the LOCOSTO peripherals mapping and accesses, See Chapter 5, *Interconnect*, and Chapter 2, *Memory Mapping*.

Table 1-1 through Table 1-3 list the LOCOSTO device peripherals.

Table 1-1. LOCOSTO MPU Peripherals

Peripheral Name	General Description
2x timers	16-bit MPU timers
1x keyboard	Keyboard interface up to 5 x 5
1x USB client	USB 2.0 full-speed interface
3x GPIOs (16)	Three sets of 16 general-purpose I/O (GPIO) ports
1x PWL	Control of the backlight of an LCD and/or a keypad
1x PWT	Generate modulated frequency signal for an external buzzer

Table 1-1. LOCOSTO MPU Peripherals (continued)

Peripheral Name	General Description
1x LPG	Produce the electrical signal for a blinking LED
1x USIM	Support of many simultaneous SmartCard applications
1x GEA	GPRS encryption algorithm (GPRS is not available on LOCOSTO Lite.)
1x GEA	GSM time processing unit (TPU) module
2x I2C	General-purpose I^2C controller
	TWL3031-dedicated I^2C controller
1x LCD	LCD controller
1x DPLL	Phase-locked loop (PLL)
1x CLKM	Power, reset, and clock management
1x PRRM	Reset module
1x ULPD	Traffic controller configuration
1x chip configuration registers	Software-programmable registers
1x WD	Watchdog timer
1x secure WD	16-bit secure watchdog timer (high-security device only)
1x INT handler	Interrupt handler
1x secure interrupt handler	Secure interrupt handler (high-security device only)
1x EMPU	Enhanced memory protection unit

Table 1-2. LOCOSTO DSP Private Peripherals

Peripheral Name	General Description
1x cipher A5	Cipher algorithm
1x I^2C	TWL3031 interfacing

Table 1-3. LOCOSTO MPU/DSP Shared Peripherals

Peripheral Name	General Description
1x CPORT	Codec port interface
1x UART	Universal asynchronous receiver/ transmitter
1x MCSI	Multichannel serial interface
1x DRP	RF transceiver
1x APC	Automatic power control

1.3.2 On-Chip Memory

The on-chip memory (OCM) consists of two separate on-chip memory controllers connected independently to an on-chip 64K-byte ROM and an on-chip 512K-byte SRAM/ROM (the OCM BOOT ROM, the OCM SRAM, and the OCM ROM, respectively).

The 512K bytes of the SRAM/ROM are divided as follows:

- 256K bytes protected SRAM
- 64K bytes nonprotected SRAM
- 192K bytes ROM

For more information about on-chip memory, See Chapter 7, *On-Chip Memory.*

1.3.3 GSM/GPRS Peripherals

1.3.3.1 Ultralow Power-Down Controller

In coordination with the CLKM manager, the ultra-low power down (ULPD) controller manages the deep-sleep mode of the baseband function. The ULPD allows the 13-MHz clock (divided by two from the DRP DCXO) to be switched off during the discontinuous reception phases in the GSM idle mode (DRX) for power saving.

During the deep-sleep mode, the ULPD controller maintains the GSM timebase activity from the low-power 32-kHz clock reference, thus keeping the mobile phone synchronization with the network. Moreover, as an extrapolation of the deep-sleep management concept, the ULPD controller monitors the chip wake-up and falling asleep sequences when switching from the off mode to the on mode and from the on mode to the off mode.

The ULPD block includes the following main functions:

- Gauging of the 32-kHz clocking signal supplied from the analog baseband (ABB) chip
- Maintenance of GSM time during deep-sleep mode with the minimum time accuracy to allow a burst demodulation at wakeup
- Programmable timer to exit deep sleep
- Delivery of the 13-MHz master clock to the CLKM/DPLL modules
- Switching between 13 MHz and 32 kHz
- Generation of the chip functional reset

For more information about the ULPD controller, see Chapter 6, *Power, Reset, and Clock Management.*

1.3.3.2 Time Processing Unit

The TPU real-time sequencer is dedicated to monitoring GSM baseband processing. Working from an events table that refers to a GSM TDMA time base, the TPU activates tasks to control DSP peripherals with respect to the timing constraints related to GSM sequencing.

The TPU schedules hardware tasks having time accuracy that is incompatible with software management. For more information about the TPU, see Chapter 16, *Time Processing Unit*.

1.3.3.3 Universal Subscriber Identity Module Interface

The LOCOSTO chip-set environment includes the universal subscriber identity module (USIM) interface, which controls the GSM SIM-card. The USIM implements the hardware interfaces to the SmartCard and a sequencer that manages the transmission protocols T0 and T1, defined in the ISO7816.3 standard, thus freeing the M-controller of the real-time constraints.

The USIM interface handles the following functions:

- Activation sequence
- Hardware interpretation of coding convention
- Timing management for the answer to reset (ATR) sequence
- T = 0 transmit and receive protocol with parity error and timer management
- T = 1 transmit and receive protocol with error (parity/EDC) and timer management
- Warm reset sequence
- Clock stop sequence
- Deactivation sequence

For more information about the USIM interface, see Chapter 24, *Universal Subscriber Identity Module*.

1.3.3.4 GPRS Encryption Module (GEA3)

Note: The GEA3 is not available on LOCOSTO Lite.

In GPRS mode, a ciphering function called the GPRS encryption algorithm (GEA) performs data confidentiality. The ciphering is executed within the link layer control (LLC) upper layer. Based on the option negotiated with the network, GEA mode 1, mode 2, or mode 3 can be selected.

The LLC conveys information between the mobile station and the serving GPRS support node. The procedures are modeled on the high-level data-link control (HDLC) concepts. The LLC supports both the acknowledge and the unacknowledge mode and implements a frame check sequence (FCS) based on the selected mode.

The GEA3 module supports the computation of the FCS based on the LLC and the ciphering/deciphering modes 1, 2, and 3.

Note: The GEA3 module is mapped to the MPU-TIPB.

For more information about GPRS encryption, see Chapter 15, *GPRS-GSM Encryption*.

1.3.3.5 GSM Cipher Processor (A5)

The cipher A5 module offers voice security, implementing both the A5/1 and A5/2 algorithms, and is enhanced with the new A5/3 algorithm (Kasumi).

These algorithms realize the protection of both user data and signaling information elements at the physical layer on the dedicated channel. The A5/3 algorithm implements additional registers to configure long key data (up to 128 bits) and to generate long encipher/decipher data.

Note: The cipher A5 module is mapped to the DSP-TIPB.

For more information about GSM encryption, see Chapter 15, *GPRS-GSM Encryption.*

1.3.4 Serial Interfaces

1.3.4.1 Universal Asynchronous Receiver/Transmitter 16C750

The LOCOSTO device contains a UART shared between the MPU and the DSP through a TIPB/OCP static switch. Two modes can be selected using a set of configuration registers, UART and IrDA.

The UART connects an external modem device through a standard wired interface or an IrDA module. The two functions can be dynamically supported with additional GPIOs and software control.

The following main features are common to both UART and IrDA modes:

- Selectable UART/IrDA modes
- Dual 64-entry first-in, first-out (FIFO) buffer for received and transmitted data payload
- Programmable and selectable transmit and receive FIFO trigger levels for DMA and interrupt generation
- Programmable sleep mode
- Complete status reporting capabilities in the normal and sleep modes
- Frequency prescaler values from 0 to 16383 to generate the appropriate baud rates
- One clock reference of 52 MHz, 48 MHz, or 13 MHz for baud setting
- Two DMA requests, one interrupt request to the MPU, and one interrupt request to the DSP

CAUTION

The UART is shared between the MPU and the DSP processors. Only one processor at a time can control the module with the allocation defined by the processor handshaking through the TIPB-switch module. By default, the UART is connected to the MPU TIPB.

For more information about the UART, see Chapter 19, *UART/IrDA.*

1.3.4.2 USB Client

The USB device controller module supports the implementation of a full-speed device that fully complies with the USB 2.0 standard. The USB device controller is compatible with the *USB Specification Revision 2.0 and USB Specification* Revision 1.1.

The USB device controller module provides an interface between the MPU and the USB device and handles USB transactions with minimal MPU intervention.

The module supports one control endpoint (EP0), up to 15 IN endpoints, and up to 15 OUT endpoints. The exact endpoint configuration is software programmable. The specific items configurable for each endpoint are the size in bytes, the direction (IN, OUT), the type (bulk/interrupt or ISO), and the associated endpoint number.

The USB device controller module also supports three DMA channels for IN endpoints and three DMA channels for OUT endpoints for either bulk/interrupt or ISO transactions. In operation, the USB requires a 48-MHz clock generated that is by a dedicated embedded analog phase-locked loop (APLL) on request. The USB module interface is peripheral virtual component interface (PVCI) compliant. Therefore, the module interfaces with the TI MPU TIPB through a TIPB-PVCI bridge.

The LOCOSTO device has no internal USB transceiver and must use the TWL3031 transceiver capability to access an external USB device.

For more information about the USB client, see Chapter 23, *USB Client.*

1.3.4.3 Master/Slave Serial Port Interface

The MSSPI module, which complies with the SPI standard, is a bidirectional, 4–line interface consisting of the following elements:

- Clock used to shift data in and out
- Multiple device enable
- Multiple data input
- Multiple data output

The LOCOSTO device has one MSSPI module based on a looped shift register that allows both transmit and receive modes. The MSSPI module operates in master mode or in slave mode using either an MPU or DMA protocol.

- In master mode, the SPI provides up to three CSs.
- In slave mode, spi_ncs0 is used as the SPI module input CS.

Note: The MSSPI module is a native 32-bit OCP bus connected to the 32-bit internal memory slow bus through a specific wrapper that permits either the ARM processor or the DMA to access SPI resources.

For more information about the MSSPI, see Chapter 26, *Master/Slave SPI*.

1.3.4.4 Multichannel Serial Interface

The MCSI, with multichannel transmission and reception capability, expands the parallel interface of the MPU or the DSP to connect to external devices, such as audio codecs and GSM system simulators.

The MCSI on the LOCOSTO device provides full-duplex communication with master or slave clock control. All transmission parameters are configurable to cover the maximum number of operating conditions:

- Master or slave clock control (transmission clock and frame synchronization pulse)
- Programmable transmission clock frequency
- Single-channel or multichannel (x16) frame structure
- Programmable word length: 3 bits to 16 bits

- Full-duplex transmission
- Programmable frame configuration
- Continuous or burst transmission
- Normal or alternate framing
- Normal or inverted frame polarity
- Short or long frame pulse
- Programmable oversize frame length
- Programmable frame length
- Programmable interrupt occurrence time (TX and RX)
- Error detection with interrupt generation on wrong frame length
- GSM digital audio interface (DAI) operating modes (radio uplink, radio downlink, and acoustics). The DAI mode is a GSM test interface used to determine the routing of speech data for the devices tested.

For more information about the MCSI, see Chapter 25, *Multichannel Serial Interface*.

1.3.4.5 I²C Master/Slave Serial Interface

The I²C is a master/slave half-duplex serial port using two lines (data and clock) for data transmission with software-addressable external devices. The interface complies with the Philips standard.

The I²C includes the following main features:

- Standard (100 kHz) and fast (400 kHz) transmission modes
- Support of burst write, single read, and combined read modes
- Transmit/receive burst buffer of two 16-bit words
- 3-bit programmable spike filtering logic
- Error-handling capability during I2C bus access

The I²C bus is used as a communication path with the TWL3031 device. The I²C bus also controls additional external components, such as FM radio, the A-GPS system, and the camera module.

For more information about the I²C interface, see Chapter 20,*I²C Module.*

1.3.5 Audio and Video Interfaces

1.3.5.1 LCD Parallel Interface

The LCD interface connects an external color graphic controller to the DBB device (LOCOSTO). This 8-bit parallel interface complies with the 8-bit 6800-series and 8086-series parallel interface standard to support a large range of LCD displays available on the market.

The configuration and data information are transferred from the MPU to the LCD interface through the TIPB by 16-bit words up to a rate of 52M words/s. The control and data information are transferred from the LCD interface to the external LCD controller at a speed that is a fraction (1, 2, 4, or 8) of the 13-MHz GSM clock. Data are transferred in 8-bit words.

A 2-port FIFO buffer of 128 x 16-bit words adapts the transfer rate between the internal SRAM and the LCD graphical RAM.

This interface targets the LCD device with the following features:

- Color LCD screens (passive or active matrix)
- 8-bit 6800-series or 8086-series parallel interface
- 16-bit (RGB 656) or 24-bit (RGB 888) color
- Up to QVGA (320 x 240) format

Note: The serial interface (MSSPI, I²C) is an alternative to the external LCD controller for limited printed circuit board (PCB) traces.

For more information about the LCD parallel interface, see Chapter 21, *LCD Interface.*

1.3.5.2 Camera Core Interface

Note: The camera core interface is not available on LOCOSTO Lite.

The camera core module supports two image-sensor interfaces: serial and parallel. Only one interface selection at a time can be set.

Both serial and parallel interfaces include the following features:

- 128 x 32-bit FIFO
  - DMA access with enable/disable capability
  - Configurable FIFO trigger level
- One interrupt line supporting programmable interrupts
  - Start/end of line
  - Start/end of image
  - FIFO overflow
  - Data threshold status
- Byte-swapping capability with 32-bit word packet supply

  The interfaces include the following properties:
- Serial interface

  The compact camera port (CCP) interfaces a unidirectional differential interface that connects to a camera sensor in compliance with MIPI Rev1.0. The physical link is the sub-LVDS (low-voltage differential signaling, clock and data) type for low electromagnetic interference (EMI) supporting external clocking signals up to 208 MHz. The CCP bit stream has the following embedded synchronization signals:
  - Frame start
  - Line start
  - Frame end
  - Line end
- Parallel interface

  This 8-bit interface includes camera horizontal/vertical synchronization signals, a camera reference clock, and a pixel-capture clock operating up to 48 MHz. The interface provides a camera reference clock (CAM_XCLK) to the camera-sensor based on the on-chip APLL (48 MHz) or MPU (52 MHz) clock sources.

  The interface supports two operating modes:
  - BT656: ITU-R BT656-compatible parallel interface. This BT656 block extracts the start-of-active video (SAV), the end-of-active video (EAV), and the image-data from the 8-bit data flow.
  - No BT: a general parallel interface with camera horizontal/vertical synchronization. The No BT mode detects valid data and includes the reference clock and the pixel-capture clock.

For more information about the camera interface, see Chapter 17, *Camera Interface*.

1.3.5.3 Voice Serial Port

The VSP is a bidirectional (transmit/receive) serial port that directly connects the DSP SPI to the voice codec of the TWL3031.

For more information about the VSP, see Chapter 4, *Section 4.2.6, Serial Port.*

1.3.5.4 CODEC Port Interface

The codec port interface (C-port) supports three data interfaces:

- AC'97 compatible (revision 1.x)
- I2S compatible (master or slave mode)

- 16-bit pulse code modulation (PCM) compatible: least-significant bit (LSB)-justified format or standard format

CAUTION

The C-port registers are shared between the MPU and the DSP processors. Only one processor at a time can control the module. The allocation to the MPU or DSP processor is configurable based on the processor mailbox/handshaking mechanism. By default, the C-port control is allocated to the DSP TIPB.

For more information about the C-port interface, see Chapter 28, *C-port.*

1.3.6 Security Peripherals

1.3.6.1 DES/3DES

This crypto-processor supports the DES and 3DES cipher algorithms and implements the encryption and decryption functions with the following features:

- ECB (electronic codebook)
- CBC (cipher block chaining)
- 8-byte input and output buffers
- 56-bit key size (up to 3)
- 16 (DES) round cycles per block
- Write and read DMA channels

For more information about the DES/3DES, see Chapter 22, *Security Features*.

1.3.6.2 SHA-1/MD5

This crypto-processor supports the MD5 and SHA-1 hashing algorithms and implements the hashing functions with the following features:

- 64-byte input data block
- 160 (SHA)/128 (MD5)-bit output digest
- 80 (SHA)/64 (MD5) round cycles per block
- Write DMA channel
- Read interrupt

For more information about the SHA-1/MD5, see Chapter 22, *Security Features.*

1.3.6.3 Random Number Generator

The RNG provides a true, nondeterministic noise source to generate keys for cryptographic algorithms.

This true RNG is based on the following:

- Dual-shot noise generators
- Nonlinear mixer

The RNG includes the following main features:

- FIPS 140-1 built-in self-test compliant
- 32-bit output register
- 224 system clock cycles for first random output after reset
- 32-bit random number produced each 160 clock cycles
- Interrupt on 32-bit generation completion

For more information about the RNG, see Chapter 22, *Security Features.*

1.3.6.4 Enhanced Memory Protection Unit

The EMPU allows internal-memory subregions to be defined, each with a separate protection attribute to allow the memory space to be partitioned into program instruction, system data, user data, stack, etc.

The application program configures the EMPU, which interfaces with the processor through the TIPB. The address bus directly issued from the processor is monitored, providing a real-time position of the memory region accessed. When a protection breach attempt occurs, the memory control signals are affected, not writing/reading the memory, and the fault condition is indicated to the processor.

The EMPU allows control of two different protected memory spaces shared by a microcontroller and a DMA:

- API memory space
- Internal memory space

For each region, the memory-mapped control registers define the following:

- Internal memory
  - Up to four programmable protected regions within a maximum memory space of 256K bytes
  - 64K-byte maximum region size (256K bytes for four regions)
  - Minimum granularity of 8 bytes on internal memory
  - Minimum granularity of 1 byte on the API
  - Privileged-code memory region (see the definition of protection mode in a following bullet)
  - Protected memory region base-address
  - Protection mode (nonuser read/write, user read-only, ROM, privileged- region write) applied to the memory subregion bounded by the starting/ending addresses
  - Starting/ending addresses within the protected memory region
- API
  - Boundary of the API-protected region
  - Protection mode (read protect, write protect, read/write protect)

The EMPU generates the following:

- An illegal-access signal for each type of protected memory space (internal and API), if the application program tries a nonauthorized access to a memory region (that is, user write access to a region is programmed for user read-only accesses)
- An MPU fault signal that is a logical function of the three protected spaces (internal and API) illegal-access signals
- A fault indication (illegal access) flagged into the EMPU status register available to the processor for fault analysis

For more information about the EMPU, see Chapter 9, *Enhanced Memory Protection Unit*.

1.3.6.5 Protected Resource Reset Management

The protected resource reset management (PRRM) module is a reset module that can perform a reset on the following:

- The 2M-bit internal memory space: by providing an address bus for the internal memory space. This address is incremented from the start address register to the end address register.
- The API memory space: by providing an address bus for the API memory space. This address is incremented from the start address register to the end address register.
- The protected resources

The PRRM can also generate the following:

- Global chip reset
- Protected resource reset
- Opcode prefetch abort

The PRRM reset is initiated on the following various reset sources:

- Global chip reset request
- Protected resource reset request
- Secure watchdog timer reset request
- Soft reset

For more information about the PRRM, see Chapter 6, *Power, Reset, and Clock Management.*

1.3.6.6 Secure Watchdog Timer (high-security device only)

See Section 1.3.8.3, *Timers and Watchdogs*.

1.3.6.7 Secure Interrupt Handler (high-security device only)

See Section 1.3.8.2.3, *Secure Interrupt Handler (high-security device only)*.

1.3.7 Other Modules

1.3.7.1 NAND Flash Interface

The NAND flash interface allows the NAND flash EEPROM to be connected as an external mass storage facility.

The interface implements an 8-bit parallel data bus in addition to the control signals for selecting chip, writing/reading, command and address latching, ready/busy status:

- 8-, 16-, and 32-bit interface from the host processor (mapped on the internal memory interface)
- Double page FIFO implementation (2 x 128 bytes)
- Programmable access time
- Interrupt and flag (polling operations) for end page or end transfer
- Fully automatic transfer process
- Programmable number of bytes for command, address, and data
- Programmable page size of NAND flash memory
- Support of hardware computation of the error code correction (ECC) in read and program mode
- Support of DMA transfer with a DMA request

For more information about the external memory interface, see Chapter 11,*EMIF.*

1.3.7.2 General Purpose I/O

The general-purpose interface of the LOCOSTO device combines three GPIO banks of 16 general-purpose pins.

Each GPIO can be individually configured as the following:

- Input or output port using register programming
- Interrupt generation on the detection of external events
- Debouncing pin reference

Note: For each GPIO module, all interrupts are mapped on a single physical channel of the microcontroller unit (MCU) interrupt handler. The debouncing is applied at only one pin.

Specific outputs of a GPIO module are reserved for buzzer and light control. For that function, GPIO modules provide the following:

- A signal light (LT) to control the power level of a backlight
- A signal buzzer (BU) to control the power level and the tone of a buzzer

Note: Only one BU signal is available on the LOCOSTO device; using the three LT signals depends on pin multiplexing selection.

For more information, see Chapter 27, *General-Purpose Interface*.

1.3.7.3 Keyboard Interface

The LOCOSTO keyboard controller implements a built-in scanning algorithm for hardware-based key press decoding and allows MPU software overhead reduction.

The keyboard controller can handle up to 5 5 keyboards, works on a 32-kHz clock, and can generate wake-up events when the chip is in sleep mode.

The keyboard controller includes the following main features:

- Support of multiconfiguration keyboards of up to 5 rows x 5 columns
- Integrated programmable timer
- Programmable interrupt generation on key events
- Event detection on both key press and key release
- Multikey press detection and decoding
- Long key detection on prolonged key press
- Programmable time-out on permanent key press or after keyboard release

For more information about the keyboard interface, see Chapter 31, *Keyboard Controller.*

1.3.7.3.1 Pulse Width Tones

The pulse width tones (PWTs) generate a modulated frequency signal for an external buzzer.

The PWT frequency is programmable between 349 Hz and 5276 Hz with 12 half-tone frequencies per octave. The volume is also programmable.

For more information about the PWT, see Chapter 29, *Light and Buzzer Control.*

1.3.7.3.2 Pulse Width Light

The pulse width light (PWL) module provides control of the power level of an LCD or keypad backlight.

The PWL voltage level control technique, based on a 4096-bit random sequence, decreases the spectral power at the modulator harmonic frequencies. The logic operates from the 32-kHz reference clock to be independent from the system activity mode. For more information about the PWL, see Chapter 29, *Light and Buzzer Control.*

1.3.7.3.3 Light Pulse Generator

The light pulse generator (LPG) module generates a signal to control a blinking indication light emitting diode (LED).

The blinking period is programmable between 125 ms and 3 s. In addition, the switched-on duration is programmable and the LED can be switched on permanently. For more information about the LPG, see Chapter 29, *Light and Buzzer Control.*

1.3.7.4 Chip Configuration Registers

Software programmable registers, memory-mapped to the MPU TIPB, are used to configure the device hardware. The registers include the following main functions: *Light and Buzzer Control.*

- Static device configuration
- Security and emulation control
- Debug and observability
- Pad functional multiplexing and configuration

For more information about the chip-configuration registers, see Chapter 18, *Configuration*.

1.3.7.5 Debug Unit

The debug unit (DU) is a hardware resource that provides additional support to a software abort-handler. The DU provides a 64-stage-deep history table of the last memory accesses performed before entering the abort mode and then permits an analysis of previous bus transactions.

The DU is a stand-alone function that needs no configuration. It is connected directly to the processor buses (address and control signals) from which it collects the data and mapped in the MPU memory space to allow the saved history table to be retrieved.

The DU includes the following main features:

- 64-word FIFO depth memory mapped
- 32-bit data words made up of the following:
  - 26-bit address
  - 8 control bits
- Continuous data record clocked on every processor fetch (instruction or data)
- Data record automatically frozen on mode switch to abort
- Disabling capability using signal control (that is, software control from a configuration register bit)
- Reuse as a general-purpose RAM storage facility when the debug function is disabled

For more information about the DU, see Chapter 8, *Debug Unit*.

1.3.8 LOCOSTO System Components

The LOCOSTO system components are dedicated to chip-level configuration and processing control.

1.3.8.1 LOCOSTO Clocks, Reset, and Power Management

1.3.8.1.1 Clocks

Except for the 32-kHz source and the external clock, all clocks dispatched through the LOCOSTO core are generated from one 13-MHz source provided by the DRP2.0 module. The clock manager (CLKM) module controls the clocks and manages the power-saving modes with the help of the ULPD module. The CLKM module receives the following four clock frequencies before dispatching the clocks through the core:

- External clock of up to 40 MHz
- Regenerated 13-MHz clock directly fed by the DRP2.0 module
- Regenerated DPLL clock at 104 MHz. This clock is synthesized by the DPLL from the DRP 13-MHz clock and is resynchronized by the DRP2.0 module.
- APLL clock at 48 MHz. The APLL clock is generated from the nonregenerated 13-MHz clock and provided to the USB, camera, UART, and MSCI modules.

For more information about LOCOSTO clocking schemes, see Chapter 6, *Power, Reset, and Clock Management.*

1.3.8.1.2 Reset

The following modules generate hardware and software reset signals:

- eFuse
- ULPD
- CLKM
- Watchdogs
- PRRM
- Secure hardware

For additional information about LOCOSTO reset schemes, see Chapter 6, *Power, Reset, and Clock Management*.

1.3.8.1.3 Power Management

The LOCOSTO device offers three power-saving modes: running, big sleep, and deep sleep. For more information about LOCOSTO power management, see Chapter 6, *Power, Reset, and Clock Management.*

1.3.8.2 Interrupt Handlers

1.3.8.2.1 MPU Interrupt Handler

The MPU interrupt handler allows the connection of up to 32 prioritized and maskable interrupts to the ARM core. The interrupt handler receives interrupts from both internal modules and the chip external environment. Each incoming interrupt is configured as a low-level sensitive or falling-edge sensitive interrupt and can be individually masked using dedicated configuration registers.

An interrupt-level register is associated with each incoming interrupt to define a priority to the corresponding interrupt. Interrupts having the same priority level are decoded based on the hardware predefined order.

Each interrupt can be configured either as an ARM fast interrupt request (FIQ) or an ARM low-priority interrupt request (IRQ).

1.3.8.2.2 DSP Interrupt Handler

The DSP interrupt handler handles up to 21 interrupts to the DSP core.

Each incoming interrupt can be configured as a low-level sensitive or fallingedge sensitive interrupt. Mask and priority level of the interrupts are configured in the DSP core.

1.3.8.2.3 Secure Interrupt Handler (high-security device only)

The secure interrupt handler handles interrupts from all protected resources, providing five prioritized and maskable interrupts. Each interrupt is configured as a low-level sensitive or falling-edge sensitive interrupt and can be individually masked using dedicated configuration registers.

Incoming secure interrupts are managed according to their level of priority and then forwarded to the MPU interrupt handler.

An interrupt level register controlled through the TIPB is associated with each incoming interrupt to define a priority to the corresponding interrupt. Interrupts having the same priority level are sent in a predefined order.

For more information about the LOCOSTO interrupt handlers, see Chapter 12, *Interrupt Handlers.*

1.3.8.3 Timers and Watchdogs

The LOCOSTO device implements the following:

- Two 16-bit general-purpose timers
- One 16-bit watchdog, configurable as a general-purpose timer
- One 16-bit secure watchdog, configurable as a secure general-purpose timer or as a nonsecure general-purpose timer/watchdog

The timers are clocked from the 13-MHz source. The watchdogs are clocked from the 13-MHz source divided by 14 (that is, 928 kHz). For more information, see Chapter 14, *Timers and Watchdogs.*

1.3.8.4 LOCOSTO Configuration Module

The LOCOSTO configuration module consists of a set of registers dedicated to chip-level configuration. The configuration module provides the following functional configurations:

- Secure mode configuration and emulation controls
- Secure resources status
- Peripheral sensitivity to MPU and/or DSP signals
- Static device configuration
- Debug and observability I/O multiplexing
- Functional I/O multiplexing
- Pad configuration (pullup or pulldown enable)

Note: Device pin limitations require I/O sharing. The LOCOSTO device configuration module handles I/O multiplexing.

For more information, see Chapter 18, *Configuration.*

1.4 LOCOSTO Device Type and Identification Registers

The registers described in this section are on-chip revision registers for the LOCOSTO device. Reset values depend on the device and are listed in Table 1-4 and Table 1-5 as TI internal data.

1.4.1 Chip ID Code Register

Table 1-4. Chip ID Code

Address Offset	0x00														
Physical Address	0xFFFE: F000			**Instance**	JTAG_ID										
Description	This register contains the revision number, the Hawkeye number, and the Manufacturer ID.														
Type	R														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VERSION				HAWKEYE											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAWKEYE				TI_IDM											–

Bits	Field Name	Description	Type	Reset
31:28	VERSION	0x1: ES 1.1 and ES 1.11 0x2: ES 2.0 and ES 2.1	R	According to the ES release
27:12	HAWKEYE	Hawkeye number	R	0xB6CD
11:1	TI_IDM	Manufacturer identity	R	0x017
0	–	Always set to 1	R	0x1

1.4.2 Version Register

Table 1-5. Version Register

Address Offset	0x48		
Physical Address	0xFFFE F048	**Instance**	Version register
Description	This register contains data on the LOCOSTO version.		
Type	R		

15	14	13	12	11	10	9	8
Reserved				ENGG_VERSION			
7	6	5	4	3	2	1	0
Reserved		PROTECTION	ULC_BIT	HARDWARE_ VERSION			

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Reserved	R	0x0
11:8	ENGG_VERSION	Revision number	R	TI internal data
7:6	Reserved	Reserved	R	0x0
5	PROTECTION	Protected mode 0x0: Enable 0x1: Disable	R	0x1
4	ULC_BIT	0: LOCOSTO 1: LOCOSTO Lite	R	0x0
3:0	HARDWARE_ VERSION	Hardware version	R	TI internal data

Memory Mapping

This chapter introduces the MPU and the DSP memory address spaces of the LOCOSTO device.

Topic **Page**

2.1 Introduction

The 32-bit microprocessor unit (MPU) address bus defines a 4G-byte space. This space is divided into regions based on the type of internal bus used.

The MPU memory space is composed of:

- 512K-byte internal memory space through the internal memory interface (IMIF)
- 124M-byte external memory space through the external memory interface (EMIF)
- Internal memory-like peripheral space through the MEM_PERIPHERAL interface (I/F)
- 32K-byte internal DSP shared memory space (the API) through an application programming interface/Texas Instruments peripherals bus (API/TIPB) bridge
- Peripheral space through an API/TIPB bridge

The LOCOSTO DSP memory space is divided into seven 128K-byte pages:

- 1 data memory page
- 5 program memory pages
- 1 XIO memory page

2.2 MPU Memory Mapping

2.2.1 Memory Address Space Overview

This section provides a global view of memory mapping in the LOCOSTO MPU memory mapping. Figure 2-1 shows the memory space divided into six spaces: the boot ROM, external memory, internal memory, memory-like peripherals, API memory, and peripherals. Table 2-1 describes the address range of each space.

The external memory and the peripheral spaces are detailed in the following subsections.

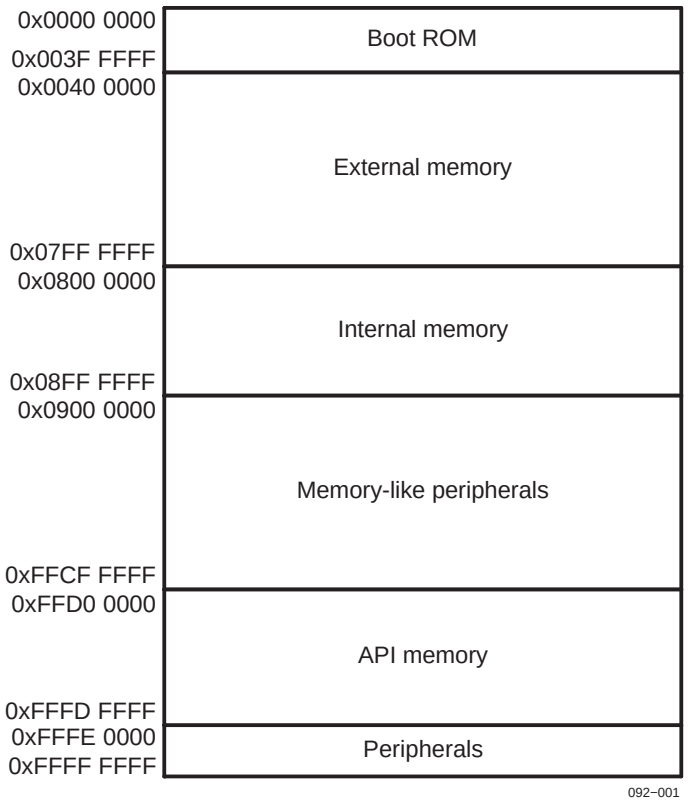


Figure 2-1. MPU Memory Mapping Overview

Table 2-1. MPU Memory Mapping Addresses

Device Name	Start Address	End Address	Size	Data Access	BUM Chapter
Boot Area 0x0000 0000 – 0x003F FFFF					
Boot ROM	0x0000 0000	0x0000 FFFF	64K bytes	8/16/32 bits R	NA
Reserved	0x0001 0000	0x003F FFFF			
External Memory 0x0400 0000 – 0x07FF FFFF					
External memory	0x0040 0000	0x07FF FFFF	124M bytes	8/16/32 bits R/W	11
Internal Memory 0x0800 0000 – 0x08FF FFFF					
Protected RAM	0x0800 0000	0x0803 FFFF	256K bytes	8/16/32 bits R/W	NA
Nonprotected RAM	0x0804 0000	0x0804 FFFF	64K bytes	8/16/32 bits R/W	NA
ROM	0x0805 0000	0x0807 FFFF	192K bytes	8/16/32 bits R	NA
Reserved	0x0808 0000	0x08FF FFFF			
Internal Memory-Like Peripherals 0x0900 0000 – 0x0FFF FFFF					
Boot ROM	0x0900 0000	0x0900 FFFF	64K bytes	8/16/32 bits R	NA

Table 2-1. MPU Memory Mapping Addresses (continued)

Device Name	Start Address	End Address	Size	Data Access	BUM Chapter
Reserved	0x0901 0000	0x096F FFFF			
Camera	0x0970 0000	0x097F FFFF	1M byte	32 bits R/W	17
SHA-1/MD5	0x0980 0000	0x098F FFFF	1M byte	32 bits R/W	22
DES/3DES	0x0990 0000	0x099F FFFF	1M byte	32 bits R/W	22
RNG	0x09A0 0000	0x09AF FFFF	1M byte	32 bits R/W	22
TI reserved (see (1))	0x09B0 0000	0x09CF FFFF			
NAND flash	0x09D0 0000	0x09DF FFFF	1M byte	8/16/32 bits R/W	10
MSSPI	0x09E0 0000	0x09EF FFFF	1M byte	32 bits R/W	26
Debug unit	0x09F0 0000	0x09FF FFFF	1M byte	32 bits R/W	8
Reserved	0x0A00 0000	0xFFCF FFFF			
API Memory 0xFFD0 0000 – 0xFFFD FFFF					
API RAM	0xFFD0 0000	0xFFD0 7FFF	32K bytes	16/32 bits R/W	NA
Reserved	0xFFD0 8000	0xFFDF FFFF			
API control register	0xFFE0 0000	0xFFE0 0001	2 bytes	16 bits R/W	4
Reserved	0xFFE0 0002	0xFFFD FFFF			
Peripherals 0xFFFE 0000 – 0xFFFF FFFF					
TIPB strobe #1	0xFFFE 0000	0xFFFE FFFF	64K bytes	8/16 bits R/W	NA
TIPB strobe #0	0xFFFF 0000	0xFFFF FFFF	64K bytes	8/16 bits R/W	NA

(1) To avoid hazardous effects, do not write in this space.

Note: The boot ROM is mapped in both the area space and the memory-like peripheral space.

2.2.2 External Memory Mapping

The EMIF can control four external devices through four independent chip-selects (CS0, CS1, CS2, and CS3) without adding any external logic. Figure 2-2 shows the link between the MPU addresses and the external memory chip-selects. Because of the boot area, the first chip-select (CS0) can only support up to 28M bytes of memory; the other three chip-selects can support up to 32M bytes of memory. For further details, see Chapter 11, *EMIF*.

Note: The nCS1 and nCS2 signals are not available at power-on reset due to multiplexed pins (respectively, gpio_36 and gpio_35). For further details, see Chapter 18, *Configuration*.

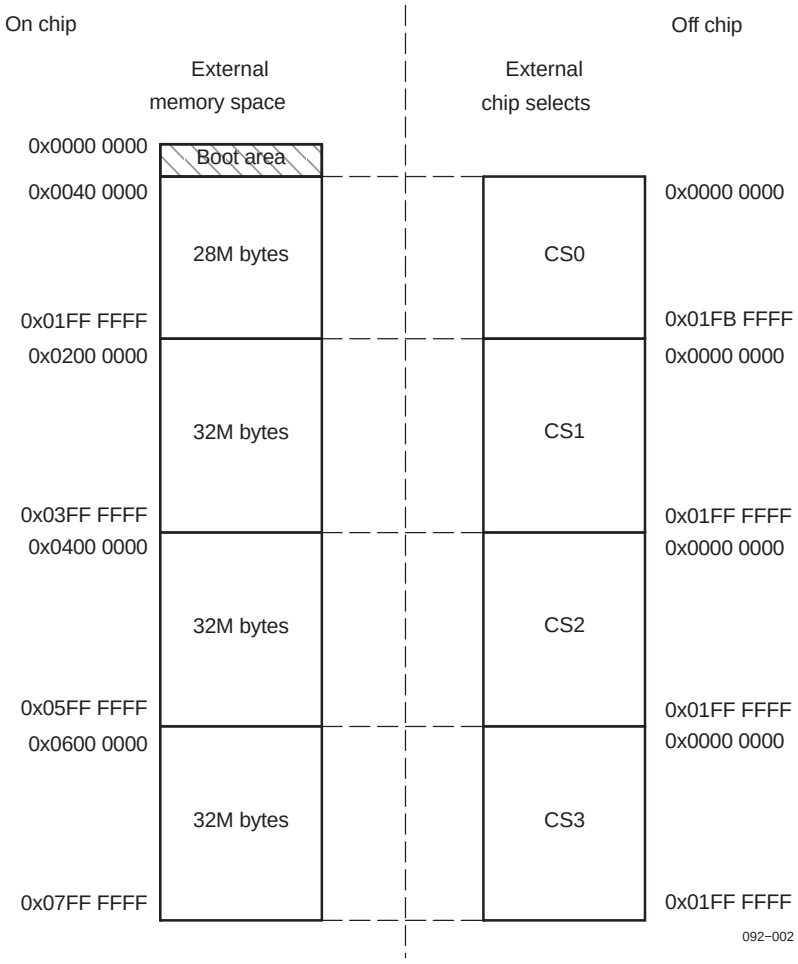


Figure 2-2. External Memory Mapping

2.2.3 API

These 32K bytes of internal dual access data RAM (DARAM) are shared between different resources: the MPU subsystem, the DSP subsystem, and the DMA subsystem. Table 2-1 lists the address space on the MPU side, and Table 2-4 lists the address space on the DSP side.

2.2.4 Peripheral and Control Registers

This space is accessible through the TIPB STROBE #0 and STROBE #1 lines. For further details, see Chapter 6, *Interconnect*.

Table 2-2 and Table 2-3 list the modules that are connected to the STROBE #0 line and the STROBE #1 line, respectively.

Table 2-2. Peripheral Space Addresses - TIPB Strobe #0

Device Name	Start Address	End Address	Size	Data Access	BUM Chapter
DRP reserved(1)	0xFFFF 0000	0xFFFF 3FFF			
DRP external memory	0xFFFF 4000	0xFFFF 4FFF	4K bytes	16 bits R/W	7

(1) To avoid hazardous effects, do not write into this space.

Table 2-2. Peripheral Space Addresses - TIPB Strobe #0 (continued)

Device Name	Start Address	End Address	Size	Data Access	BUM Chapter
APC[2]	0xFFFF 5000	0xFFFF 57FF	2K bytes	16 bits R/W	See Note below table
Reserved	0xFFFF 5800	0xFFFF 6FFF			
UART[2]	0xFFFF 7000	0xFFFF 727F	640 bytes	8 bits R/W	19
UART TIPB/OCP switch	0xFFFF 7280	0xFFFF 77FF	1.375K bytes	8 bits R/W	5
MCSI[2]	0xFFFF 7800	0xFFFF 7FFF	2K bytes	16 bits R/W	25
DRP dynamic switch	0xFFFF 8000	0xFFFF 87FF	2K bytes	16 bits R/W	5
APC TIPB static switch	0xFFFF 8800	0xFFFF 881F	32 bytes	16 bits R/W	5
MCSI TIPB static switch	0xFFFF 8820	0xFFFF 883F	32 bytes	16 bits R/W	5
C-port TIPB static switch	0xFFFF 8840	0xFFFF 885F	32 bytes	16 bits R/W	5
Reserved	0xFFFF 8860	0xFFFF 8FFF			
TPU RAM	0xFFFF 9000	0xFFFF 97FF	2K bytes	16 bits R/W	16
DPLL	0xFFFF 9800	0xFFFF 9FFF	2K bytes	16 bits R/W	6
LCD interface	0xFFFF A000	0xFFFF A7FF	2K bytes	16 bits R/W	21
USIM interface	0xFFFF A800	0xFFFF AFFF	2K bytes	16 bits R/W	24
USB	0xFFFF B000	0xFFFF B7FF	2K bytes	16 bits R/W	23
I^2C	0xFFFF B800	0xFFFF BFFF	2K bytes	16 bits R/W	20
GEA3	0xFFFF C000	0xFFFF C7FF	2K bytes	16 bits R/W	15
I^2C Triton Lite	0xFFFF C800	0xFFFF CFFF	2K bytes	16 bits R/W	20
C-port (control registers)[2]	0xFFFF D000	0xFFFF D7FF	2K bytes	16 bits R/W	28
C-port (FIFO registers)[2]	0xFFFF D800	0xFFFF DFFF	2K bytes	16 bits R/W	28
TI reserved[1]	0xFFFF E000	0xFFFF E7FF			
DMA controller	0xFFFF E800	0xFFFF EFFF	2K bytes	16 bits R/W	13
TPU	0xFFFF F000	0xFFFF F7FF	2K bytes	16 bits R/W	16
Watchdog	0xFFFF F800	0xFFFF F87F	128 bytes	16 bits R/W	14
Secure watchdog	0xFFFF F880	0xFFFF F8FF	128 bytes	16 bits R/W	14
API/TIPB bridge	0xFFFF F900	0xFFFF F9FF	256 bytes	16 bits R/W	5
Interrupt handler	0xFFFF FA00	0xFFFF FA7F	128 bytes	16 bits R/W	12
Secure interrupt handler	0xFFFF FA80	0xFFFF FAFF	128 bytes	16 bits R/W	12
EMIF	0xFFFF FB00	0xFFFF FB2D	46 bytes	16 bits R/W	11
API_RHEA_CNTL register	0xFFFF FB2E	0xFFFF FB2F	2K bytes	16 bits R/W	3
BOOT_MODE_CONF register	0xFFFF FB30	0xFFFF FB31	2K bytes	16 bits R/W	30
Reserved	0xFFFF FB32	0xFFFF FBFF			
PRRM	0xFFFF FC00	0xFFFF FCFF	256 bytes	16 bits R/W	6
CLKM	0xFFFF FD00	0xFFFF FDFF	256 bytes	16 bits R/W	6
Reserved	0xFFFF FE00	0xFFFF FEFF			
EMPU	0xFFFF FF00	0xFFFF FFFF	256 bytes	16 bits R/W	9

(2) Module behind a configurable switch

Note: The codec port (C-port) FIFO registers are accessed in read/write mode only through the MPU TIPB, whereas they can be accessed in read-only mode only through the TIPB static switch. The other control registers are accessed in read/write mode only through the TIPB static switch. For further details, see Chapter 28, *C-Port*.

For more information about the APC module, see the *LOCOSTO APC Application Note*, which is available through your TI representative.

Table 2-3. Peripheral Space Addresses - TIPB Strobe #1

Device Name	Start Address	End Address	Size	Data Access	BUM Chapter
Reserved	0xFFFE 0000	0xFFFE 07FF			
TPU–2–OCP	0xFFFE 0800	0xFFFE 0FFF	2K bytes	16 bits R	16
Reserved	0xFFFE 1000	0xFFFE 1FFF			
ULPD	0xFFFE 2000	0xFFFE 27FF	2K bytes	16 bits R/W	6
Reserved	0xFFFE 2800	0xFFFE 37FF			
TIMER1	0xFFFE 3800	0xFFFE 3FFF	2K bytes	16 bits R/W	14
Reserved	0xFFFE 4000	0xFFFE 47FF			
GPIO	0xFFFE 4800	0xFFFE 4FFF	2K bytes	16 bits R/W	27
GPIO1	0xFFFE 5000	0xFFFE 57FF	2K bytes	16 bits R/W	27
GPIO2	0xFFFE 5800	0xFFFE 5FFF	2K bytes	16 bits R/W	27
Reserved	0xFFFE 6000	0xFFFE 67FF			
TIMER2	0xFFFE 6800	0xFFFE 6FFF	2K bytes	16 bits R/W	14
Reserved	0xFFFE 7000	0xFFFE 77FF			
LPG	0xFFFE 7800	0xFFFE 7FFF	2K bytes	8 bits R/W	29
PWL	0xFFFE 8000	0xFFFE 87FF	2K bytes	8 bits R/W	29
PWT	0xFFFE 8800	0xFFFE 8FFF	2K bytes	8 bits R/W	29
Software debug registers	0xFFFE 9000	0xFFFE 97FF	2K bytes	16 bits R/W	18
Reserved	0xFFFE 9800	0xFFFE B7FF			
Keyboard controller	0xFFFE B800	0xFFFE BFFF	2K bytes	16 bits R/W	31
Reserved	0xFFFE C000	0xFFFE EFFF			
JTAG PART_ NUMBER (IDCODE[27:12])	0xFFFE F000	0xFFFE F001	2 bytes	16 bits R	1
JTAG VERSION (IDCODE[31:28])	0xFFFE F002	0xFFFE F003	2 bytes	16 bits R	1
DieID [63:0]	0xFFFE F004	0xFFFE F00B	8 bytes	16 bits R	1
Man_pub Key	0xFFFE F00C	0xFFFE F01B	16 bytes	16 bits R	22
LOCOSTO core configuration	0xFFFE F01C	0xFFFE F03F	36 bytes	16 bits R/W	18
DieID [127:64]	0xFFFE F040	0xFFFE F047	8 bytes	16 bits R	1
LOCOSTO version register	0xFFFE F048	0xFFFE F049	2 bytes	16 bits R	18
Reserved	0xFFFE F04A	0xFFFE F0FF			
I/O configuration registers	0xFFFE F100	0xFFFE F1FF	256 bytes	16 bits R/W	18
Reserved	0xFFFE F200	0xFFFE FFFF			

2.3 DSP Memory Mapping

Figure 2-3 presents an overview of the DSP memory space, and Table 2-4 summarizes the address range of each memory.

The 16-bit addressing DSP memory space is divided into seven pages: data, Prog 0, Prog 1, Prog 2, Prog 3, Prog 4, and XIO. The program memory space contains the instructions to be executed and the tables used during execution. The data memory space stores data used by the instructions. The I/O memory space interfaces to external memory-mapped peripherals.

A page can be composed of different memories:

- API: Dual access data RAM, used as a communication memory between the lead CPU and the ARM host processor
- DARAM: Dual access data RAM
- PDRAM: Program or data RAM mapped on both the data page and the program page
- PROM: Program ROM, always in the program page
- DROM: Data ROM, always in the data page
- PDROM: Program or data ROM

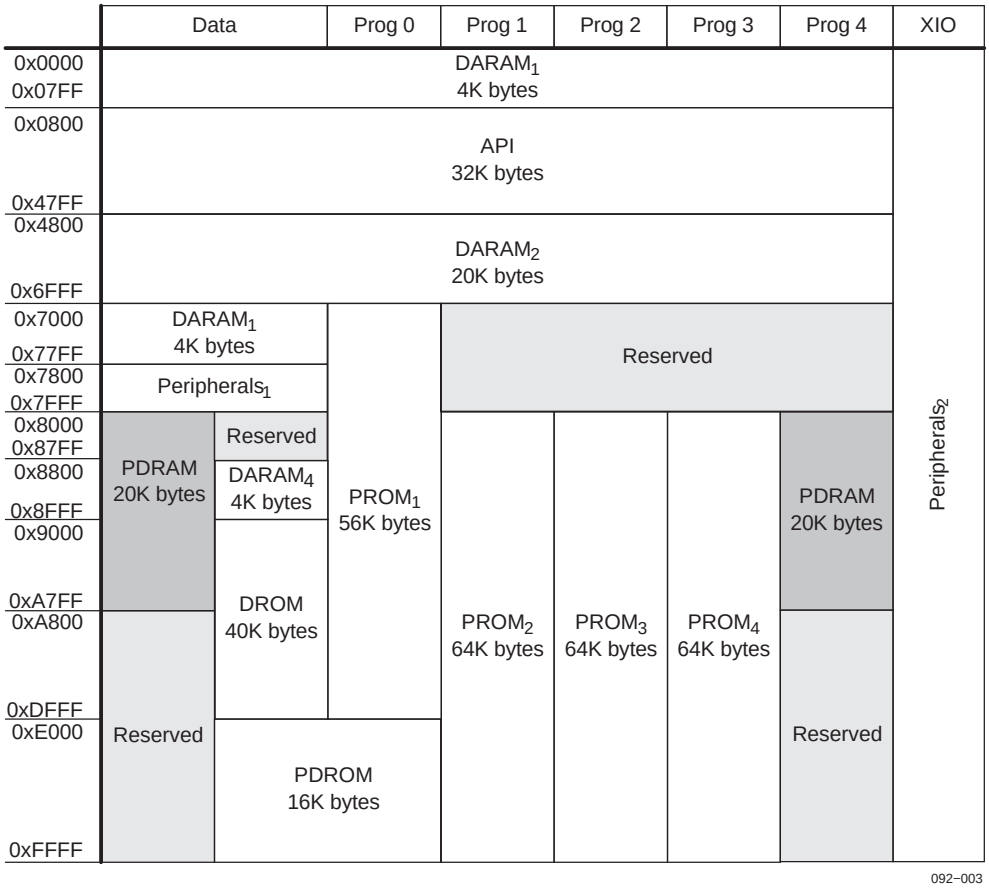


Figure 2-3. 16-Bit Addressing DSP Memory Space

Table 2-4. DSP Memory Address Ranges

Memory	Start Address	End Address	Size	Pages					
				Data	Prog 0	Prog 1	Prog 2	Prog 3	Prog 4
API	0x0800	0x47FF	32K bytes	ü	ü OVLY=1	ü OVLY=1	ü OVLY=1	ü OVLY=1	ü OVLY=1

Table 2-4. DSP Memory Address Ranges (continued)

Memory	Start Address	End Address	Size	Pages					
				Data	Prog 0	Prog 1	Prog 2	Prog 3	Prog 4
DARAM	0x0000	0x07FF	4K bytes	ü	ü OVLY=1	ü OVLY=1	ü OVLY=1	ü OVLY=1	ü OVLY=1
	0x4800	0x6FFF	20K bytes	ü	ü OVLY=1	ü OVLY=1	ü OVLY=1	ü OVLY=1	ü OVLY=1
	0x7000	0x77FF	4K bytes	ü					
	0x8800	0x8FFF	4K bytes	ü DROM=1					
PDRAM	0x8000	0xA7FF	20K bytes	ü DROM=0					ü
DROM	0x9000	0xDFFF	40K bytes	ü DROM=1					
PDROM	0xE000	0xFFFF	16K bytes	ü DROM=1	ü				
PROM	0x7000	0xDFFF	56K bytes		ü				
	0x8000	0xFFFF	64K bytes			ü			
	0x8000	0xFFFF	64K bytes				ü		
	0x8000	0xFFFF	64K bytes					ü	

- From 0x0000 to 0x6FFF, DARAM1, DARAM2, and the API memories can be shared between all the pages using the DSP. PMST[5] OVLY bit. (For more information, see the *TMS320C54x Customizable DSP (cDSP CPU Ref Set, Volume 1)* (SPRU251)).
- 20K bytes of PDRAM are shared between the Prog 4 page and the data page when the DROM bit is set to 0 (see the *TMS320C54x Customizable DSP (cDSP CPU Ref Set, Volume 1)* (SPRU251)). If the DSP.PMST[3] DROM bit is set to 1, the 40K-byte DROM, the 16K-byte PDROM, and the 4K-byte DARAM (0x8800 – 0x8FFF) are accessible in the data page. In this configuration, the 16K-byte PDROM is shared between the data page and the Prog 0 page.
- Different PROMs can be accessed in Prog 0, Prog 1, Prog 2, and Prog 3.

For further details on the DSP memory space configuration, see Chapter 4, *DSP Subsystem*.

2.3.1 Peripheral Mapping

Table 2-5 and Table 2-6 list the peripheral mapping in the data page and in the XIO page, respectively.

Table 2-5. DSP Data Page - Peripheral Space Addresses

Peripheral Name	Start Address	End Address	Size	Access	BUM Chapter
MCSI(1)	0x7800	0x787F	256 bytes	16 bits R/W	25
I^2C Triton Lite	0x7880	0x78FF	256 bytes	16 bits R/W	20
UART(1)	0x7900	0x793F	128 bytes	16 bits R/W	19
UART TIPB/OCP switch	0x7940	0x797F	128 bytes	16 bits R/W	5
APC(1)	0x7980	0x79FF	256 bytes	16 bits R/W	See note below table.
DRP external memory	0x7A00	0x7AFF	512 bytes	16 bits R/W	7
DRP internal memory	0x7B00	0x7BFF	512 bytes	16 bits R/W	7
C-port (FIFO registers and control registers)(1)	0x7C00	0x7C7F	256 bytes	16 bits R/W	28
APC TIPB static switch	0x7C80	0x7C81	4 bytes	16 bits R/W	5
Reserved	0x7C82	0x7C9F			

(1) Module behind a configurable switch

Table 2-5. DSP Data Page - Peripheral Space Addresses (continued)

Peripheral Name	Start Address	End Address	Size	Access	BUM Chapter
MCSI TIPB static switch	0x7CA0	0x7CA1	4 bytes	16 bits R/W	5
Reserved	0x7CA2	0x7CBF			
C-port TIPB static switch	0x7CC0	0x7CC1	4 bytes	16 bits R/W	5
Reserved	0x7CC2	0x7DFF			
DRP reserved(2)	0x7E00	0x7EFF			

(2) To avoid hazardous effects, do not write into this space.

Note: For more information about the APC module, see the *LOCOSTO APC Application Note*, which is available through your TI representative.

Table 2-6. DSP XIO Page - Peripheral Space Addresses

Peripheral Name	Start Address	End Address	Size	Access	BUM Chapter
Reserved	0x0000	0x07FF			
MCSI(1)	0x0800	0x0FFF	4K bytes	16 bits R/W	25
Reserved	0x1000	0x27FF			
CipherA5	0x2800	0x2FFF	4K bytes	16 bits R/W	15
Reserved	0x3000	0x7FFF			
DRP reserved(2)	0x8000	0x9FFF			
DRP external memory	0xA000	0xA7FF	4K bytes	16 bits R/W	7
Reserved	0xA800	0xE7FF			
DMA controller	0xE800	0xEFFF	4K bytes	16 bits R/W	13
Reserved	0xF000	0xF7FF			
XI0/TIPB bridge	0xF800	0xF8FF	512 bytes	16 bits R/W	5
API control	0xF900	0xF9FF	512 bytes	16 bits R/W	5
Interrupt handler	0xFA00	0xFAFF	512 bytes	16 bits R/W	12
NMI_REG register	0xFB00	0xFBFF	512 bytes	16 bits R/W	4
Reserved	0xFC00	0xFFFF			

(1) Module behind a configurable switch
(2) To avoid hazardous effects, do not write into this space.

- The MCSI module can be accessed in either the data page or the XIO page.
- The 4K-byte digital RF processing (DRP) external memory is in the DRP wrapper. The last 512 bytes of this memory are mapped to the data page (0x7A00 – 0x7AFF), and the whole memory is mapped to the XIO page (0xA000 – 0xA7FF).

MPU Subsystem

This chapter introduces and briefly defines the main features of the MPU subsystem of the LOCOSTO device.

Topic **Page**

3.1 MPU Subsystem Overview

The microprocessor unit (MPU) subsystem consists mainly of two modules:

- A microprocessor computing unit based on an ARM7TDMIE
- A memory interface that translates addresses and directs data to the following interfaces:
  - Internal memory and memory-like peripherals memory
  - EMIF
  - API/TIPB bridge
  - MPU TIPB

The MPU subsystem interacts with different modules, such as the following:

- ARM port interface (API), which permits high-bandwidth parallel access by the MPU and system DMA to 32K-byte digital signal processor (DSP) resources
- Two-level interrupt handler to configure the priorities, masks, and level sensitivity of secure and nonsecure interrupts generated by the peripherals
- API/TIPB bridge, which provides Texas Instruments peripheral bus (TIPB) interfaces to access MPU private and shared peripherals
- Internal and external memories
- Memory-like peripherals
- Clock manager, which provides the functional 104-MHz clock for the subsystem

The API is a 16-bit data parallel port used to interface the MPU with the DSP. Information is exchanged through a shared memory that is accessible by both the MPU and the DSP (see Chapter 4, *DSP Subsystem*). Two methods of sharing are accessible:

- Host-only mode (HOM): Only the MPU and the DMA controller can access memory.
- Shared access mode (SAM): The DSP, the MPU, and the DMA controller can access shared memory.

The MPU subsystem has two interrupt handler levels:

- The secure interrupt handler (level 2) services up to five secure interrupts coming from secured modules such as RNG, DES3DES, etc. For more details, see Chapter 22, *Security Features*.
- The MPU interrupt handler (level 1) allows the configuration masks, priorities, and level sensitivity of 30 interrupts. It also provides the routing of all interrupts directly to the MPU core via an interrupt request (IRQ) or a fast interrupt request (FIQ). For more information, see Chapter 12, *Interrupt Handlers*.

The API/TIPB bridge converts MPU requests into TIPB or API requests to access private or shared peripherals.

- The MPU private peripherals are as follows:
  - CLKM
  - PRRM
  - Secure and nonsecure watchdog
  - DMA controller
  - I^2C
  - USB
  - USIM
  - LCD
  - Keyboard
  - Light and buzzer
  - GEA
  - GPIO
  - Timers
  - TPU
  - ULPD

  - Secure interrupt handler
  - Interrupt handler
- The MPU shared peripherals are as follows:
  - DRP 2 OCP memory
  - APC
  - UART
  - MCSI
  - C-port

The API/TIPB bridge module allows to share two resources (TIPB with two strobe lines, and an API bus with data and control strobe lines) between two bus masters (the MPU memory interface and the DMA controller). For more information, see Chapter 5, *Interconnect*.

The 4M bytes of internal static RAM and ROM can be extended with an external component via the external memory interface (EMIF), which manages memory adaptation. The EMIF supports most common memory interface protocols. A flexible programming and timing control allows the support of any type of internal or external IC target module (see Chapter 11, *EMIF*).

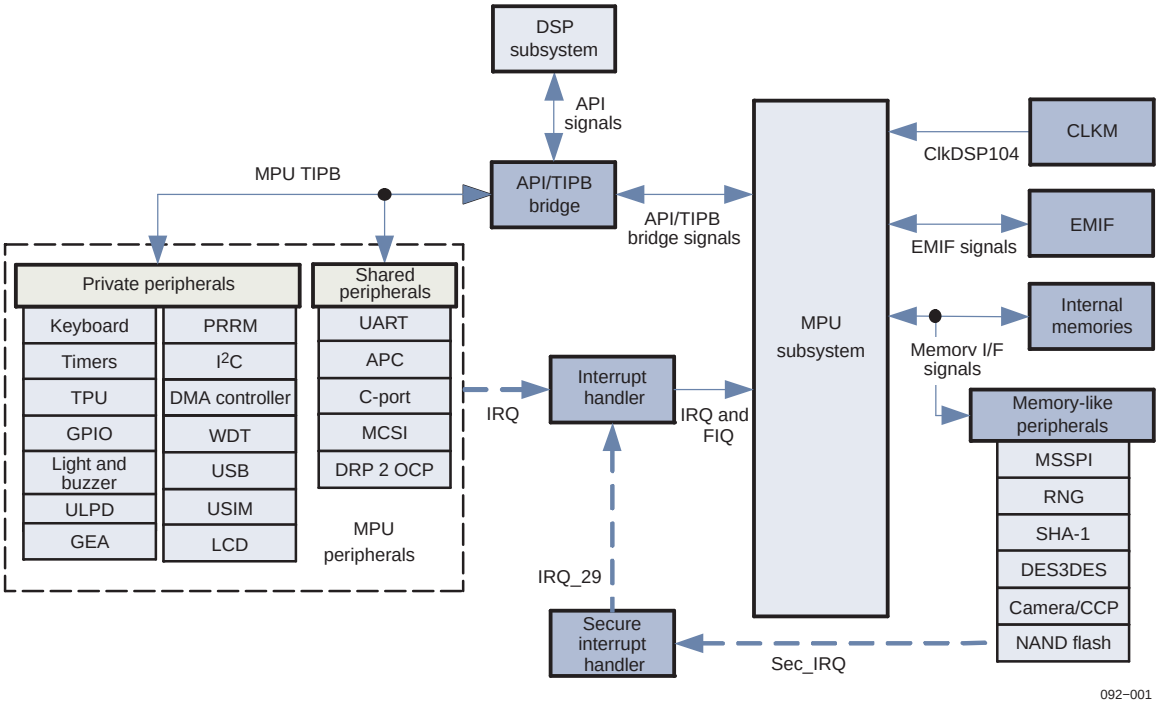
A set of peripherals are mapped as memory:

- SHA-1
- DES3DES
- RNG
- MSSPI
- Camera/CCP
- NAND flash interface

Note: Camera/CCP is not available on the LOCOSTO Lite device.

Figure 3-1 shows the MPU subsystem.

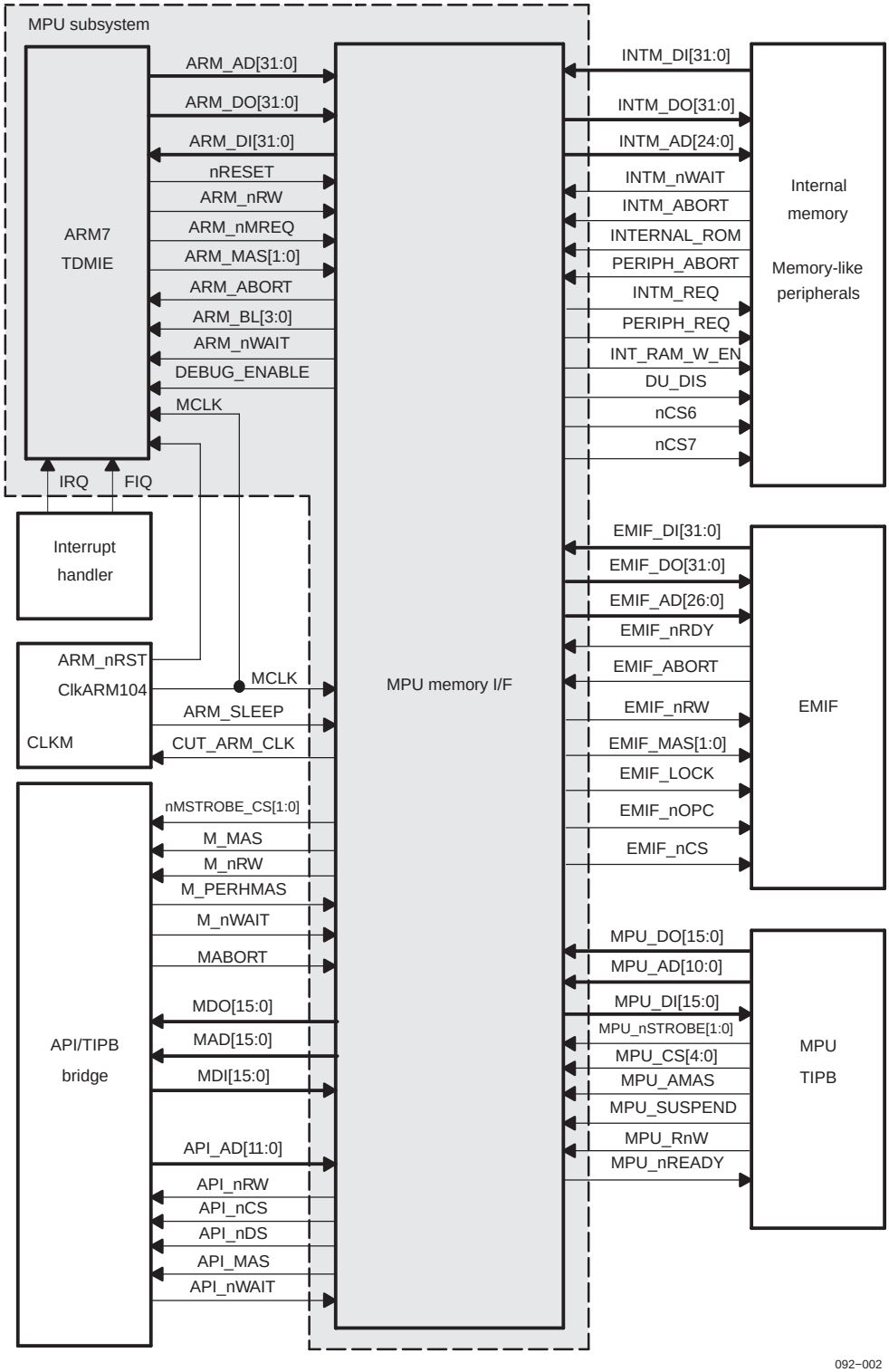
Figure 3-1. MPU Subsystem Overview



3.1.1 Signals and I/O Description

Figure 3-2 shows the external and internal signals of the MPU subsystem.

Figure 3-2. MPU Subsystem Interface Signals



3.1.1.1 MPU Signals

Table 3-1 describes the internal signals between the ARM core and the memory interface in the MPU subsystem.

Table 3-1. MPU Interface Signals

Name	Description
ARM_AD[31:0]	32-bit address bus
ARM_DI[31:0]	Unidirectional bus used to transfer instructions and data from the memory interface to the ARM core
ARM_DO[31:0]	Unidirectional bus used to transfer data from the ARM core to the memory interface
MCLK	Master clock (MPU clock)
nRESET	Master reset signal, active low
ARM_nMREQ	Not memory request. When low, it indicates that the MPU requires a memory access during the following cycle.
ARM_nRW	Not read/write. When high, indicates a processor write access; otherwise, the MPU performs a read access.
ARM_MAS[1:0]	Memory access size 00: 1-byte transfer 01: 2-byte transfer 10: 4-byte transfer 11: Reserved
ARM_ABORT	Memory abort. Set high if the current access has been aborted. An MPU access is aborted if an invalid address is detected, if a write access to a read-only device is done, or if a TIPB time-out condition occurs.
ARM_BL[3:0]	Byte latch. These signals control when data and instruction bytes are latched from the API bus. BL[3] high: MPU DI[31:24] latched on MCLK falling edge BL[2] high: MPU DI[23:16] latched on MCLK falling edge BL[1] high: MPU DI[15:8] latched on MCLK falling edge BL[0] high: MPU DI[7:0] latched on MCLK falling edge
ARM_nWAIT	MPU is held, active low.
DEBUG_ENABLE	Control MPU debug enable signal

3.1.1.2 Internal Memory and Memory-Like Peripherals Interface Signals

Table 3-2 describes the signals between the internal memory/memory-like peripheral interface and the MPU memory interface.

Table 3-2. Internal Memory Interface Signals

Name	Description
INTM_DI[31:0]	Internal memory input data bus
INTM_DO[31:0]	Internal memory output data bus
INTM_AD[24:0]	Internal memory address bus
INTM_nWAIT	Internal memory nwait signals
INTM_ABORT	Internal memory abort signals
PERIPH_ABORT	Peripheral abort signals
INTM_REQ	Asynchronous internal main memory request
PERIPH_REQ	Asynchronous memory-like peripheral request
INT_RAM_W_EN	Internal RAM write enable from TIPB configuration register
DU_DIS	Debug unit control signal
INTERNAL_ROM	When high, CS0 is considered as an internal ROM.

Table 3-2. Internal Memory Interface Signals (continued)

Name	Description
nCS6	Internal memory chip-select, active low
nCS7	Memory-like peripherals chip-select, active low

3.1.1.3 EMIFS Signals

Table 3-3 describes the signals between the EMIF module and the MPU memory interface.

Table 3-3. EMIF Signals

Name	Description
EMIF_DIN[31:0]	External memory input data bus
EMIF_DOUT[31:0]	External memory output data bus
EMIF_AD [26:0]	External memory address bus
EMIF_nRDY	EMIF wait signal
EMIF_ABORT	External memory abort signals
EMIF_nRW	Not read and write signal
EMIF_MAS[1:0]	Access size
EMIF_LOCK	Locked access indicator to implement read-modify write instructions
EMIF_nCS	External memory chip-select, active low
EMIF_nOPC	Locked access indicator to implement read-modify write instructions

3.1.1.4 API/TIPB Bridge Interface Signals

Table 3-4 and Table 3-5 describe the signals between the API/TIPB bridge and the MPU memory interface.

The MPU API/TIPB interface signals are composed of TIPB signals and API signals. These sets of signals are connected to the bridge, which manages the connection between the shared and private peripherals, the API memory, the DMA controller, and the MPU.

3.1.1.4.1 API Signals

Table 3-4 describes the API signals. For more information about the API bus, see Chapter 5, *Interconnect*.

Table 3-4. API Bus Signals From the MPU to the API/TIPB Bridge

Name	Description
API_AD[11:0]	API memory address bus. Valid during all API accesses.
API_nWAIT	This flag is delivered by the API to postpone the MPU access during the API access. Active low.
API_nRW	Transaction type: 0: Read 1: Write
API_nCS	API memory chip-select for control access. Active low.
API_nDS	API memory chip-select for data access. Active low.
API_MAS	API word access length: 0: 8 bits 1: 16 bits

3.1.1.4.2 TIPB Signals

Table 3-5 describes the TIPB signals between the MPU memory interface and the API/TIPB bridge. For more information on the TIPB, see Chapter 5, *Interconnect*.

Table 3-5. MPU TIPB Signals From the MPU to the API/TIPB Bridge

Name	Description
MDO[15:0]	Data output bus. Used for TIPB and API write access.
MDI[15:0]	Data input bus. Used for TIPB and API read access.
MAD[15:0]	Address in the TIPB memory range
M_nRW	Transaction type: 0: Read 1: Write
M_MAS	Memory access size: 0: 8 bits 1: 16 bits
M_PERHMAS	Peripheral accessed size: 0: 8 bits 1: 16 bits
nMSTROBE_CS[1:0]	TIPB chip-select. Active low.
nMWAIT	This flag is delivered by the API/TIPB bridge to postpone the MPU access during a TIPB access. Active low.
MABORT	Current access is aborted. Active high.

3.1.1.5 MPU TIPB Signals

Table 3-6 describes the MPU TIPB signals. For more information on the TIPB, see Chapter 5, *Interconnect*.

Table 3-6. MPU TIPB Signals From the API/TIPB Bridge to Peripherals

Name	Description
MPU_AD[10:0]	Address 2K bytes per peripheral
MPU_CS[4:0]	Chip-select allows connecting up to 32 peripherals.
MPU_DO[15:0]	Data from bridge to peripheral Clocked in peripheral on nSTROBE rising edge (if READY sent by the peripheral is active)
MPU_DI[15:0]	Date from peripheral to bridge Stored in bridge on ClkBridge52 rising edge related to nSTROBE rising edge (if READY sent by the peripheral is active)
MPU_RnW	Transaction type: 1: Read 0: Write

Table 3-6. MPU TIPB Signals From the API/TIPB Bridge to Peripherals (continued)

Name	Description
MPU_nSTROBE[1:0]	Output intended to drive the clocks of all attached peripherals; these clocks are active only when data are being transferred. Each clock addresses a different memory space. The pulse duration depends on the capabilities of the peripherals. It can be chosen with the access factor parameters, which are stored in TIPB_CNTL, one for each strobe line. Fast peripherals: Access factor = 0 nSTROBE is generated from the ClkBridge52 clock. The nSTROBE low-level pulse duration is the ClkBridge52 low-level duration. In case of an access with 0 wait-state, only one pulse is generated. Otherwise, one additional pulse is generated per wait-state inserted by the target. Slow peripherals: Access factor ≠ 0 In this case, the access factor represents the number of ClkBridge52 periods used to generate a low-level or high-level pulse for nSTROBE. Wait-states can also be inserted by the target.
MPU_READY	Peripheral ready to accept or send data. Active low. It is evaluated by the bridge on the ClkBridge52 rising edge related to the nSTROBE rising edge.
MPU_SUSPEND	Indicates that the processor has suspended execution for an emulation breakpoint. Active low. May be used by a peripheral to suspend the internal state-machines.
MPU_AMAS	Memory access length: 0: 8 bits 1: 16 bits
MPU_PERHMAS	Peripheral word access length Allows the bridge to determine the word size of the addressed peripheral 0: 8 bits 1: 16 bits

3.1.2 Clocking, Reset and Power-Management Scheme

3.1.2.1 Clocks

The MPU subsystem module has two clocks:

- ClkARM104: Typically 104 MHz; comes from the CLKM
- ClkARM52: Typically 52 MHz; divided from the ClkARM104 clock

The MPU memory interface works in a dual-frequency because the internal and external memory accesses work at 104 MHz, and the remaining domains (API, TIPB, and memory-like peripherals) operate at 52 MHz. Therefore, the incoming clocks are adapted for slow mechanisms (see Chapter 6, *Power, Reset, and Clock Management*).

Table 3-7 describes the clock signals for the MPU memory subsystem.

Table 3-7. MPU Subsystem Clock

Type	Name	Source	Frequency	Description
Functional	ClkARM104	CLKM	104 MHz	Used for ARM core and MPU memory interface
Functional	ClkARM52	ClkARM 104	52 MHz	Used to interface with buses and slower peripherals such as TIPB and external memory-like peripherals

3.1.2.2 Power Management

Table 3-8 describes the CLKM interface signals.

Table 3-8. CLKM Interface Signals

Name	I/O	Description
ARM_SLEEP	I	Set to 1 by the CLKM unit during the MPU sleep mode sequence before it cuts the MPU clock. The memory interface ends the current access and grants it with CUT_ARM_CLK.
CUT_ARM_CLK	O	This signal is set to 1 when the memory interface is ready to enter the MPU sleep mode.

3.1.2.3 Hardware and Software Resets

Table 3-9 describes the reset strategy for this module. For more details about these signals, see Chapter 6, *Power, Reset, and Clock Management*.

Table 3-9. MPU Subsystem Reset

Type	Name	Source	Activation	Domain
Hardware	ARM_nRST	CLKM	0	Global reset of ARM core and memory interface registers

3.1.3 Hardware Requests

The MPU subsystem does not generate interrupt or DMA requests but receives interrupts from the MPU interrupt handler (see Table 3-10).

Table 3-10. Hardware Requests

Type	Name	Source	Description
Abort	ABORT	PRRM	Abort signal generated by PRRM (see Chapter 6, *Power, Reset, and Clock Management*)
Interrupt	FIQ	MPU interrupt handler	Fast interrupt request
Interrupt	IRQ	MPU interrupt handler	Interrupt request lower priority than FIQ

3.2 MPU Subsystem Functional Description

3.2.1 Block Diagram

The MPU subsystem is built around two main modules:

- AMR7TDMIE
- Memory interface

The ARM7TDMIE consists of the following features:

- 32-/16-bit RISC architecture (ARM v4T)
- 32-bit ARM instruction set for maximum performance and flexibility
- 16-bit thumb instruction set for increased code density
- Unified bus interface; 32-bit data bus carries both instructions and data
- Three-stage pipeline
- 32-bit arithmetic logic unit (ALU)

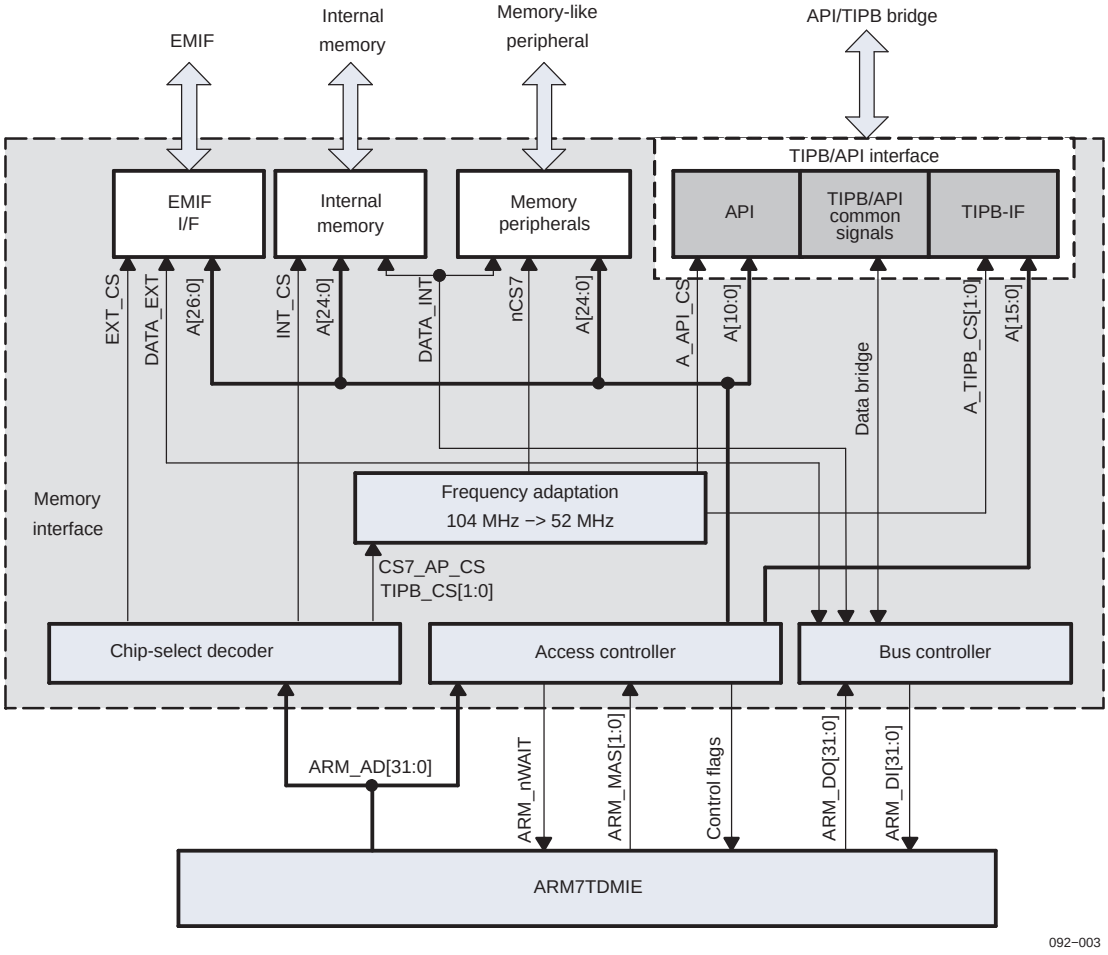
For more details, see the ARM7TDMIE reference manual.

The MPU memory interface is responsible for the following:

- Internal MPU memory access management
- Internal MPU memory-like peripheral access management: fixed 32-bit access
  - Request adaptation to 52-MHz peripheral clock
- EMIF access management
- MPU-to-API memory access management:
  - MPU access size adaptations for API read and write access
  - Address signal-timing adaptation to comply with the API requirement
  - Request adaptation to the 52-MHz bridge-API clock
- MPU-to-TIPB bridge access management:
  - MPU access size adaptation for TIPB access. The access size adaptation is done regarding the TIPB peripheral minimal transaction size. If the TIPB peripheral cannot handle byte transaction, then no byte write transaction can be done. The ADAPT1 bit MPU.API_ RHEA_CNTL[3] and ADAPT0 bit MPU.API_RHEA_CNTL[1] control registers are used for this purpose.
  - Request adaptation to the 52-MHz bridge-TIPB clock
- MPU nWAIT and access control flags generation (byte-latch, etc.)

Figure 3-3 shows an overview of the MPU subsystem with interfaces that allow connections to external buses and modules.

Figure 3-3. MPU Subsystem Block Diagram



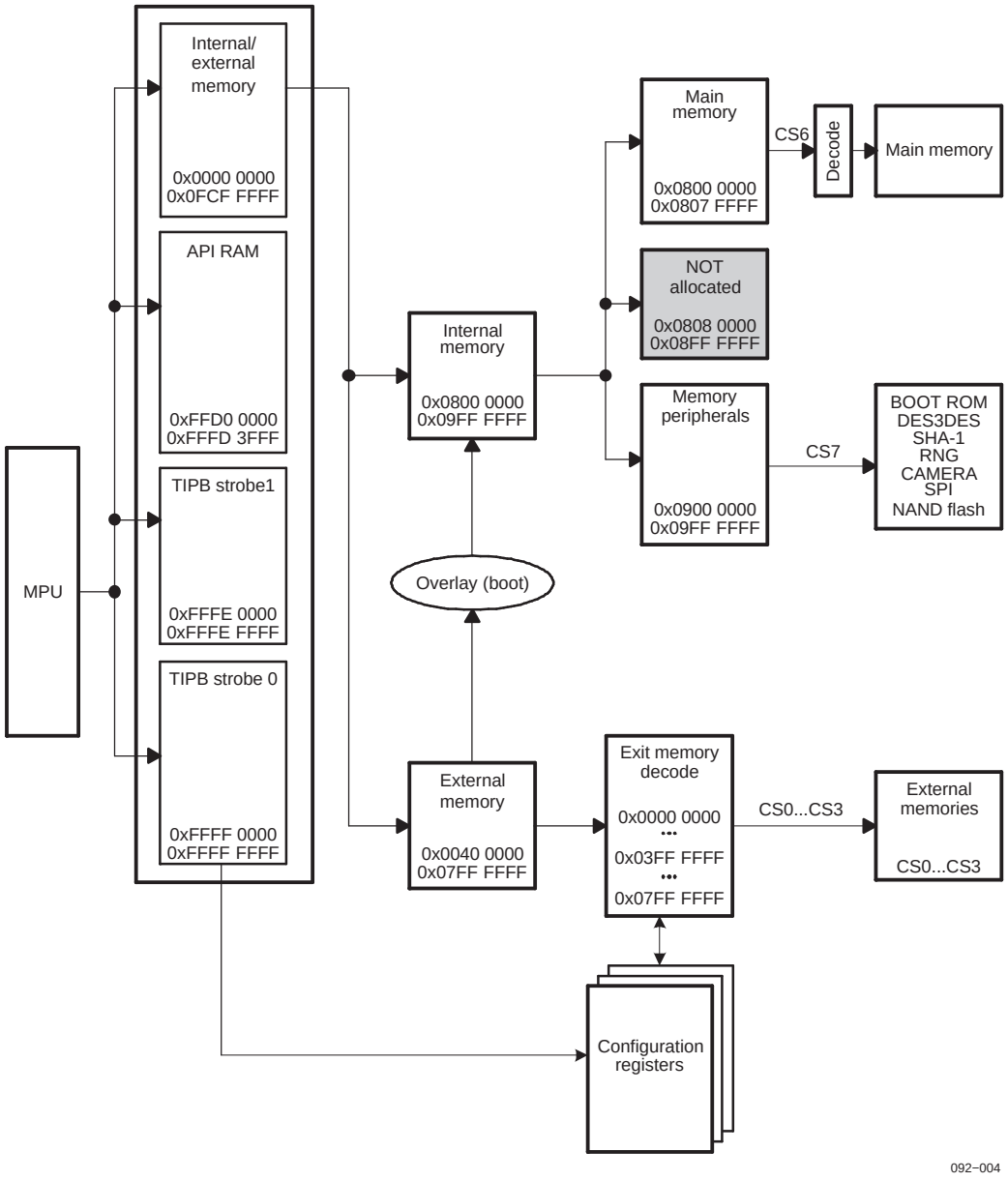
3.2.2 Memory Organization

The MPU memory interface is responsible for the program/data and control and status accesses with the off-chip and on-chip memories and the memory-mapped resources. The memory space is divided into five parts. Figure 3-4 shows an overview of the MPU memory space.

The MPU memory interface also supports communication between the MPU processor and the internal memory/peripheral, EMIF, API, the bridge TIPB, TIPB, and the CLKM interfaces.

Figure 3-4 describes MPU memory mapping.

Figure 3-4. Address Decoding and Range



NOTE: Camera/CCP is not available on the LOCOSTO Lite device.

For information on the MPU memory mapping addresses, see Table 2-1 in Chapter 2, *Memory Mapping*.

3.2.3 Chip-Select Decoder

3.2.3.1 External Memory Chip-Select

The synchronous/asynchronous EMIF supports most common memory interface protocols through flexible programming and control of the timing signals.

The EMIF can control up to four memory devices for a total address range of 124M bytes (1 x 28M bytes + 3 x 32M bytes), without adding any external logic.

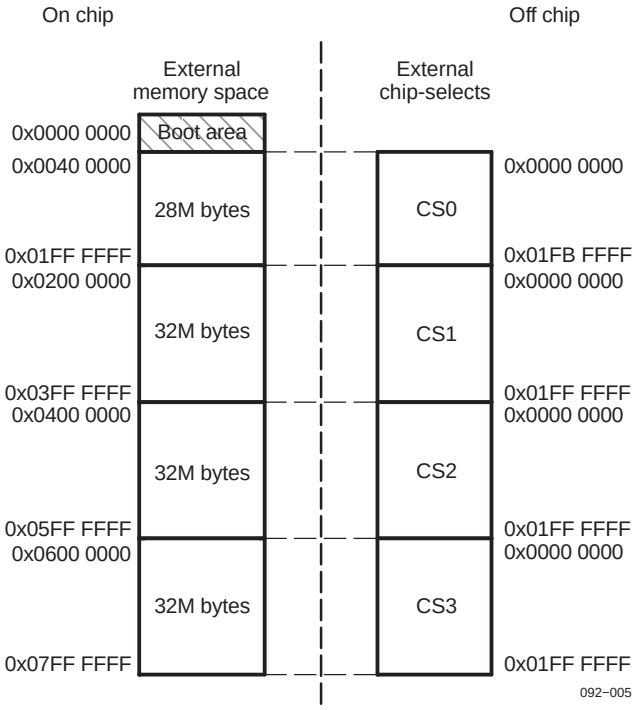
The number of memory chip-select signals is configurable from one to four (the default is two chip-selects).

The address space of each chip-select has dedicated wait-state and/or dummy-cycle insertion configuration registers to fulfill the protocol and timing constraints of the attached memory devices.

The MPU memory interface (MPU-EMIF) does not perform an external memory decode. It depends on the EMIF module to manage the external accesses. The MPU-EMIF simply decodes the request. If the request lies within the memory map for the external memory, the MPU-EMIF passes the address bus, other control signals, and the external memory chip-select (EXT_nCS) to the EMIF module to manage the request.

Four chip-selects (CS0…CS3) are provided for external memories. Figure 3-5 describes the external memory mapping.

Figure 3-5. External Memory Organization



3.2.3.2 Internal Memory Chip-Select

Internal memory is 4M bits of on-chip memory used as the program and data. It is accessed in read and write mode in 8-, 16-, or 32-bit format. It is divided into 2.5M bits of SRAM and 1.5M bits of ROM. The first 2M bits of RAM can be partially or entirely protected with the enhanced memory protection unit (EMPU) and cleared at power-on reset. The remaining 0.5M bits of RAM cannot be used as protected memory and cannot be reset. Writes to RAM are done with the help of a write buffer, to avoid any wait-state generation to the MPU. No writes are permitted in ROM. An abort is generated if an attempt is made to write in the ROM area.

The internal memory can be accessed by the MPU or DMA. The arbitration between the MPU access and DMA access is done with the use of the DMA arbiter. This makes these accesses transparent to the internal memory. The arbiter generates the required address, data, and control signal for reading from or writing to the internal memory.

The access control signal is generated from the EMPU to disable any access in case of protection fault. When access is disabled, no write occurs in the memory; in case of a read, all zeros are driven on the data bus.

3.2.3.3 Internal Memory-Like Peripherals Chip-Select

The internal memory-like peripheral interface is used as an alternative to the TIPB interface to connect high-speed modules (no strobe, no access factor, and 32-bit size).

This memory space is divided into two sections:

- The boot ROM memory

  The device boot sequence is executed from the internal 64K-byte boot ROM. The boot ROM is connected to the internal-memory slow bus and is then accessed for execution in read mode up to 52 MHz in 8-, 16-, or 32-bit format. The boot ROM content is protected and cannot be read by “firmware” code. Basically, the secure boot ROM code authenticates and verifies the integrity of the external firmware or the download FLASH programmer before it is executed on the device.

  In secure mode, the boot space always maps to the secure ROM. For more information on the booting features of the LOCOSTO, see Chapter 30, *Initialization*.
- The memory-like peripherals

3.2.4 Access Controller

The access controller controls the following:

- Access size adaptation for API transactions:

  To allow the MPU to execute code from the API memory, the MPU memory access size (8, 16, or 32 bits) must be adapted to the API memory size (16 bits).

> **CAUTION**
>
> For API 32-bit access, the adaption is done in the MPU memory interface, and the access is split into two 16-bit transactions. The ADAPTAPI bit MPU.API_RHEA_CNTL[5] must be set to 1.

- Access size adaptation for TIPB transactions:

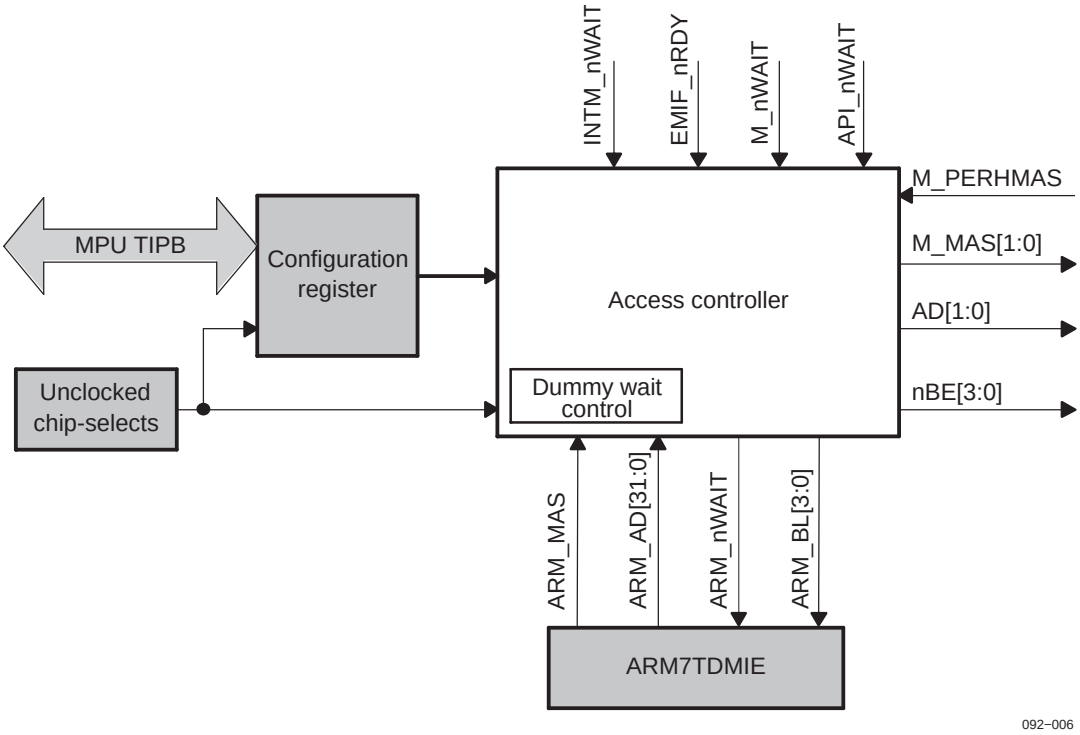
  For the API access size adaptation, the 32-bit MPU access can be split into two 16-bit TIPB accesses. This adaptation can be done independently for TIPB strobe 0 and TIPB strobe 1.

> **CAUTION**
>
> To enable the adaptation of the 32-bit MPU to 16-bit TIPB, set the ADAPT0 bit MPU.API_RHEA_CNTL[1] and the ADAPT1 bit MPU.API_RHEA_CNTL[3] for TIPB strobe 0 and TIPB strobe 1, respectively.

- Access control flag generation (ARM_BL, nWAIT, nBE, MPU_AMAS)
- Access control signal generation (API_nRW, API_MAS for the API access)

Figure 3-6 shows the access controller.

Figure 3-6. Access Controller



3.2.4.1 Wait-States

Wait-states are introduced to stall the MPU when a peripheral takes more than one cycle to serve a request.

The MPU address mapping can access various types of peripherals/ memories and can encounter disparate numbers of wait cycles. The types of MPU accesses and the corresponding stall reasons are as follows:

- External memory access

  All external access requests are forwarded to the EMIF module to serve. This block works on a preprogrammed timing specification for each chipselect to access the external memory. Therefore, as long as the external memory does not respond, the MPU is held in stall.
- Internal memory access

  The internal memory space, which comprises 2.5M bits SRAM and 1.5M bits ROM, is accessible by both the MPU and the DMA. Internal memory accesses are 0 wait-state. This condition is not true in two situations and generates a stall:
  - If the DMA is also accessing internal memory and the MPU loses arbitration
  - In case of a PRRM reset, this reset is always generated at reset and on other conditions, such as protection fault, etc. The internal memory is reset to all zeros. Under this condition, if the MPU attempts an access to the 2M bits of protected internal memory space, its access is stalled. Access to the other 0.5M bits and ROMs is not stalled, however. For more information on the PRRM, see Chapter 6, *Power, Reset, and Clock Management*.
- Memory-like peripheral access

  The memory-like peripherals are instantiated directly on the MPU bus and are accessible in a low-latency mode by both the MPU and the DMA. The MPU bus is in the 52-MHz domain and therefore involves access adaptation from the 104-MHz clock domain. This generates dummy waits to the MPU. The wait to the MPU can also be generated on arbitration lost to the DMA or on a stall generated by the secure RAM.
- TIPB/API access

  The number of wait-states to be inserted is unknown. It is controlled by the TIPB bridge by API_nWAIT

and M_nWAIT. These flags are set high during the last MCLK period of the API/TIPB transaction.

The frequency and request management wait-states are inserted just as in the case of memory-like peripherals. To manage the adaptation from 104 MHz to 52 MHz, a dummy wait-state mechanism is built to allow the 52-MHz devices to respond at their normal rate.

3.2.4.2 Address and Byte Control Signals

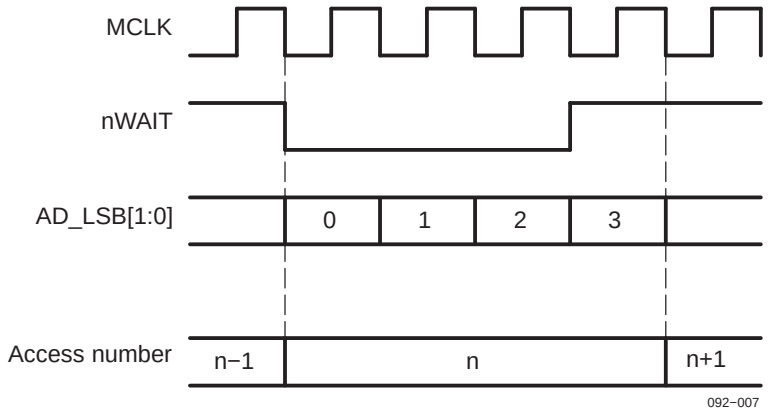
3.2.4.2.1 Address Generation

During a multiple access, the MPU generates only the valid address. Therefore, during a 32-bit MPU data access, the addresses AD[0] and AD[1] are always set to 0; during a 16-bit MPU data access, address AD[0] is always set to 0.

3.2.4.2.2 Address Generation Waveform

Figure 3-7 shows a 0 wait-state device programmed with 4 accesses. In this example, the 8-bit device is accessed by a 32-bit MPU data access.

Figure 3-7. Address Generation for 4 Accesses With 0 Wait-State



During the first access (one clock period in this example), the memory interface generates the first address, and so on until the end of the cycle. When wait-states are inserted, the memory interface waits for the API_nWAIT or M_nWAIT flag to generate the next address.

3.2.4.2.3 Address Generation Table

The memory interface generates addresses depending on the device data size and the MPU data access (see Table 3-11).

Table 3-11. Address Generation (Multiple Access)

MPU Access Size	Device Size	Access Number	Device Address	
			AD[1]	AD[0]
8	8	1	ARM_AD[1]	ARM_AD[0]
16	8	1	ARM_AD[1]	0
		2	ARM_AD[1]	1
32	16	1	ARM_AD[1]	0
32	8	1	0	0
		2	0	1
		3	1	0
		4	1	1

Table 3-11. Address Generation (Multiple Access) (continued)

MPU Access Size	Device Size	Access Number	Device Address	
			AD[1]	AD[0]
32	16	1	0	0
		2	1	0
32	32	1	0	0

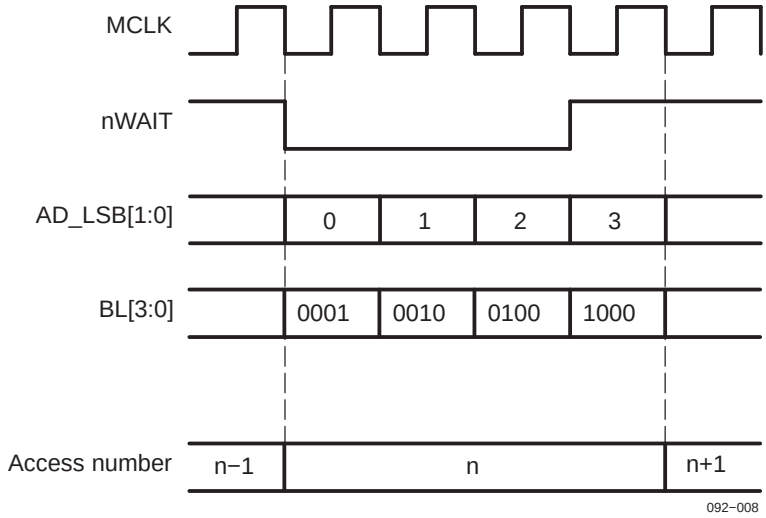
For TIPB accesses, the target peripheral size is unknown during the first access. Therefore, the first access is done assuming that the peripheral is 16 bits. At the end of the access, if the M_PERHMAS signal is at 1, the access ends; otherwise, it continues with the second 32-bit-to-8-bit access configuration.

3.2.4.3 Byte-Latch Generation

The byte-latch signals are generated during a read cycle to latch the byte or half-word in the right byte lane inside the MPU core. They are generated by the memory interface during multiple accesses at the same time as the least significant bit (LSB) address. These signals control the data and instruction latch.

Figure 3-8 shows a 0 wait-state device programmed with a 4 access number (8-bit device accessed with a 32-bit MPU data access).

Figure 3-8. Byte-Latch Generation for 4 Accesses With 0 Wait-State



The memory interface generates byte-latch depending on the device transaction size and the generated address LSBs.

Table 3-12. Byte-Latch Generation (32-Bit MPU Access)

		BL[3:0]		
		Access Size		
AD[1]	AD[0]	8 Bits	16 Bits	32 Bits
0	0	0001	0011	1111
0	1	0010	- - -	- -
1	0	0100	1100	- -
1	1	1000	- -	- -

3.2.4.4 Byte Selection for 16- or 32-Bit Memory Device

Byte access to an external 16- or 32-bit device requires supplementary control signals (nBE[3:0]) (see Table 3-13). These signals allow individual bytes to be write or read.

Table 3-13. nBE for 16-Bit Device

Access	A[0]	nBE[3:0]
8 bits	0	10
8 bits	1	01
16 bits	X	00

Table 3-14. nBE for 32-Bit Device

Access	A[1: 0]	nBE[3:0]
8 bits	00	1110
8 bits	01	1101
8 bits	10	1011
8 bits	11	0111
16 bits	0X	1100
16 bits	1X	0011
32 bits	XX	0000

For timing reasons, the nBE generation for internal memory and memory-like peripherals is latched (transparent on the low level of MCLK) instead of being registered on the falling edge of MCLK.

3.2.5 Bus Controller

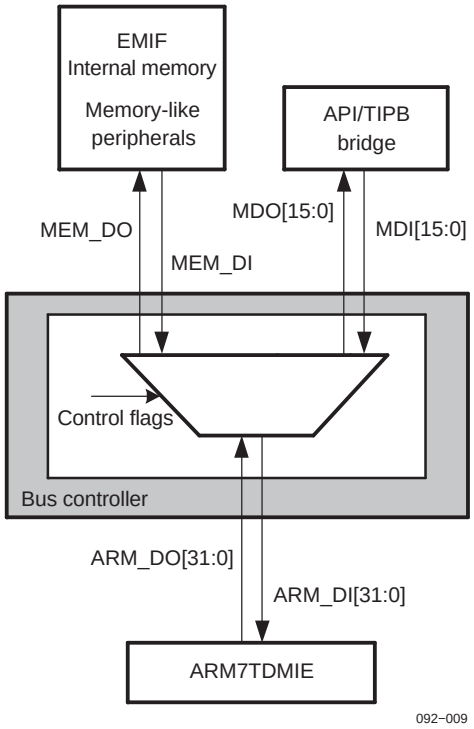
The MPU core consists of two 32-bit unidirectional data buses that must be connected to one of the following buses:

- External 32-bit data bus
- Internal TIPB/API 16-bit data bus

The bus controller switches the internal/external data buses to/from the ARM core, depending on the device data size and the MPU data access mode.

Figure 3-9 shows the functions of the bus controller.

Figure 3-9. Functional Description of the Bus Controller



The data on the output buses (MEM_DO, ARM_DO, and MDO) depend on the MPU access size and on the device access size. Table 3-15 through Table 3-18 list the positions of the data in the buses.

Table 3-15. ARM_DO→MEM_DO Adaptation

MPU Access Size	Device Access Size	Address LSB[1:0]	MEM_DO [31:24]	MEM_DO [23:16]	MEM_DO [15:8]	MEM_DO [7:0]
8	8	00	-	-	-	ARM_DO [7:0]
	16	00	-	-	ARM_DO [15:8]	ARM_DO [7:0]
	32	00	ARM_DO [31:24]	ARM_DO [23:16]	ARM_DO [15:8]	ARM_DO [7:0]
16	8	00	-	-	-	ARM_DO [7:0]
		01	-	-	-	ARM_DO [15:8]
	16	00	-	-	ARM_DO [15:8]	ARM_DO [7:0]
	32	00	ARM_DO [31:24]	ARM_DO [23:16]	ARM_DO [15:8]	ARM_DO [7:0]
32	8	00	-	-	-	ARM_DO [7:0]
		01	-	-	-	ARM_DO [15:8]
		10	-	-	-	ARM_DO [23:16]
		11	-	-	-	ARM_DO [31:24]
	16	00	-	-	ARM_DO [15:8]	ARM_DO [7:0]
		10	-	-	ARM_DO [31:24]	ARM_DO [23:16]
	32	00	ARM_DO [31:24]	ARM_DO [23:16]	ARM_DO [15:8]	ARM_DO [7:0]

Table 3-16. ARM_DO→MDO Adaptation

MPU Access Size	Device Access Size	Address LSB[1:0]	MDO[15:8]	MDO[7:0]
8	8	00	-	ARM_DO[7:0]
	16	00	ARM_DO[15:8]	ARM_DO[7:0]
16	8	00	-	ARM_DO[7:0]
		01	-	ARM_DO[15:8]
	16	00	ARM_DO[15:8]	ARM_DO[7:0]
32	8	00	-	ARM_DO[7:0]
		01	-	ARM_DO[15:8]
		10	-	ARM_DO[23:16]
		11	-	ARM_DO[31:24]
	16	00	ARM_DO[15:8]	ARM_DO[7:0]
		10	ARM_DO[31:24]	ARM_DO[23:16]

Table 3-17. MEM_DI→ARM_DI

Device Access Size	MPU Access Size	Address LSB[1:0]	ARM_DI [31:24]	ARM_DI [23:16]	ARM_DI [15:8]	ARM_DI [7:0]
8	8/16/32	00	MEM_DI[7:0]	MEM_DI [7:0]	MEM_DI [7:0]	MEM_DI [7:0]
16	8	00	MEM_DI[7:0]	MEM_DI [7:0]	MEM_DI [7:0]	MEM_DI [7:0]
		01	MEM_DI [15:8]	MEM_DI [15:8]	MEM_DI [15:8]	MEM_DI [15:8]
	16/32	00	MEM_DI [15:8]	MEM_DI [7:0]	MEM_DI [15:8]	MEM_DI [7:0]
32	8	00	MEM_DI [7:0]	MEM_DI [7:0]	MEM_DI [7:0]	MEM_DI [7:0]
		01	MEM_DI [15:8]	MEM_DI [15:8]	MEM_DI [15:8]	MEM_DI [15:8]
		10	MEM_DI [23:16]	MEM_DI [23:16]	MEM_DI [23:16]	MEM_DI [23:16]
		11	MEM_DI [31:24]	MEM_DI [31:24]	MEM_DI [31:24]	MEM_DI [31:24]
	16	00	MEM_DI [15:8]	MEM_DI [7:0]	MEM_DI [15:8]	MEM_DI [7:0]
		10	MEM_DI [31:24]	MEM_DI [23:16]	MEM_DI [31:24]	MEM_DI [23:16]
	32	00	MEM_DI [31:24]	MEM_DI [23:16]	MEM_DI [15:8]	MEM_DI [7:0]

Table 3-18. MDI→ARM_DI Adaptation

Device Access Size	MPU Access Size	Address LSB[1:0]	ARM_DI [31:24]	ARM_DI [23:16]	ARM_DI [15:8]	ARM_DI [7:0]
8	8/16/32	00	MDI [7:0]	MDI [7:0]	MDI [7:0]	MDI [7:0]
16	8	00	MDI [7:0]	MDI [7:0]	MDI [7:0]	MDI [7:0]
		01	MDI [15:8]	MDI [15:8]	MDI [15:8]	MDI [15:8]
	16/32	00	MDI [15:8]	MDI [7:0]	MDI [15:8]	MDI [7:0]

3.2.6 Error Reporting

Errors are relative to memory access or memory protection. For example, an MPU access to memory is aborted if an invalid address is detected, if a write access to a read-only device is done, or if the TIPB time-out condition occurs.

The access is aborted by setting high the ARM_ABORT signal after receiving one of the following signals:

- INTM_ABORT: The INTM_ABORT signal is generated by the internal memory module when a write attempt is made to the ROM area.

- EMIF_ABORT: The EMIF_ABORT signal is generated in response to an unauthorized write access to the protected memory area or in response to a time-out. Two registers are then used: EMIF_ADDRD (contains the abort address) and EMIF_ATYPER (informs about the abort cause). For more details, see Chapter 11, *EMIF*.
- MABORT: The MABORT signal is generated by a TIPB bridge when the maximum time that a peripheral can stall the MPU is exceeded. The TIMEOUT bit of the TIPB_CNTL[15:8] register is used to define the maximum time, and the TIMEOUT_EN bit of the ENHANCED_TIPB_CNTL[0] register is used to enable or to disable the TIMEOUT feature. For more details, see Chapter 5, *Interconnect*.
- PERIPH_ABORT: The PERIPH_ABORT signal is generated in case of security violations in secure hardware, access violation in peripherals, or illegal access to the internal memory according to the EMPU setting.

3.3 MPU Subsystem Register Manual

3.3.1 Module Register Mapping Summary

Table 3-19. Register Offset Address

Register Name	Type	Register Width (Bits)	Physical Address
API_RHEA_CNTL	R/W	16	0xFFFF FB2E

3.3.2 Register Description

Table 3-20. API_RHEA_CNTL Register

Register Name	Type	Register Width (Bits)	Physical Address
API_RHEA_CNTL	R/W	16	0xFFFF FB2E

Table 3-21. API_RHEA_CNTL

Address Offset	-		
Physical address	0XFFFF FB2E	**Instance**	API_RHEA_CNTL register
Description	API TIPB control		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										ADAPTAPI	RESERVED	ADAPT1	RESERVED	ADAPT0	RESERVED

Bits	Field Name	Description	Type	Reset
15:6	RESERVED	Write has no effect. Read returns 0x0.	R	0x0
5	ADAPTAPI	API access size adaptation 0: No action 1: A 32-bit MPU API access is split into 16-bit API transactions.	R/W	0
4	RESERVED	Write has no effect. Read returns 0x0.	R	0x0
3	ADAPT1	TIPB strobe 1 access size adaptation: 0: No action 1: A 32-bit MPU TIPB access is split into 16-bit TIPB transactions.	R/W	0
2	RESERVED	Write has no effect. Read returns 0x0.	R	0x0
1	ADAPT0	TIPB strobe 0 access size adaptation: 0: No action 1: A 32-bit MPU TIPB access is split into 16-bit TIPB transactions.	R/W	0
0	RESERVED	Write has no effect. Read returns 0x0.	R	0x0

DSP Subsystem

This chapter discusses the digital signal processor (DSP) subsystem of the LOCOSTO device.

Topic **Page**

4.1 DSP Subsystem Overview

The digital signal processor (DSP) subsystem interacts with various modules, including the following:

- ARM port interface (API) to permit high-bandwidth parallel access to the 32K-byte DSP DARAM resources by the microprocessor unit (MPU) and system direct memory access (sDMA)
- XIO/TIPB bridge to provide Texas Instruments peripheral bus (TIPB) interfaces to access DSP private and shared peripherals
- Interrupt handler to configure level sensitivity of the interrupts INT_0 to INT_11 generated by the peripherals
- Clock manager to provide the functional 104-MHz clock for the subsystem

The API is a 16-bit data parallel port used to interface the MPU to the DSP. Information is exchanged through a shared memory accessible by both the MPU and the DSP. The following two methods are available to access the memory:

- Host-only mode (HOM): Only the MPU and DMA controller can access memory.
- Shared access mode (SAM): The DSP, MPU, and DMA controller can access shared memory.

The DSP subsystem has one interrupt handler, which services up to 12 interrupts from internal modules. The DSP interrupt handler allows the level sensitivity to be selected and all interrupts to be routed directly to the DSP core. Contrary to the MPU interrupts, the DSP interrupts are cleared, masked, and prioritized at the DSP core level.

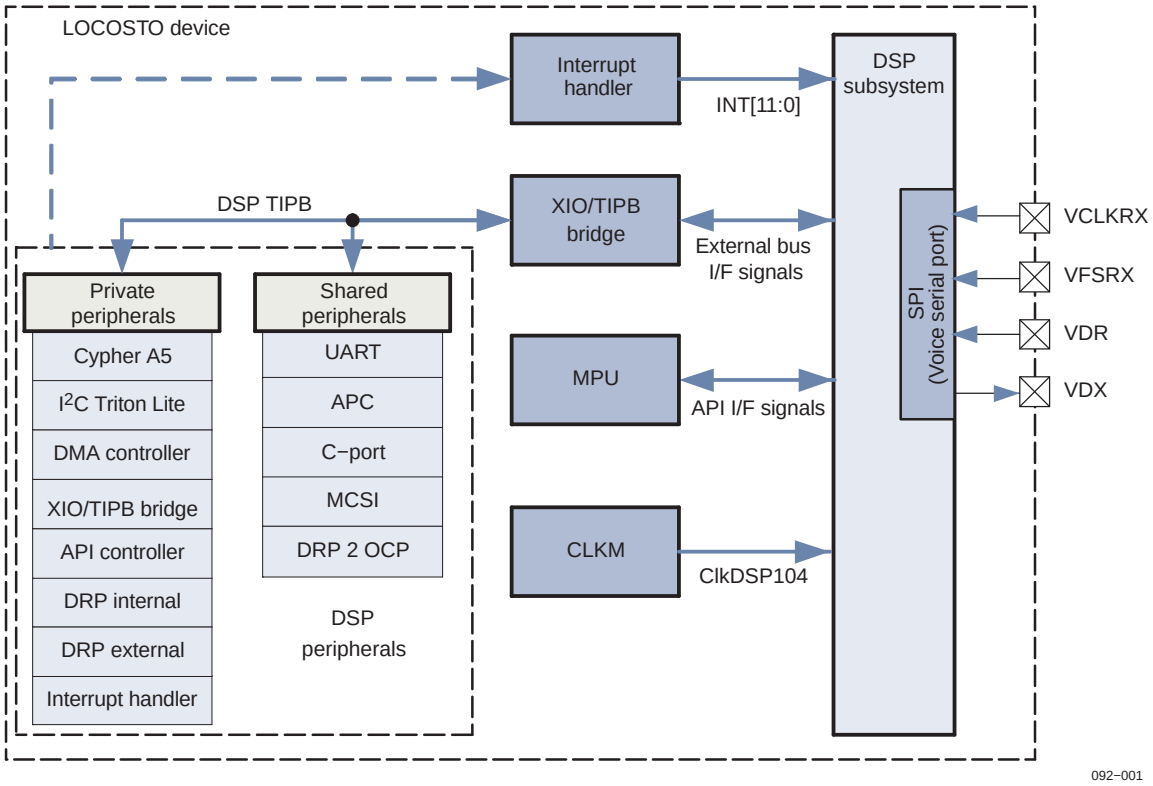
The XIO/TIPB bridge converts DSP requests into TIPB requests to access private or shared peripherals.

- DSP private peripherals:
  - Cipher A5
  - I^2C Triton Lite
  - DRP external/internal memory
  - DMA controller
  - XIO/TIPB bridge control register
  - API controller
  - DSP Interrupt handler
- DSP shared peripherals:
  - DRP 2 OCP memory
  - APC
  - UART
  - MCSI
  - C-port

This chapter describes the API and the memory mapping configuration. The interrupt handler, XIO/TIPB bridge, and peripherals are detailed in their respective chapters.

Figure 4-1 shows an overview of the DSP subsystem and the modules with which it interacts.

Figure 4-1. DSP Subsystem Overview



4.1.1 Signals and I/O Description

Figure 4-2 shows the DSP subsystem I/O.

Figure 4-2. DSP Subsystem I/O Description

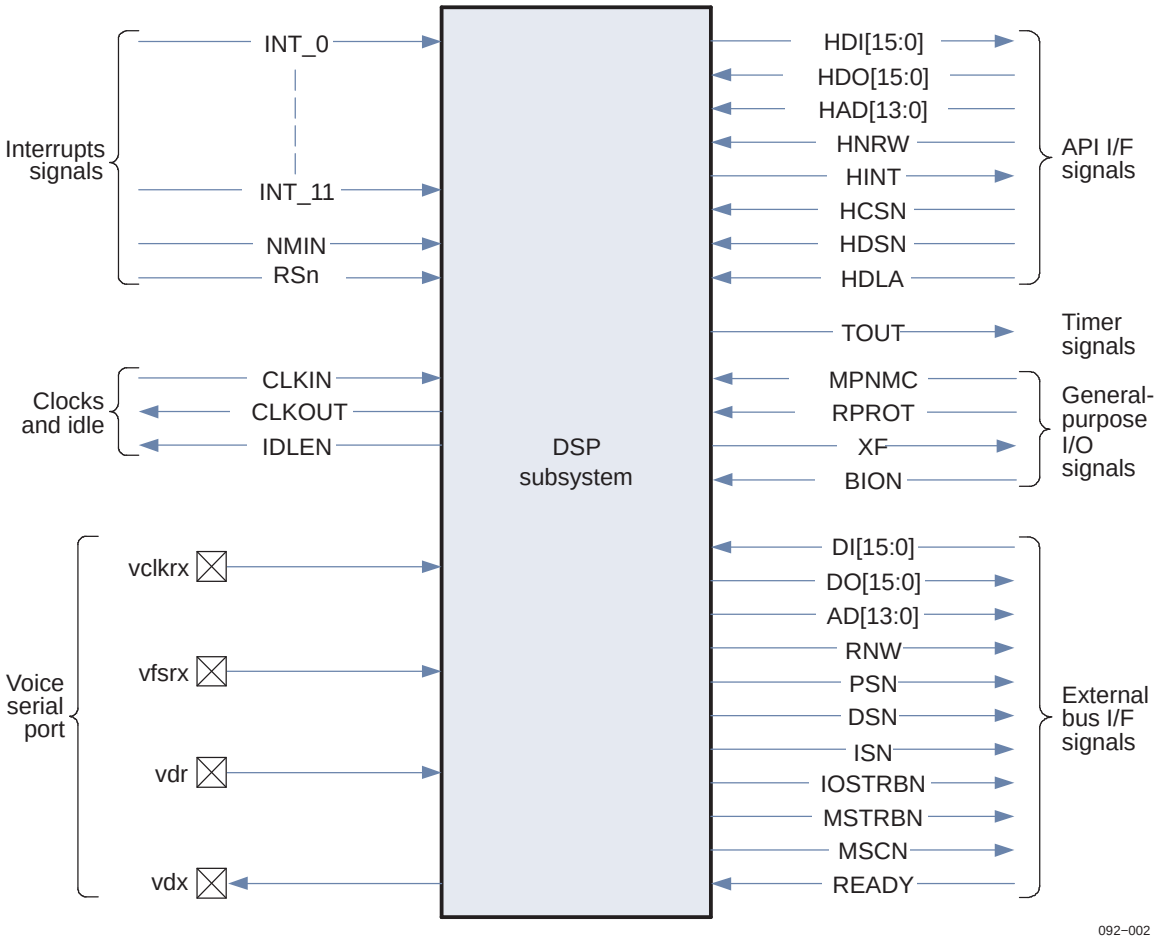


Table 4-1 describes the API I/O.

Table 4-1. API I/O Description

Signal Name	Description
HDI[15:0]	API data bus output
HDO[15:0]	API data bus input
HAD[13:0]	API address bus
HNRW	API read not write, transaction type 0: Read; 1: Write
HINT	API interrupt from DSP to MPU. Active high pulse.
HCSN	API control registers select; indicates API control register access. Active low.
HDSN	API data select; indicates API access to memory. Active low.
HDLA	API data latch enable; indicates end of read cycle (performed with either HCSN or HDSN strobe). Active high.

Table 4-2 describes the buffered serial port (BSP) interface I/O.

Table 4-2. BSP Interface I/O Description

Signal Name	Description
vclkrx	Receive clocks. External clock signal for clocking data from the vdr (data receive) pin into the RSR (serial port receive shift register).
vfsrx	Frame synchronization pulse for receive input. The falling edge of the vfsrx pulse initiates the data receive process, which begins the clocking of the RSR.
vdr	Serial data receive input. Serial data is received in the RSR through the vdr pin.
vdx	Serial port transmit output. Serial data is transmitted from the XSR via the vdx pin.

Table 4-3 describes the timer I/O.

Table 4-3. Timer I/O Description

Signal Name	Description
TOUT	Timer output when the on-chip timer counts down past 0. The pulse is a CLKOUT cycle wide.

Table 4-4 describes the general-purpose I/O.

Table 4-4. General-Purpose I/O Description

Signal Name	Description
MPNMC	Microprocessor/microcomputer mode select pin. If active low at reset (microcomputer mode), the pin causes the internal program ROM to be mapped to the upper words of the program memory space. In the microprocessor mode, the upper words of the program memory are mapped externally.
RPROT	ROM protection signal. Protects internal memories from external access when high.
XF	External flag output (latched software-programmable signal). This signal is set high or low by specific instruction or by loading status register 1 (ST1). Used to signal other processors in multiprocessor configurations or as a general purpose output pin. This pin is set high at reset.
BION	Branch control input. Samples as the BIO condition. If low, the device executes the conditional instruction. The BIO condition is sampled during the decode phase of the pipeline for XC instruction, and all other instructions sample the BIO during the read phase of the pipeline.

Table 4-5 describes the interrupts I/O.

Table 4-5. Interrupts I/O Description

Signal Name	Description
RSn	Reset input. Causes the device to terminate execution and forces the program counter to 0FF80h. When RSn is brought to a high level, execution begins at location 0FF80h of program memory. RSn affects various registers and status bits.
NMIN	Nonmaskable interrupt. External interrupt that cannot be masked via the INTM or the IMR. When NMIN is activated, the processor traps to the appropriate vector location.
INT_0 to INT_11	External user interrupt inputs. Prioritized and maskable by IMR and INTM. Can be polled and reset via the interrupt flag register.

Table 4-6 describes the clocks and idle I/O.

Table 4-6. Clocks and Idle I/O Description

Signal Name	Description
CLKIN	Input clock to the DSP subsystem
CLKOUT	Master clock output signal. This signal cycles at the machine-cycle rate of the CPU. The internal machine cycle is bounded by the rising edges of this signal.
IDLEN	Active low to indicate that the configurable DSP (cDSP) core is in the IDLE3 state.

Table 4-7 describes the TIPB signals from the DSP to the XIO/TIPB bridge.

Table 4-7. TIPB Signals From the DSP to the XIO/TIPB Bridge

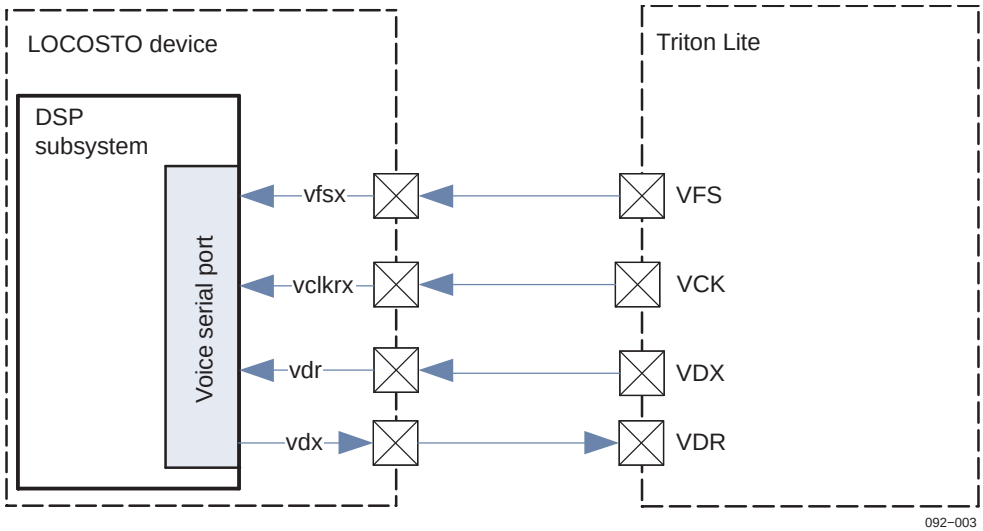
Signal Name	Description
IOSTRBN	I/O strobe signal. Always high unless low level is asserted to indicate an external bus access to an I/O device.
PSN	Program space select signals. Always high unless low level is asserted to communicate to a particular external space.
DSN	Data space select signals. Always high unless low level is asserted to communicate to a particular external space.
ISN	I/O space select signals. Always high unless low level is asserted to communicate to a particular external space.
RnW	Read/write signal. Normally in read mode (high), unless low level is asserted to perform a write operation.
DI[15:0]	Parallel data input bus
DO[15:0]	Parallel data output bus. This signal is not valid when HZDO is active high, but is always driven high or low.
AD[15:0]	Parallel address bus A15(MSB) through A0(LSB). Multiplexed to address external data/program memory or I/O.
READY	Data ready input. Indicates that an external device is prepared for the bus transaction to be completed. If the device is not ready (READY is low), the processor waits one cycle and checks READY again. The processor performs ready detection if at least two software wait-states are programmed. The READY signal is not sampled until the completion of the software wait-states.
MSTRBN	Memory strobe signal. Always high unless low level is asserted to indicate an external bus access to data/program memory.
MSCN	Microstate complete signal. This output goes low when the last wait-state of two or more internal software wait-states programmed is executed. If connected to the READY line, it forces one external wait-state after the last internal wait-state is complete.

4.1.2 Environment

Figure 4-3 shows a typical connection between the LOCOSTO device and the Triton Lite for voice transmission.

For more information about protocols, see *C54x cDSP Reference Set Vol 2: Peripherals*.

Figure 4-3. Typical Connection Between LOCOSTO and Triton Lite for Voice Transmission



4.1.3 Clocking, Reset, and Power-Management Scheme

4.1.3.1 Clocks

Table 4-8 describes the characteristics of the DSP subsystem functional clock.

The DSP subsystem operates from a single and fixed functional clock of 104 MHz generated by the digital phase-locked loop (DPLL) through the CLKM module (for more information, see Chapter 6, *Power, Reset, and Clock Management*).

Table 4-8. DSP Subsystem Clock

Type	Name	Source	Frequency	Description
Functional	CLKIN	ClkDSP104	104 MHz	Used to clock the cDSP core and its internal peripherals

4.1.3.2 Power Management

The DSP subsystem has a power-down mode based on the TMS320C54x cDSP core. In this mode, the subsystem enters into a dormant state and dissipates less power than in normal operation while maintaining the CPU contents. This allows operations to continue unaltered when the power-down mode is terminated.

Power-down modes can be invoked either by executing the IDLE1, IDLE2, or IDLE3 instructions or by driving the HOLDN signal low with the HM status bit set to 1.

- IDLE1 mode halts all CPU activities except the system clock. Because the system clock remains applied to the peripheral modules, the peripheral circuits continue operating and the CLKOUT port remains active. Thus, peripherals such as BSPs and timers can take the CPU out of its power-down state. Use the IDLE1 instruction to enter the IDLE1 mode. To terminate IDLE1, use a wake-up interrupt.
- IDLE2 mode halts the on-chip peripherals as well as the CPU. Because the on-chip peripherals are stopped in this mode, they cannot be used to generate the interrupt to wake up the cDSP core as with the IDLE1 mode. However, power is significantly reduced because the device is completely stopped. Use the IDLE2 instruction to enter the IDLE2 mode. To terminate IDLE2, activate any of the external interrupt ports (RSn, NMIN, and INT_11-INT_0) with a 10-ns minimum pulse.
- IDLE3 mode functions like IDLE2 but can also halt the DPLL. IDLE3 is used to completely shut down the cDSP core. This mode reduces power dissipation more than IDLE2. Use the IDLE3 instruction to enter the IDLE3 mode. To terminate IDLE3, any of the external interrupt ports (RSn, NMIN, and INT_11-INT_0) must be activated with a 10-ns minimum pulse.
- The hold mode is another power-down mode. It enables the address, data, and control lines to be put into the high-impedance state. Depending on the value of the HM bit, the CPU can be halted. This power-down mode is initiated by the HOLDN signal. If HM = 1, the CPU stops executing and address, data, and control lines go into the high-impedance state for additional power reduction. If HM = 0, the address, data, and control signals are put into the high-impedance state, but the CPU continues to execute internally. This mode does not stop the operation of on-chip peripherals, such as the timer and the BSP. This mode is terminated when HOLDN becomes inactive.
- External bus off allows the C54x cDSP core to disable the internal clock of the external bus interface, thus placing the interface into a lower power consumption mode. The external bus interface clock is disabled by setting to 1 the EXIO bit BSCR[0] of the C54x cDSP core.

Table 4-9 lists the idle modes with their power-down functions.

Table 4-9. Idle Mode

Idle Mode	C54x cDSP Core	Timer and BSP	API Memory	Memories
Idle 1	Off	On	On	Off
Idle 2	Off	Off	Off	Off
Idle 3	Off	Off	Off	Off

Note: The API memory can be awakened by the MPU when access is required during an idle.

CAUTION

When IDLE2 or IDLE3 terminates, all peripherals must be reset, especially when externally clocked.

4.1.3.3 Hardware and Software Resets

The DSP subsystem has one hardware reset and two software resets:

- The RSn input performs hardware reset by resetting the C54x cDSP core to terminate execution and forces the program counter (PC) to 0xFF80h. RSn puts the CPU in a predefined state and also affects various CPU/peripherals. RSn is active low and must be active for at least four clock cycles.
- The RESET instruction performs a nonmaskable software reset that can be used at any time to put the C54x cDSP core into a known state. When the RESET instruction is executed, some operations occur. The MPNMC pin is not sampled during this software reset.
- The DSP_RESET bit CNTL_REG[2] in the CLKM domain register performs software reset. To reset the DSP subsystem, set this bit to 1.

Table 4-10 lists the characteristics of each reset.

Table 4-10. DSP Subsystem Resets

Type	Name	Source	Activation	Domain
Hardware	RSn	DSP_nRST	0	Global reset
Software	RESET	Instructions of C54x cDSP core	n/a	Reset C54x cDSP core (see *TMS320C54x DSP Reference Set Vol 2: Mnemonic Instruction Set*).
Software	DSP_RESET	CNTL_REG[2] in CLKM domain registers	1	Global reset (see Chapter 6, *Power, Reset, and Clock Management*).

4.1.4 Hardware Requests

Table 4-11 shows the 14 external sources of interrupts in the DSP subsystem. Table 4-12 shows the DSP subsystem internal interrupts.

Table 4-11. DSP Subsystem External Requests

Type	Name	Source	Destination	Description
Interrupt	RSn	External	DSP subsystem	See Chapter 12, *Interrupt Handlers*.
Interrupt	NMIN	XIO/TIPB bridge	DSP subsystem	
Interrupt	INT_0	DRP	DSP subsystem	
Interrupt	INT_1	DRP	DSP subsystem	
Interrupt	INT_2	UART and DRP	DSP subsystem	
Interrupt	INT_3	MCSI	DSP subsystem	
Interrupt	INT_4	MCSI	DSP subsystem	
Interrupt	INT_5	MCSI	DSP subsystem	
Interrupt	INT_6	MCSI and C-port	DSP subsystem	
Interrupt	INT_7	Cipher A5	DSP subsystem	
Interrupt	INT_8	TPU	DSP subsystem	
Interrupt	INT_9	TPU	DSP subsystem	
Interrupt	INT_10	DMA	DSP subsystem	
Interrupt	INT_11	I^2C	DSP subsystem	

Table 4-12. DSP Subsystem Internal Requests

Type	Name	Source	Destination	Description
Interrupt	TINT	Internal timer	C54x cDSP core	Timer interrupt
Interrupt	RINT	BSP	C54x cDSP core	BSP receive interrupt
Interrupt	XINT	BSP	C54x cDSP core	BSP transmit interrupt
Interrupt	DSP_INT	API	C54x cDSP core	API interrupt generated

4.2 DSP Subsystem Functional Description

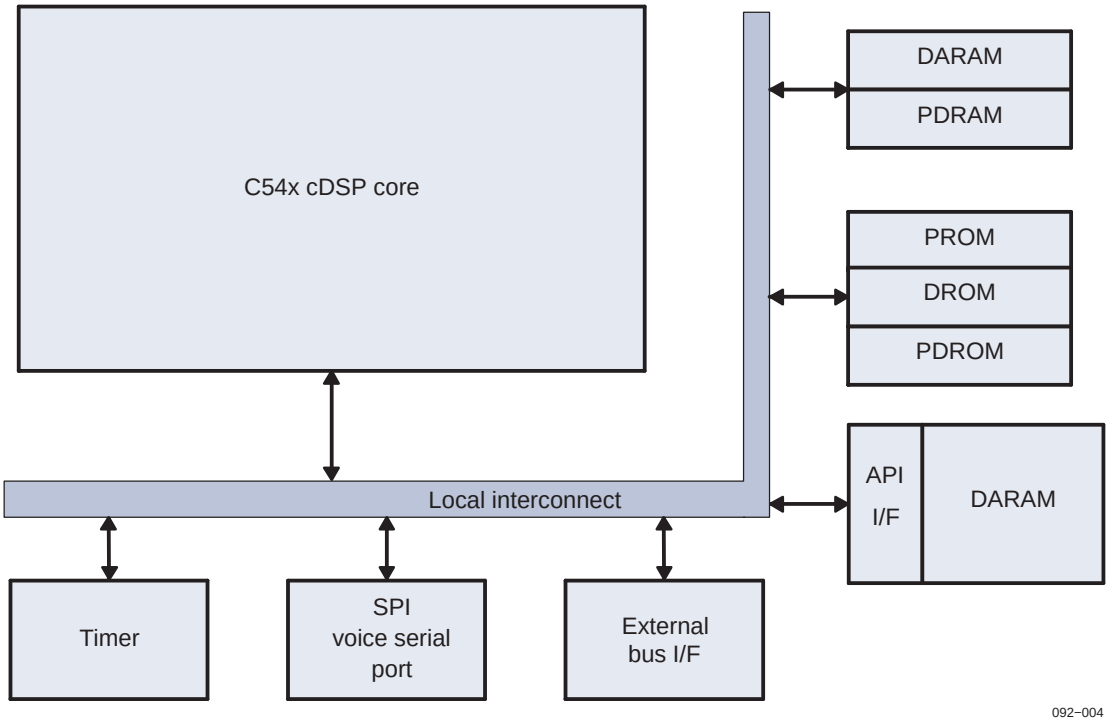
4.2.1 Block Diagram

The DSP subsystem is built around the following elements:

- C54x cDSP core
- Tightly coupled memory blocks of 308K bytes of ROM, 60K bytes of RAM (including API memory), and 20K bytes of extended external RAM
- API to share 32K bytes of RAM between the MPU and the DSP
- Timer
- Serial-port interface (SPI) with Triton Lite for serial voice

Figure 4-4 shows an overview of the DSP subsystem. The link between these modules is an interconnect composed of eight bus addresses and program/data.

Figure 4-4. DSP Subsystem Block Diagram



4.2.2 C54x cDSP Overview

The C54x cDSP architecture is built around the following elements:

- Eight major 16-bit buses (four program/data buses and four address buses):
  - The program bus (PB) carries the instruction code and immediate operands from program memory.
  - Three data buses (CB, DB, and EB) interconnect to various elements, such as the CPU, data address-generation logic, program address-generation logic, on-chip peripherals, and data memory.
  - CB and DB carry the operands read from data memory.
  - EB carries the data to be written to memory.
- Four address buses (PAB, CAB, DAB, and EAB) carry the addresses needed for instruction execution.

The C54x cDSP uses the two auxiliary register arithmetic units (ARAU0 and ARAU1) to generate up to two data memory addresses per cycle.

The PB can carry data operands stored in program space (for example, a coefficient table) to the multiplier and adder for multiply/accumulate operations or to a destination in data space for data move instructions (MVPD and READA). In conjunction with the dual-operand read feature, this capability supports the execution of single-cycle three-operand instructions, such as the FIRS instruction.

The C54x cDSP also has an on-chip bidirectional bus for accessing on-chip peripherals; this bus is connected to DB and EB through the bus exchanger in the CPU interface. Accesses that use this bus can require two or more cycles for reads and writes, depending on the structure of the peripheral.

Table 4-13 summarizes the buses used during the various types of accesses.

Similarly to the standard TMS320C54x devices, the CPU has the following features:

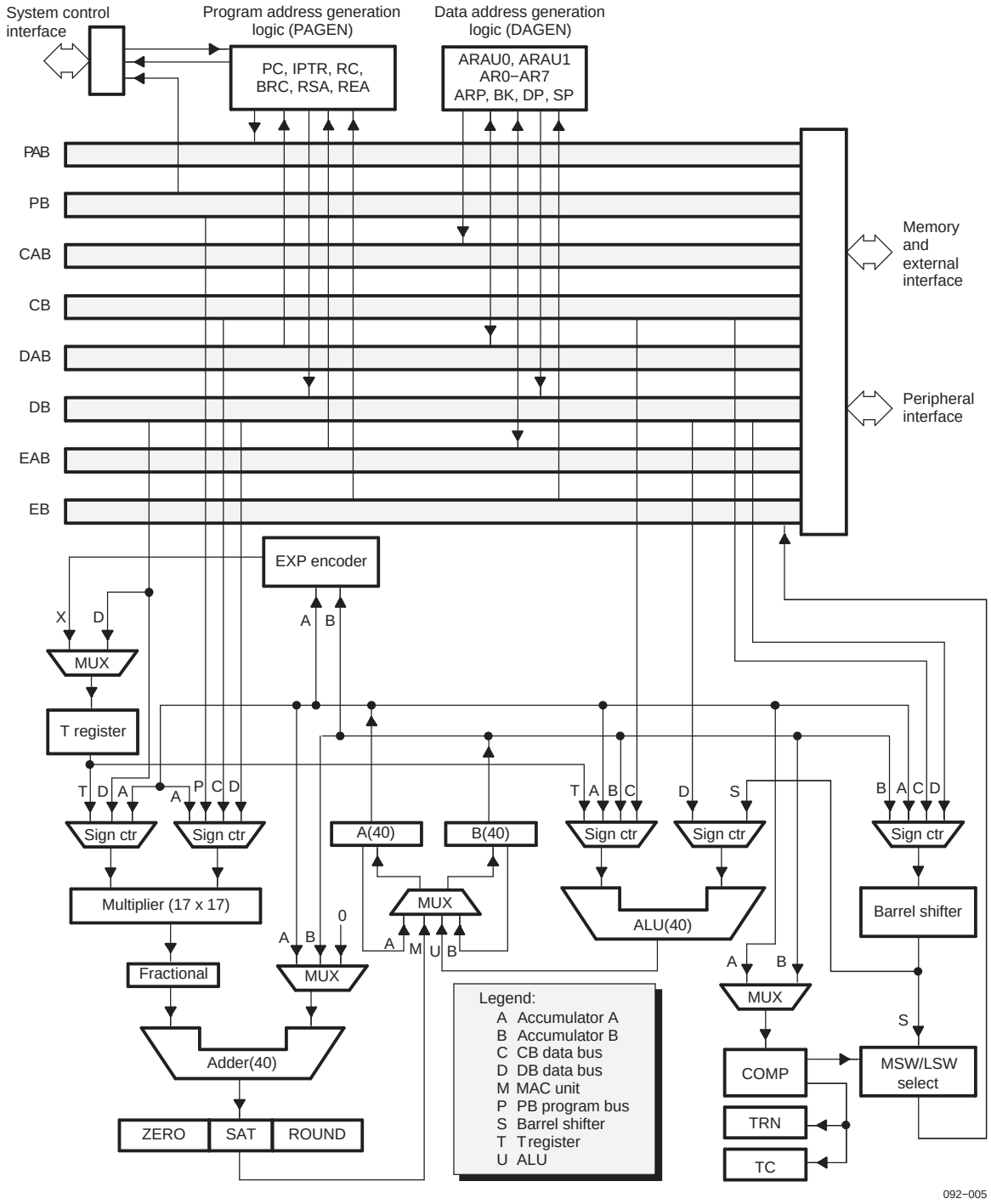
- 40-bit arithmetic logic unit (ALU)
- Two 40-bit accumulators
- Barrel shifter
- 17 x 17-bit multiplier
- Compare, select, and store unit (CSSU)

Table 4-13. Bus Use for Read and Write Accesses

Access Type	Address Bus				Data Bus			
	PAB	CAB	DAB	EAB	PB	CB	DB	EB
Program read	√				√			
Program write	√							√
Data single read			√				√	
Data dual read		√	√			√	√	
Data long (32-bit) read		√/(hw)[1]	√/(lw)			√/(lw)	√/(hw)	
Data single write				√				√
Data read/data write			√	√			√	√
Dual read/coefficient read	√	√	√		√	√	√	
Peripheral read			√				√	
Peripheral write				√				√

Figure 4-5 shows the C54x cDSP architecture.

Figure 4-5. C54x cDSP Architecture Overview



4.2.3 Memory Organization

The DSP subsystem memory is organized into three individual selectable spaces: program, data, and I/O (see Table 4-14).

The program memory space contains the instructions to execute as well as the tables used during execution. The data memory space stores data used by instructions. The I/O memory space interfaces to external memory-mapped peripherals.

Table 4-14. DSP Subsystem Memory Spaces

<table>
<tr><th></th><th colspan="2">Data</th><th>Prog#0</th><th>Prog#1</th><th>Prog#2</th><th>Prog#3</th><th>Prog#4</th><th>XIO</th></tr>
<tr><td>0x0000 0x07FF</td><td colspan="6">DARAM 4K bytes</td><td rowspan="10">Peripherals</td></tr>
<tr><td>0x0800 0x47FF</td><td colspan="6">API 32K bytes</td></tr>
<tr><td>0x4800 0x6FFF</td><td colspan="6">DARAM 20K bytes</td></tr>
<tr><td>0x7000 0x77FF</td><td colspan="2">DARAM 4K bytes</td><td rowspan="6">PROM 56K bytes</td><td colspan="4" rowspan="2">Reserved</td></tr>
<tr><td>0x7800 0x7FFF</td><td colspan="2">Peripherals</td></tr>
<tr><td>0x8000 0x87FF</td><td rowspan="3">PDRAM 20K bytes</td><td>Reserved</td><td rowspan="5">PROM 64K bytes</td><td rowspan="5">PROM 64K bytes</td><td rowspan="5">PROM 64K bytes</td><td rowspan="3">PDRAM 20K bytes</td></tr>
<tr><td>0x8800 0x8FFF</td><td>DARAM 4K bytes</td></tr>
<tr><td>0x9000 0xA7FF</td><td rowspan="2">DROM 40K bytes</td></tr>
<tr><td>0xA800 0xDFFF</td><td rowspan="2">Reserved</td><td rowspan="2">Reserved</td></tr>
<tr><td>0xE000 0xFFFF</td><td>PDROM 16K bytes</td><td>PDROM 16K bytes</td></tr>
</table>

Several on-chip memory types are available on the DSP subsystem:

- DARAM (dual-access data RAM): Always mapped to the data space
- APIRAM (dual-access data RAM): Always mapped to the data space and can be overlaid in program space using the OVLY bit
- DROM (data ROM): Always mapped to the data space
- PROM (program ROM): Always mapped to the program space
- PDROM (program or data ROM): Always mapped to the program space but can be simultaneously mapped to both the data and program space by setting the DROM bit to 1
- PDRAM (program/data RAM): Mapped to both the program and the data space of the XIO interface

Three features of the C54x cDSP provide the flexibility to enable or disable the on-chip memories in the program and data space. These features are controlled through the MPNMC, OVLY, and DROM bits of PMST[6,5,3] (for more information, see *C54x cDSP Reference Set Vol 1: CPU*).

- MPNMC (microprocessor/microcontroller) bit: Setting MPNMC to 0 maps the internal ROMs to the program space.
- OVLY (RAM overlay) bit: If OVLY is cleared to 0, the internal RAM is mapped only to the data space; if OVLY is set to 1, the internal RAM is mapped to both the data and program spaces.
- DROM (Data ROM) bit: If DROM is set to 1, a part of the internal DROM memory to mapped to the data space.

4.2.4 API

The API provides an interface between the MPU ARM7TDMIE and the C54x cDSP CPU. The interface is achieved using a shared block of DARAM.

The API has two modes of operation selected by the C54x cDSP:

- Shared access mode (SAM)
- Host-only mode (HOM)

The mode is configured using the writable SMODE bit BSCR[3] in the C54x cDSP core. This bit field is mirrored on read access on the API_CTRL_DSP and API_CTRL_MPU registers.

CAUTION

API_CTRL_DSP is not a writable register. Writing is done using the BSCR register in the C54x cDSP core. Section 4.3, *DSP Subsystem Register Manual*, describes the BSCR register.

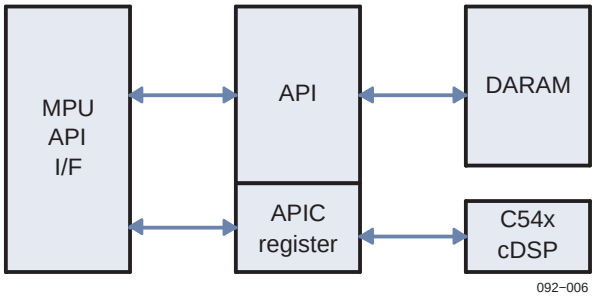
It is possible to generate two interrupts using the following bits:

- Use DSPINT bit API_CTRL_MPU[2] to interrupt the DSP from the MPU.
- Use HINT bit BSCR[3] to interrupt the MPU from the DSP.

The NMI status register provides the state of the TIPB and the API on the DSP side.

Figure 4-6 shows the API block diagram.

Figure 4-6. API Overview



4.2.4.1 Shared Access Mode (SAM)

In SAM, both the C54x cDSP and the MPU can access the API memory. Asynchronous host accesses from the MPU are resynchronized internally. If the C54x cDSP and the MPU try to perform a simultaneous access, the host has access priority and the C54x cDSP waits one cycle. SAM can be run when the C54x cDSP is in IDLE1 mode.

The API supports high-speed, back-to-back host accesses. In SAM, the API can handle 1 byte every four C54x cDSP clock periods. Thus, the number of host wait-states depends only on the DSP/host clock frequency ratio.

While the C54x cDSP is in active mode, the usual mode of operation is SAM because the C54x cDSP wants to access the API RAM. However, in some cases, HOM might be preferred to allow faster access to the API RAM.

4.2.4.2 Host-Only Mode (HOM)

In HOM, only the MPU can access the API memory. When HOM is selected, API memory blocks are disabled for C54x cDSP access. The C54x cDSP cannot read or write to the API memory in HOM.

For the host processor, HOM accesses are faster than SAM accesses. During a C54x cDSP reset, HOM is automatically selected. After reset, the mode automatically changes to SAM to allow a boot from the API.

When the C54x cDSP is in IDLE1, IDLE2, or IDLE3 mode, HOM lets the host processor access the API RAM even though the API memory is not being clocked. When the C54x cDSP is in IDLE3 mode, HOM lets the MPU access the API RAM while the C54x cDSP is dissipating less power.

4.2.5 Timer

The on-chip timer is a software-programmable timer consisting of three registers that can be used to periodically generate interrupts. The timer resolution is the CLKOUT rate of the DSP and has a 20-bit dynamic range.

The timer is an on-chip down-counter that can be used to periodically generate DSP interrupts using the TINT interrupt. The timer is driven by a prescaler that is decremented by 1 at every CLKOUT cycle. Each time the counter decrements to 0, a timer interrupt (TINT) is generated and the down-counter is reloaded with the period value.

4.2.6 Serial Port

The serial port interfaces provide full duplex, bidirectional, communication with the Triton Lite device for voice transmission. Both receive and transmit operations are double-buffered, allowing a continuous communications stream with 8-, 10-, 12-, or 16-bit data packets.

The continuous mode provides operation that, once initiated, requires no additional frame synchronization pulse VFSRX when transmitting at maximum packet frequency. The serial ports are fully static and thus function at arbitrarily low clocking frequencies.

When the voice serial port is in reset, the device can be configured to turn off the internal serial port clock, which allows the device to run in a lower power mode.

For more information on configuration and operation modes, see *C54x cDSP Reference Set Vol 2: Peripherals*.

4.3 DSP Subsystem Register Manual

Table 4-15 and Table 4-16 summarize the DSP and MPU access instances.

Table 4-15. DSP Access Instance Summary

Module Name	Base Address	Size
API controller	0xF900	512 byte
NMI status register	0xFB00	512 bytes

Table 4-16. MPU Access Instance Summary

Module Name	Base Address	Size
API controller	0xFFE0 0000	2 bytes

4.3.1 Module Register Mapping Summary

Table 4-17 through Table 4-19 list the module registers.

Table 4-17. Register Offset Address with DSP Access

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
API_CTRL_DSP	R	16	0x00	0xF900
NMI status register	R	16	0x00	0xFB00

Table 4-18. Register Offset Address with C54x cDSP Access

Register Name	Type	Register Width (Bits)	Address Offset
BSCR	RW	16	0x29

Table 4-19. Register Offset Address with MPU Access

Register Name	Type	Register Width (Bits)	Address Offset
BSCR	RW	16	0x29

4.3.2 Register Descriptions

Table 4-20 through Table 4-23 describe the DSP subsystem register bits.

Table 4-20. API_CTRL_DSP

Address Offset	0x000		
Physical address	0xF900	**Instance**	API controller
Description	DSP mode status		
Type	R		
Write latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												HINT	DSPINT	SMODE	Reserved

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	R	Undefined
3	HINT	Interrupt from DSP to MPU 0: No interrupt 1: Interrupt generated to MPU by DSP	R	0x0
2	DSPINT	Interrupt from MPU to DSP 0: No interrupt 1: Interrupt generated to DSP by MPU	R	0x0
1	SMODE	Shared access mode 0: No interrupt 1: Interrupt generated to DSP by MPU	R	0x0
0	Reserved	Reserved	R	Undefined

Table 4-21. NMI Status Register

Address Offset	0x000		
Physical address	0xFB00	**Instance**	NMI status register
Description	Status of the TIPB and API access		
Type	R		
Write latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												API_RST	TIMEOUT_RHEA	Reserved	

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	R	Undefined
3	API_RST	Status of API access 0: No reset on API 1: API is reset. This bit is cleared on read access, or when a new NMI occurs.	R	0x0
2	TIMEOUT_RHEA	Status of TIP access 0: No timeout access on TIPB 1: Maximum TIPB access time is reached. This bit is cleared on read access, or when a new NMI occurs.	R	0x0
1:0	Reserved	Reserved	R	Undefined

Table 4-22. BSCR

Address Offset	0x029		
Physical address	Refer to *TMS320C54x DSP Reference Set Vol.1*	**Instance**	C54x cDSP core
Description	Programmable bank switching logic		
Type	RW		
Write latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNKCMP				PS-DS	Reserved							HINT	SMODE	Reserved	EXIO

Bits	Field Name	Description	Type	Reset
15:12	BNKCMP	Bank compare determines the external memory-bank size. BNKCMP is used to mask the 4 MSBs of an address. For example, if BNKCMP = 1111b, the 4 MSBs (bits 15-12) are compared, resulting in a bank size of 4K words. Bank sizes from 4K words to 64K words are allowed.	RW	0x0F
11	PS-DS	Program read-data read access. Inserts an extra cycle between consecutive accesses of program-memory read and data-memory read, or data-memory read and program-memory read. 0: No extra cycles are inserted by this feature. 1: One extra cycle is inserted between consecutive accesses of program-memory read and data-memory read, or data-memory read and program-memory read	RW	0x1
10:4	Reserved	Reserved	R	Undefined
3	HINT	Interrupt from DSP to MPU 0: No interrupt 1: Interrupt generated to MPU by DSP	R	0
2	SMODE	Shared access mode 0: Shared access mode (SAM) 1: Host-only mode (HOM)	R	0
1	Reserved	Reserved	R	Undefined
0	EXIO	External bus interface off. Controls the external-bus-off function. 0: The external-bus-off function is disabled. The address bus, data bus, and control signals are enabled. The DROM, MP/MC, and OVLY bits in PMST and the HM bit in ST1 can be modified. 1: The external-bus-off function is enabled. The address bus, data bus, and control signals become inactive after completing the current bus cycle. The DROM, MP/MC, and OVLY bits in PMST and the HM bit in ST1 cannot be modified.	RW	0

Table 4-23. API_CTRL_MPU

Address Offset	0x000		
Physical address	0xFFE0 0000	**Instance**	API controller
Description	DSP mode control		
Type	RW		
Write latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												HINT	DSPINT	SMODE	Reserved

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	R	Undefined
3	HINT	Interrupt from DSP to MPU 0: No interrupt 1: Interrupt generated to MPU by DSP	R	0
2	DSPINT	Interrupt from MPU to DSP 0: No interrupt 1: Interrupt generated to DSP by MPU	RW	0x0
1	SMODE	Shared access mode 0: Shared access mode (SAM) 1: Host-only mode (HOM)	R	0x1
0	Reserved	Reserved	R	Undefined

Interconnect

This chapter describes the interconnection of modules in the LOCOSTO device.

Topic **Page**

5.1 Interconnect Overview

The chip-level interconnect consists of two types of communication between all modules and processor subsystems:

- Memory communications handles many types of data transfer, especially exchange with system on-chip/external memories and memory-like peripherals.
- The TI peripheral bus (TIPB) interconnect uses two bridges to provide all of the peripheral interconnections between the MPU, the DSP, and the DMA.

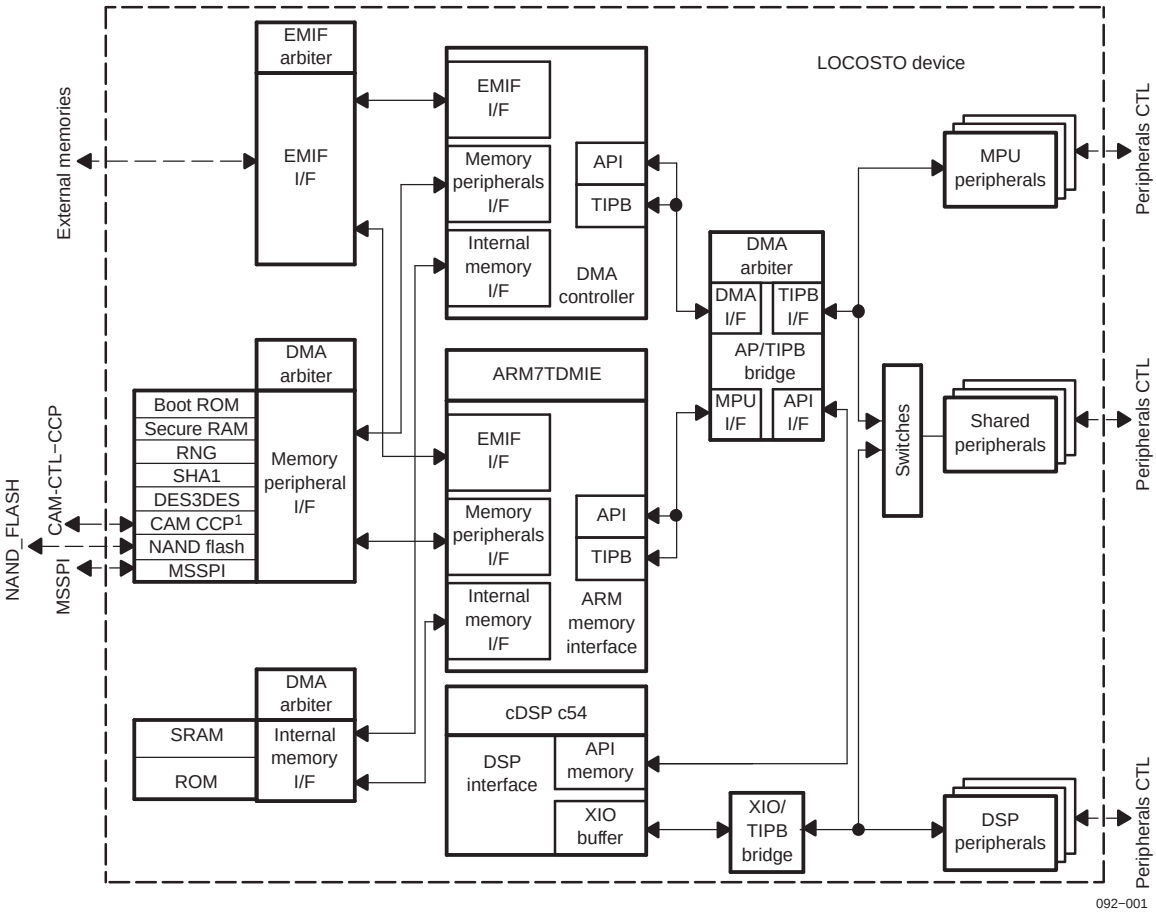
Interconnect layers enable communication between all instances of the LOCOSTO device and assume routing of all data exchanges at the chip level.

5.1.1 LOCOSTO Interconnect Overview

The LOCOSTO device is built around the MPU and DSP core (for more information, see Chapter 3, *MPU Subsystem*, and Chapter 4, *DSP Subsystem*). The LOCOSTO device includes several peripherals allocated to the MPU, the DSP, or both processors.

Figure 5-1 shows a general overview of the LOCOSTO interconnects between the processors and peripherals using buses.

Figure 5-1. LOCOSTO Interconnect Overview



(1) The camera interface is not available in LOCOSTO Lite.

The LOCOSTO device includes peripherals that support one of the following three bus protocols:

- The TIPB is used for communication between processors and peripherals.

- The application programming interface (API) bus allows memory sharing between the MPU and the DSP.
- The open core protocol (OCP) bus is used as an interface between the peripheral and the TIPB using a TIPB/OCP adapter.

The interconnection of the LOCOSTO device resides mainly in four elements:

- The API/TIPB bridge converts MPU or DMA requests to API or TIPB requests.
- The XIO/TIPB bridge converts DSP requests to TIPB requests.
- The switches allow multiple allocations of peripherals.
- The memory interface is used to access memory, such as peripherals and internal memory.

Three types of switches allow peripherals to be accessed by the MPU or the DSP:

- TIPB static switches are used to share TIPB native peripherals.
- TIPB/OCP static switches are used to share OCP native peripherals.
- Dynamic switches are used to share OCP native peripherals.

Each switch has a different arbitration scheme: a hardware scheme for dynamic switches and a software scheme for static switches.

5.1.2 Signals I/O and Description

5.1.2.1 MPU TIPB

Figure 5-2 and Table 5-1 describe a classical connection between the TIPB interface bridge and the TIPB native peripherals.

Figure 5-2. MPU TIPB Bus

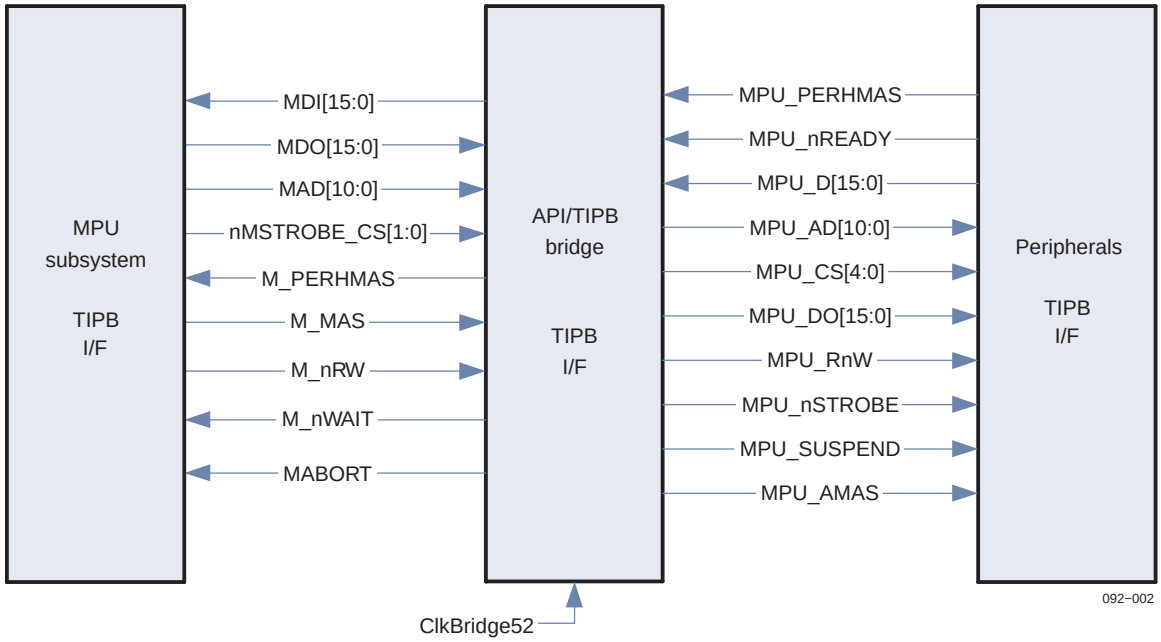


Table 5-1. MPU TIPB Signals from the API/TIPB Bridge to the Peripherals(1)

Name	Description
MPU_AD[10:0]	Address 2K bytes per peripheral
MPU_CS[4:0]	Chip-select allows connecting up to 32 peripherals.

(1) The MPU_PERHMAS, MPU_nREADY, and MPU_DI[15:0] signals from the TIPB interface are the resulting signals from the TIPB router.

Table 5-1. MPU TIPB Signals from the API/TIPB Bridge to the Peripherals (continued)

Name	Description
MPU_DO[15:0]	Data from bridge to peripheral: Clocked in the peripheral on the nSTROBE rising edge (if naready sent by the peripheral is active).
MPU_DI[15:0]	Data from peripheral to bridge: Stored in the bridge on the ClkBridge52 rising edge related to the nSTROBE rising edge (if naready sent by the peripheral is active).
MPU_RnW	Transaction type 1 => read _UnicodeEncodeError_ 0 => write
MPU_nSTROBE[1:0]	Output intended to drive the clocks of all attached peripherals; these clocks are active only when data are being transferred. Each clock addresses a different memory space. The pulse duration depends on the peripherals capabilities and can be chosen with the Access Factor parameters stored in TIPB_CNTL, one for each strobe line. **Fast peripherals: Access Factor = 0:** nSTROBE is generated from the ClkBridge52 clock. The nSTROBE low-level pulse duration is the ClkBridge52 low-level duration. In case of an access with 0 wait-state, only one pulse is generated. Otherwise, one additional pulse is generated per wait-state inserted by the target. **Slow peripherals: Access Factor ≠ 0:** In this case, the Access Factor represents the number of ClkBridge52 periods used to generate the low-level or high-level pulse for nSTROBE. Wait-states can also be inserted by the target.
MPU_nREADY	Peripheral ready to accept or to send data. Active low. It is evaluated by bridge on ClkBridge52 rising edge related to nSTROBE rising edge.
MPU_SUSPEND	Indicates that the processor has suspended execution for an emulation breakpoint; active low. May be used by a peripheral to suspend internal state-machines
MPU_AMAS	Memory access length 0 => 8 bits, 1=> 16 bits
MPU_PERHMAS	Peripheral word access length Allows the bridge to determine the word size of the addressed peripheral 0 => 8 bits, 1=> 16 bits

Table 5-2 describes the connection between the MPU and the TIPB interface bridge.

Table 5-2. MPU TIPB Signals From the MPU to the API/TIPB Bridge

Name	Description
MDO[15:0]	Data output bus; used for TIPB and API write access.
MDI [15:0]	Data input bus; used for TIPB and API read access.
MAD[15:0]	Address in the TIPB memory range
M_nRW	Transaction type 0: read 1: write
M_MAS	Memory access size 0: 8 bits 1: 16 bits
M_PERHMAS	Peripheral accessed size 0: 8 bits 1: 16 bits
nMSTROBE_CS[1:0]	TIPB chip-select; active low.

Table 5-2. MPU TIPB Signals From the MPU to the API/TIPB Bridge (continued)

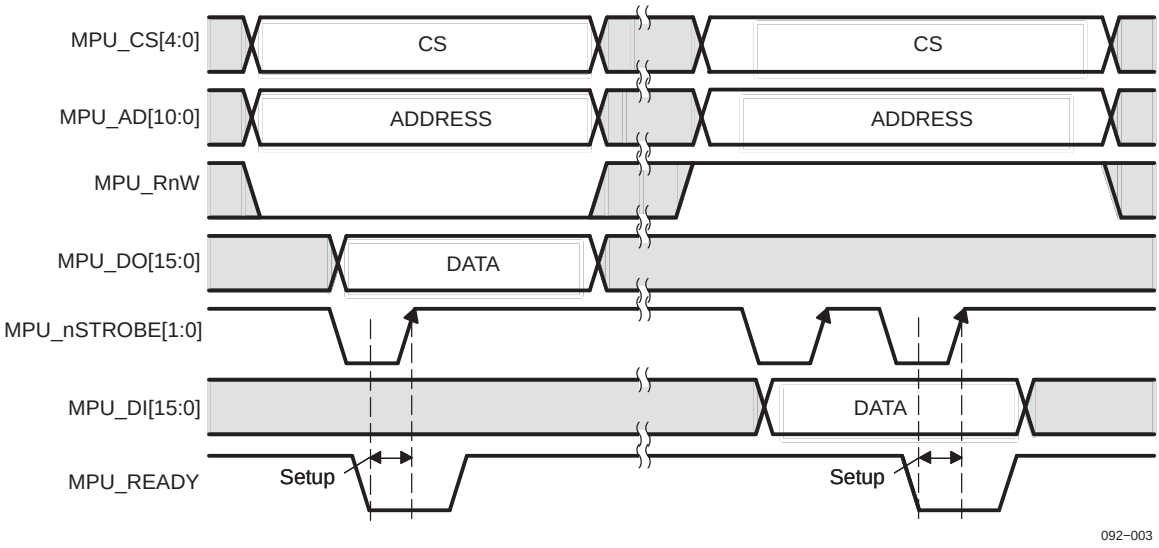
Name	Description
nMWAIT	This flag is delivered by the API/TIPB bridge to postpone the MPU access during a TIPB access; active low.
MABORT	Current access is aborted; active high.

Figure 5-3 shows a protocol example of the TIPB during a read and write transaction.

Note: When the peripherals do not return a ready signal, the local host strobe signal (nMSTROBE) is regenerated; the chip-select, the address bus, and the data bus are maintained.

All input signals of the TIPB bridge (from the peripherals) are sampled on the rising edge of the strobe signal.

Figure 5-3. TIPB Protocol Example



5.1.2.2 DSP TIPB

Figure 5-4 and Table 5-3 describe a classical connection between the XIO/ TIPB interface bridge and the TIPB native peripherals or nonnative peripherals with the TIPB interface.

Figure 5-4. DSP TIPB

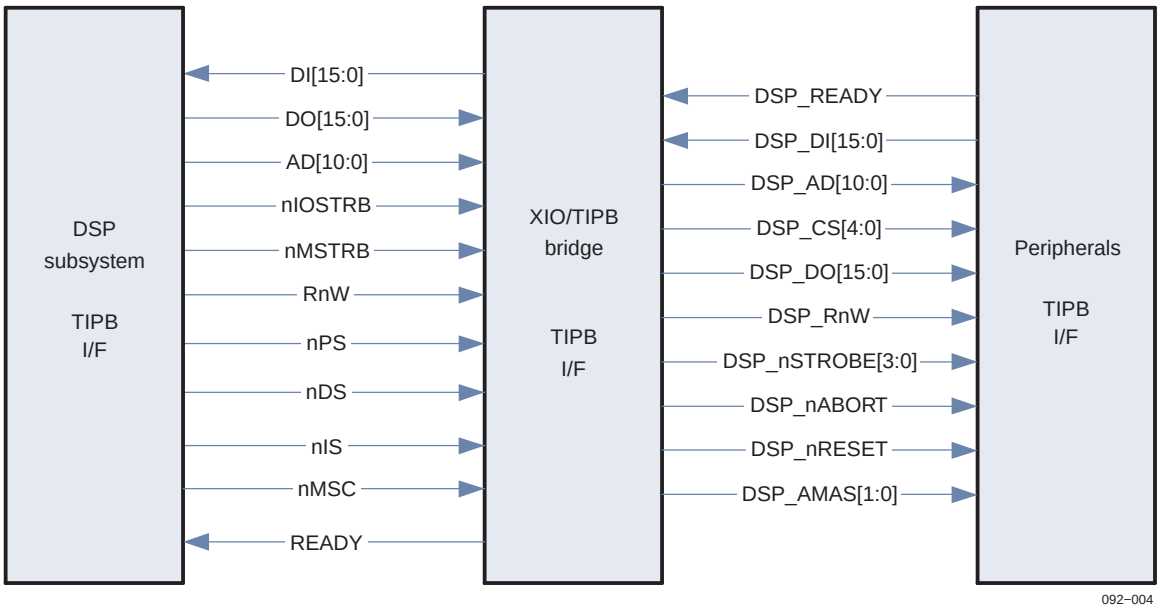


Table 5-3. DSP TIPB Signals From the XIO/TIPB Bridge to the Peripherals

Name[1]	Description
DSP_AD[10:0]	Address 2K bytes per peripheral
DSP_CS[4:0]	Chip-select allows connecting up to 32 peripherals.
DSP_DO[15:0]	Data from bridge to peripheral: Received in the peripheral on the nSTROBE rising edge
DSP_DI[15:0]	Data from peripheral to bridge: Stored in the bridge on the nSTROBE rising edge
DSP_RnW	Transaction type 1: read 0: write
DSP_nSTROBE[3:0	Transfer clocks for all attached peripherals The XIO TIPB finishes transfers by setting the nSTROBE to a high level to avoid bus clashes on successive accesses. In case of a zero wait-state access, only one pulse is generated. Otherwise, one additional pulse is generated per wait-state inserted by the target. When the peripheral inserts wait-states via nREADY, data are not stored in the peripheral (write) or in the bridge (read) until that nSTROBE rising edge when nREADY is in the active-low (ready) condition.
DSP_AMAS[1:0]	Memory access size: Wired to 01 for 16-bit access
DSP_READY	Peripheral is ready to accept or to send data; active low. It is evaluated on the nSTROBE rising edge. When nREADY is at a high level when nSTROBE transitions high, a wait-state is added to the current transfer.
DSP_nRESET	Asynchronous reset, active low
DSP_nABORT	Abort current TIPB transaction; active low:

(1) The DSP_READY and DSP_DI[15:0] signals from the TIPB interface are the resulting signals from the TIPB router.

Table 5-3. DSP TIPB Signals From the XIO/TIPB Bridge to the Peripherals (continued)

Name[1]	Description
	This signal is set low when the accessed peripheral fails to respond after a prescribed period of time has elapsed. DSP_nABORT is used to reset the peripheral bus interfaces.

Table 5-4 describes the connection between the DSP and the XIO/TIPB interface bridge.

Table 5-4. DSP TIPB Signals From the DSP to the XIO/TIPB Bridge

Name	Description
nIOSTRB	I/O strobe signal. Always high unless low level is asserted to indicate an external bus access to I/O device.
nPS	Program space-select signals. Always high unless low level is asserted to communicate to a particular external space.
nDS	Data space select signals. Always high unless low level is asserted to communicate to a particular external space.
nIS	I/O space select signals. Always high unless low level is asserted to communicate to a particular external space.
RnW	Read/write signal. Normally in read mode (high), unless low level is asserted to perform a write operation.
DI[15:0]	Parallel data input bus
DO[15:0]	Parallel data output bus. Not valid when HZDO is active high but is always driven high or low.
AD[15:0]	Parallel address bus A15(MSB) through A0(LSB). Multiplexed to address external data/program memory or I/O.
READY	Data ready input. Indicates that the external device is prepared for the bus transaction to be completed. If the device is not ready (READY is low), the processor waits one cycle and checks READY again. The processor performs ready detection when at least two software wait-states are programmed. The READY signal is not sampled until completion of the software waitstates.
nMSTRB	Memory strobe signal. Always high unless low level is asserted to indicate an external bus access to data/program memory.
nMSC	Microstate complete signal. This output goes low when the last wait-state of 2 or more internal software wait-states programmed is executed. If connected to the READY line, it forces one external wait-state after the last internal wait-state completes.

5.1.2.3 API Bus

Figure 5-5 and Table 5-5 describe the connection between interface bridge and the DSP.

Figure 5-5. API Bus

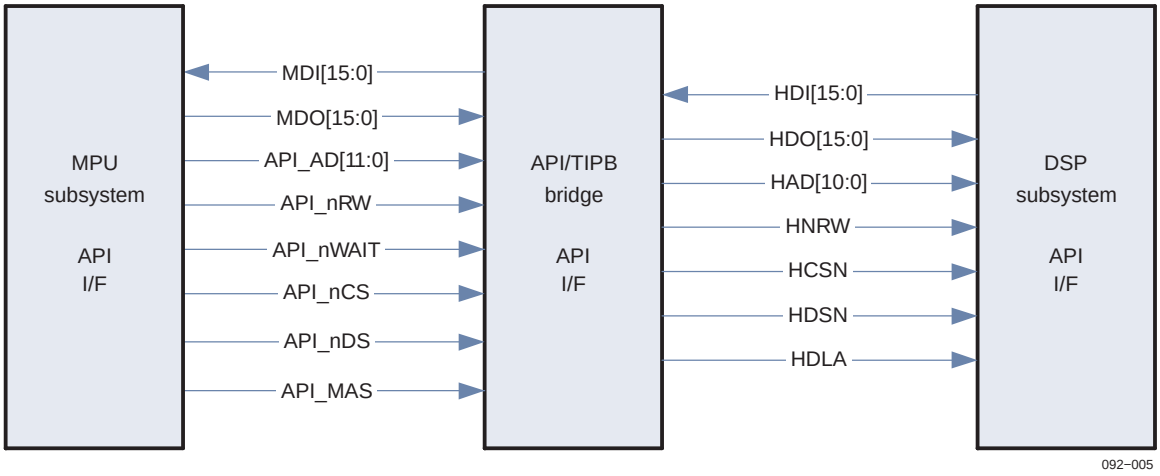


Table 5-5. API Bus Signals From the API/TIPB Bridge to the DSP Subsystem Interface

Name	Description
HDI[15:0]	API data bus
HDO[15:0]	API data bus
HAD[10:0]	API addresses bus
HNRW	API read not write, transaction type 0: read 1: write
HCSN	API control registers select, indicates API control register access; active low.
HDSN	API data select, indicates API access to memory; active low.
HDLA	API data latch enable. Active high. Indicates end of read cycle (performed with either the hcsn or hdsn strobe) **Note about wait-states:** API cannot delay the transfer because there is no ready mechanism. However, wait-states can be inserted, their number programmable by the MPU onto the bridge. Number of ClkBridge52 periods used to generate hcsn/hdsn and possibly hdla (for a read access) = Nb wait-states +1. For 0 wait-state access:

Table 5-5. API Bus Signals From the API/TIPB Bridge to the DSP Subsystem Interface (continued)

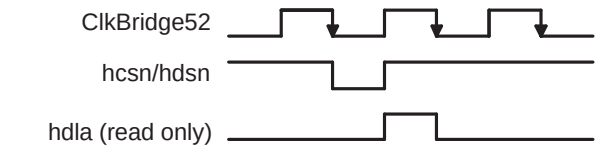
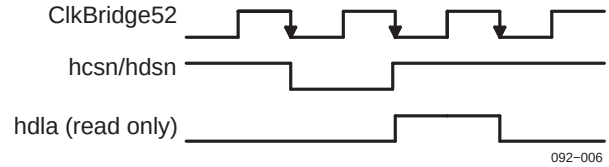
Name	Description
	For 0 wait-state access:  For 1 wait-state access:  Like hcsn/hdsn and hdla pulses must have the same duration: _UnicodeEncodeError_ The falling edge of hcsn/hdsn always appears on the ClkBridge52 falling edge. _UnicodeEncodeError_ The rising edge of hcsn/hdsn can appear on the ClkBridge52 rising or falling edge. _UnicodeEncodeError_ The rising edge of hdla can appear on the ClkBridge52 rising or falling edge. _UnicodeEncodeError_ The falling edge of hdla always appears on the ClkBridge52 falling edge.

Table 5-6 describes the API bus signals from the MPU to the API/TIPB interface bridge.

Table 5-6. API Bus Signals From the MPU Subsystem to the API/TIPB Bridge

Name	Description
API_AD[11:0]	API memory address bus; valid during all the API access.
API_nWAIT	This flag is delivered by the API interface to postpone the MPU access during API access; active low.
API_nRW	Transaction type 0 => read _UnicodeEncodeError_ 1 => write
API_nCS	API memory chip-select for control access; active low.
API_nDS	API memory chip-select for data access; active low.
API_MAS	API word access length 0 => 8 bits, 1=> 16 bits

5.1.2.4 DMA Access

Table 5-7 describes the TIPB signals from the DMA to the API/TIPB interface bridge.

Table 5-7. TIPB Signals From the DMA to the API/TIPB Bridge

Name	Description
DMA_ADD[10:0]	DMA address
DMA_CS[4:0]	DMA chip-select
DMA_RnW	Access type 1 => read - 0 => write

Table 5-7. TIPB Signals From the DMA to the API/TIPB Bridge (continued)

Name	Description
DMA_TIPB	When active, RHEA bus is under control of DMA controller. (ARM accesses to RHEA bus are disabled.) Active high, synchronous with the ClkBridge52 rising edge.
DMA_nREQ	The DMA is transferring data from or to the RHEA bus, or wants to (depending on dma_rhea). Active low, synchronous with ClkBridge52 rising edge.
DMA_ADO[15:0]	Data from DMA to bridge
DMA_ADI[15:0]	Data from bridge to DMA
DMA_MAS	DMA access size 0 => 8 bits, 1=> 16 bits
DMA_TIPB_BUSY	The MPU is transferring data from or to the TIPB toward the bridge, or wants to (depending on DMA_TIPB). Active high, synchronous with ClkBridge52 rising edge.
DMA_TIPB_DONE	This signal informs the DMA controller of the end of the current cycle on the TIPB; for MPU or DMA accesses (depending on DMA_TIPB). Active high on ClkBridge52 rising edge.

Table 5-8 describes the API bus signals from the DMA to the API/TIPB interface bridge.

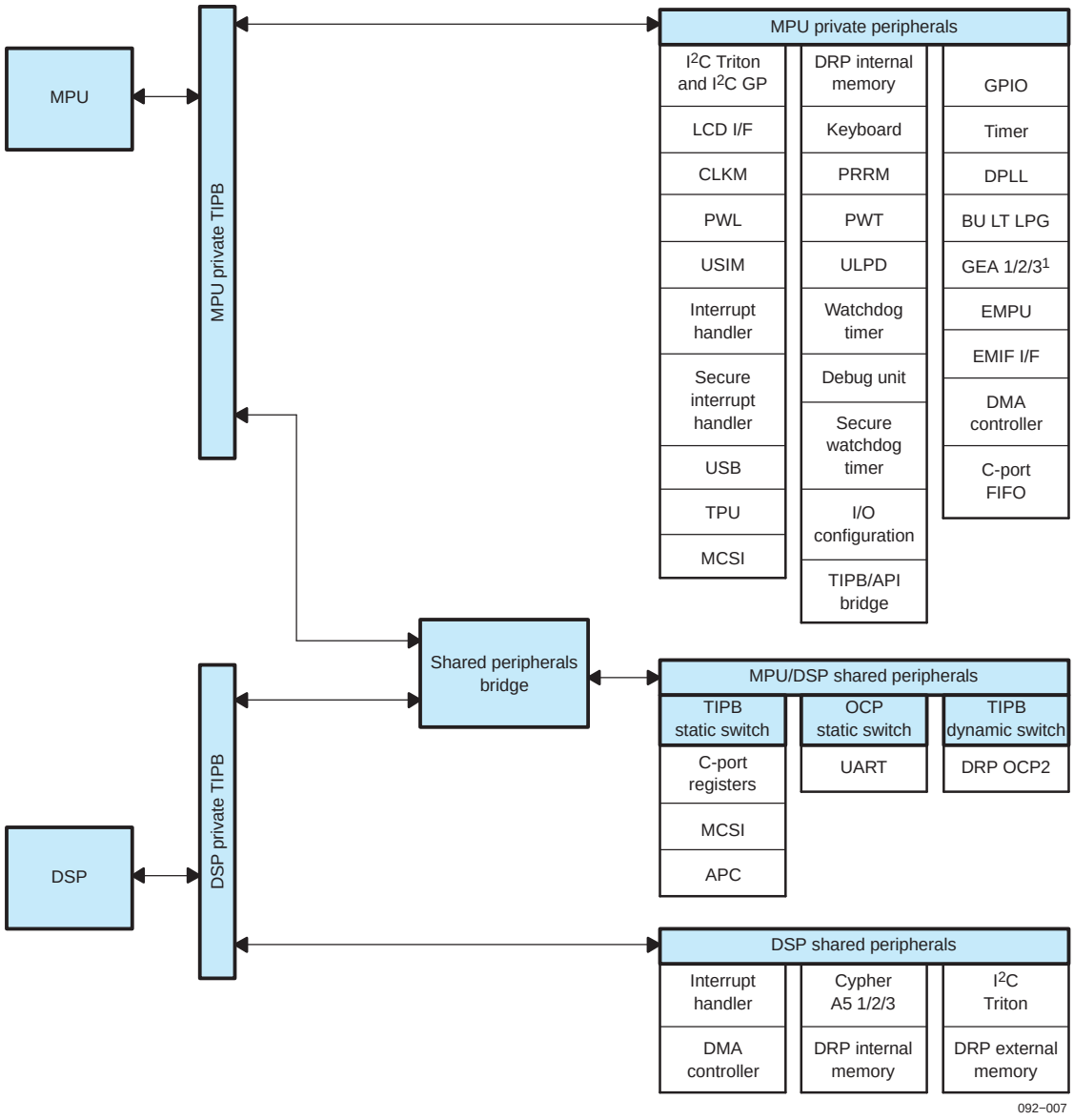
Table 5-8. API Bus Signals From the DMA to the API/TIPB Bridge

Name	Description
DMA_HAD[11:0]	DMA address
DMA_HMAS	DMA access size 0: 8 bits 1:16 bits
DMA_HnRW	Access type 0: read 1: write
DMA_API	When active, the API bus is under control of the DMA controller. (MPU accesses to the API bus are disabled); active high, synchronous with the ClkBridge52 rising edge.
DMA_HDO[15:0]	Data bus from the DMA to the bridge
DMA_HDI[15:0]	Data bus from the bridge to the DMA
DMA_API_BUSY	ARM is transferring data from or to the API bus toward the bridge, or wants to (depends on DMA_API); active high.
DMA_API_DONE	This signal informs the DMA controller of the end of the current cycle on the API bus. For ARM or DMA accesses (depends on dma_api); active high on the ClkBridge52 rising edge.

5.1.3 Peripherals Addresses and Widths

Figure 5-6 shows all of the peripherals and their access with respect to the processor and the type of switches used for shared peripherals.

Figure 5-6. TIPB Peripherals for MPU and DSP Access



(1) The camera interface and GPRS modules are not available in LOCOSTO Lite.

Table 5-9 through Table 5-11 list the following information for each peripheral:

- Address range for accessing the peripheral
- For shared peripherals between the MPU and the DSP, the type of switch used (dynamic, static, TIPB/OCP static)
- Valid access size

Table 5-9. MPU Memory and Peripherals Accesses

Device Name	Base Address	Size	Data Access	Access Type	DMA Request
Boot ROM	0x0000 0000	64K bytes	8/16/32 bits R		

Table 5-9. MPU Memory and Peripherals Accesses (continued)

Device Name	Base Address	Size	Data Access	Access Type	DMA Request
External memory	0x0040 0000	124M bytes	8/16/32 bits R/W	External I/F	
Protected RAM	0x0800 0000	256K bytes	8/16/32 bits R/W	Internal I/F	
Nonprotected RAM	0x0804 0000	64K bytes	8/16/32 bits R/W	Internal I/F	
ROM	0x0805 0000	192K bytes	8/16/32 bits R	Internal I/F	
Boot ROM	0x0900 0000	64K bytes	8/16/32 bits R	Peripheral I/F	
Camera CCP(1)	0x0970 0000	1M bytes	32 bits R/W	Peripheral I/F	√
SHA1/MD5	0x0980 0000	1M bytes	32 bits R/W	Peripheral I/F	√
DES/3DES	0x0990 0000	1M bytes	32 bits R/W	Peripheral I/F	√
RNG	0x09A0 0000	1M bytes	32 bits R/W	Peripheral I/F	
NAND Flash	0x09D0 0000	1M bytes	8/16/32 bits R/W	Peripheral I/F	√
MSSPI	0x09E0 0000	1M bytes	32 bits R/W	Peripheral I/F	√
Debug unit	0x09F0 0000	1M bytes	32 bits R/W	Peripheral I/F	
API control register	0xFFE0 0000	2 bytes	16 bits R/W	TIPB	
TPU-2-OCP	0xFFFE 0800	2K bytes	16 bits R	TIPB	
ULPD	0xFFFE 2000	2K bytes	16 bits R/W	TIPB	
TIMER1	0xFFFE 3800	2K bytes	16 bits R/W	TIPB	
GPIO	0xFFFE 4800	2K bytes	16 bits R/W	TIPB	
GPIO1	0xFFFE 5000	2K bytes	16 bits R/W	TIPB	
GPIO2	0xFFFE 5800	2K bytes	16 bits R/W	TIPB	
TIMER2	0xFFFE 6800	2K bytes	16 bits R/W	TIPB	
LPG	0xFFFE 7800	2K bytes	8 bits R/W	TIPB	
PWL	0xFFFE 8000	2K bytes	8 bits R/W	TIPB	
PWT	0xFFFE 8800	2K bytes	8 bits R/W	TIPB	
Keyboard controller	0xFFFE B800	2K bytes	16 bits R/W	TIPB	
TPU RAM	0xFFFF 9000	2K bytes	16 bits R/W	TIPB	
DPLL	0xFFFF 9800	2K bytes	16 bits R/W	TIPB	
LCD interface	0xFFFF A000	2K bytes	16 bits R/W	TIPB	√
USIM interface	0xFFFF A800	2K bytes	16 bits R/W	TIPB	√
USB	0xFFFF B000	2K bytes	16 bits R/W	TIPB	√
I^2C	0xFFFF B800	2K bytes	16 bits R/W	TIPB	√
GEA1/2/3(1)	0xFFFF C000	2K bytes	16 bits R/W	TIPB	
I^2C Triton	0xFFFF C800	2K bytes	16 bits R/W	TIPB	√
C-Port (FIFO registers)	0xFFFF D800	2K bytes	16 bits R/W	TIPB	√
DMA controller	0xFFFF E800	2K bytes	16 bits R/W	TIPB	
TPU	0xFFFF F000	2K bytes	16 bits R/W	TIPB	
Watchdog	0xFFFF F800	128 bytes	16 bits R/W	TIPB	
Secure watchdog	0xFFFF F880	128 bytes	16 bits R/W	TIPB	
API/TIPB bridge	0xFFFF F900	256 bytes	16 bits R/W	TIPB	
Interrupt handler	0xFFFF FA00	128 bytes	16 bits R/W	TIPB	
Secure interrupt handler	0xFFFF FA80	128 bytes	16 bits R/W	TIPB	
EMIF	0xFFFF FB00	46 bytes	16 bits R/W	TIPB	
PRRM	0xFFFF FC00	256 bytes	16 bits R/W	TIPB	
CLKM	0xFFFF FD00	256 bytes	16 bits R/W	TIPB	
JTAG IDCODE	0xFFFF FE00	4 bytes	16 bits R/W	TIPB	
EMPU	0xFFFF FF00	256 bytes	16 bits R/W	TIPB	

(1) The camera interface and GPRS modules are not available in LOCOSTO Lite.

Table 5-10. DSP Peripherals Access

Device Name	Base Address	Size	Data Access	Access Type	Memory Space
Cipher A5	0x2800	4K bytes	16	TIPB	I/O
I^2C Triton	0x7880	256 bytes	16	TIPB	Data
DRP external memory	0x7A00	512 bytes	16	TIPB	Data
DRP internal memory	0x7A00	512 bytes	16	TIPB	Data
DMA controller	0xE800	4K bytes	16	TIPB	I/O
XIO/TIPB bridge	0xF800	512 bytes	16	TIPB	I/O
API control register	0xF900	512 bytes	16	TIPB	I/O
Interrupt handler	0xFA00	512 bytes	16	TIPB	I/O

Table 5-11. Peripherals Shared by the MPU and the DSP

Module Name	MPU Domain		Type of Bridge	DSP Domain			DMA Request	Reset Allocation
	Base Address	Size in Bytes		Base Address	Size in Bytes	Memory Space		
DRP memory	0xFFFF 0000	16K	Dynamic	0x8000	16K	I/O		DSP
APC	0xFFFF 5000	2K	TIPB	0x7980	256	Data		DSP
UART	0xFFFF 7000	2K	TIPB/ OCP	0x7900	256	Data	√	MPU
MCSI	0xFFFF 7800	2K	TIPB	0x7800	256	Data		DSP
C-Port (conf reg)	0xFFFF D000	2K	TIPB	0x7C00	256	Data		DSP

Table 5-12 lists the switches used with the shared peripherals.

Table 5-12. Switches for Shared Peripherals

Module Name	MPU Domain		DSP Domain		
	Base Address	Size in Bytes	Base Address	Size in Bytes	Memory Space
UART TIPB/OCP switch	0xFFFF 7280	2	0x7940	128	Data
DRP dynamic switch	0xFFFF 8000	2K	No access	-	-
APC TIPB switch	0xFFFF 8800	32	0x7C80	2	Data
MCSI TIPB switch	0xFFFF 8820	32	0x7CA0	2	Data
C-port TIPB switch	0xFFFF 8840	32	0x7CC0	2	Data

Note: The MPU can access the C-port at two points:

- 0xFFFF D000, to initialize the configuration registers; access is shared between the MPU and the DSP through the TIPB switch.
- 0xFFFF D800, to read the FIFO; write access is reserved to the MPU only.

The DSP can access:

- FIFOs only in read access
- Configuration registers in read/write

5.2 Interconnect Functional Description

5.2.1 API/TIPB Bridge

5.2.1.1 API/TIPB Bridge Functional Description

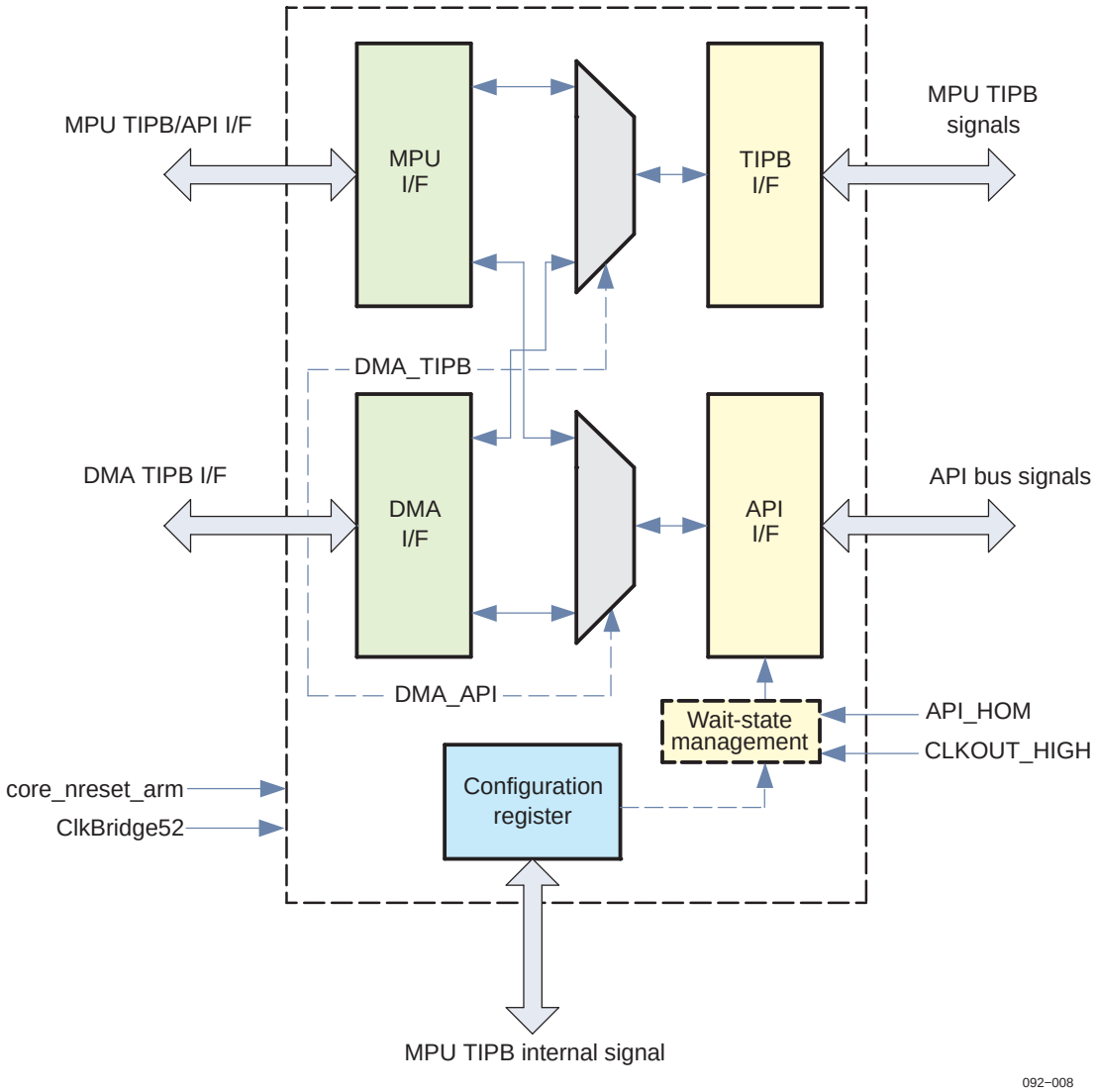
The bridge module allows two resources (the TIPB with two strobe lines and the API bus with data and control strobe lines) to be shared between two bus masters (the MPU and the DMA). The following data transfers are possible:

- MPU↔TIPB
  - On both strobe lines (0 and 1)
  - 8/16 bits data bus length
  - With or without buffer for write access
  - In user or privilege mode
- MPU↔API
  - On both strobe lines (HCSN strobe for control access and HDSN strobe for data access)
  - 8/16 bits data bus length
- DMA↔TIPB
  - On strobe line 0 only
  - 8/16 bits data bus length
  - No buffer for write access
  - In user mode only
- DMA↔API
  - On data strobe line only (HDSN)
  - 8/16 bits data bus length
- TIPB↔API in two steps, using the DMA controller: One step transfers data from the source to the internal buffer in the DMA controller. The other step transfers data from the internal buffer to the target.

Note: For more information on STROBE lines 0 and 1, see Chapter 2, *Memory Mapping.*

Figure 5-7 shows the block diagram of the API/TIPB bridge.

Figure 5-7. API/TIPB Bridge Block Diagram



The API/TIPB bridge consists of five main blocks:

- MPU interface (MPU I/F): Performs a multiplexing function for the data and control signals to/from the MPU memory interface. These signals are directed to either the API interface or the TIPB interface.

  The MPU I/F also manages the frequency difference between the MPU and the bridge (for more information, see Chapter 3, *MPU Subsystem*). MPU accesses can then be detected by the TIPB and API interfaces just as DMA accesses are detected.
- DMA interface (DMA I/F): Performs a multiplexing function for the data and control signals to/from the DMA controller. These signals are directed to either the API interface or the TIPB interface.
- MPU TIPB interface (TIPB I/F): Allows access to peripherals compatible with the TIPB definition.
- The commands and data are oriented to/from either the MPU or the DMA controller based on the DMA_RHEA signal.
- The MPU can perform accesses on both strobes lines (0 and 1); the DMA controller can perform accesses only on strobe line 0.
- Peripherals can delay completion. The bridge informs the other host of unfinished transactions between the peripheral and the master host by asserting wait or done signals.
- A mechanism limits the maximum time a peripheral can stall the processor. This time-out value is programmable by software and must be adapted based on the ClkBridge52 frequency.
- All transfers are performed at the ClkBridge52 frequency for fast access time. Moreover, unused

ClkBridge52 cycles in an MPU access (at the beginning or end of the transfer) can be used for DMA accesses.

- To access slow peripherals, the nASTROBE frequency can be divided based on a programmable register (one access factor parameter per strobe line).
- Privilege mode is possible for an MPU access only.

CAUTION

Because of a timing issue, it is recommended that both strobes signals be programmed to the same frequency.

- API interface (API I/F): Allows access to the DSP API memory and the API control register
  - The DMA_API signal directs the API interface to transfer data from/to the API memory to/from the MPU memory interface or the DMA controller.
  - All transfers are performed at the ClkBridge52 frequency for fast access time. Moreover, unused ClkBridge52 cycles in an MPU access (at the beginning or end of the transfer) can be used for DMA accesses.
- Configuration registers: Four registers are controlled directly by the MPU internal TIPB (for more information, see Section 5.4.2, *Register Description*).

The MPU is generally the bus master and can access the API or TIPB on request (DMA_RHEA and DMA_API are inactive). Nevertheless, when the DMA controller must perform an access, it can directly control the TIPB or API bus by positioning the DMA_RHEA or DMA_API signal active (an arbiter in the DMA controller allows the priority to be resolved between the different DMA channels and the processor).

After the bus master (either the MPU or the DMA controller) starts a transfer, only a time-out can abort the transfer.

When requesting control of a bus, the DMA controller or the MPU must wait for the end of the current transfer by monitoring two available signals for each API and TIPB to determine if the DMA is finished or if the current access between the MPU and peripherals is finished.

In the same way, when the MPU tries to access a bus under control of the DMA controller, the same wait-states are generated to the MPU until the end of the current access.

For TIPB accesses, because the memory interface manages memory continuity, the bridge manages the access time problem. The bridge must stall the processor when a wait signal based on the nREADY signal is driven by a peripheral. The nREADY signal indicates the state of the peripheral to accept or send data.

For API accesses, the bridge must also manage the access time problem based on the API_WS_X registers and the API wait-state management signals. Moreover, the bridge must manage the access size adaptation for 8-bit accesses (the API can be accessed only in 16-bit mode).

5.2.1.2 API Wait-State Insertion

When accessing the API memory in shared access mode (SAM) and for synchronization reasons, it is necessary to respect a ratio between the frequencies of the DSP clock and the HDSN.

To achieve this requirement, a wait-state generator must be included in the API interface to increase the strobe pulse duration.

The maximum number of wait-states depends on the minimum Lead frequency and the maximum MPU frequency and can be estimated using the following formula:

Maximum wait-state = (Max MCLK / Min DSP) * 2

For example, if MCLK = 104 MHz and lead = 104 MHz, the maximum wait-state is 2.

To speed access to the API memory as much as possible, the number of waitstates must be adapted based on:

- Host only mode (HOM): When the API is configured in HOM (no DSP access), the insertion of wait-states is unnecessary. This information is provided by the API_HOM signal from the DSP interface (for more information, see Chapter 4, *DSP Subsystem*).
- SAM: When in SAM (MPU and DSP access), wait-states might be needed.

Note: The number of wait-states is kept constant during an entire access (16 bits read, 16 bits write, or 8 bits read/modify/write operation).

5.2.1.3 Access Size Adaptation

To allow the MPU to execute code from the API memory, it is necessary to adapt the MPU memory access size (8/16 or 32 bits) to the API memory size (16 bits). Adaptation is performed in the MPU memory interface or in the bridge, as described in Table 5-13.

Table 5-13. MPU Access Size Adaptation

	Access Size		
	32 bits	**16 bits**	**8 bits**
MPU→MPU memory I/F	√	√	√
MPU memory I/F→API/ TIPB bridge	Conversion in the MPU memory interface: two consecutive 16-bit accesses to the bridge.	√	√
API/TIPB bridge→API bus	N/A	√	Conversion in the API/TIPB bridge: a 16-bit MPU read/modify/ write access.

The DMA controller can also perform 8-bit or 16-bit accesses on the API memory (16 bits). Adaptation might be necessary in the API/TIPB bridge, as described in Table 5-14.

Table 5-14. DMA Controller Access Size Adaptation

	Access Size	
	16 bits	**8 bits**
DMA controller → API/TIPB bridge	√	√
API/TIPB bridge → API bus	√	Conversion in the API/TIPB bridge: a 16-bit DMA read/ modify/write access

5.2.1.3.1 8-Bit Accesses

To convert, the API interface receives two types of information from the MPU memory interface or the DMA controller:

- Address
- Access size

Read and write access is explained as follows:

- 8-bit read access:
  - The bridge performs 16-bit read access on the API memory without considering the least-significant bit (LSB) address bit.
  - Depending on the LSB address bit, the bridge updates the MPU or the DMA controller bus with either the least-significant (LS) or the mostsignificant (MS) part of the read value.
- 8-bit write access read/modify/write access:
  - Bridge performs 16-bit read access on the API memory without considering the LSB address bit.
  - Depending on the LSB address bit, the bridge updates either the LS or the MS part of the read value with the 8 LSBs data bus value (MPU or DMA controller).

– Finally, the bridge writes back the modified value to the API memory.

Note: The read/modify/write does not ensure protection from DSP writes while in the update cycle.

5.2.1.4 Buffer Mechanism for the MPU Interface

To avoid stalling the MPU, an internal buffer is available when accessing a slow peripheral on the TIPB using the W_BUF_EN_0 and W_BUF_EN_1 bits MPU_TIPB_CNTL[1:0].

The following steps make up the writing procedure using the internal buffer:

1. The MPU writes to an internal buffer: With DMA_RHEA inactive, all signals relating to this access are saved in buffers.
2. The bridge performs the real access: On the next ClkBridge52 cycle, the buffered values are sent to the addressed peripheral until it accepts them. During this second step, the MPU core is not stalled and can still access memory or execute internal cycles.

Note: On the TIPB specification, peripherals send their data bus length (using the A_PERHMAS signal) to the bridge only during the nASTROBE low level.

Therefore, when the bridge executes an MPU write access bufferization, this information is unknown. (The real access begins only after bufferization is complete.) Consequently, for all MPU writes with buffer accesses, the bridge sends to the MPU memory interface a fixed value for R_PERHMAS, regardless of real access size. The chosen value is 1↔16 bits access.

The result is that the MPU memory interface cannot use the size auto-adaptability feature for a write with the buffer to an 8-bit peripheral.

Figure 5-8 and Figure 5-9 show timing diagrams for write accesses.

Figure 5-8. Write Access With W_BUF_EN_x = 1 and ACC_FACTOR_x = 0

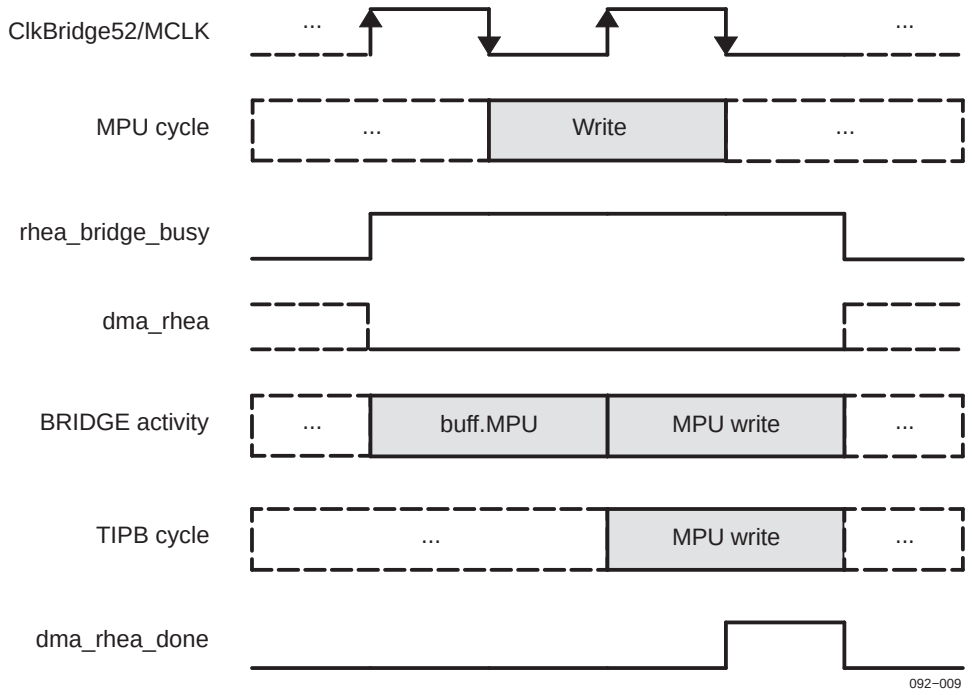
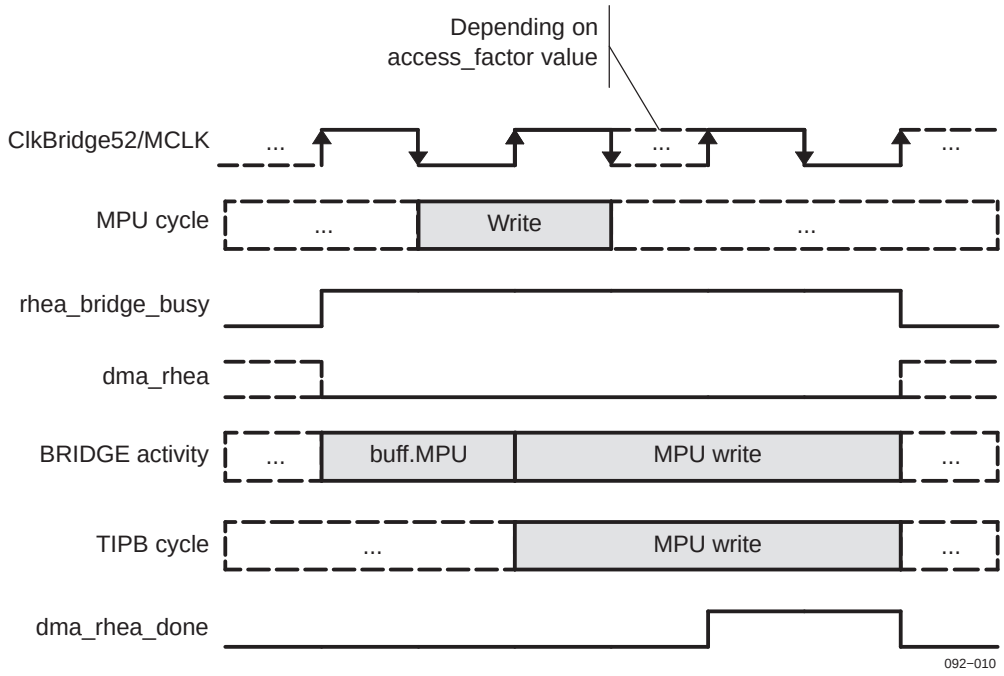


Figure 5-9. Write Access With W_BUF_EN_x = 1 and ACC_FACTOR_x ≠ 0



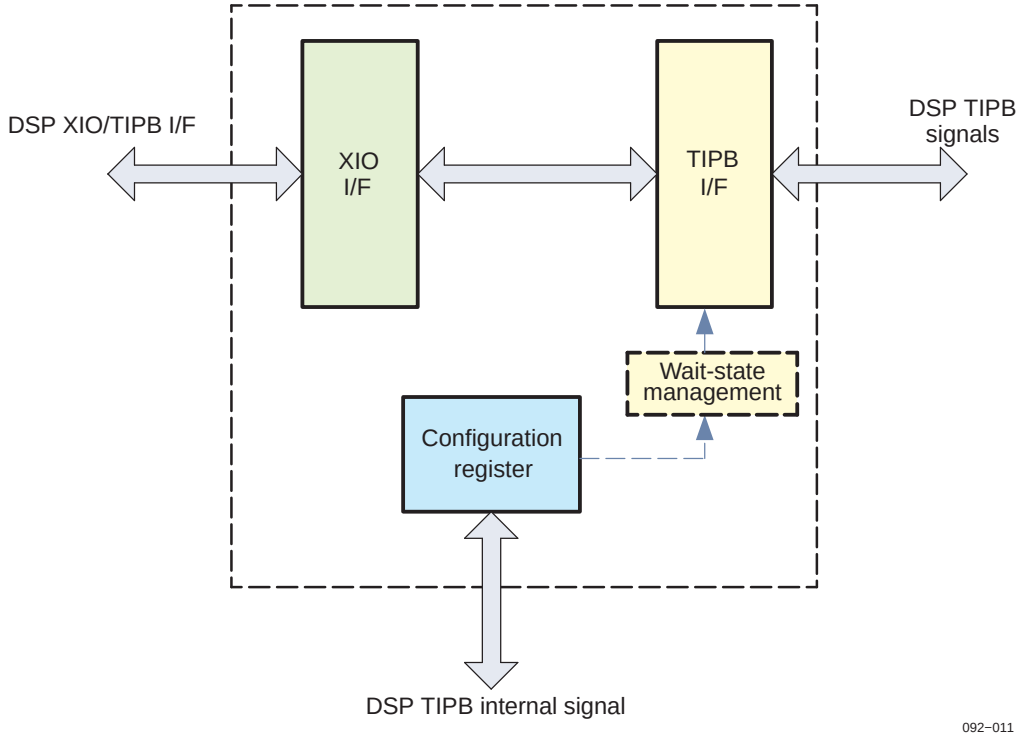
5.2.2 XIO/TIPB Bridge

The XIO/TIPB bridge module allows the DSP to access private or shared peripheral resources. The following data transfer is possible:

- DSP↔TIPB
  - Four strobe lines
  - 16-bit data bus length on the XIO and TIPB side
  - Programmable TIPB transfer clock division
  - Peripheral wait-state insertion support

Figure 5-10 shows the XIO/TIPB bridge block diagram.

Figure 5-10. XIO/TIPB Bridge Block Diagram



- I/O interface (XIO I/F): Performs a multiplexing function for the data and control signals to/from the DSP memory interface. These signals are directed to the TIPB interface.
- DSP TIPB interface (TIPB I/F): Allows access to peripherals compatible with the TIPB definition.
  - The DSP can perform accesses on four strobe lines.
  - Peripherals can delay completion. The bridge informs the DSP of an unfinished transaction by asserting wait or done signals.
  - A mechanism is provided to limit the maximum time a peripheral can stall the processor. This time-out value is programmable and must be adapted based on the ClkBridge52 frequency.
  - To access slow peripherals, the nSTROBE frequency can be divided using a programmable register (see Section 5.2.2.2, *Reduced Speed Transaction*).
- Configuration registers: The internal DSP TIPB directly controls the two available registers (see Section 5.1.2.2, *DSP TIPB*).

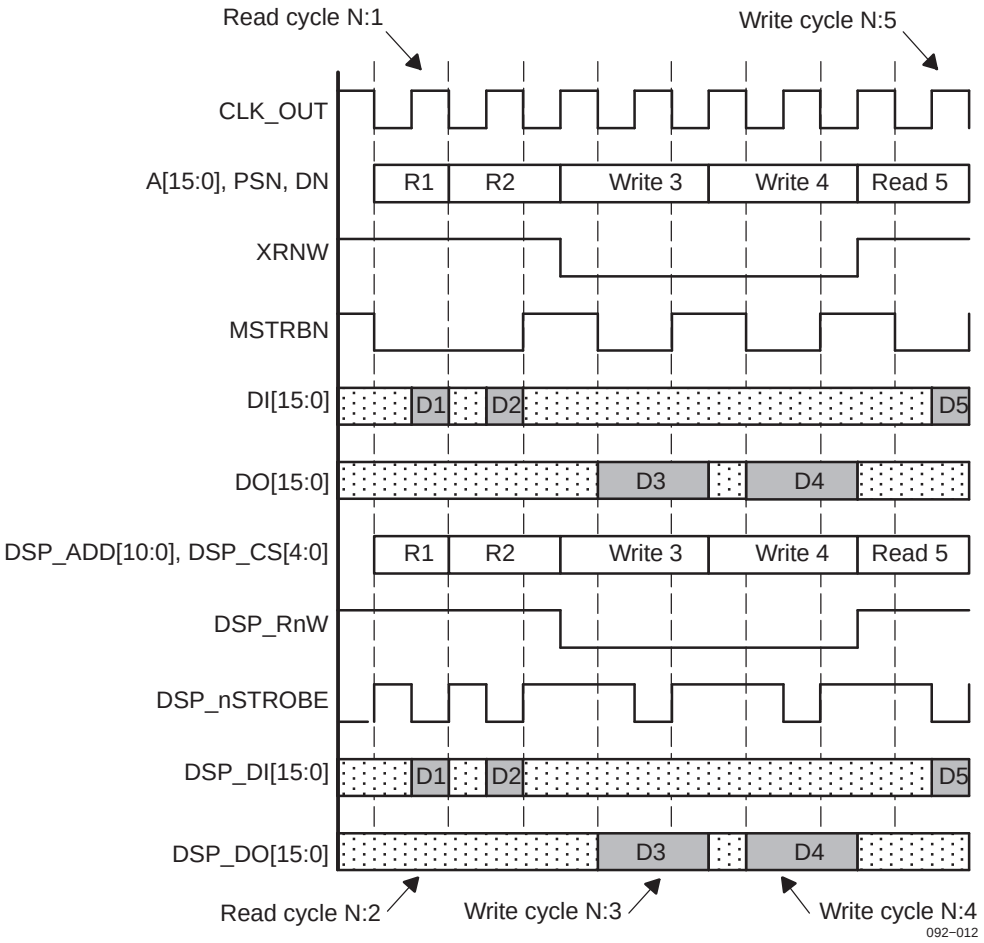
5.2.2.1 Full-Speed Access Transactions

During a full-speed TIPB transaction, the DSP does not sample the READY line to insert external wait-states because there are no internal wait-states; thus, peripherals cannot delay the transfer completion using the READY line. Transfer rate division must be used to connect a low-speed peripheral to the bus. This feature is described in Section 5.2.2.2, *Reduced Speed Transaction.*

5.2.2.1.1 Memory Space Access

Figure 5-11 shows full-speed read and write XIO cycles without wait-states. DSP memory accesses are performed in one clock cycle for read transactions and two clock cycles for write transactions. An additional half-clock period is inserted between read and write memory cycles. For all DSP transactions, the DSP address and flag signals are valid before the CLK_OUT rising edge when MSTRBN is low.

Figure 5-11. XIO/TIPB Bridge Timing for Memory Space Access



From the TIPB side, all full-speed accesses are performed in one DSP cycle. During the first part (nSTROBE high), address and control flags (DSP_ADD, DSP_RnW, and DSP_CS) are delivered to the TIPB.

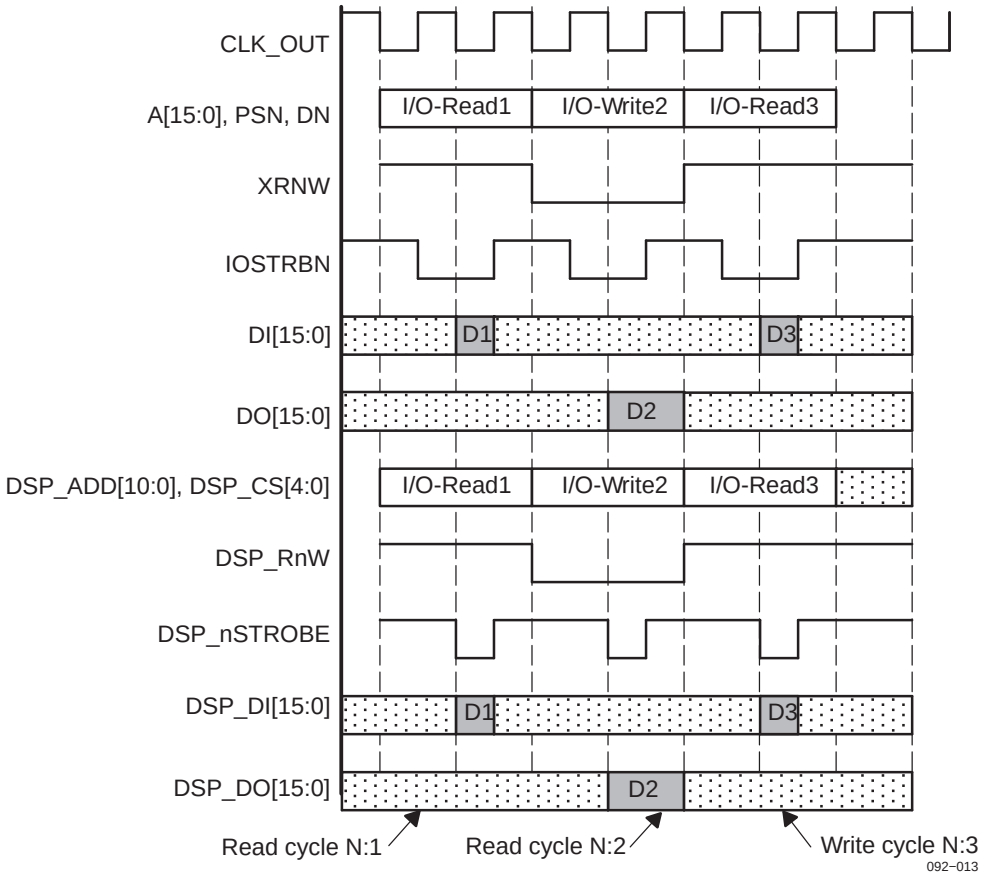
The data transfer is performed in the second part (nSTROBE low). Peripherals cannot insert an extra wait-state using the DSP_READY flag because the DSP does not sample its READY line.

5.2.2.1.2 I/O Space Access

DSP accesses to the I/O memory space are always performed in two CLK_OUT cycles for both read and write.

Figure 5-12 shows an I/O-memory read-write-read sequence.

Figure 5-12. XIO/TIPB Bridge Timing for I/O Space Access



The TIPB transaction is performed during the IOSTRBN low level. The CLK_OUT cycle is divided into two halves. The first half is the bus set-up phase, bus signals valid, and nSTROBE=1. The second half is the transfer phase (nSTROBE low). Peripherals cannot insert an extra transfer cycle using the READY line.

5.2.2.2 Reduced Speed Transaction

Connecting slower devices to the TIPB requires a reduced transfer clock rate. One way to reduce the transfer rate is to reduce the clock controlling the transfers. The ability to reduce clock rates is provided but might not be the best solution because other peripherals on the DSP TIPB might require higher speed transfers. This transfer rate versatility is supported by allowing addressed peripherals to insert wait-states.

5.2.2.2.1 Access Timing with Transfer Clock Division

Transfer rate division is one way to enable low-speed peripherals to connect to the TIPB. The transfer rate is programmable and can be different for each nSTROBE line.

For the DSP bus controller, two cases are present:

- Half-rate transfer is performed in two DSP cycles. The DSP_READY flag cannot be used to insert this extra cycle. Thus, the transfer must be delayed using internal DSP wait-states. Furthermore, peripherals cannot delay the transfer using the READY line.
- For rates other than full rate and half rate, DSP accesses are delayed using both the internal XIO/TIPB bridge generated wait-states and external wait-states inserted with the DSP_READY line. Peripherals can postpone the transfer completion by asserting a high READY level that inserts an extra transfer period.

The nSTROBE rate division factors are programmed using the TRANS_CYCLE_n bit field TRANSFER_RATE, where n is one of the four nSTROBE lines. TRANS_CYCLE_n defines the duration of the high and low parts of the associated nSTROBE line in CLK_OUT half periods. The maximum transfer duration is 16 DSP clock periods.

For example, if TRANS_CYCLE_2 is set to 3, TIPB transactions on the nSTROBE[2] line are performed in three CLK_OUT cycles, divided into three half-clock periods for address and flags decoding (nSTROBE high), and three half-clock periods for data transfer (nSTROBE low). A 0 TRANS_CYCLE parameter means a full-speed transfer.

Table 5-15 defines the number of internal DSP cycles required per TIPB access, given the number of software wait-states (SWSs), the number of external WS (RWS, for READY wait-states, or the number of times the peripheral fails to activate READY before the rising edge of nSTROBE), and the value of the TRANS_CYCLE parameter.

Table 5-15. Total Internal DSP Cycles for Access

TRANS_CYCLE_n Value	SWS	RWS	Total Transaction		
			Memory Read	Memory Write	I/O Read / Write
0	0	N/A	1	2	2
1	1	N/A	2	3	3
Y	SWS	RWS	[(Y + 1) * (1 + RWS)] + 1	[(Y + 1) * (1 + RWS)] + 2	[(Y + 1) * (1 + RWS)] + 2

Note: When TRANS_CYCLE_n = 0 or 1:

- SWS must equal TRANS_CYCLE_n.
- RWS is not applicable.

When TRANS_CYCLE_n > 1:

- 2 ≤ SWS ≤ TRANS_CYCLE_n + 1
- RWS is applicable.

The number of DSP software wait-states is programmed for each TIPB nSTROBE using the SWWSR DSP register (see the *cDSP 54x volume 1: Reference Set*).

Figure 5-13 through Figure 5-16 present XIO/TIPB transactions with varying TRANS_CYCLE parameters.

Figure 5-13 shows a TRANS_CYCLE=1 read access leading to a TIPB transaction duration of two CLK_OUT cycles. The DSP software wait-state generator must be programmed to insert one wait-state for the associated nSTROBE memory space. The read access is followed by a three CLK_OUT cycle write access.

Figure 5-13. Memory Access With TRANS_CYCLE = 1 and SWWSR = 1

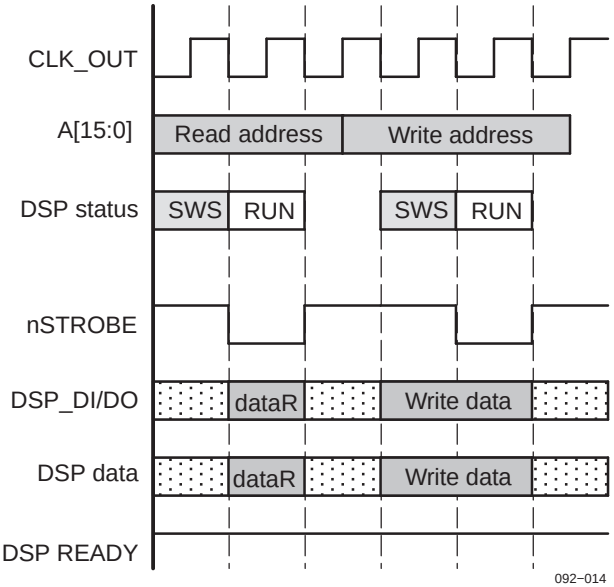


Figure 5-14 shows two XIO/TIPB memory read accesses. A wait cycle is requested by the peripheral on the second read access by maintaining DSP_READY high on the first rising edge of nSTROBE. BWS refers to a wait cycle inserted by the XIO/TIPB bridge.

The DSP is programmed to insert two software wait-states, and the TRANS_CYCLE bit field is set to 2.

Figure 5-14. Access Without and With External Wait-State TRANS_CYCLE = 2 and SWWSR = 2

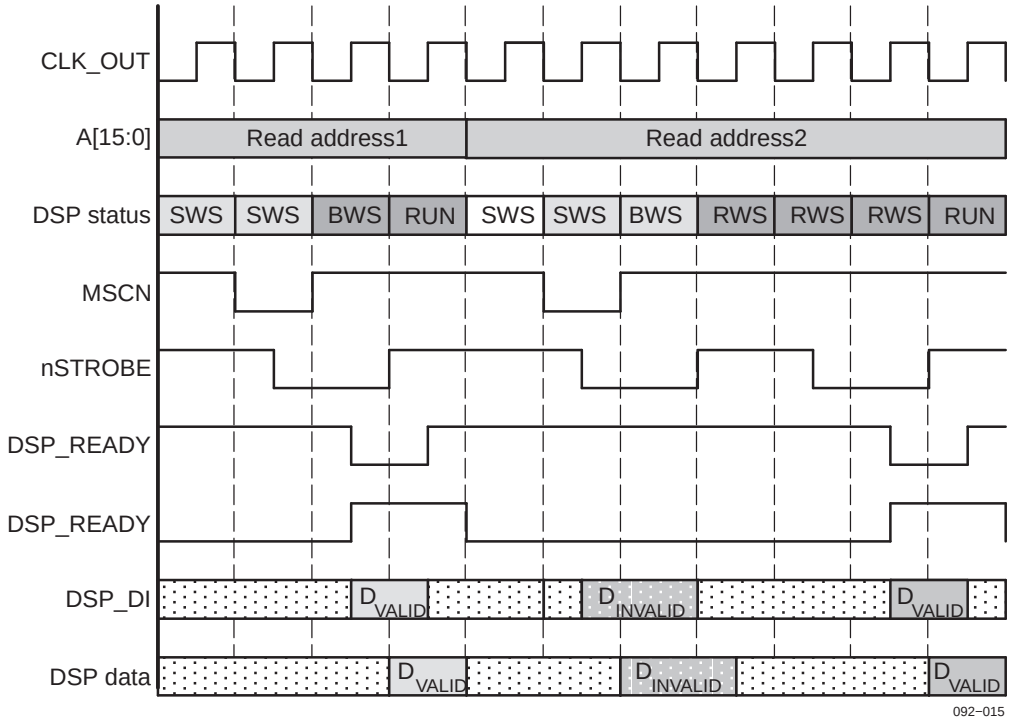


Figure 5-15 shows an I/O space read access. DSP I/O space read accesses are performed in two cycles. The internal and READY wait-states are inserted before the second cycle. For Figure 5-15, the DSP is programmed to insert two software wait-states, and the XIO/TIPB bridge TRANS_CYCLE bit field is set to 3.

Note: TRANS_CYCLE_n can be set to greater than the value of the programmed software wait-states, but this is only true if the value of the programmed software wait-states is 2 or more.

Figure 5-15. I/O Space Access With TRANS_CYCLE = 3 and SWWSR = 2

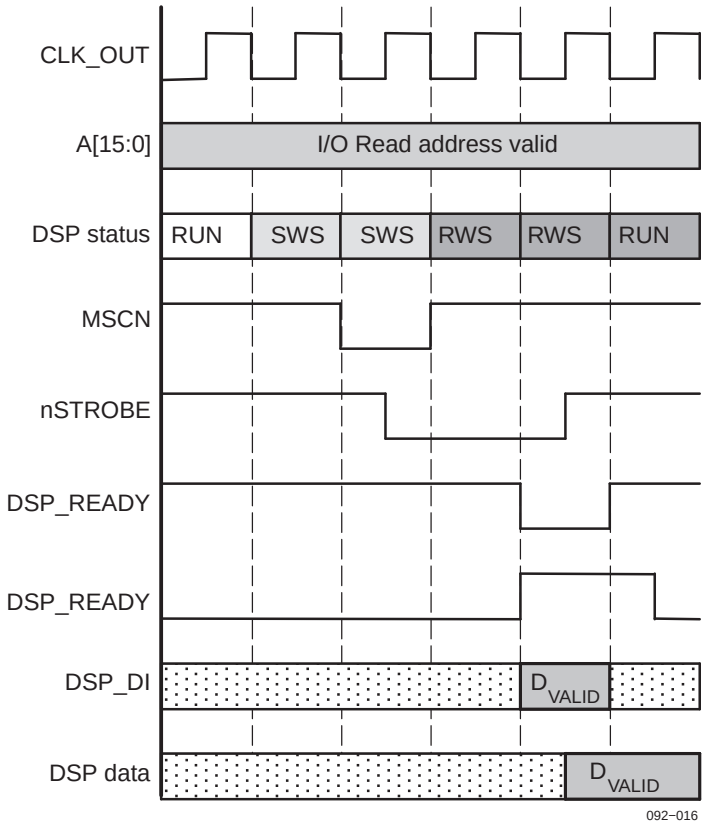
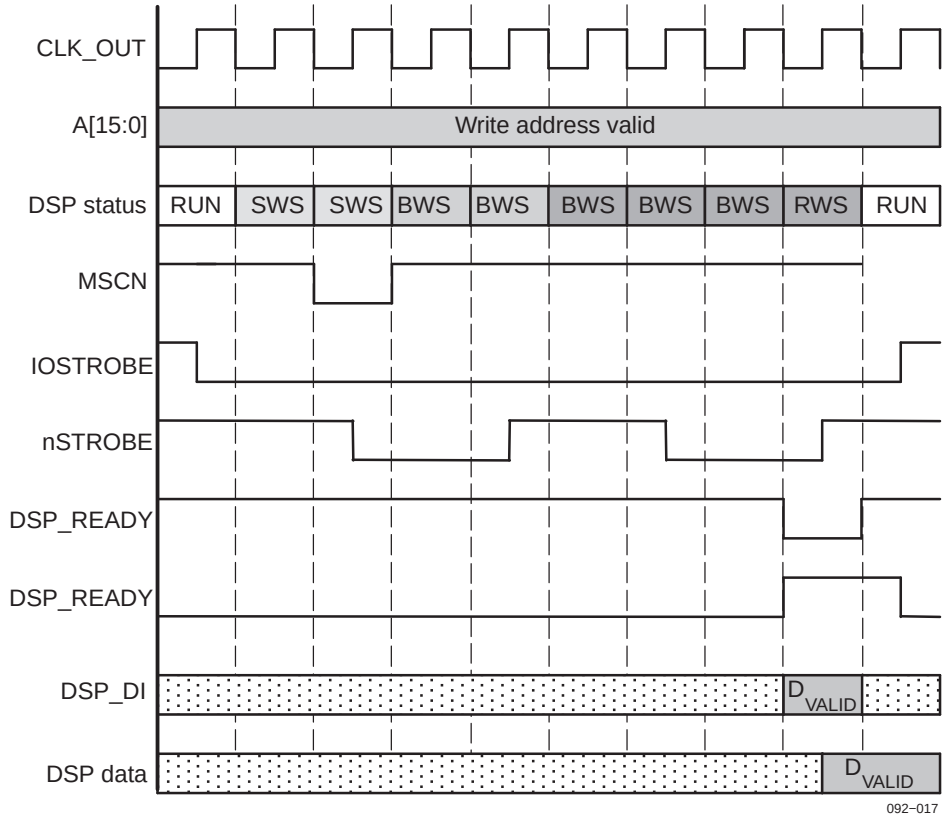


Figure 5-16 shows the same I/O space read access as shown in Figure 5-15 but with an extra nSTROBE cycle inserted. This extra cycle is the result of the target peripheral failing to set READY low before the rising edge of the first nSTROBE pulse.

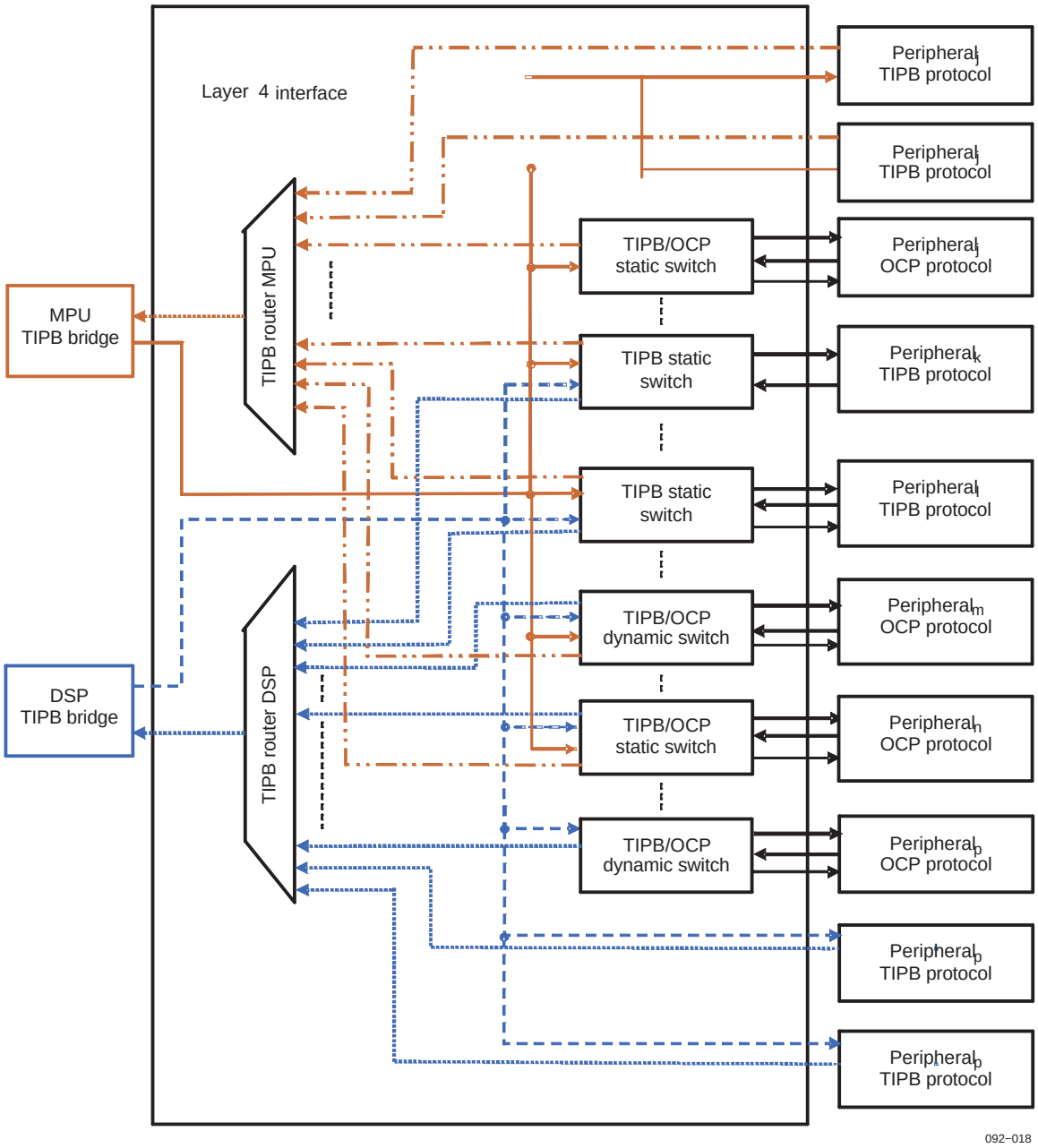
Figure 5-16. I/O Space Read Access With TRANS_CYCLE = 3 and RWS = 1



5.2.3 Peripherals Interconnect

The peripherals interconnect manages accesses to the LOCOSTO peripherals through the MPU TIPB and the DSP TIPB. The peripherals interconnect is composed of a set of dynamic and static switches and wrappers dedicated to the access protocol of each peripheral (see Figure 5-17).

Figure 5-17. Layer 4 Block Diagram



Three different switches are implemented, including two types of static switches for peripheral native protocols and one dynamic switch:

- The TIPB/OCP static switch for OCP native peripherals groups an OCP/ TIPB and TIPB/OCP wrapper and a switch followed by a TIPB router
- The TIPB static switch for TIPB native peripherals includes a TIPB router only.
- The TIPB/OCP dynamic switch includes resynchronization on MPU and DSP accesses and an OCP/TIPB, TIPB/OCP wrapper.

The common protocol on the MPU/DSP side is the TIPB protocol. For each TIPB bridge, a dedicated router is implemented to allow the multiplexing of all the return paths from the peripherals.

In both cases, the TIPB router is a simple multiplexer that transmits and returns information coming from the peripherals to the host (DSP or MPU). The information concerns a current read or write access on a peripheral.

Regarding software implementation, there is no difference between accessing the TIPB or OCP native peripherals. Peripheral access is performed through reading and writing registers inside the TIPB address space.

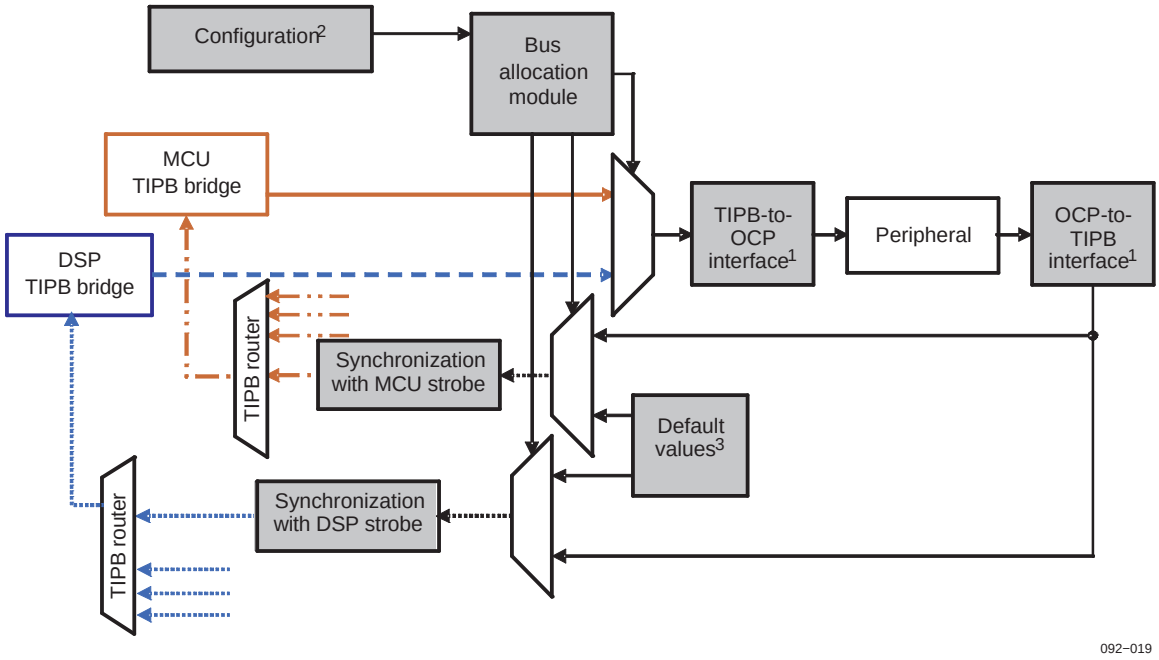
However, error and abort handling is slightly different, depending on the peripheral category, because the OCP/TIPB wrappers have their own way of handling TIPB errors.

The peripherals interconnect requires little configuration. Only the TIPB and OCP static switches must be configured, depending on the host (MPU or DSP) needing to access a peripheral. See Section 5.2.3.2, *TIPB Static Switch*, and Section 5.3.1, *TIPB Static Switch*, for TIPB static switches. Section 5.2.3.3, *TIPB/OCP Static Switch*, and Section 5.3.2, *TIPB/OCP Static Switch*, explain TIPB/OCP static switches. Finally, Section 5.2.3.4, *TIPB/OCP Dynamic Switch*, resumes the use of dynamic switches.

5.2.3.1 Static and Dynamic Switches

Figure 5-18 shows a simplified block diagram of a dynamic switch and a static switch. The differences (see annotations 1, 2, and 3 in Figure 5-18) in the switches used in the LOCOSTO are in the configuration of the bus allocation and in the presence or absence of the TIPB/OCP/TIPB interface.

Figure 5-18. Static and Dynamic Switch Block Diagram



- Interfaces
  - For OCP native peripherals, an OCP/TIPB interface is included with the switch to convert TIPB signals into OCP standard format.
  - For native TIPB peripherals, no interface is implemented.
- Configuration of bus allocation
  - For a dynamic switch, if both bridges request access to a peripheral simultaneously, the arbitration for the allocation of the peripheral is based on the DSW_CONF register value. Only the MPU can access the priority configuration of the switch.
  - The TIPB/OCP static switch allows peripherals to be shared using a basic protocol controlled by the software through two registers: SSW_MPU_CONF for the MPU and SSW_DSP_CONF for the DSP. These registers are accessible by both processors using their respective TIPB.
  - TIPB static switches are similar to TIPB/OCP static switches in functionality. Two sets of registers are available: TSW_xxx_CONF for bus allocation and TSW_xxx_STA for status.
- Default value: To simplify the TIPB router implementation and reduce toggling on buses, a default value is returned to the host that does not access the peripheral.

5.2.3.2 TIPB Static Switch

Figure 5-19 shows the peripherals accessed using the TIPB static switch.

Note: The registers for the TIPB static switch use the following naming convention:

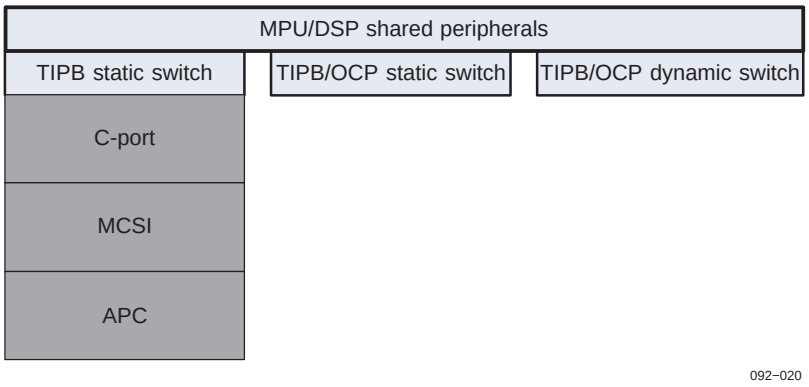
xxx_TSW_yyy_CONF

where:

- xxx denotes the name of the peripheral.
- yyy denotes the name of the processor.

For example, APC_TSW_MPU_CONF

Figure 5-19. Shared Peripherals Using the TIPB Static Switch



5.2.3.2.1 Bus Allocation

The switch between TIPBs is implemented using a basic protocol. The hardware that implements the functionality is reduced to the minimum controls to ensure safe operation. The software must ensure successful operation by thoroughly following the protocol instructions described in Section 5.3.1, *TIPB Static Switch*.

Compared to the OCP/TIPB static switch, the TIPB switch has the functionality to report errors or pending interrupts using the xxx_TSW_yyy_STA register.

Four registers control the accessibility from the host to each statically shared peripheral:

- xxx_TSW_yyy_CONF allows the allocation of the peripheral to a host to be configured.
- xxx_TSW_yyy_STA reports the status on pending interrupt and errors.

5.2.3.2.2 Conflicting and Error Transactions

When the switch is locked by both processors, an error bit is set in the xxx_TSW_yyy_STA register of both processors for user information. The TSW_BOTH_LCK_ERR field of the xxx_TSW_yyy_STA register is set to 1 for both processors.

Note: The software protocol described in Section 5.3.1, *TIPB Static Switch*, should prevent this type of error.

The software must check that there is no interrupt or DMA transfer pending during switch configuration. Nevertheless, a pending command on either an interrupt request or a DMA request is hardware-checked, but this check does not ensure that the switch operation is performed successfully. Hardware checks occur on the falling edges of both the MPU and DSP LOCK bits.

For a pending interrupt request, the TSW_ITPEND_ERR field of the xxx_TSW_yyy_STA register is set to 1 for both processors.

For a pending DMA request, the TSW_DMAREQ_ERR field of the xxx_TSW_yyy_STA register is set to 1 for both processors.

CAUTION

Reallocating a peripheral that has a pending interrupt or a DMA request can produce unexpected results and set the system to an unsteady state.

To clear errors in the xxx_TSW_yyy_STA register, the TSW_ERR_nIRQ bit must be set high. This results in clearing the relevant processor IRQ line and all other status bits. Each processor is in charge of clearing its own status register.

CAUTION

Simultaneous access when setting both the DSP_PERIPH_LOCK and MPU_PERIPH_LOCK bits can lead to unpredictable behavior and must be prevented.

5.2.3.2.3 Reset Methodology

A reset on the TIPB static switch configuration register resets the DSP_PERIPH_LOCK bit and sets the MPU_PERIPH_LOCK bit, which turns the TIPB static switch to the MPU position.

5.2.3.3 TIPB/OCP Static Switch

Figure 5-20 shows the peripheral accessed using the TIPB/OCP static switch.

Note: The registers for the TIPB/OCP static switch use the following naming convention:

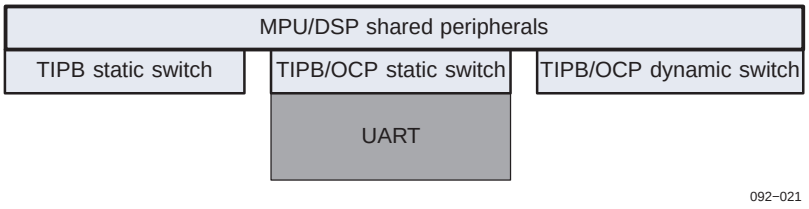
xxx_SSW_xxx_CONF

where:

- xxx denotes the name of the associated peripheral.
- yyy denotes the name of the associated processor.

For example, UART_SSW_MPU_CONF

Figure 5-20. Shared Peripheral Using the TIPB/OCP Static Switch



5.2.3.3.1 Bus Allocation

The static switch module allows a given peripheral to be switched between the DSP and MPU processors. The accessibility between the two TIPBs is implemented using a basic protocol controlled carefully by the software.

Note: Ensure that the ongoing DMA transfers from/to the peripheral are completed before updating the peripheral ownership. There is no protection from the static switch hardware.

If the static switch is not properly programmed before an access to a shared peripheral register, the corresponding ready signal is not sent back to the host. The result is a timeout error generated by the XIO/TIPB bridge or the API/TIPB bridge.

Two registers control the accessibility from the host to each statically shared peripheral:

- xxx_SSW_MPU_CONF in the MPU peripheral address space: To perform an MPU access, the MPU software must set the MPU_SWITCH bit in the SSW_MPU_CONF register.
- xxx_SSW_DSP_CONF in the DSP peripheral address space: To perform a DSP access, the DSP software must set the DSP_SWITCH bit in the SSW_DSP_CONF register.

Figure 5-21 shows the transition from the MPU to DSP allocation.

Figure 5-21. Transition State From MPU to DSP Allocation

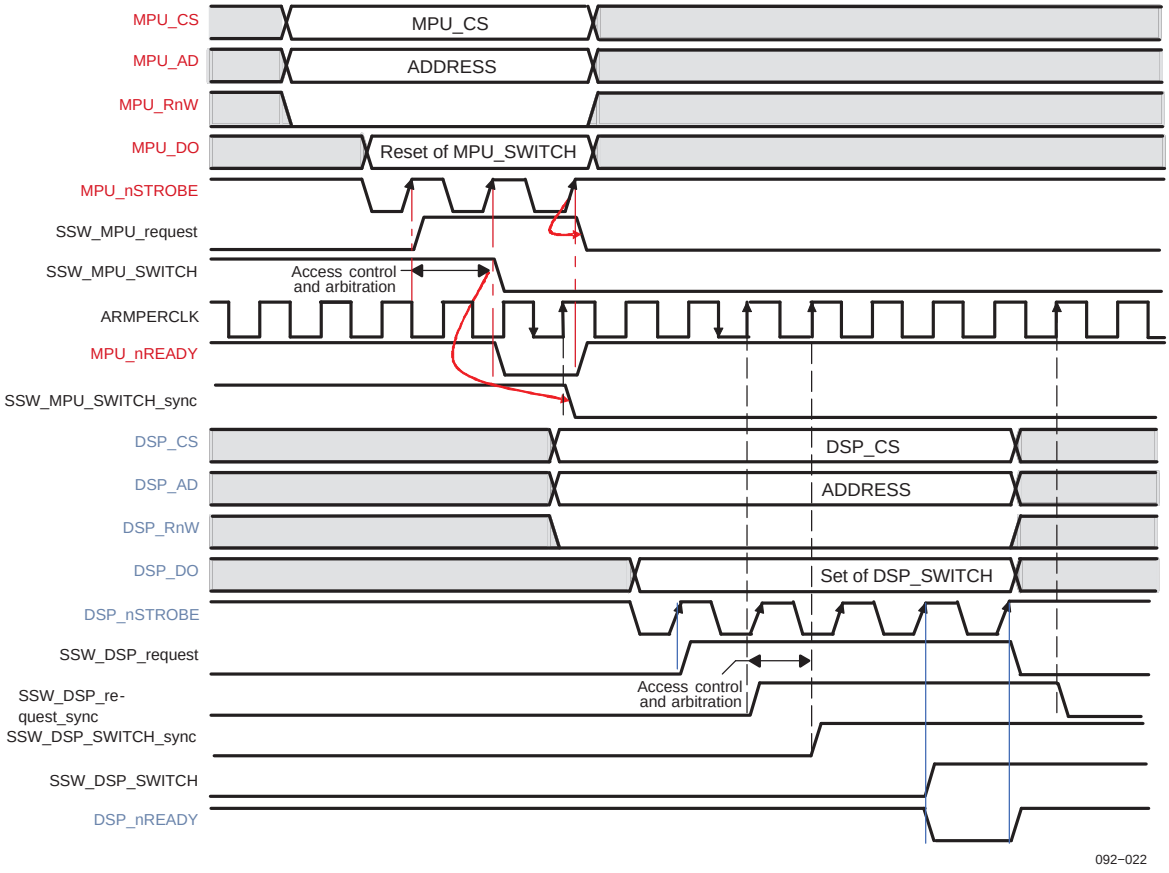


Figure 5-22 shows the static switch DSP read transaction.

Figure 5-22. Static Switch DSP Read Transaction

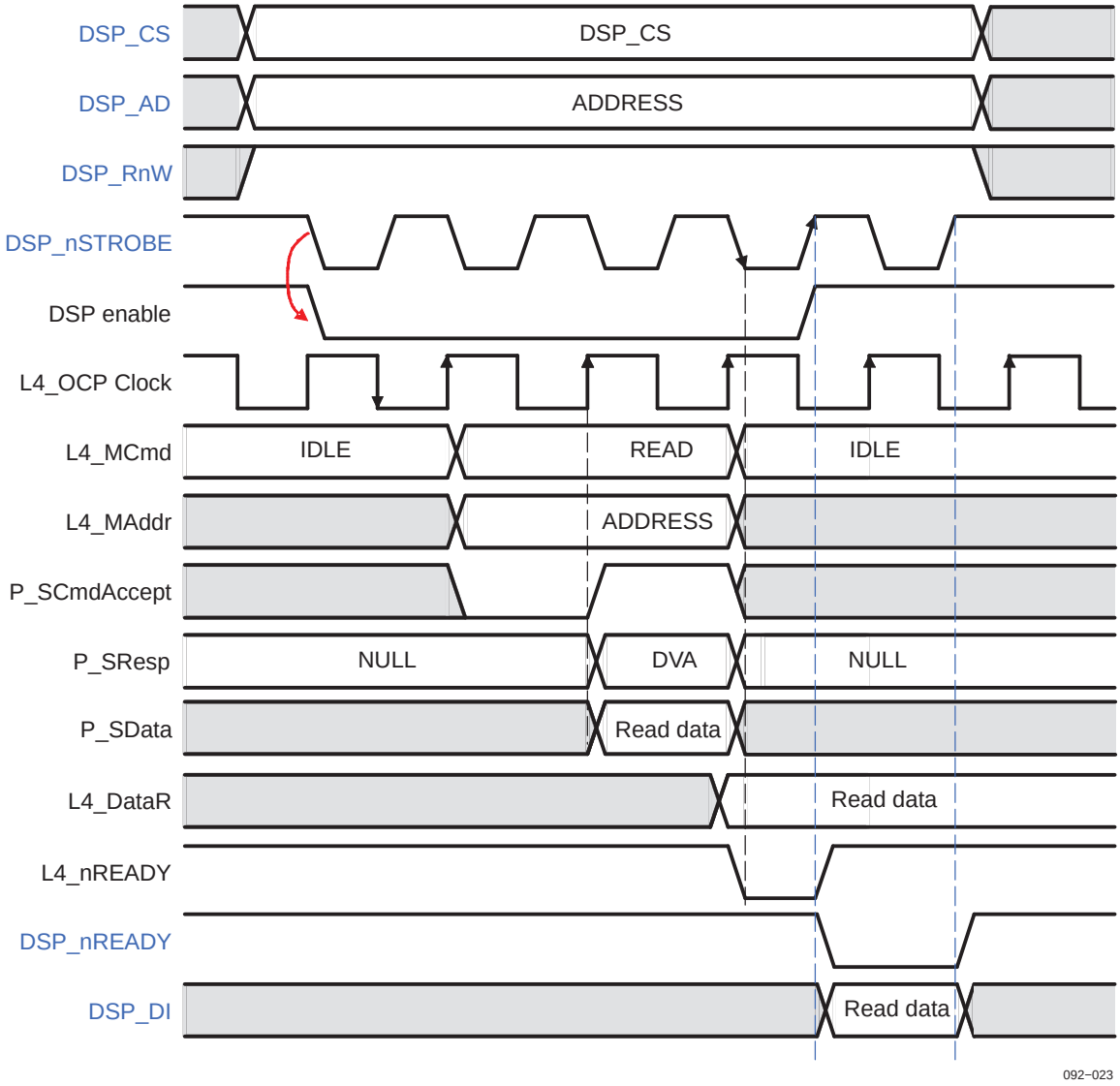
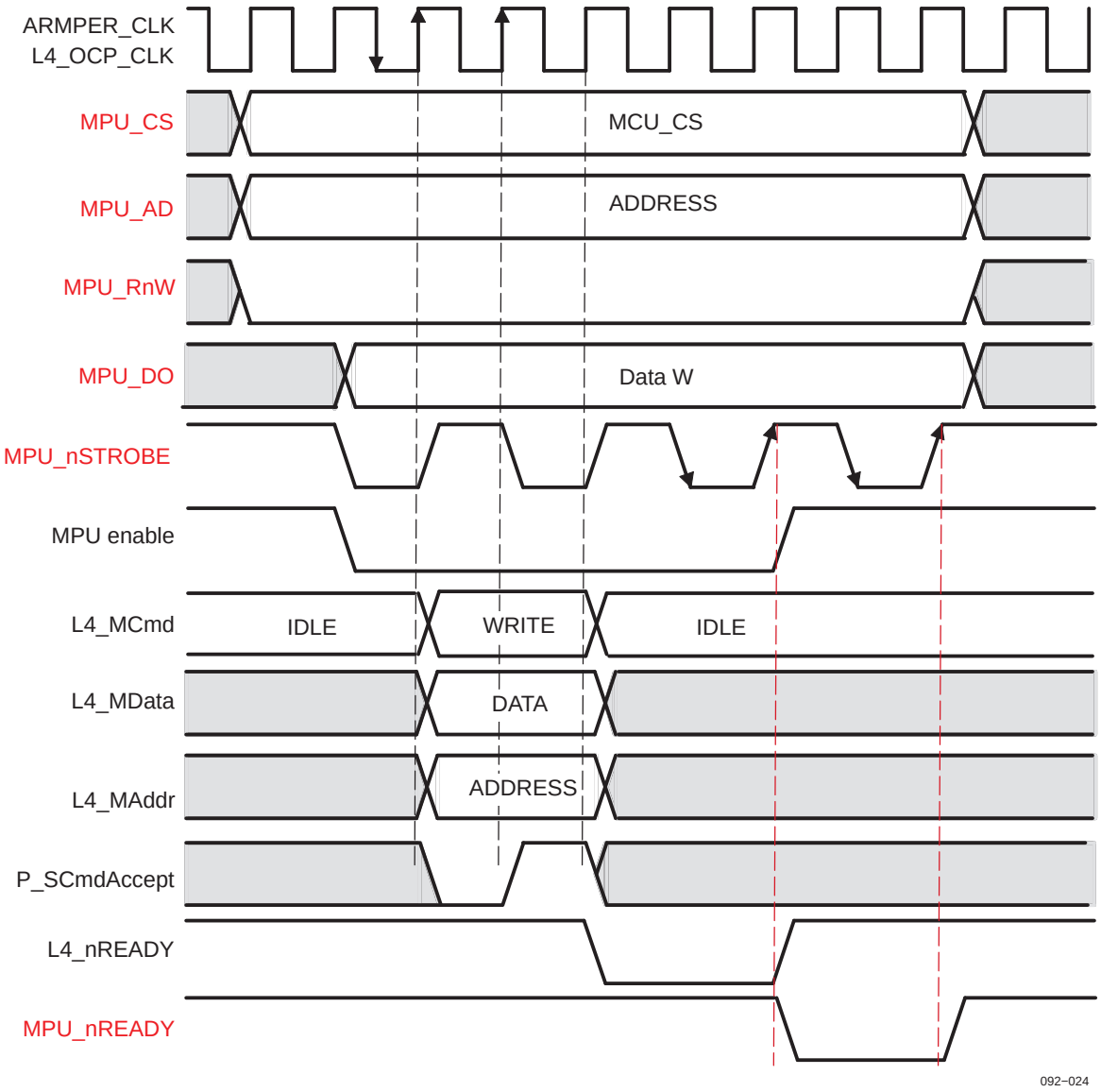


Figure 5-23 shows the static switch MPU write transaction.

Figure 5-23. Static Switch MPU Write Transaction



5.2.3.3.2 16-Bit Accesses on 32-Bit Atomic Registers

CAUTION

There is no protection inside the static switch to ensure the atomic reading or writing of one 32-bit register with two 16-bit accesses. The nonoverlapping requests from the two hosts must be controlled by software.

5.2.3.3.3 Conflicting and Abort Transactions

If the MPU wants to access the xxx_SSW_MPU_CONF register when the DSP_SWITCH bit is set, the access is completed but a write access has no effect on the register. Symmetrically, if the DSP wants to access the xxx_SSW_DSP_CONF register when the MPU_SWITCH bit is set, the access is completed but a write access has no effect on the register.

Setting both the DSP_SWITCH and MPU_SWITCH bits is not possible by design. An access control and arbitration phase in the ARMPER_CLK clock domain ensures that this programming conflict never occurs (for more information, see Chapter 6, *Power, Reset, and Clock Management*). If two write requests are simultaneous, only the MPU write access is served.

5.2.3.3.4 Reset Methodology

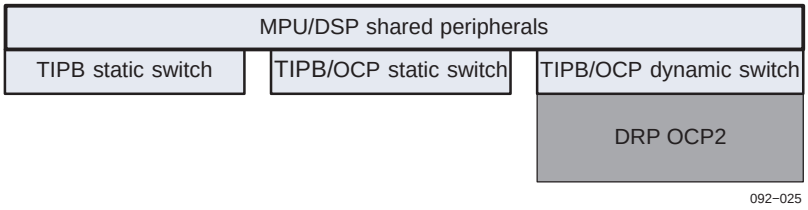
An MPU or DSP initiator reset is always used internally by the static switch. A reset on the static switch configuration register resets the DSP_SWITCH bit and sets the MPU_SWITCH bit, which turns the static switch in an MPU allocation.

A No Reset signal is forwarded through the switch to the connected peripheral.

5.2.3.4 TIPB/OCP Dynamic Switch

Figure 5-24 shows the peripheral accessed using the TIPB/OCP dynamic switch.

Figure 5-24. Shared Peripheral Using the TIPB/OCP Dynamic Switch



5.2.3.4.1 Bus Allocation

The DRP OCP2 can be accessed by the MPU and DSP using a dynamic switch. For this peripheral, the hardware determines which initiator accesses the peripheral based on a simple arbitration scheme. Software implementation for the dynamic switches is not required, but the software must avoid dynamically accessing the peripheral simultaneously with the MPU and the DSP.

One register controls the priority allocated for simultaneous access: DSW_CONF is accessible only by the MPU.

Note: The names of the registers for the dynamic switch begins with the name of the module.

For example, the name of the dynamic switch for the DRP peripheral is DRP_DSW_CONF.

Figure 5-25 and Figure 5-26 represent read and write transactions from the DSP to/from the peripheral.

Figure 5-25. Dynamic Switch DSP Read Transaction

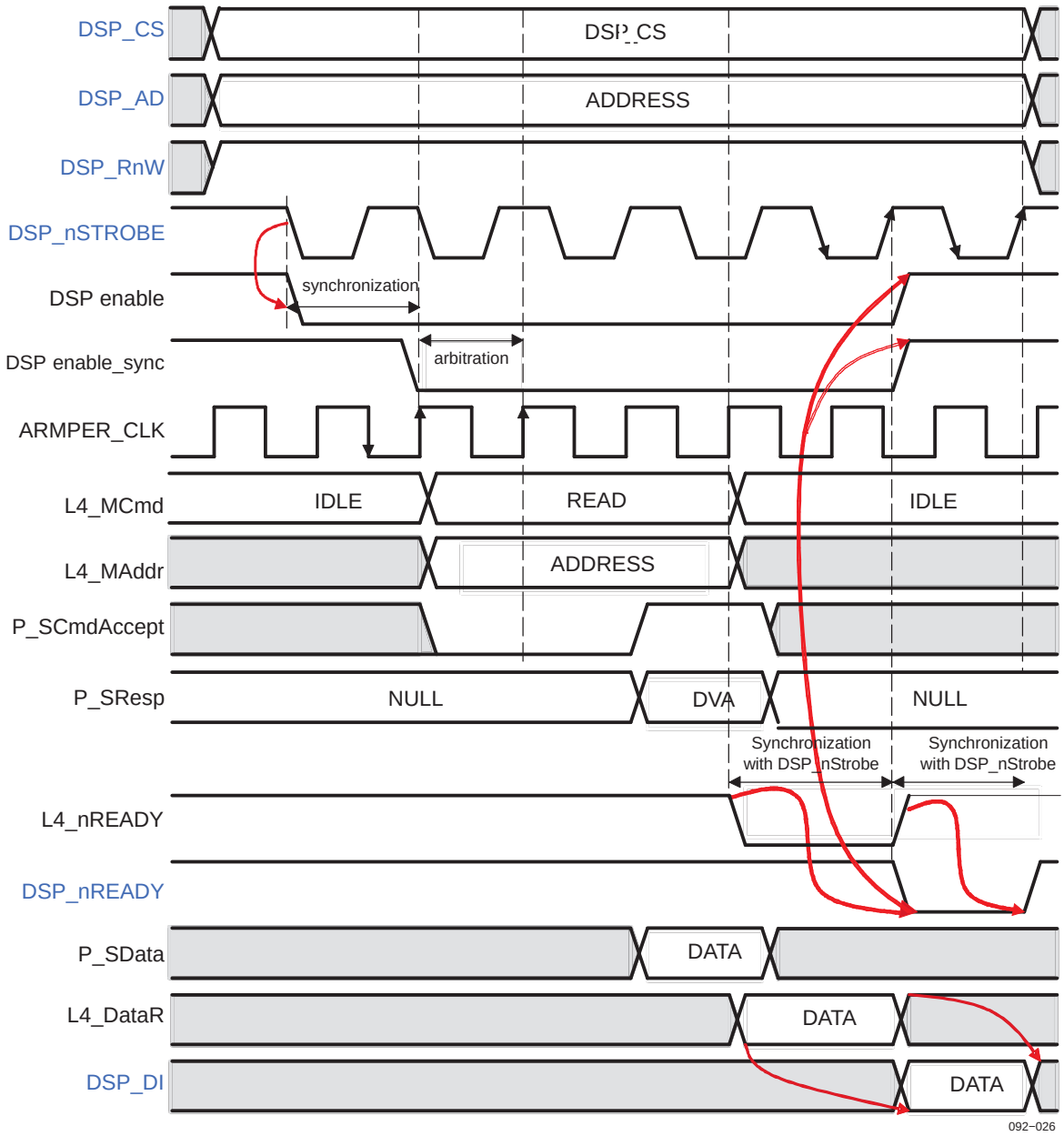
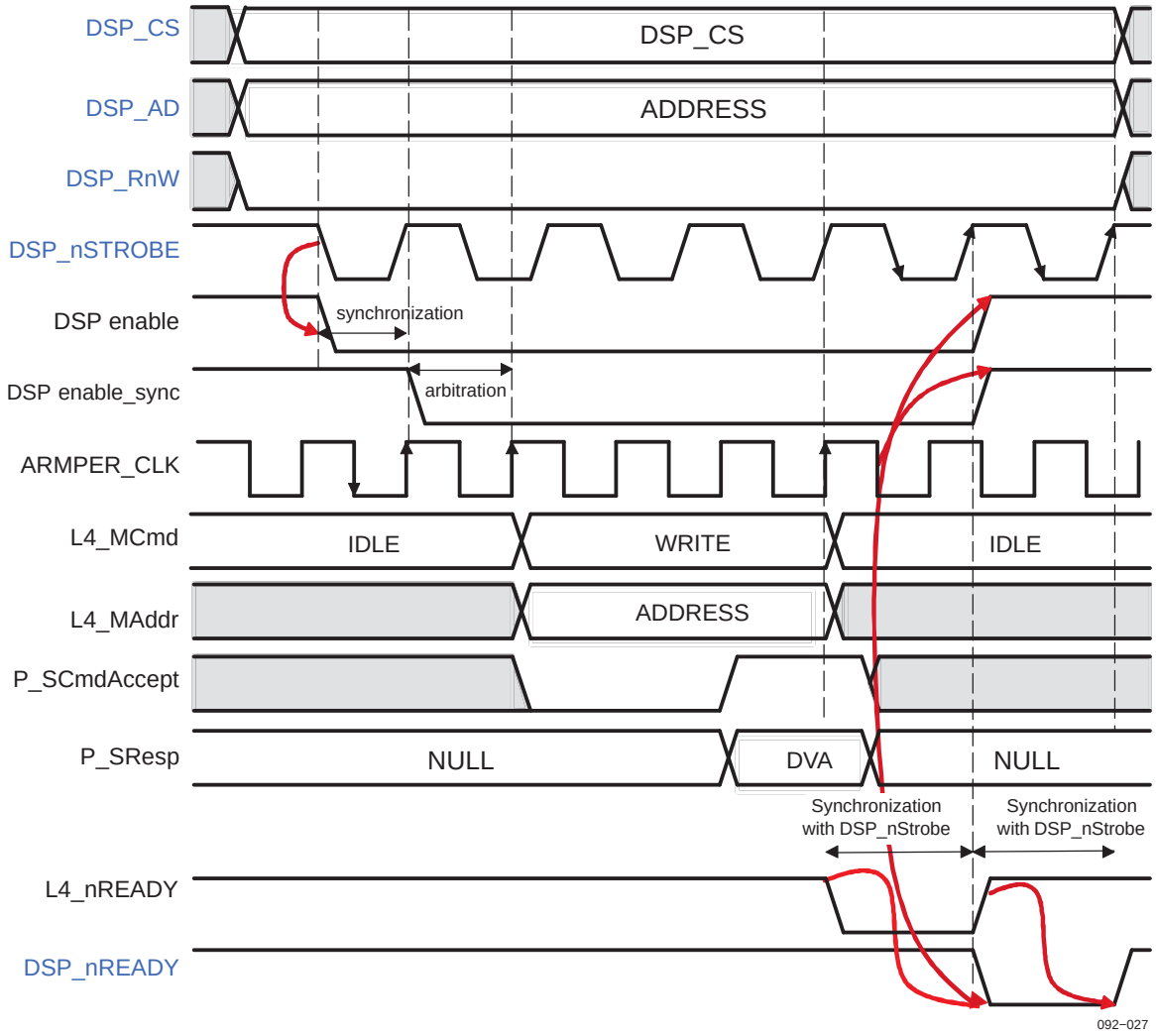


Figure 5-26. Dynamic Switch DSP Write Transaction



5.2.3.4.2 16-Bit Accesses on 32-Bit Atomic Registers

CAUTION

There is no protection inside the dynamic switch to ensure the atomic reading or writing of one 32-bit register with two 16-bit accesses. The nonoverlapping requests from the two hosts must be controlled by software.

5.2.3.4.3 Conflicting and Abort Transactions

When one host requires an access to the peripheral, an enable_host signal is set (synchronously with the host strobe). The enable_host is then resynchronized with the dynamic switch clock to perform arbitration. Arbitration occurs during one dynamic switch clock cycle. For simultaneous accesses, the master is selected based on the setting of DSW_REG:

- If DEW_REG is 0, the MPU TIPB bridge has priority and the DSP XIO/ TIPB bridge waits until the ready is sent to the MPU API/TIPB bridge.
- If DEW_REG is 1, the DSP XIO/TIPB bridge has priority and the MPU API/ TIPB bridge waits until the ready is sent to the DSP XIO/TIPB bridge.

The next master then performs its access.

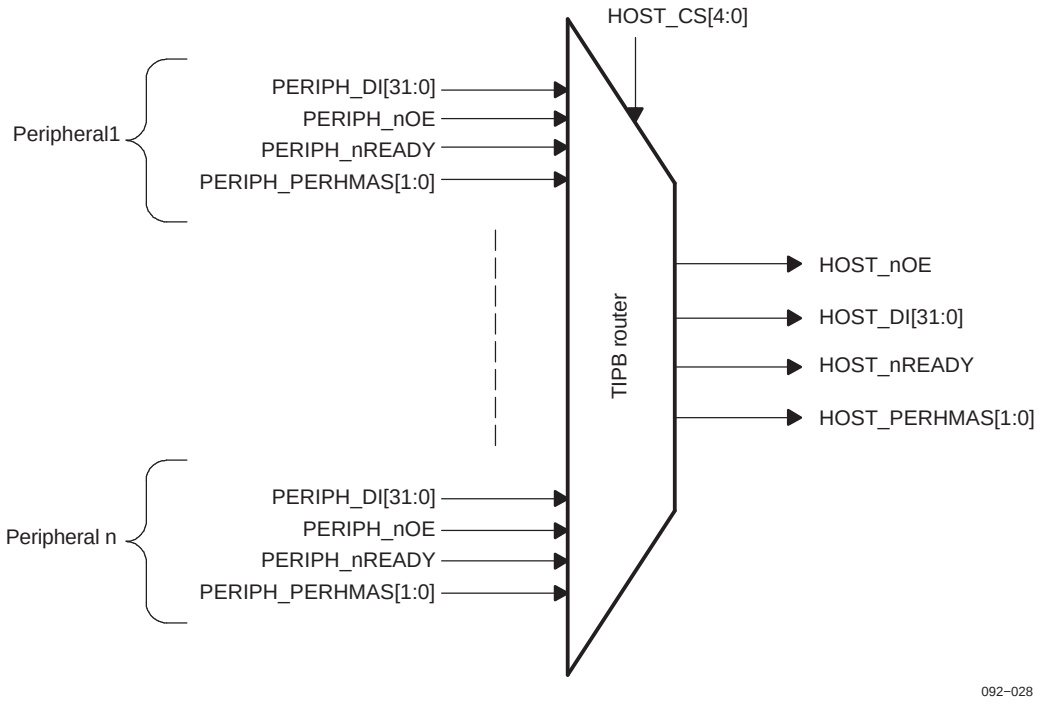
5.2.3.4.4 Reset Methodology

An MPU initiator reset is always used with the dynamic switch to reset an eventual MPU ongoing transaction. A DSP initiator reset is also used with the dynamic switch to reset an eventual DSP ongoing transaction. A No Reset signal is forwarded through the switch to the connected peripheral.

5.2.3.5 Routers TIPB

As shown in Figure 5-27, the TIPB router is a simple multiplex gate that returns information to the host (DSP or MPU) during a read or write access on a peripheral.

Figure 5-27. DSP and MPU TIPB Router

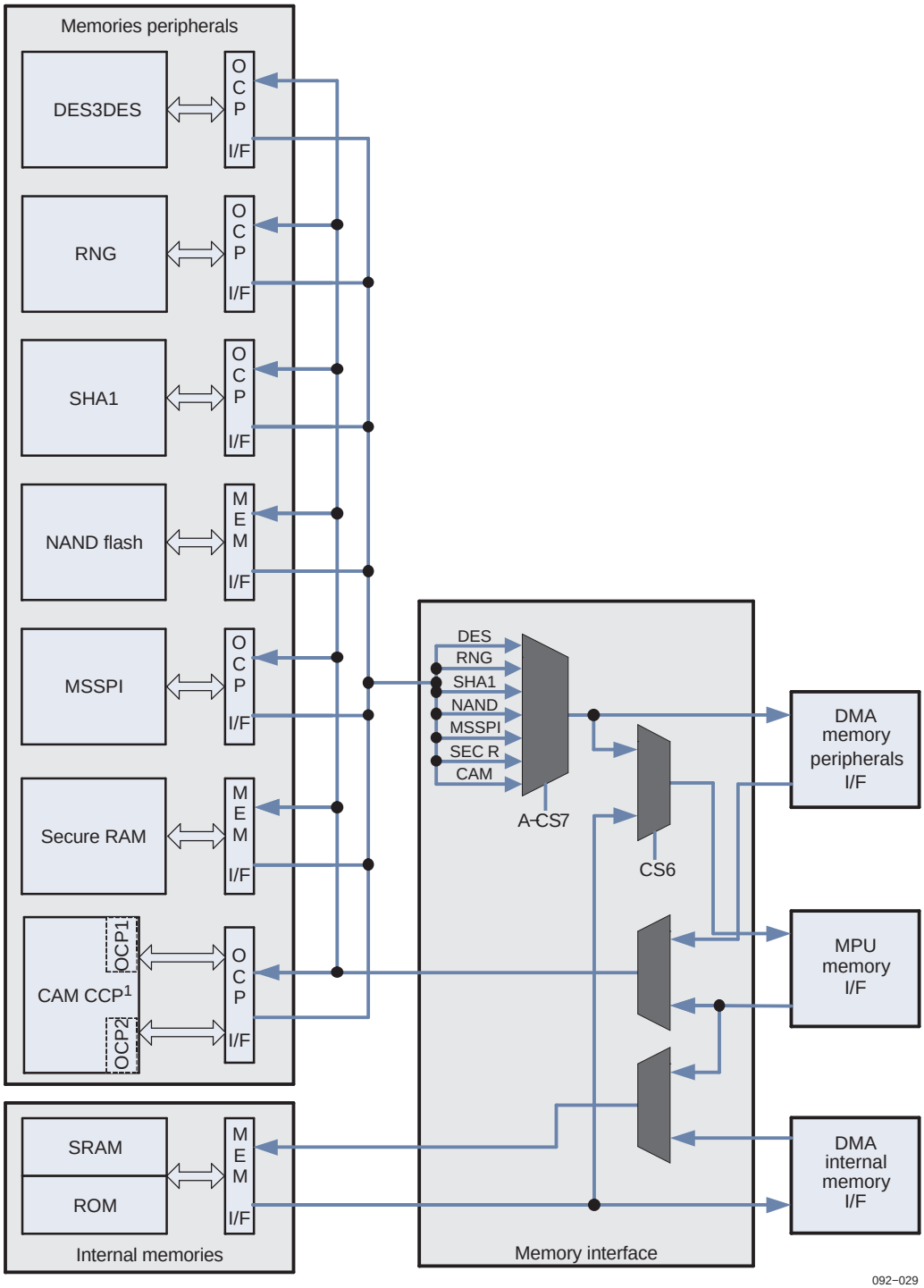


5.2.4 Memory Interconnect

The memory interface is the LOCOSTO device central interconnect that manages all accesses on the OCP bus for internal and memory peripherals (see Figure 5-28).

The role of the memory interface is to control the flow of data to or from the memories. This action is performed by a set of multiplexers driven by a chipselect coming from the MPU memory interface (for more information, see Chapter 3, *MPU Subsystem)*.

Figure 5-28. Memory Interface



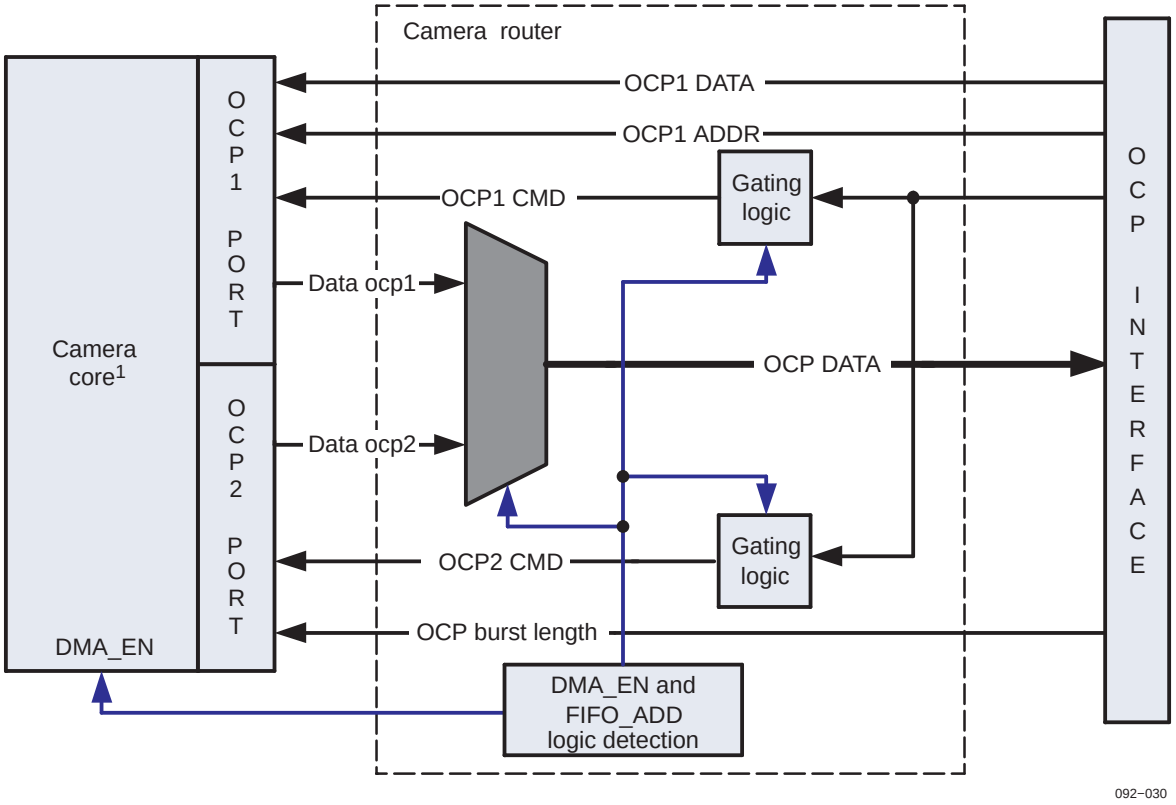
(1) The camera interface is not available in LOCOSTO Lite.

Two chip-selects are used to retrieve data going to the MPU:

- CS7: Selects the peripheral having access to the bus
- CS6: Because the MPU has a single port for internal and memory-like peripherals, it is necessary to multiplex the data going to the MPU using the CS6 chip-select.

Each peripheral has an interface that converts the signal from the MPU or the DMA to an OCP protocol, except the camera core module, which is a native dual-OCP bus (OCP1-MPU and OCP2-DMA), as shown in Figure 5-29. The camera interface is not available on LOCOSTO Lite device.

Figure 5-29. OCP Interconnect for CCP Camera



(1) The camera interface and GPRS modules are not available in LOCOSTO Lite.

The camera core module is connected to the 32-bit, internal-memory slow bus through a specific router that permits the MPU or the DMA to access the proper camera core resources (OCP1/OCP2) from a unique 32-bit bus.

The target ports are defined as follows:

- OCP1-MPU: Configures the camera core module or read/write FIFO
- OCP2-DMA: Used by the DMA controller to read the data from the FIFO

5.2.5 Error Reporting

A TIPB abort can occur because of a time-out on a TIPB access: If a peripheral does not activate its ready signal after a predetermined number of strobe cycles, the TIPB bridge generates an abort. In this case, the bridge sends an abort signal (AABORT for peripherals, DMA_ABORT for the DMA controller, ARM_ABORT for the MPU, and DSP_ABORT for the DSP). The module uses the abort to kill the current access. The cause of a time-out might be an access to an unmapped address or a peripheral hanging.

The maximum number of strobe cycles is programmable for the XIO/TIPB bridge within the TIMEOUT bit field BRIDGE_CNTL[7:0] and for the API/TIPB bridge within the TIMEOUT bit field TIPB_CNTL[15:8].

5.3 Interconnect Programming Model

5.3.1 TIPB Static Switch

The following programming sequence enables MPU access to a peripheral:

1. Check that the MPU_PERIPH_LOCK bit is reset in the TSW_MPU_CONF register (peripheral already attached).
2. Verify the status of TSW_MPU_STA for unprocessed errors.
3. Check that the DSP_PERIPH_LOCK and MPU_PERIPH_LOCK bits are reset in the TSW_MPU_CONF register (no access is ongoing on the peripheral).
4. Set the MPU_PERIPH_LOCK bit in the TSW_MPU_CONF register (MPU S/W write 1).
5. Check that the MPU_PERIPH_LOCK bit is set (in case of priority conflict with the DSP software).
6. Perform the shared peripheral access through the MPU TIPB.
7. After verifying the status of the TSW_MPU_STA register for unprocessed errors, reset the MPU_PERIPH_LOCK bit in the TSW_MPU_CONF register.

Figure 5-30 shows a flowchart that allocates the peripheral through the TIPB static switch to the MPU. The same process can be repeated for an allocation to the DSP by changing the TSW_MPU_CONF register to TSW_DSP_CONF and the MPU_PERIPH_LOCK register to DSP_PERIPH_LOCK.

The following programming sequence enables DSP access to the peripheral:

1. Check that the DSP_PERIPH_LOCK bit is reset in the TSW_DSP_CONF register (peripheral already attached).
2. Verify the status of TSW_DSP_STA for unprocessed errors.
3. Check that the DSP_PERIPH_LOCK and MPU_PERIPH_LOCK bits are reset in the TSW_DSP_CONF register (no access is ongoing on the peripheral).
4. Set the DSP_PERIPH_LOCK bit in the TSW_DSP_CONF register (DSP S/W write 1).
5. Check that the DSP_PERIPH_LOCK is set (in case of priority conflict with the MPU software).
6. Perform the shared peripheral access through the DSP TIPB.
7. After verifying the status of the TSW_DSP_STA register for unprocessed errors, reset the DSP_PERIPH_LOCK bit in the TSW_DSP_CONF register.

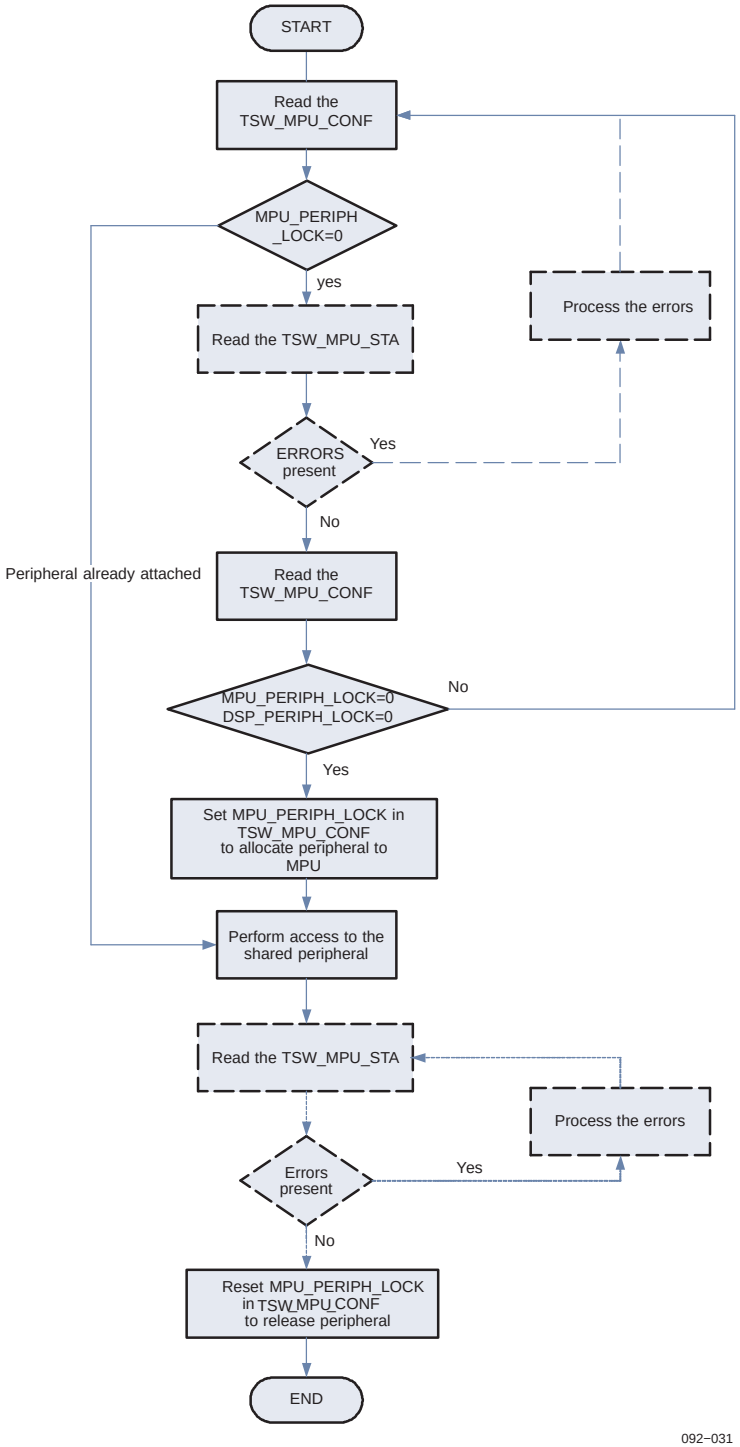
Note: The dashed boxes in Figure 5-30 indicate actions that, although not mandatory, offer more security for additional operations.

CAUTION

Setting both the LOCK MPU and the LOCK DSP bits high can cause unexpected spikes and glitches on the peripheral clock (but not on the strobe) when one LOCK bit is asserted or deasserted.

This can occur because the peripheral clock is gated and not multiplexed. When both LOCK bits are set, the MPU still owns the peripheral and the peripheral clock receives the MCLK clock.

Figure 5-30. Programming Guide for the TIPB Static Switch



5.3.2 TIPB/OCP Static Switch

The following programming sequence enables DSP access to the peripheral:

1. Check that the MPU_SWITCH bit is reset (no MPU access is ongoing on the same peripheral).
2. Set the DSP_SWITCH bit in the SSW_DSP_CONF register (DSP S/W write 1).
3. Check that the DSP_SWITCH is set (in case of a priority conflict with the MPU software).

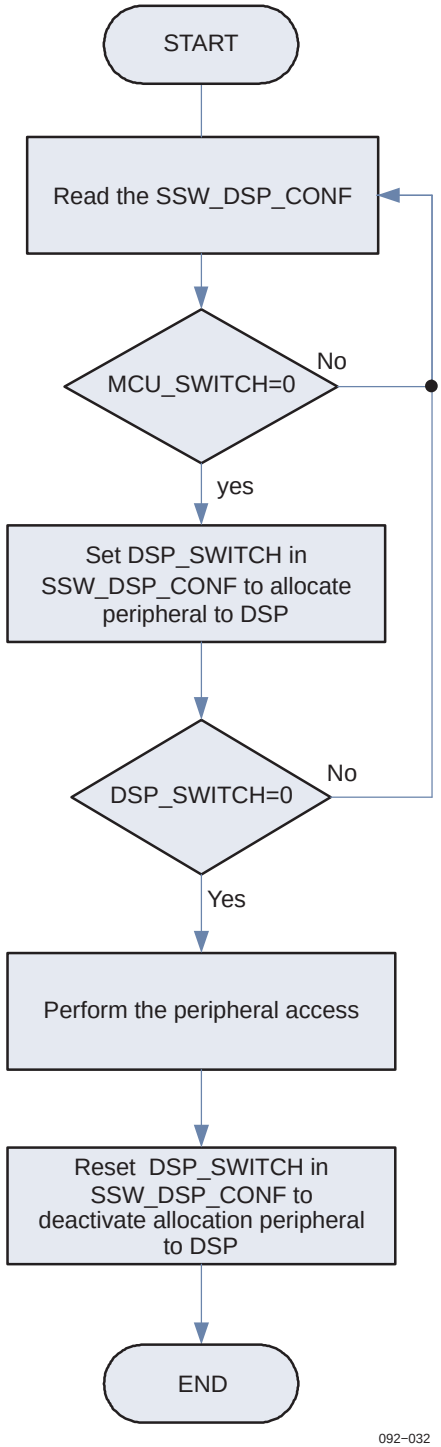
4. Perform the shared peripheral access through the DSP TIPB.
5. After completing access, reset the DSP_SWITCH bit in the SSW_DSP_CONF register.

The following programming sequence enables MPU access to the peripheral:

1. Check that the DSP_SWITCH bit is reset (no DSP access is ongoing on the same peripheral).
2. Set the MPU_SWITCH bit in the SSW_MPU_CONF register (MPU S/W write 1).
3. Check that the MPU_SWITCH is set (in case of a priority conflict with the DSP software).
4. Perform the shared peripheral access through the MPU TIPB.
5. After completing access, reset the MPU_SWITCH bit in the SSW_MPU_CONF register.

Figure 5-31 shows a flowchart allocating the TIPB/OCP switch to the DSP. The same process can be repeated for an allocation to the MPU by changing the SSW_DSP_CONF register to SSW_MPU_CONF.

Figure 5-31. Programming Guide TIPB/OCP Static Switch



5.4 Interconnect Register Manual

This section describes the registers used for both MPU and DSP access.

Table 5-16 and Table 5-17 summarize the MPU access and DSP access instances.

Table 5-16. MPU Access Instance Summary

Module Name	Base Address	Size
UART TIPB/OCP switch	0xFFFF 7280	2 bytes
DRP dynamic switch	0xFFFF 8000	2K bytes
APC TIPB static switch	0xFFFF 8800	32 bytes
MCSI TIPB static switch	0xFFFF 8820	32 bytes
C-PORT TIPB static switch	0xFFFF 8840	32 bytes
API/TIPB bridge	0xFFFF F900	256 bytes

Table 5-17. DSP Access Instance Summary

Module Name	Base Address	Size
UART TIPB/OCP switch	0x7900	256 bytes
APC TIPB Static Switch	0x7C80	4 bytes
MCSI TIPB Static Switch	0x7CA0	4 bytes
C-PORT TIPB Static Switch	0x7CC0	4 bytes
XIO/TIPB bridge	0xF800	512 bytes

5.4.1 Module Register Mapping Summary

Table 5-18 through Table 5-24 summarize the module register mapping.

Table 5-18. API/TIPB Bridge Register Offset Address Only MPU Access

Register Name	Type	Register Width (Bits)	Address offset	Physical Address
TIPB_CNTL	R/W	16	0x00	0xFFFF F900
API_WS	R/W	16	0x02	0xFFFF 9002
MPU_TIPB_CNTL	R/W	16	0x04	0xFFFF 9004
ENHANCED_TIPB_CNTL	R/W	16	0x06	0xFFFF 9006

Table 5-19. TIPB Static Switch Register Offset Address With MPU Access

Register Name	Type	Register Width (Bits)	Address offset	Physical Address
APC_TSW_MPU_CONF	R/W	16	0x00	0xFFFF 8800
APC_TSW_MPU_STA	R/W	16	0x02	0xFFFF 8804
MCSI_TSW_MPU_CONF	R/W	16	0x00	0xFFFF 8820
MCSI_TSW_MPU_STA	R/W	16	0x02	0xFFFF 8824
CPORT_TSW_MPU_CONF	R/W	16	0x00	0xFFFF 8840
CPORT_TSW_MPU_STA	R/W	V	0x02	0xFFFF 8844

Table 5-20. TIPB/OCP Static Switch Register Offset Address With MPU Access

Register Name	Type	Register Width (Bits)	Address offset	Physical Address
UART_SSW_MPU_CONF	R/W	16	0x00	0xFFFF 7280

Table 5-21. Dynamic Switch Register Offset Address

Register Name	Type	Register Width (Bits)	Address offset	Physical Address
DRP_DSW_CONF	R/W	16	0x00	0xFFFF 8000

Table 5-22. XIO/TIPB Bridge Register Offset Address Only DSP Access

Register Name	Type	Register Width (Bits)	Address offset	Physical Address
TRANSFER_RATE	R/W	16	0x00	0xF800
BRIDGE_CNTL	R/W	16	0x01	0xF801

Table 5-23. TIPB Static Switch Register Offset Address With DSP Access

Register Name	Type	Register Width (Bits)	Address offset	Physical Address
APC_TSW_DSP_CONF	R/W	16	0x00	0x7C80
APC_TSW_DSP_STA	R/W	16	0x00	0x7C81
MCSI_TSW_DSP_CONF	R/W	16	0x00	0x7CA0
MCSI_TSW_DSP_STA	R/W	16	0x01	0x7CA1
CPORT_TSW_DSP_CONF	R/W	16	0x00	0x7CC0
CPORT_TSW_DSP_STA	R/W	16	0x01	0x7CC1

Table 5-24. TIPB/OCP Static Switch Register Offset Address With DSP Access

Register Name	Type	Register Width (Bits)	Address offset	Physical Address
UART_SSW_DSP_CONF	R/W	16	0x00	0x7900

5.4.2 Register Description

Table 5-25 through Table 5-45 describe the register bits.

Table 5-25. TIPB_CNTL

Address Offset	0x00		
Physical Address	0xFFFF F900	**Instance**	API/TIPB bridge
Description	Set up of TIPB time-out and access factor		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEOUT								ACCESS_FACTOR1				ACCESS_FACTOR0			

Bits	Field Name	Description	Type	Reset
15:8	TIMEOUT	Value to limit the maximum time a peripheral can stall the processor. When starting a cycle on TIPB, the time out counter is loaded with this value. If the current cycle is not finished when the counter reach 0, abort is generated.	R/W	0xFF
7:4	ACCESS_FACTOR1	Division factor of MPU_nSTROBE(1) allowing access to slow peripherals by reducing the access frequency nASTROBE low level pulse duration: • Access_factor = 0 → ClkBridge52 low level. • Access_factor ≠ 0→ access_factor ↔ number of ClkBridge52 periods to use.	R/W	0xF
3:0	ACCESS_FACTOR0	Division factor of MPU_nSTROBE(0) allowing access to slow peripherals by reducing the access frequency.	R/W	0xF

Bits	Field Name	Description	Type	Reset
		nASTROBE low level pulse duration: • Access_factor = 0 → ClkBridge52 low level. • Access_factor ≠ 0 → access_factor ↔ number of ClkBridge52 periods to use.		

Table 5-26. API_WS

Address Offset	0x02		
Physical Address	0xFFFF F902	**Instance**	API/TIPB bridge
Description	Number of wait-states asserted for each API access.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								API_WS_H						API_WS_L	

Bits	Field Name	Description	Type	Reset
15:10	Reserved	Reserved	R	Undefined
9:5	API_WS_H	Indicates the number of wait-states inserted for each API access when lead frequency is the highest one (CLKOUT_HIGH = 1)	R/W	0x1F
4:0	API_WS_L	Indicates the number of wait-states inserted for each API access when lead frequency is not the highest one (CLKOUT_HIGH = 0)	R/W	0x1F

Table 5-27. MPU_TIPB_CNTL

Address Offset	0x04		
Physical Address	0xFFFF F904	**Instance**	API/TIPB bridge
Description	Enable write buffer		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														W_BUF_EN_1	W_BUF_EN_0

Bits	Field Name	Description	Type	Reset
15:2	Reserved	Reserved	R	Undefined
1	W_BUF_EN_1	1: Write buffer is enabled for strobe domain 1 0: Write buffer is bypassed.	R/W	0x1
0	W_BUF_EN_0	1: Write buffer is enabled for strobe domain 0 0: Write buffer is bypassed.	R/W	0x1

Table 5-28. ENHANCED_TIPB_CNTL

Address Offset	0x06		
Physical Address	0xFFFF F906	**Instance**	API/TIPB bridge
Description	Enable Time _UnicodeEncodeError_out		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															TIMEOUT_EN

Bits	Field Name	Description	Type	Reset
15:1	Reserved	Reserved	R	Undefined
0	TIMEOUT_EN	Enable/Disable TIPB TIMEOUT watchdog: 1: Enable the TIMEOUT feature 0: Disable the TIMEOUT feature	R/W	1

Table 5-29. APC_TSW_MPU_CONF

Address Offset	0x00		
Physical Address	0xFFFF 8800	**Instance**	APC TIPB static switch
Description	To allocate APC peripheral to MPU		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DSP_PERIPH_LOCK	MPU_PERIPH_LOCK

Bits	Field Name	Description	Type	Reset
15:2	Reserved	Reserved	R	0x00
1	DSP_PERIPH_LOCK	1 : Peripheral allocate to DSP TIPB 0 : Peripheral not allocate to DSP TIPB.	R	1
0	MPU_PERIPH_LOCK	1 : Allocate peripheral to MPU TIPB or peripheral already allocate to MPU. 0 : Peripheral not allocate to MPU TIPB	R/W	0

Table 5-30. APC_TSW_MPU_STA

Address Offset	0x02		
Physical Address	0xFFFF 8802	**Instance**	APC TIPB static switch
Description	Status of the APC TIPB static switch		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TSW_BOTH_LCK_ERR	TSW_ITPEND_ERR	TSW_DMAREQ_ERR	TSW_ERR_niRQ

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	R	0x000
3	TSW_BOTH_LCK_ERR	1: Error, both LOCK MPU and DSP are set. 0: No error occurs	R	0
2	TSW_ITPEND_ERR	1: A switch has occurred and a peripheral Interrupt was pending and not fully processed. 0: No DMA request pending during switch.	R	0
1	TSW_DMAREQ_ERR	1: A switch has occurred and a peripheral DMA request was pending and not fully processed. 0: No DMA request pending during switch.	R	0
0	TSW_ERR_niRQ	1: When written, clear bits [1:3]. 0: When read, NOR of bits [1:3].	R/W	1

Table 5-31. MCSI_TSW_MPU_CONF

Address Offset	0x00		
Physical Address	0xFFFF 8820	**Instance**	MCSI TIPB static switch
Description	To allocate MCSI peripheral to MPU		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DSP_PERIPH_LOCK	MPU_PERIPH_LOCK

Bits	Field Name	Description	Type	Reset
15:2	Reserved	Reserved	R	0x00
1	DSP_PERIPH_LOCK	1: Peripheral allocate to DSP TIPB 0: Peripheral not allocate to DSP TIPB.	R	1
0	MPU_PERIPH_LOCK	1 : Allocate peripheral to MPU TIPB or peripheral already allocate to MPU. 0: Peripheral not allocate to MPU TIPB	R/W	0

Table 5-32. MCSI_TSW_MPU_STA

Address Offset	0x02		
Physical Address	0xFFFF 8822	**Instance**	MCSI TIPB static switch
Description	Status of the MCSI TIPB static switch		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TSW_BOTH_LCK_ERR	TSW_ITPEND_ERR	TSW_DMAREQ_ERR	TSW_ERR_niRQ

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	R	0x000
3	TSW_BOTH_LCK_ERR	1: Error, both LOCK MPU and DSP are set. 0: No error occurs	R	0
2	TSW_ITPEND_ERR	1: A switch has occurred and a peripheral Interrupt was pending and not fully processed. 0: No DMA request pending during switch.	R	0
1	TSW_DMAREQ_ERR	1: A switch has occurred and a peripheral DMA request was pending and not fully processed. 0: No DMA request pending during switch.	R	0
0	TSW_ERR_niRQ	1: When written, clear bits [1:3]. 0: When read, NOR of bits [1:3].	R/W	1

Table 5-33. CPORT_TSW_MPU_CONF

Address Offset	0x00		
Physical Address	0xFFFF 8840	**Instance**	C-port TIPB static switch
Description	To allocate C-port registers peripheral to MPU		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DSP_PERIPH_LOCK	MPU_PERIPH_LOCK

Bits	Field Name	Description	Type	Reset
15:2	Reserved	Reserved	R	0x00
1	DSP_PERIPH_LOCK	1: Peripheral allocated to DSP TIP. 0: Peripheral not allocated to the DSP TIPB.	R	1
0	MPU_PERIPH_LOCK	1: Peripheral allocated to MPU TIPB or peripheral already allocated to MPU. 0: Peripheral not allocated to MPU TIPB. **Note:** When 1 is written, the ARM gains access to the CPORT and the DSP gives up control by writing 0 in the DSP_PERIPH_LOCK bit of the CPORT_TSW_DSP_CONF register.	R/W	0

Table 5-34. CPORT_TSW_MPU_STA

Address Offset	0x02		
Physical Address	0xFFFF 8842	**Instance**	C-port TIPB static switch
Description	Status of the C-port registers TIPB static switch		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TSW_BOTH_LCK_ERR	TSW_ITPEND_ERR	TSW_DMAREQ_ERR	TSW_ERR_niRQ

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	R	0x000
3	TSW_BOTH_LCK_ERR	1: Error, both LOCK MPU and DSP are set. 0: No error occurs	R	0
2	TSW_ITPEND_ERR	1: A switch has occurred and a peripheral Interrupt was pending and not fully processed. 0: No DMA request pending during switch.	R	0
1	TSW_DMAREQ_ERR	1: A switch has occurred and a peripheral DMA request was pending and not fully processed. 0: No DMA request pending during switch.	R	0
0	TSW_ERR_niRQ	1: When written, clear bits [1:3]. 0: When read, NOR of bits [1:3].	R/W	1

Table 5-35. UART_SSW_MPU_CONF

Address Offset	0x00		
Physical Address	0xFFFF 7280	**Instance**	UART TIPB/OCP static switch
Description	To allocate UART peripheral to MPU		
Type	R/W		
Write Latency	Not relevant		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														DSP_SWITCH	MPU_SWITCH

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Reserved	R	0x00
1	DSP_SWITCH	1: Peripheral allocate to DSP TIPB 0: Peripheral not allocate to DSP TIPB.	R	0
0	MPU_SWITCH	1: Allocate peripheral to MPU TIPB or peripheral already allocate to MPU. 0: Peripheral not allocate to MPU TIPB	R/W	1

Table 5-36. DRP_DSW _CONF

Address Offset	0x00		
Physical Address	0xFFFF 8000	**Instance**	DRP dynamic static switch
Description	To select priority for DRP dynamic switch.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															DSW_REG

Bits	Field Name	Description	Type	Reset
15:1	Reserved	Reserved	R	0x0000
0	DSW_REG	1: DSP TIPB bridge has priority. 0: MPU TIPB bridge has priority.	R/W	1

Table 5-37. APC_TSW_DSP_CONF

Address Offset	0x00		
Physical Address	0x7C80	**Instance**	APC TIPB static switch
Description	To allocate APC peripheral to DSP		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DSP_PERIPH_LOCK	MPU_PERIPH_LOCK

Bits	Field Name	Description	Type	Reset
15:3	Reserved	Reserved	R	0x0000
1	DSP_PERIPH_LOCK	1: Allocate peripheral to DSP TIPB or peripheral already allocate to DSP. 0: Peripheral not allocate to DSP TIPB.	R/W	1
0	MPU_PERIPH_LOCK	1: Peripheral allocates to MPU TIPB 0: Peripheral not allocate to MPU TIPB	R	0

Table 5-38. APC_TSW_DSP_STA

Address Offset	0x01		
Physical Address	0x7C81	**Instance**	APC TIPB static switch
Description	Status of the APC TIPB static switch		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TSW_BOTH_LCK_ERR	TSW_ITPEND_ERR	TSW_DMAREQ_ERR	TSW_ERR_niRQ

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	R	0x000
3	TSW_BOTH_LCK_ERR	1: Error, both LOCK MPU and DSP are set. 0: No error occurs	R	0
2	TSW_ITPEND_ERR	1: A switch has occurred and a peripheral Interrupt was pending and not fully processed. 0: No DMA request pending during switch.	R	0
1	TSW_DMAREQ_ERR	1: A switch has occurred and a peripheral DMA request was pending and not fully processed. 0: No DMA request pending during switch.	R	0
0	TSW_ERR_niRQ	1: When written, clear bits [1:3]. 0: When read, NOR of bits [1:3].	R/W	1

Table 5-39. MCSI_TSW_DSP_CONF

Address Offset	0x00		
Physical Address	0x7CA0	**Instance**	MCSI TIPB static switch
Description	To allocate MCSI peripheral to DSP		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DSP_PERIPH_LOCK	MPU_PERIPH_LOCK

Bits	Field Name	Description	Type	Reset
15:2	Reserved	Reserved	R	0x00
1	DSP_PERIPH_LOCK	1: Allocate peripheral to DSP TIPB or peripheral already allocate to DSP. 0: Peripheral not allocate to DSP TIPB.	R/W	1
0	MPU_PERIPH_LOCK	1: Peripheral allocates to MPU TIPB 0: Peripheral not allocate to MPU TIPB	R	0

Table 5-40. MCSI_TSW_DSP_STA

Address Offset	0x01		
Physical Address	0x7CA1	**Instance**	MCSI TIPB static switch
Description	Status of the MCSI TIPB static switch		

Table 5-40. MCSI_TSW_DSP_STA (continued)

Type	R/W
Write Latency	Not relevant

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TSW_BOTH_LCK_ERR	TSW_ITPEND_ERR	TSW_DMAREQ_ERR	TSW_ERR_niRQ

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	R	0x000
3	TSW_BOTH_LCK_ERR	1: Error, both LOCK MPU and DSP are set. 0: No error occurs	R	0
2	TSW_ITPEND_ERR	1: A switch has occurred and a peripheral Interrupt was pending and not fully processed. 0: No DMA request pending during switch	R	0
1	TSW_DMAREQ_ERR	1: A switch has occurred and a peripheral DMA request was pending and not fully processed. 0: No DMA request pending during switch.	R	0
0	TSW_ERR_niRQ	1: When written, clear bits [1:3]. 0: When read, NOR of bits [1:3].	R/W	1

Table 5-41. CPORT_TSW_DSP_CONF

Address Offset	0x00		
Physical Address	0x7CC0	**Instance**	CPORT TIPB static switch
Description	To allocate C_UnicodeEncodeError_PORT registers peripheral to DSP		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DSP_PERIPH_LOCK	MPU_PERIPH_LOCK

Bits	Field Name	Description	Type	Reset
15:2	Reserved	Reserved	R	0x00
1	DSP_PERIPH_LOCK	1: Peripheral allocated to DSP TIPB or peripheral already allocated to DSP. 0: Peripheral not allocated to DSP TIPB. **Note:** When 0 is written, the DSP gives up control.	R/W	1
0	MPU_PERIPH_LOCK	1: Peripheral allocated to MPU TIPB. 0: Peripheral not allocated to MPU TIPB.	R	0

Table 5-42. CPORT_TSW_DSP_STA

Address Offset	0x01		
Physical Address	0x7CC1	**Instance**	CPORT TIPB static switch
Description	Status of the C_UnicodeEncodeError_Port registers TIPB static switch		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TSW_BOTH_LCK_ERR	TSW_ITPEND_ERR	TSW_DMAREQ_ERR	TSW_ERR_niRQ

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	R	0x000
3	TSW_BOTH_LCK_ERR	1: Error, both LOCK MPU and DSP are set. 0: No error occurs.	R	0
2	TSW_ITPEND_ERR	1: A switch has occurred and a peripheral Interrupt was pending and not fully processed. 0: No DMA request pending during switch	R	0
1	TSW_DMAREQ_ERR	1: A switch has occurred and a peripheral DMA request was pending and not fully processed. 0: No DMA request pending during switch	R	0
0	TSW_ERR_niRQ	1: When written, clear bits [1:3]. 0: When read, NOR of bits [1:3].	R/W	1

Table 5-43. UART_SSW_DSP_CONF

Address Offset	0x00		
Physical Address	0x7900	**Instance**	UART TIPB/OCP static switch
Description	To allocate UART peripheral to DSP		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DSP_SWITCH	MPU_SWITCH

Bits	Field Name	Description	Type	Reset
15:2	Reserved	Reserved	R	0x00
1	DSP_SWITCH	1: Peripheral allocated to DSP TIPB 0: Peripheral not allocated to DSP TIPB.	R	0
0	MPU_SWITCH	1: Allocate peripheral to MPU TIPB or peripheral already allocated to MPU. 0: Peripheral not allocated to MPU TIPB	R/W	1

Table 5-44. TRANSFER_RATE

Address Offset	0x00		
Physical Address	0xF800	**Instance**	XIO/TIPB bridge
Description	Set_UnicodeEncodeError_up the duration of nSTROBE in DSP cycles for DSP access		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRANS_CYCLE_3				TRANS_CYCLE_2				TRANS_CYCLE_1				TRANS_CYCLE_0			

Bits	Field Name	Description	Type	Reset
15:12	TRANS_CYCLE_3	Defines the duration, in half period DSP cycles, of nSTROBE[3] The transfer rate is governed by programming the duration of nSTROBE[3]. • The programmable range is from full speed (0) - which is a duration of half a DSP clock cycle - down to slowest speed (5) - a duration of 16 DSP clock cycles	R/W	0xF
11:8	TRANS_CYCLE_2	Defines the duration, in half period DSP cycles, of nSTROBE[2] The transfer rate is governed by programming the duration of nSTROBE[2]. • The programmable range is from full speed (0) - which is a duration of half a DSP clock cycle - down to slowest speed (5) - a duration of 16 DSP clock cycles	R/W	0xF
7:4	TRANS_CYCLE_1	Defines the duration, in half period DSP cycles, of nSTROBE[1] The transfer rate is governed by programming the duration of nSTROBE[1]. • The programmable range is from full speed (0) - which is a duration of half a DSP clock cycle - down to slowest speed (5) - a duration of 16 DSP clock cycles	R/W	0xF
3:0	TRANS_CYCLE_0	Defines the duration, in half period DSP cycles, of nSTROBE[0] The transfer rate is governed by programming the duration of nSTROBE[0]. • The programmable range is from full speed (0) - which is a duration of half a DSP clock cycle - down to slowest speed (5) - a duration of 16 DSP clock cycles	R/W	0xF

Table 5-45. BRIDGE_CNTL

Address Offset	0x01		
Physical Address	0xF801	**Instance**	XIO/TIPB bridge
Description	To set the time-out value for XIO/TIPB access		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						NSUPV	TIMEOUT_EN	TIMEOUT							

Bits	Field Name	Description	Type	Reset
15:10	Reserved	Reserved	R	0x00
9	NSUPV	RHEA bus supervisor flag. This flag is used to control the access to privileged peripheral registers. • If NSUPV is set to 0, access can be done on privileged registers. • If NSUPV is set to 1, access is forbidden.	R/W	0
8	TIMEOUT_EN	Enable/Disable TIPB TIMEOUT watchdog: 1: Enable the TIMEOUT feature and NMI interrupt. 0: Disable the TIMEOUT feature	R/W	0
7:0	TIMEOUT	TIPB access time out. Limits, by counting nSTROBE cycles, the maximum time a peripheral can stall the processor. When starting a transaction on the TIPB, the time out counter is reset at zero. If the current cycle is not finished when the counter reaches a count of TIMEOUT + 1, the transaction is aborted by sending nXABORT to the peripheral and an NMI interrupt if TIMEOUT_EN is set.	R/W	0x7F

Power, Reset, and Clock Management

This chapter discusses power, reset, and clock management for the LOCOSTO device.

Topic **Page**

6.1 Introduction

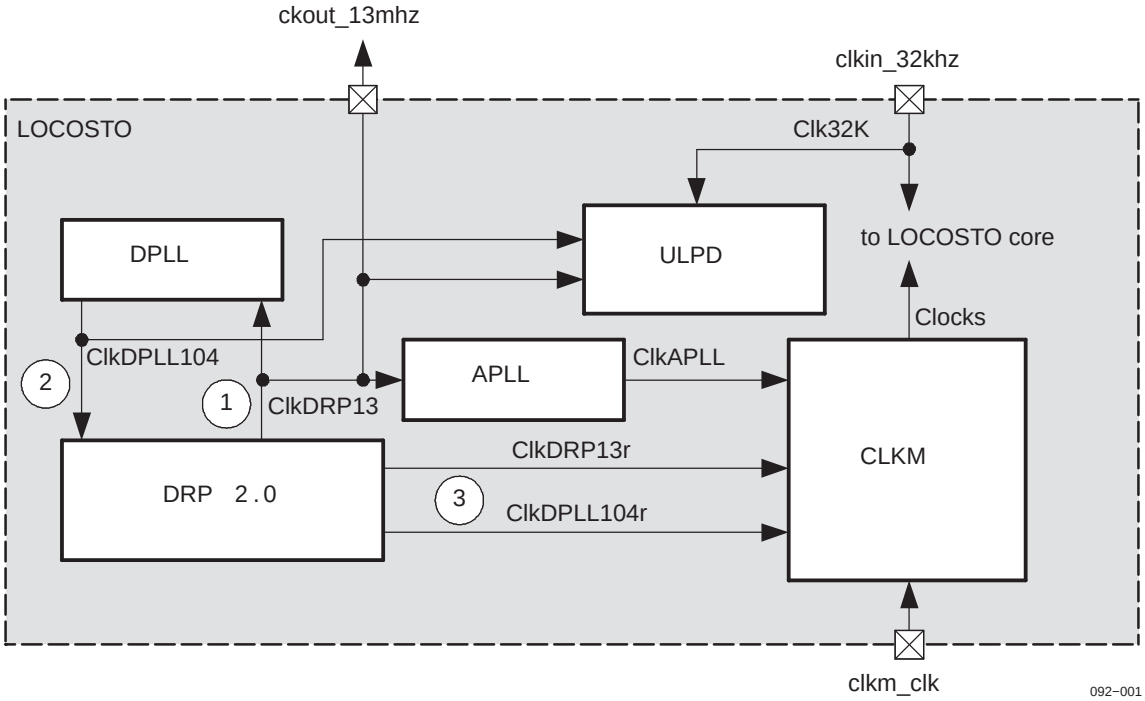
This chapter describes clock generation, presents an overview of clock distribution through the LOCOSTO core, and includes an exhaustive inventory of clock configurations. The power management section emphasizes power-saving configurations. This chapter also describes the reset signals inside the LOCOSTO device and how they are generated.

6.2 Clock

6.2.1 Overview

Except for the 32-kHz source (clkin_32khz) and the external clock (clkm_clk), the clocks dispatched through the LOCOSTO core are generated from one 13-MHz source (ClkDRP13) provided by the digital radio processor module, version 2.0 (DRP2.0) (see Figure 6-1).

Figure 6-1. Clock Generation Overview



The clock manager (CLKM) module controls these clocks and manages the power-saving modes with the help of the ultralow power-down (ULPD) module. The CLKM module receives four different clock frequencies before dispatching the clocks through the core:

- External clock (clkm_clk) up to 40 MHz
- Regenerated 13-MHz clock (clkDRP13r) directly fed by the DRP2.0 module
- Regenerated digital phase-locked loops (DPLLs) clock (ClkDPLL104r), typically 104 MHz. This clock is synthesized by the DPLL from the digital radio processor (DRP) 13-MHz clock (ClkDRP13) and resynchronized by the DRP2.0 (ClkDPLL104r). (The circled numbers in Figure 6-1 highlight this sequence.)
- Analog phase-locked loop (APLL) clock, typically 48 MHz, generated from the non-regenerated ClkDRP13 clock. The ClkAPLL clock is provided to the universal serial bus (USB), camera, universal asynchronous receiver/ transmitter (UART), and multichannel serial interface (MCSI) modules.

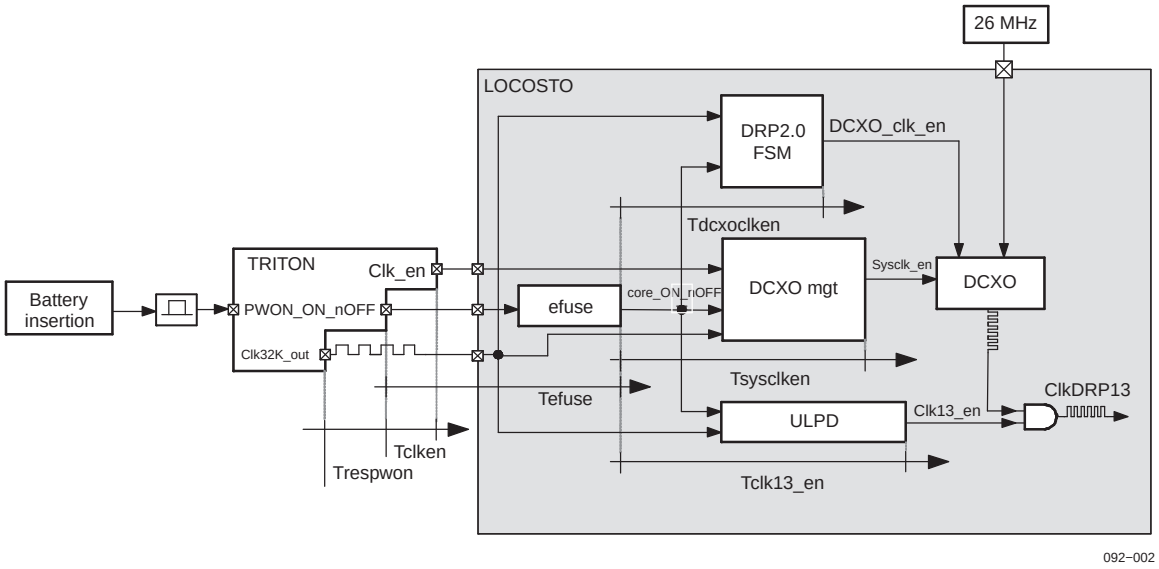
Clock regeneration avoids digital noise on the digital baseband (DBB) chip. The DRP integrates regenerating circuitry to resynchronize the main clocks of the DBB part on the DRP local oscillator only when the latter is in tracking mode. Otherwise, the ClkDRP13r and ClkDPLL104r clocks are equivalent to the ClkDRP13 clock and to the ClkDPLL104 clock, respectively.

The ULPD module also receives the ClkDPLL104 and the ClkDRP13 clocks (non-regenerated clocks) for global system for mobile communications (GSM) and gauging features.

6.2.2 Power-Up

This section details the ClkDRP13 generation from the battery insertion event. Figure 6-2 shows the block and timing diagram of the power-up sequence.

Figure 6-2. Power-Up: 13-MHz Clock-Generation Sequence



The LOCOSTO device power-up is managed by the TWL3031 device in cooperation with the LOCOSTO device power-management finite state-machine (FSM). When the main battery is initially plugged in, a small portion of the system is supplied and clocked after the 32-kHz oscillator start-up phase.

When the ON/OFF button is pressed, the TWL3031 power-management FSM accepts the start condition as valid only after a stable low-level period of the PWON signal. Once a valid switch-on condition is recognized, the TWL3031 FSM provides the correct sequence for power supplies rising up.

When the VRIO power domain is stable, the FSM enables the 32-kHz clock to the external word; after the Trespwon time interval, the FSM releases the system reset, driving the ON_nOFF signal to a high level. As a final action, the TWL3031 FSM enables the system clock generation, thus driving the Clk_en signal to a high logical level at Tclken time after reset release (see Table 6-1).

On the LOCOSTO device side, after the ON_nOFF signal is driven high by the TWL3031 device, the Efuse module releases the core_ON_nOFF signal after a Tefuse delay (see Table 6-1).

The following three state-machines then start in parallel to activate the 13-MHz clock (ClkDRP13):

- The DCXO management module generates the Sysclk_en signal to the DRP DCXO module following a Tsysclken delay.
- The DRP2.0 FSM activates the DCXO clock with the DCXO_Clk_en following a Tdcxoclken delay.
- Meanwhile, the ULPD FSM enables the 13-MHz clock by releasing the Clk13_en signal following a Tclk13en delay.

Note: Because the three FSMs run in parallel, the values of the different delays must satisfy a consistent enable of the system clock; that is, the DCXO_Clk_en signal must be released before the Sysclk_en signal, which must be released before the Clk13_en signal.

Table 6-1. Power-Up Delay Settings

Delay	Default Value	Configuration
Trespwon	7.1411 ms	See TWL3031 documentation
Tclken	19 x 32 kHz clock periods (0.5798 ms)	See TWL3031 documentation
Tefuse	744 x 32 kHz clock periods (22.705 ms)	None
Tdcxoclken	183 x 32 kHz clock periods (5.585 ms)	None
Tsysclken	510 x 32 kHz clock periods (15.564 ms)	Reset value
Tclk13en	8253 x 32 kHz clock periods (251.86 ms)	Reset value

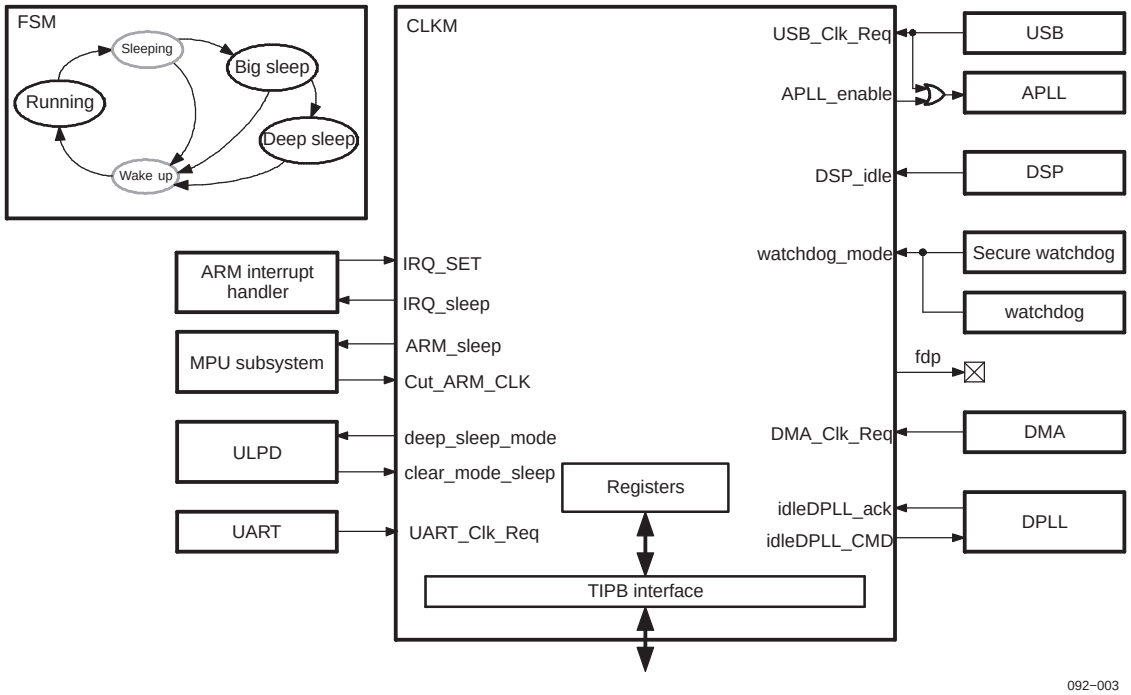
6.2.3 Power-Saving Modes

To reduce dynamic power consumption, software can program the CLKM module to selectively cut off clocks to various parts of the device, either by manually cutting off clocks or by entering power-saving modes (also called sleep modes). This section describes the different sleep modes. Section 6.2.5.2, *ARM104MHz Domain*, through Section 6.2.5.19, *ckout_13mhz Output Clock*, present the configurations of the registers used to manually cut the clocks.

Figure 6-3 shows the environment of the CLKM module, including the state machine of the sleep modes:

- Running mode: All clocks, except those that are manually cut off, are enabled.
- Sleeping mode: Transition mode—the CLKM module waits for acknowledgement from the MPU subsystem.
- Big sleep mode: The ARM clock (at a minimum) and selectively other clocks are cut off based on activity and configuration. In this mode, the ClkDRP13 and the ClkDRP13r clocks are not stopped and continue to run.
- Deep sleep: Only the 32-kHz clock is running in the device.
- Wake-up mode: A transition mode in which the clocks are restarted.

Figure 6-3. CLKM Module Environment



6.2.3.1 Sleeping Mode

Clearing the CNTL_ARM_CLK[0] BIG_SLEEP bit initiates the ARM clock cut and moves LOCOSTO to the sleeping mode. The CLKM module sends a sleep request to the ARM subsystem (ARM_sleep) and waits for an acknowledgement (Cut_ARM_CLK).

6.2.3.2 Big Sleep Mode

After receiving MPU acknowledgement in the sleeping mode, the state machine moves to big sleep mode, and the ARM clock is cut off. Other clocks are also cut off, depending on the clock control configuration (see Section 6.2.5.2 through Section 6.2.5.19).

Note: The BIG_SLEEP bit is automatically set to 1 when the FSM moves to wakeup mode.

6.2.3.3 Deep Sleep Mode

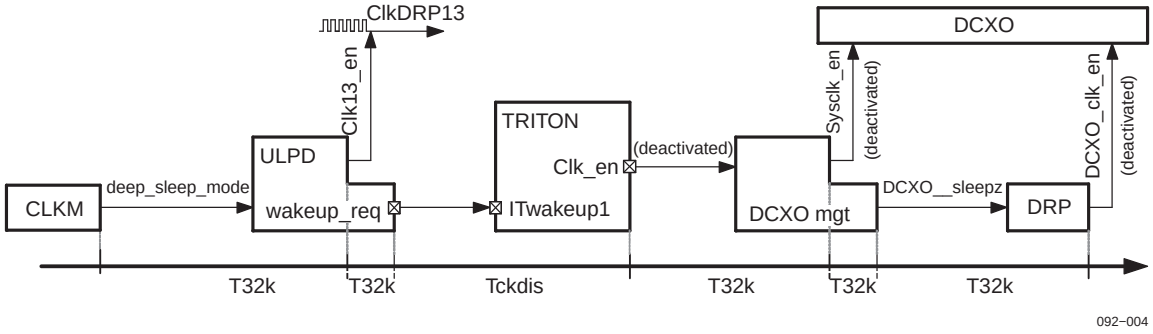
6.2.3.3.1 Enter Deep Sleep Mode

The ULPD and CLKM modules work together to put the LOCOSTO device in deep sleep mode. Before initiating the deep sleep sequence, the DPLL module must be in the idle mode (see Section 6.2.5.1). The CLKM module initiates the deep sleep request by clearing the CNTL_ARM_CLK[12] DEEP_SLEEP bit.

Note: The BIG_SLEEP bit does not have to be set before the DEEP_SLEEP bit is set.

Because the ClkDRP13 clock is cut off after the sequence of actions shown in Figure 6-4, only the 32-kHz clock is running in deep sleep mode.

Figure 6-4. 13-MHz Clock Shut-Down Sequence



To start the deep sleep procedure, the CLKM module sends a request to the ULPD module (deep_sleep_mode). After one 32-kHz clock period, the ULPD module disables the 13-MHz clock and, one more 32-kHz clock period later, the wakeup_req signal is driven to a low level.

At this time, the TWL3031 device deactivates the Clk_en signal following a Tckdis delay. The DCXO management module then deactivates the Sysck_en signal after one 32 kHz-clock period. Finally, one 32-kHz clock period later, the DRP2.0 FSM deactivates the DCXO module using the DCXO_Clk_en signal.

Table 6-2 shows the default value of each delay and the associated configuration.

Table 6-2. Deep Sleep Delay Settings

Delay	Default Value	Configuration
T32k	1 x 32 kHz clock period (0.0305 ms)	None
Tckdis	19 x 32 kHz clock period (0.5798 ms)	See TWL3031 documentation

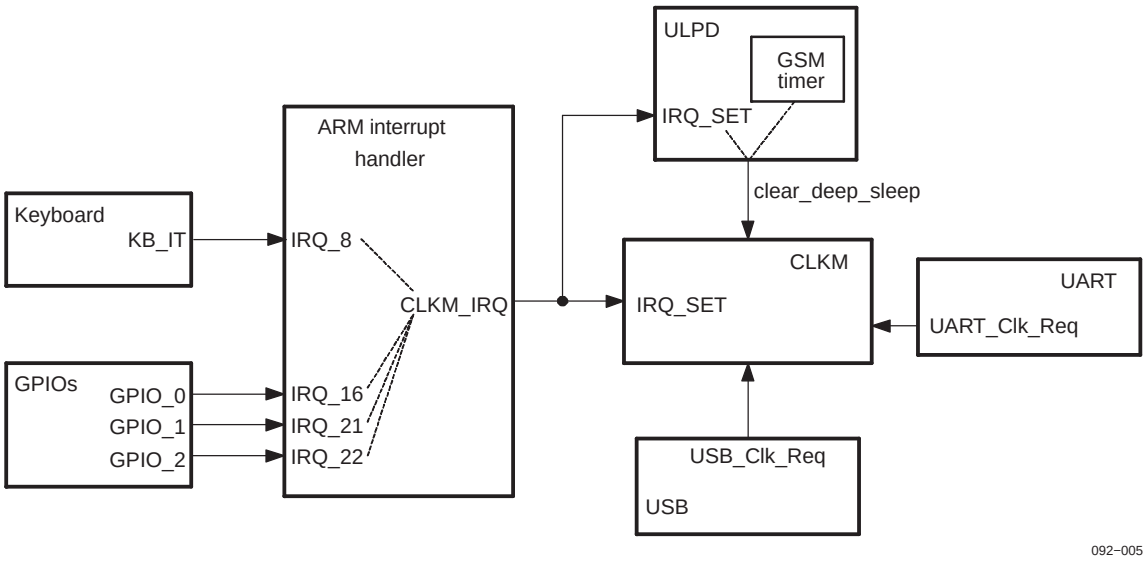
6.2.3.3.2 Exit Deep Sleep Mode

The following conditions cause the LOCOSTO device to exit deep sleep mode (see Figure 6-5):

- The programmed number of GSM frames elapses and the device must wake up for radio frequency (RF) activity.
- The USB and UART modules initiate a wake-up request based on hardware and software events on their ports.
- An unmasked interrupt is generated by the keyboard controller (see Chapter 31, *Keyboard Controller)* or by the GPIO_0, GPIO_1, and GPIO_2 (see Chapter 27, *General-Purpose Interface*).

When one of these events occurs before the ULPD module receives the deep_sleep_mode request, the LOCOSTO core enters wake-up mode, thus allowing the DEEP_SLEEP bit to be set to 0.

Figure 6-5. Deep Sleep Wake-Up Initiators



For more information about the deep-sleep mode, see the *DEEP SLEEP Application Note* (APN 210, LOCOSTO Program) available through your TI representative.

6.2.3.4 Wake-Up Mode

The BIG_SLEEP bit and the DEEP_SLEEP bit are automatically set to 1, except when the LOCOSTO core is interrupted while going into deep sleep mode. When the state-machine comes from deep sleep mode, all clocks restart, including the 13-MHz clock (ClkDRP13).

The 13-MHz clock has the following restart sequence:

1. By the time the wake-up phase is initiated (see Figure 6-6), the ULPD module drives the wakeup_req signal to a high level.
2. The TWL3031 device activates the Clk_en signal following a T'cken delay.
3. The DCXO management module starts its FSM machine, activating both the DCXO_sleepz and the Sysclk_en signals.
4. The DRP2.0 module enables the DCXO module after receiving the DCXO_sleepz signal.
5. Finally, the ULPD module enables the 13-MHz clock (ClkDRP13) using the Clk13_en signal.

Note: Like the power-up phase, three FSMs run in parallel. The values of the different delays must satisfy a consistent enabling of the system clock; that is, the DCXO_Clk_en signal must be released before the Sysclk_en signal, which must be released before the Clk13_en signal (see Table 6-3 for the timings).

Figure 6-6. 13-MHz Clock Generation Wake-Up Sequence

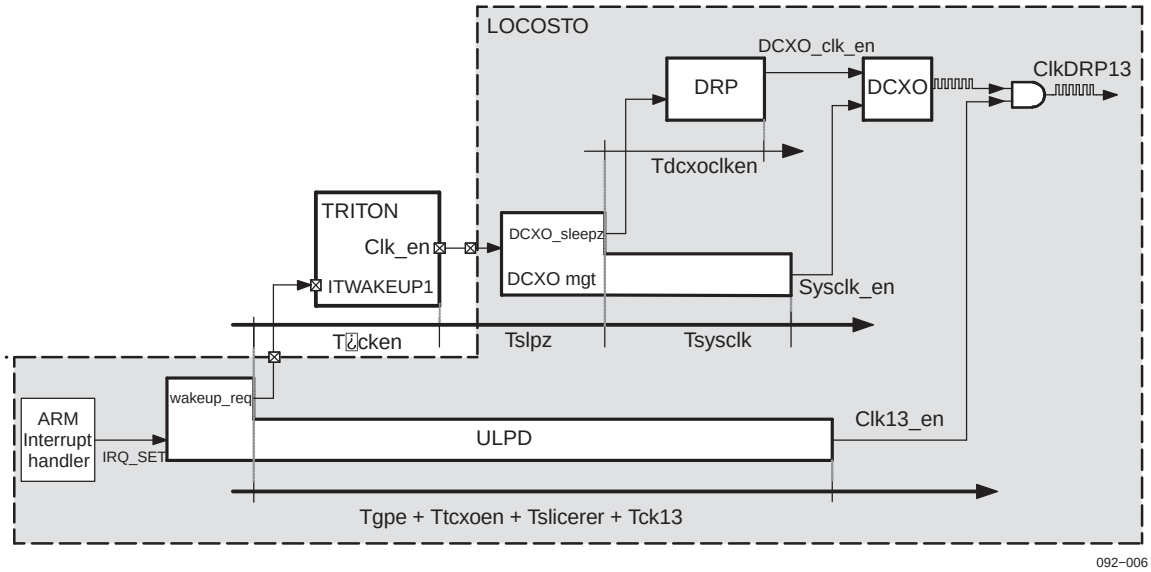


Table 6-3. Wake-Up Delay Settings

Delay	Default Value	Configuration
T'cken	88 x 32 kHz clock periods	See TWL3031 documentation
Tdcxoclken	183 x 32 kHz clock periods	None
Tslpz	255 x 32 kHz clock periods	ULPD_DCXO_SETUP_SLEEPN register
Tsysclk	255 x 32 kHz clock periods	ULPD_DCXO_SETUP_SYSCLKEN register
Tgpe	0 x 32 kHz clock periods	SETUP_RF_REG register
Ttcxoen	4095 x 32 kHz clock periods	SETUP_VTCXO_REG register
Tsliceren	4095 x 32 kHz clock periods	SETUP_SLICER_REG register
Tclk13	63 x 32 kHz clock periods	SETUP_CLOCK_13MHZ_REG register

6.2.3.5 NOR Flash Sleep Mode

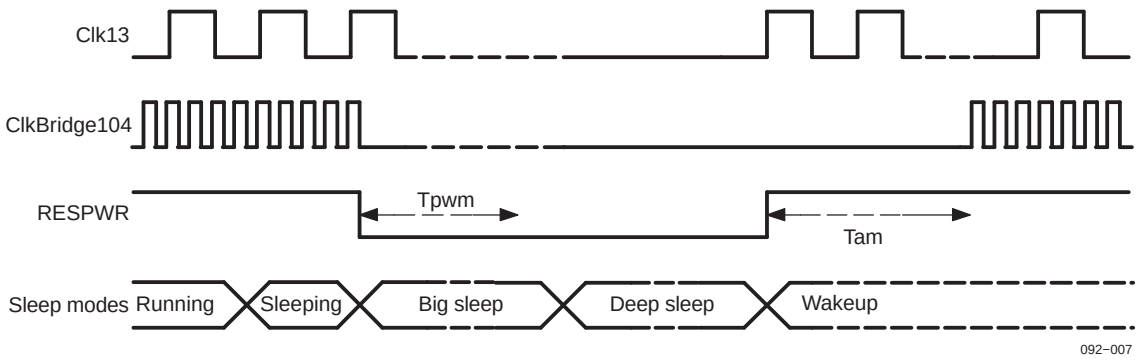
NOR Flash memories support a sleep mode to conserve power during periods of inactivity. This sleep mode is achieved by indicating to the memory (using the fdp signal) that no access will be made in the immediate future and that the memory can go to a low-power consumption state. The CNTL_CLK[15] FLASH_CNTL bit controls this feature.

Figure 6-7 shows the case in which memory is allowed to go into sleep mode (that is, the FLASH_CNTL bit is set to 0). In this case, even if an interrupt tries to wake up the system, the memory is ensured to be maintained for a minimum duration of time (Tpwm) equaling 16 clock periods of the 13-MHz clock

On exiting the sleep mode, the first access to the memory should be after a Tam time defined by the CNTL_ARM_CLK[11:8] DEEP_POWER field; the ClkBridge104 clock restarts after this delay. (For more information about the ClkBridge104 clock, see Section 6.2.4 and Section 6.2.5.5.)

Note: If the FLASH_CNTL bit is set to 0, neither of the Tpwm and Tam delays are considered.

Figure 6-7. NOR Flash Sleep Mode



6.2.4 Clock Distribution

Figure 6-8 shows the functional clock domains of the LOCOSTO device while Figure 6-9 shows the interface clock domains. Each domain is different based on its clock frequency or its control on the clock. The difference between the interface clocks and the functional clocks is that the first ones are used to configure the module through registers and the second ones are provided to the state machines of the module.

Figure 6-8. Functional Clock Diagram

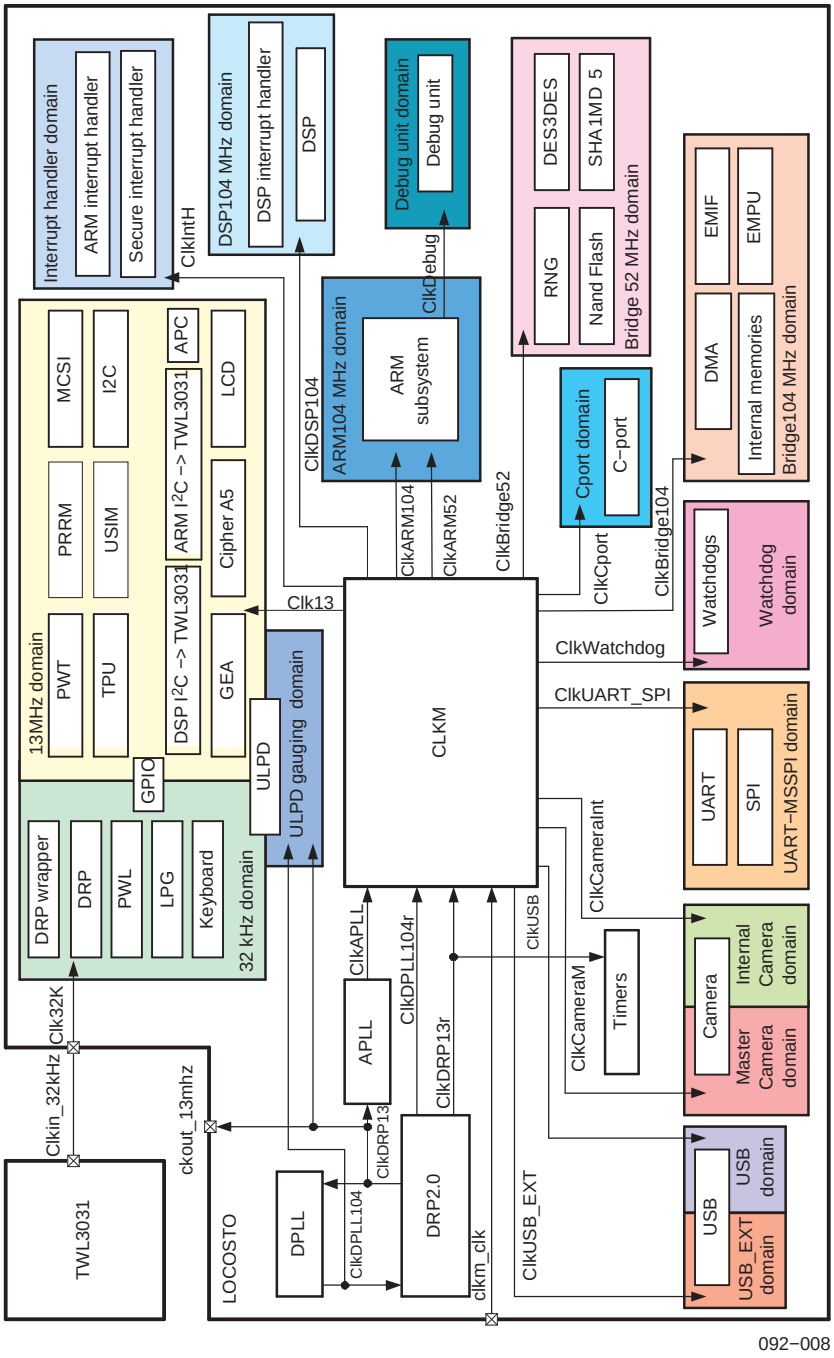
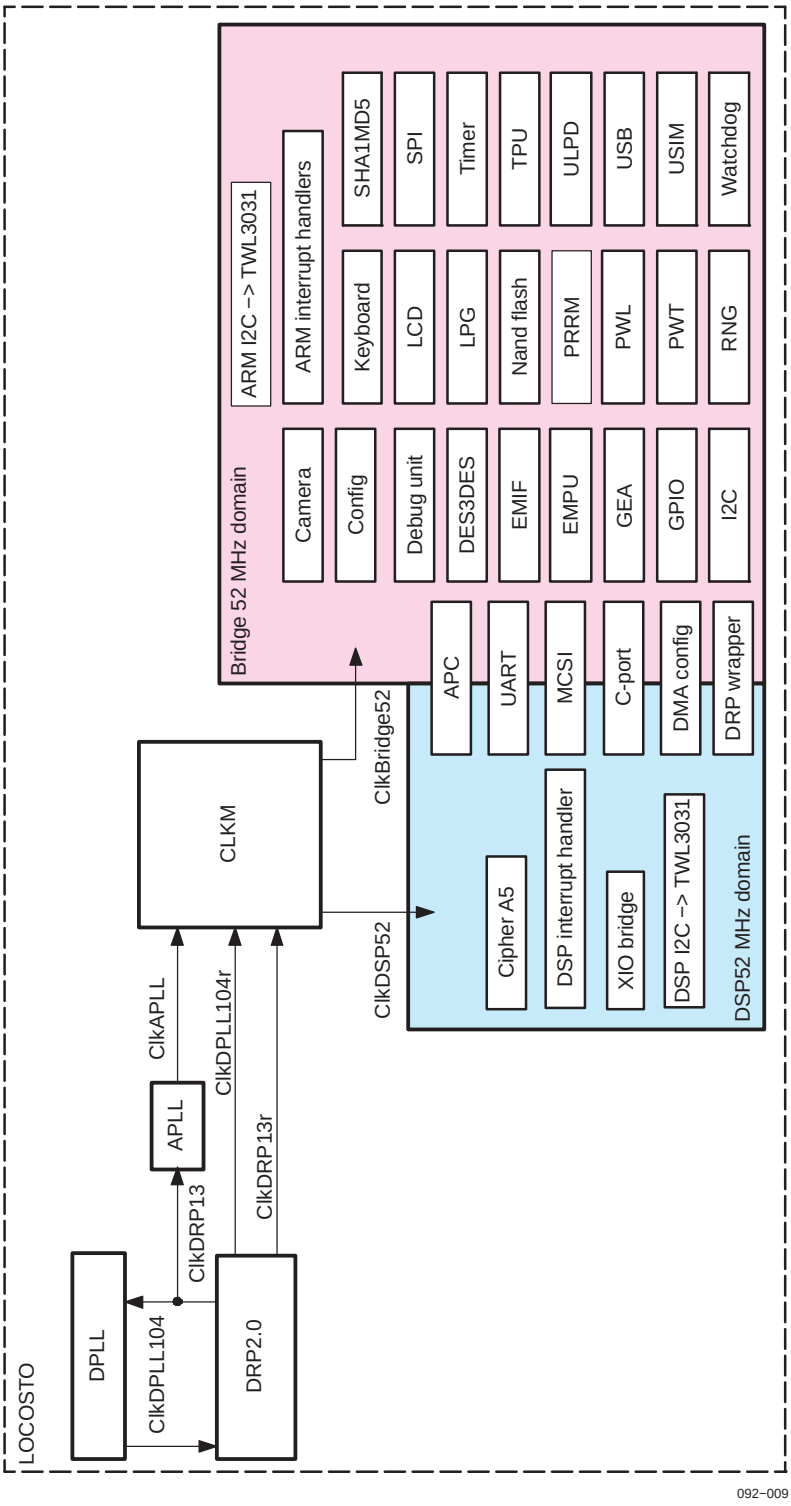


Figure 6-9. Interface Clock Diagram



6.2.5 Clock Controls

Both the DPLL module and the CLKM module control the LOCOSTO clocks. The DPLL module synthesizes the ClkDPLL104 clock, the frequency being configured through the control registers. The CLKM module disables/enables the clocks while dispatching them.

Notes:

- The automatic on/off switch in deep sleep mode is the only control of the ClkDRP13r and ClkDRP13 clocks.
- The CLKM module controls the ClkAPLL clock.

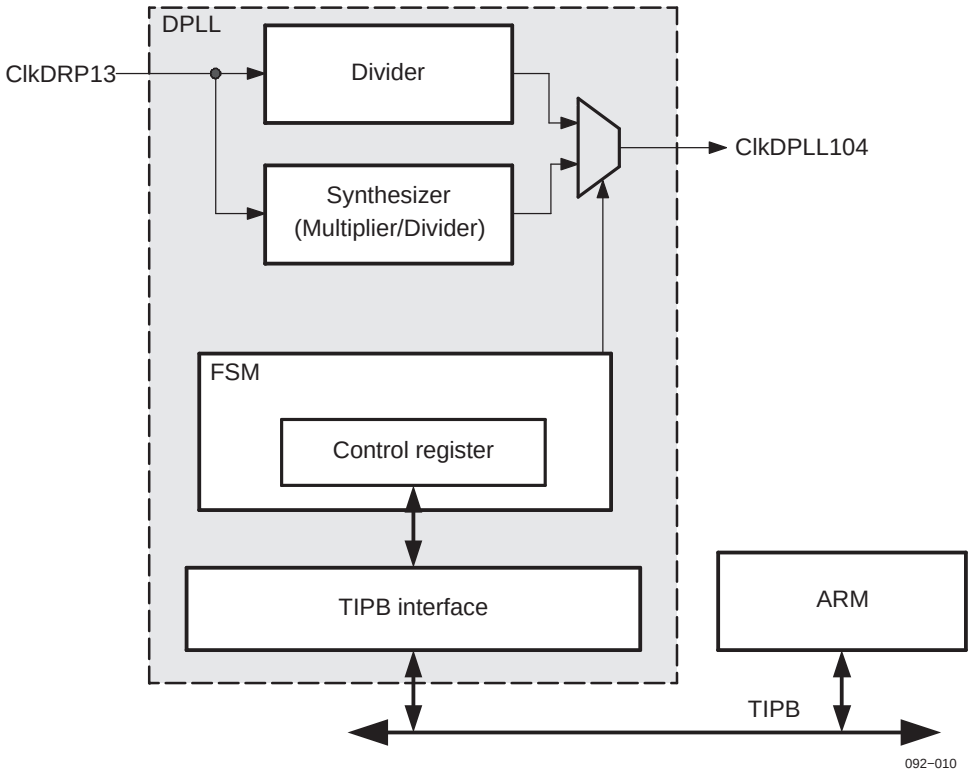
Section 6.2.5.1 describes the DPLL module. Section 6.2.5.2 through Section 6.2.5.18 provide an exhaustive list of the clock frequencies for all clock domains.

6.2.5.1 DPLL

The LOCOSTO device uses the DPLL module as either a clock divider or a clock synthesizer (see Figure 6-10). The DPLL FSM, which makes the selection, is composed of three modes:

- Lock mode (The DPLL provides a synthesized clock.)
- Bypass mode (The DPLL module provides a divided clock.)
- Idle mode (The DPLL does not provide a clock.)

Figure 6-10. DPLL Overview



6.2.5.1.1 *Lock Mode*

The DPLL module provides a synthesized output frequency that is locked to the input reference. The lock mode is entered when the DPLL_CNTL_REG[4] PLL_ENABLE bit is set and the locking sequence is completed. In the lock mode, the output clock ClkDPLL104 corresponds to a synthesized clock frequency as defined by the following:

Case 1: PLL_MULT[4:0] = 0x0 or PLL_MULT[4:0] = 0x1

$$\text{ClkDPLL104} = \frac{1}{\text{PLL\_DIV}[1:0] + 1} \times \text{CLKDRP13} \quad (6\text{-}1)$$

Case 2: 1< PLL_MULT[4:0] ≤ 31

$$\text{ClkDPLL104} = \frac{\text{PLL\_MULT}[4:0]}{\text{PLL\_DIV}[1:0] + 1} \times \text{CLKDRP13} \quad (6\text{-}2)$$

Note: PLL_MULT and PLL_DIV are fields of the DPLL_CNTL_REG register.

Each time the DPLL control register is written to, it switches automatically to the bypass mode. Depending on the new control register content, the DPLL either initiates a new lock sequence or remains in the bypass mode.

Recommendations:

- To determine if the DPLL is locked, scan the DPLL_CNTL_REG[0] LOCK bit.
- To achieve the typical clock frequency of 104 MHz, set the PLL_MULT field to 0x10 and the PLL_DIV field to 0x1.

6.2.5.1.2 Bypass Mode

The bypass mode can be used to save power because the DPLL is disabled (the synthesizer is shut down). The bypass mode also provides an output clock while the DPLL circuitry is locking.

In the bypass mode, the output clock ClkDPLL104 is equal to the ClkDRP13 clock divided by 1, 2, or 4, depending on the value of the CNTL_REG[3:2]BYPASS_ DIV field (see Section 6.5.2.4).

6.2.5.1.3 Idle Mode

When the CNTL_CLK[3] DPLL_DIS bit is set to 1, the CLKM module requests the DPLL to enter the idle mode. Before the idle mode can be entered, the following conditions must be met:

- The DSP must be in the idle3 mode (see Chapter 4, *DSP Subsystem*).
- The bridge104-MHz domain must be disabled (see Section 6.2.5.5).
- The DMA, NAND Flash IF, EMIF, and UART have no pending operations.

These conditions stop the ClkDPLL104 clock. When the DPLL is in the lock mode before the idle signal is asserted, the lock mode is reacquired on deassertion of the signal.

6.2.5.2 ARM104MHz Domain

The ARM104MHz domain consists of the ARM subsystem module.

The ClkARM52 clock is generated from the ClkARM104 clock, only a divider separates the 2 clocks.

Table 6-4 describes the ClkARM104 clock configurations.

Table 6-4. ClkARM104 Clock Configurations

ClkARM104	Frequency	Parameters
		Source Clock
Running	13 MHz	ClkDRP13r
	clkm_clk up to 40 MHz	clkm_clk
	ClkDPLL104r 104 MHz (typ)	ClkDPLL104r
Big Sleep	0 MHz	Don't care
Deep Sleep	0 MHz	Don't care

Table 6-5 describes the ClkARM104 clock parameters.

Table 6-5. ClkARM104 Clock Parameters

Parameter	Description	Value	Register Configuration
Source clock	Three root clocks can be selected: ClkDPLL104 ClkDRP13 d ClkDPLL104r, ClkDRP13r, and clkm_clk	ClkDPLL104r	CNTL_ARM_CLK[1]=0x0
		ClkDRP13r	CNTL_ARM_CLK[1]=0x1 and CNTL_ARM_CLK[2]=0x0
		clkm_clk	CNTL_ARM_CLK[1]=0x1 and CNTL_ARM_CLK[2]=0x1

6.2.5.3 DSP104Mhz Domain

The DSP104MHz domain consists of the following modules:

- DSP (functional clock)
- DSP interrupt handler (functional clock)

Table 6-6 describes the ClkDSP104 clock configurations.

Table 6-6. ClkDSP104 Clock Configurations

ClkDSP104	Frequency	Parameters	
		DSP	DPLL
Running	0 MHz	Idle3	Don't care
	ClkDPLL104r 104 MHz (typ)	Not idle3	Don't care
Big sleep	0 MHz	Idle3	Don't care
	ClkDPLL104r 104 MHz (typ)	Not idle3	Not idle
Deep sleep	0 MHz	Idle3	Idle

Table 6-7 describes the ClkDSP104 clock parameters.

Table 6-7. ClkDSP104 Clock Parameters

	Description	Value	Register Configuration
DSP	DSP can be put in idle3 mode to cut its clock.	Idle3	See Chapter 4, *DSP Subsystem*
		Not idle3	
DPLL	The DPLL module in idle mode cuts off the ClkDPLL104r clock.	Idle	CNTL_CLK[3]=0x1
		Not idle	CNTL_CLK[3]=0x0

6.2.5.4 DSP52MHz Domain

The DSP52MHz domain consists of the following modules:

- CipherA5 (interface clock)
- DSP interrupt handler (interface clock)
- XIO Bridge (interface clock)
- DSP I2C _UnicodeEncodeError_> TWL3031 (interface clock)
- APC (interface clock)
- UART (interface clock)
- MCSI (interface clock)
- C_UnicodeEncodeError_Port (interface clock)
- DMA config (interface clock)
- DRP wrapper (interface clock)

6.2.5.5 bridge104MHz Domain

The bridge104MHz domain consists of the following modules:

- DMA (functional clock)
- EMIF (functional clock)
- EMPU (functional clock)
- Internal memories (functional clock)

Table 6-8 describes the ClkBridge104 clock configurations.

Table 6-8. ClkBridge104 Clock Configurations

ClkBridge104	Frequency	Parameters			
		Source clock	Bridge	DPLL	Request ?
Running	= ClkARM104 (Running)	See ClkARM104	Don't care	Don't care	Don't care
Big sleep	0 MHz	Don't care	Disable	Don't care	No
	13 MHz	ClkDPLL104r	Disable	Idle	Yes
		ClkDRP13r	Running	Don't care	Don't care
		ClkDRP13r	Disable	Don't care	Yes
	clkm_clk up to 40 MHz	clkm_clk	Running	Don't care	Don't care
		clkm_clk	Disable	Don't care	Yes
	ClkDPLL104r 104 MHz typ)	ClkDPLL104r	Running	Not idle	Don't care
		ClkDPLL104r	Disable	Not idle	Yes
Deep Sleep	0MHz	Don't care	Don't care	Idle	Don't care

Table 6-9 describes the ClkBridge104 clock parameters.

Table 6-9. ClkBridge104 Clock Parameters

	Description	Value	Register Configuration
Source clock	Three source clocks can be l t d selected.	ClkDPLL104r	CNTL_ARM_CLK[1]=0x0
		ClkDRP13r	CNTL_ARM_CLK[1]=0x1 and CNTL_ARM_CLK[2]=0x0
		clkm_clk	CNTL_ARM_CLK[1]=0x1 and CNTL_ARM_CLK[2]=0x1
Bridge	The bridge can be disabled in big sleep mode if no request occurs.	Disable	CNTL_CLK[1]=0x1
		Running	CNTL_CLK[1]=0x0
DPLL	The DPLL module in idle mode cuts off the ClkDPLL104r clock.	Idle	CNTL_CLK[3]=0x1
		Not idle	CNTL_CLK[3]=0x0
Request?	Requests can be generated either by DMA or NAND flash IF or EMIF or USB or UART.	No/Yes	None

The ClkBridge104 clock can be cut off when entering big sleep mode and is automatically switched on when a DMA request is detected.

When the bridge104MHz domain is in the big sleep mode with an active DPLL module, the DMA clock request initiates a transfer at the bridge104MHz domain frequency rate. Otherwise, a switching mechanism allows the ClkBridge104 clock to be driven directly from the ClkDRP13r clock when the DPLL module is in idle mode.

If the bridge104MHz domain enters the big sleep mode during an ongoing DMA transfer and with a DPLL idle request, the current DMA transfer is completed using the 104-MHz clock. The ClkBridge104 clock is effectively switched to 13 MHz at the end of the transfer.

When the bridge104MHz domain goes into wake-up mode during an ongoing DMA transfer, the DMA transfer rate switches from 13 MHz to the DPLL source as soon as the switch mechanism resynchronizes to the ClkDPLL104 clock.

6.2.5.6 bridge52MHz Domain

The bridge52MHz domain consists of the following modules:

- RNG (functional clock)
- DES3DES (functional clock)
- Nand Flash interface (functional clock)
- SHA1MD5 (functional clock)
- APC (interface clock)
- UART (interface clock)
- MCSI (interface clock)
- C_UnicodeEncodeError_PORT (interface clock)
- DMA config (interface clock)
- DRP Wrapper (interface clock)
- Camera (interface clock)
- Config (interface clock)
- Debug Unit (interface clock)
- DES3DES (interface clock)
- EMIF (interface clock)
- EMPU (interface clock)
- GEA (interface clock)
- GPIO (interface clock)
- I2C (interface clock)
- ARM I^2C _UnicodeEncodeError_> TWL3031 (interface clock)
- ARM interrupt handler (interface clock)
- Keyboard (interface clock)
- LCD (interface clock)
- LPG (interface clock)
- Nand Flash (interface clock)
- PRRM (interface clock)
- PWL (interface clock)
- PWT (interface clock)
- RNG (interface clock)
- SHA1MD5 (interface clock)
- SPI (interface clock)
- Timer (interface clock)
- TPU (interface clock)
- ULPD (interface clock)
- USB (interface clock)
- USIM (interface clock)
- Watchdog (interface clock)

Table 6-10 describes the ClkBridge52 clock configurations.

Table 6-10. ClkBridge52 Clock Configurations

ClkBridge52	Frequency	Conditions
Running	= ClkBridge104/2 (Running)	See ClkBridge104
Big sleep	= ClkBridge104/2 (Big sleep)	See ClkBridge104
Deep sleep	0 MHz	None

6.2.5.7 UART-MSSPI Domain

The UART-MSSPI domain consists of the following modules:

- UART (functional clock)
- MSSPI (functional clock)

Table 6-11 describes the ClkUART_SPI clock configurations.

Table 6-11. ClkUART_SPI Clock Configurations

ClkDSP104	Frequency	Parameters		
		Source Clock	APLL	DPLL
Running	0 MHz	No clock selected	Don't care	Don't care
		ClkAPLL	Disable	Don't care
	13 MHz	ClkDRP13r	Don't care	Don't care
	48 MHz	ClkAPLL	Enable	Don't care
	ClkDPLL104r/2	Clk52	Don't care	Don't care
Big sleep	0 MHz	No clock selected	Don't care	Don't care
		ClkAPLL	Disable	Don't care
		Clk52	Clk52	Idle
	13 MHz	ClkDRP13r	Don't care	Don't care
	48 MHz	ClkAPLL	Enable	Don't care
	ClkDPLL104r/2	Clk52	Don't care	Not idle
Deep sleep	0 MHz	Don't care	Don't care	Idle

Table 6-12 describes the ClkUART_SPI clock parameters.

Table 6-12. ClkUART_SPI Clock Parameters

	Description	Value	Register Configuration
Source clock	Four source clocks can be selected.	No clock	CNTL_CLK_PROG_FREE_RUNNING[1:0]=0x3
		ClkAPLL	CNTL_CLK_PROG_FREE_RUNNING[1:0]=0x1
		ClkDRP13r	CNTL_CLK_PROG_FREE_RUNNING[1:0]=0x0
		Clk52	CNTL_CLK_PROG_FREE_RUNNING[1:0]=0x2
DPLL	The DPLL module in idle mode cuts off the ClkDPLL104r clock.	Idle	CNTL_CLK[3]=0x1
		Not idle	CNTL_CLK[3]=0x0
APLL	The APLL module can be enabled or disabled.	Disable	CNTL_APLL_DIV_CLK[0]=0x0
		Enable	CNTL_APLL_DIV_CLK[0]=0x1

6.2.5.8 Watchdog Domain

The watchdog domain consists of the following modules:

- Watchdog (functional clock)
- Secure watchdog (functional clock)

Table 6-14 describes the ClkWatchdog clock configurations.

Table 6-13. ClkWatchdog Clock Configurations

ClkWatchdog	Frequency	Parameters	
		Watchdog	Mode
Running	928 kHz	Don't care	Don't care
Big sleep	0 kHz	Disable	Timer mode
	928 kHz	Running	Don't care
		Don't care	Watchdog mode
Deep sleep	0 MHz	Don't care	Don't care

Table 6-14 describes the ClkWatchdog clock parameters.

Table 6-14. ClkWatchdog Clock Parameters

	Description	Value	Register Configuration
Watchdog	Can be disabled in big sleep mode	Disable	CNTL_CLK[2]=0x1
		Running	CNTL_CLK[2]=0x0
Mode	The timer can be configured either in watchdog mode or in timer mode.	Timer mode	See Chapter 14, *Timers and Watchdogs*
		Watchdog mode	

Note: The 13-MHz clock is divided by 14 (928 kHz) for the watchdogs only. The timers receive the ClkDRP13r clock.

6.2.5.9 Interrupt Handler Domain

The interrupt handler domain consists of the following modules:

- ARM interrupt handler (functional clock)
- Secure interrupt handler (functional clock)

Table 6-15 describes the ClkIntH clock configurations.

Table 6-15. ClkIntH Clock Configurations

ClkIntH	Frequency	Source clock	Parameters	
			Interrupt Handler	DPLL
Running	= ClkARM104 (Running)	See ClkARM104	Don't care	Don't care
Big sleep	0 MHz	Don't care	Disable	Don't care
		ClkDPLL104r	Don't care	Idle
	13 MHz	ClkDRP13r	Running	Don't care
	clkm_clk	clkm_clk	Running	Don't care
	ClkDPLL104r 104 MHz (typ)	ClkDPLL104r	Running	Not idle
Deep sleep	0 MHz	Don't care	Don't care	Idle

Table 6-16 describes the ClkIntH clock parameters.

Table 6-16. ClkIntH Clock Parameters

	Description	Value	Register Configuration
Source clock	Three source clocks can be l t d selected.	ClkDPLL104r	CNTL_ARM_CLK[1]=0x0
		ClkDRP13r	CNTL_ARM_CLK[1]=0x1 and CNTL_ARM_CLK[2]=0x0
		ClkDRP13r	CNTL_ARM_CLK[1]=0x1 and CNTL_ARM_CLK[2]=0x1
Interrupt handler	Can be disabled in big sleep mode	Disable	CNTL_CLK[0]=0x1
		Running	CNTL_CLK[0]=0x0
DPLL	The DPLL module in idle mode cuts off the ClkDPLL104r clock.	Idle	CNTL_CLK[3]=0x1
		Not idle	CNTL_CLK[3]=0x0

6.2.5.10 Internal Camera Domain

The internal camera domain consists of the camera module (functional clock).

Table 6-17 describes the ClkCameraInt clock configurations.

Table 6-17. ClkCameraInt Clock Configurations

ClkCameraInt	Frequency	Parameters	
		Source Clock	Camera
Running	0 MHz	Don't care	Disable
		No clock	Enable
	48 MHz	ClkAPLL	Enable
	= ClkBridge52 (Running)	ClkBridge52	Enable
Big sleep	0 MHz	Don't care	Disable
		No clock	Enable
	48 MHz	ClkAPLL	Enable
	= ClkBridge52 (Big sleep)	ClkBridge52	Enable
Deep sleep	0 MHz	Don't care	Don't care

Table 6-18 describes the ClkCameraInt clock parameters.

Table 6-18. ClkWatchdog Clock Parameters

	Description	Value	Register Configuration
Source lclock	Three source clocks can be selected.	No clock	CNTL_CAMERA_DIV_CLK[2:1]=0x0 or 0x3
		ClkAPLL	CNTL_CAMERA_DIV_CLK[2:1]=0x1
		ClkBridge52	CNTL_CAMERA_DIV_CLK[2:1]=0x2
Camera	The camera can be enabled or disabled.	Disable	CNTL_CAMERA_DIV_CLK[0]=0x0
		Enable	CNTL_CAMERA_DIV_CLK[0]=0x1

6.2.5.11 Master Camera Domain

The master camera domain consists of the camera module (functional clock).

Table 6-19 describes the ClkCameraM clock configurations.

Table 6-19. ClkCameraM Clock Configurations

ClkCameraInt	Frequency	Parameters
		APLL
Running	0 MHz	Disable
	48 MHz	Enable
Big sleep	0 MHz	Disable
	48 MHz	Enable
Deep sleep	0 MHz	Don't care

Table 6-20 describes the ClkCameraM clock parameters.

Table 6-20. ClkCameraM Clock Parameters

	Description	Value	Register Configuration
APLL	The APLL module can be enabled or disabled.	Disable	CNTL_APLL_DIV_CLK[0]=0x0
		Enable	CNTL_APLL_DIV_CLK[0]=0x1

6.2.5.12 USB Domain

The USB domain consists of the USB module (functional clock).

Table 6-21 describes the ClkUSB clock configurations.

Table 6-21. ClkUSB Clock Configurations

ClkCameraInt	Frequency	Parameters
		Source Clock
Running	0 MHz	No clock
	13 MHz	ClkDRP13r
	= ClkBridge52 (Running)	ClkBridge52
Big sleep	0 MHz	No clock
	13 MHz	ClkDRP13r
	= ClkBridge52 (Big sleep)	ClkBridge52
Deep sleep	0 MHz	None

Table 6-22 describes the ClkUSB clock parameters.

Table 6-22. ClkUSB Clock Parameters

	Description	Value	Register Configuration
Source clock	Three source clocks can be selected.	No clock	CNTL_CLK_USB[1:0]=0x3
		ClkDRP13r	CNTL_CLK_USB[1:0]=0x0
		ClkBridge52	CNTL_CLK_USB[1:0]=0x2

6.2.5.13 USB_EXT Domain

The USB_EXT domain consists of the USB module (functional clock).

Table 6-23 describes the ClkUSB_EXT clock configurations.

Table 6-23. ClkUSB_EXT Clock Configurations

ClkUSB_EXT	Frequency	Parameters	
		APLL	USB Request?
Running	0 MHz	Disable	No
	48 MHz	Enable	Don't care
		Don't care	Yes
Big sleep	0 MHz	Disable	No
	48 MHz	Enable	Don't care
		Disable	Yes
Deep sleep	0 MHz	Don't care	No

Table 6-24 describes the ClkUSB_EXT clock parameters.

Table 6-24. ClkUSB_EXT Clock Parameters

	Description	Value	Register Configuration
APLL	The APLL module can be enabled or disabled.	Disable	CNTL_APLL_DIV_CLK[0]=0x0
		Enable	CNTL_APLL_DIV_CLK[0]=0x1
USB Request?	The APLL module can be automatically enabled by a USB request.	Yes/No	Automatic

6.2.5.14 Debug Unit Domain

The debug unit domain consists of the debug unit module (functional clock).

Table 6-25 describes the ClkDebug clock configurations.

Table 6-25. ClkDebug Clock Configurations

ClkDebug	Frequency	Conditions
Running	0 MHz	No request generated by ARM
	= ClkARM104 (Running)	Request generated by ARM (one pulse per request)
Big sleep	0 MHz	None
Deep sleep	0 MHz	None

6.2.5.15 Cport Domain

The Cport domain consists of the C-port module (functional clock).

Table 6-26 describes the ClkCport clock configurations.

Table 6-26. ClkCport Clock Configurations

ClkCport	Frequency	Parameters
		C-Port
Running	0 MHz	Disable
	13 MHz	Enable
Big sleep	0 MHz	Disable
	13 MHz	Enable
Deep sleep	0 MHz	Don't care

Table 6-27 describes the ClkCport clock parameters.

Table 6-27. ClkCport Clock Parameters

	Description	Value	Register Configuration
C-port	The C-port module can be enabled or disabled.	Disable	CNTL_CLK[4]=0x0
		Enable	CNTL_CLK[4]=0x1

6.2.5.16 ULPD Gauging Domain

The ULPD gauging domain consists of the ULPD module (functional clock).

Table 6-28 describes the ULPD gauging clock configurations.

Table 6-28. ULPD Gauging Clock Configurations

	Frequency	Parameters	
		APLL	USB Request?
Running	13 MHz (Not regenerated)	ClkDRP13	Don't care
	ClkDPLL104 104 MHz (typ)	ClkDPLL104	Don't care
Big sleep	0 MHz	ClkDPLL104	Idle
	13 MHz (Not regenerated)	ClkDRP13	Don't care
	ClkDPLL104 104 MHz (typ)	ClkDPLL104	Not idle
Deep sleep	0 MHz	Don't care	Idle

Table 6-29 describes the ULPD gauging clock parameters.

Table 6-29. ULPD Gauging Clock Parameters

	Description	Value	Register Configuration
Source clock	Two source clocks can be selected.	ClkDRP13	GAUGING_CTRL_REG[2]=0x0
		ClkDPLL104	GAUGING_CTRL_REG[2]=0x1
DPLL	The DPLL module in idle mode cuts off the ClkDPLL104r clock.	Idle	CNTL_CLK[3]=0x1
		Not idle	CNTL_CLK[3]=0x0

6.2.5.17 32kHz Domain

The 32kHz domain consists of the following modules:

- DRP wrapper (functional clock)
- DRP (functional clock)
- PWL (functional clock)
- LPG (functional clock)

- Keyboard (functional clock)
- ULPD (functional clock)
- GPIO (functional clock)

Table 6-30 describes the Clk32K clock configurations.

Table 6-30. Clk32K Clock Configurations

ClkDebug	Frequency	Conditions
Running	32 kHz	None
Big sleep	32 kHz	None
Deep sleep	32 kHz	None

6.2.5.18 13MHz Domain

The 13MHz domain consists of the following modules:

- PWT (functional clock
- Protected resource reset management (PRRM) (functional clock)
- MCSI (functional clock)
- TPU (functional clock)
- USIM (functional clock)
- I^2C (functional clock)
- DSP I2C → TWL3031 (functional clock)
- ARM I2C → TWL3031 (functional clock)
- GEA (functional clock)
- CIPHER A5 (functional clock)
- LCD (functional clock)
- GPIO (functional clock)
- APC (functional clock)

Table 6-31 describes the Clk13 clock configurations.

Table 6-31. Clk13 Clock Configurations

ClkDebug	Frequency	Conditions
Running	13 kHz	None
Big sleep	13 kHz	None
Deep sleep	0 kHz	None

6.2.5.19 ckout_13mhz Output Clock

The ClkDRP13 clock is available at pad ckout_13mhz by setting the CONF_LOCOSTO_DEBUG[3] CUT_CLK13M bit to 0.

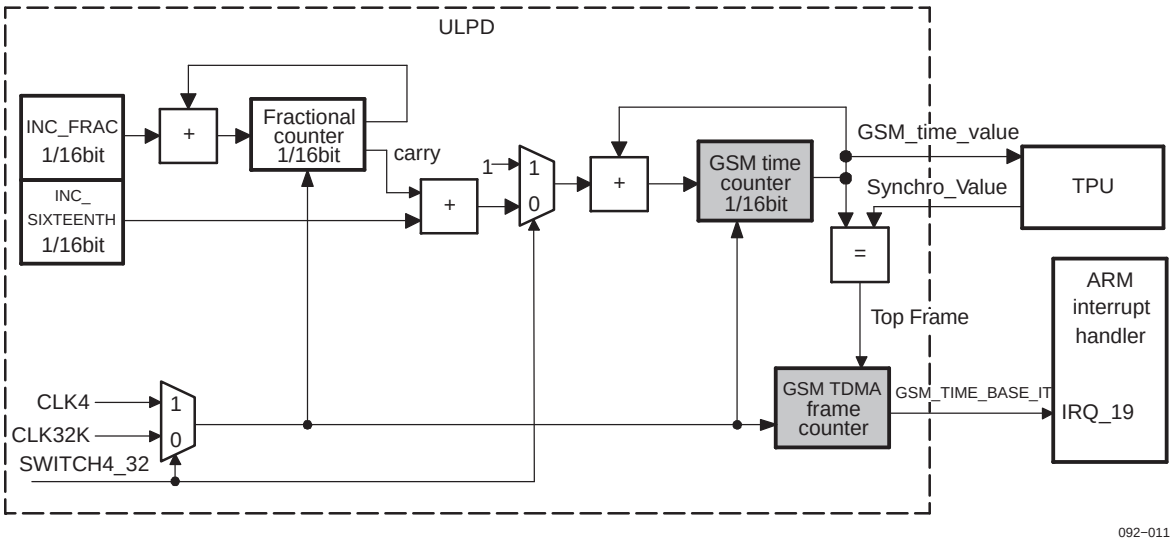
6.2.6 GSM_UnicodeEncodeError_Gauging

6.2.6.1 GSM Time Base

The main function of the GSM time base is to track the GSM frame count through its internal GSM TDMA frame-down counter and to generate the GSM_TIME_BASE_IT interrupt when the GSM TDMA frame counter reaches 0.

Figure 6-11 shows the GSM time-base block diagram.

Figure 6-11. GSM Time-Base Block Diagram



The GSM time base consists of three counters:

- The GSM time counter is a downcounter modulo 20000 expressed in 1/16 GSM bit.
- The fractional counter is a 16-bit downcounter expressed in 1/16 GSM bit.
- The GSM TDMA frame counter is a 16-bit downcounter expressed in TDMA frames (up to 65,536 frames or 302.5 s) used to schedule deep sleep to wake-up state transitions.

The first two counters provide the GSM time value, which represents the GSM time in a TDMA frame expressed in 1/16 GSM bit units. The value 20000 corresponds to 1 TDMA frame (20000 x 1/16 GSM bit = 1250 GSM bits = 8 x 156.25 GSM bits = 8 slots = 1 TDMA frame).

6.2.6.2 GSM Time Counter

The GSM time counter runs based on the state of the LOCOSTO device:

- When in wake-up mode, the GSM time value is provided to the time processing unit (TPU) to monitor GSM baseband processing.
- When in deep sleep mode, the GSM time value is useful for scheduling the GSM TDMA frame countdown by providing a top-frame signal.

Note: The GSM time value exported to the TPU in the GSM interface is in 1/4 GSM bit units (internally divided by 4).

The GSM time base source clock is either the 32-kHz clock in deep sleep mode or the 4.33-MHz clock (this is the 13-MHz clock divided by 3 using an internal divider) in the other modes. Consequently, the GSM time counter increment value depends on the reference clock:

- In deep sleep mode, the reference clock is 32 kHz (Clk32K signal, see Figure 6-11). The increment then is T32kHz/T4.33MHz, which is a ratio computed from gauging the 32-kHz clock (for more information, see Section 6.2.6.4, *Gauging the 32kHz Clock*).
- The reference clock in the other modes is 4.33 MHz (represented by the CLK4 signal, see Figure 6-11); the increment is 1.

The SWITCH4_32 signal generated by the ULPD state-machine switches between the two reference clocks.

The T32kHz/T4.33MHz ratio is programmed in the GSM time-base registers and consists of two parts, the integer part and the fractional part of the ratio:

- INC_SIXTEENTH (integer part of the ratio; up to 4096)
- INC_FRAC (fractional part of the ratio multiplied by 65,536 [216] and rounded to an integer)

Example:

T32kHz / T4.33MHz = 132.2428385417

INC_SIXTEENTH = 132

Fractional = 0. 2428385417 * 65536 = 15914.66666689

INC_FRAC = 15914

The GSM time value is then compared with Synchro_value, which is set in a register of the TPU module (for more information, see Chapter 16, *Time Processing Unit)*, and a top-frame signal is generated when GSM_time_value > Synchro_value.

The top-frame signal clocks the down-counting of the GSM TDMA frame counter.

6.2.6.3 GSM-TDMA Frame Counter

The initial value of the GSM-TDMA frame counter is programmed in the GSM_TIMER_INIT register. The GSM_TIMER_CTRL register allows this value to be loaded by setting the GSM_TIMER_CTRL[0] LOAD bit. The downcounting starts as soon as the LOAD bit is reset (that is, when loading is over). The MPU can also freeze the GSM-TDMA frame counter to delay or suspend the down-counting by setting the GSM_TIMER_CTRL[1] FREEZE bit.

When the GSM-TDMA frame counter reaches 0, the counter is frozen and the GSM_TIME_BASE_IT interrupt is generated (the GSM_TIMER_IT_ REG[0] GSM_TIMER_EVENT_IT bit is set to 1). This interrupt is low-level sensitive and is released by reading the GSM_TIMER_IT_STATUS register.

The transition from deep sleep mode to wake-up mode can be scheduled by programming the SETUP_FRAME register. The value programmed in the SETUP_FRAME register represents the equivalent time in TDMA frame units required to leave the deep sleep mode. This allows being in wake-up mode when the GSM_TIME_BASE_IT interrupt occurs.

6.2.6.4 Gauging the 32kHz Clock

Because the exact Clk32K clock frequency is unknown, it is necessary to gauge it by comparing it to a higher-frequency clock. The higher-frequency clock is selected by programming the ULPD registers.

Gauging can be performed only during an active period and consists of two counters: one counter counts the 32kHz clock periods; the other counter counts the higher-frequency clock periods. Gauging is performed by comparing the counter values.

6.2.6.4.1 Software Limitations

The counters are not resynchronized on the TIPB clock. Therefore, the values are not readable while the counters are running (when gauging is enabled). The correct procedure follows:

The correct procedure follows:

1. Disable the gauging (that is, set the GAUGING_CTRL_REG[0] GAUGING_EN bit to 0).
2. Wait for the IT_GAUGING occurrence.
3. Read the high-frequency and 32-kHz counter values.

To select the higher-frequency clock, program the GAUGING_CTRL_REG[2] SELECT_HI_FREQ_CLOCK bit. After selection, the gauging can start by setting the GAUGING_CTRL_REG[0] GAUGING_EN bit.

The gauging is stopped by resetting the GAUGING_EN bit. The counter values are readable as soon as the GAUGING_IT interrupt occurs (the GAUGING_STATUS_REG[0] GAUGING_IT bit) one cycle after Clk32K. The values are calculated as follows:

- nb_32kHz = COUNTER_32_MSB_REG x 65536 + COUNTER_32_LSB_REG
- nb_hi_freq = COUNTER_HIGH_FREQ_MSB_REG x 65536 + COUNTER_HIGH_FREQ_LSB_REG

With:

- COUNTER_32_MSB_REG: upper value of the number of periods of the 32-kHz clock during gauging time (4 representative bits)

- COUNTER_32_LSB_REG: lower value of the number of periods of the 32-kHz clock during gauging time (16 representative bits)
- COUNTER_HIGH_FREQ_MSB_REG: upper value of the number of periods of the high clock during gauging time (6 representative bits)
- COUNTER_HIGH_FREQ_LSB_REG: lower value of the number of periods of the high-frequency clock during gauging time (16 representative bits)

Comparing these two values allows the software to update the increment values (integer part and fractional part) of the GSM time-base counters (INC_SIXTEENTH and INC_FRAC registers).

6.3 Power Management

6.3.1 Power Domains

The LOCOSTO device can be supplied with an external companion chip. Texas Instruments provides a global solution with the LOCOSTO device associated with an external power device: the TWL3031 device. For more information on the TWL3031 device, contact your TI representative.

The LOCOSTO device is split into the following power domains:

VDD_CORE — DBB ASIC core power domain. This power domain includes the internal SRAM memory and the internal ROM.

> **Note:** The CNTL_CLK_10x register controls an additional retention mode implemented for leakage current reduction (see Table 6-74).

VDD_IO — System I/O supply. This power domain is shared by all system I/Os (including system high- and low-frequency clock I/Os) and interfaces.

VDD_MEM — External memory and EMIF power domain

VDD_PLL — DBB PLL power domain. This power domain is separated from the VDD_CORE domain, thus having the same voltage, to avoid noise injection potential cause of PLL unlocks.

VDD_USIM — DBB SIM power domain. This power domain is provided by the TWL3031 to the LOCOSTO SIM dedicated power domain and used to supply the LOCOSTO SIM card interface.

VCORE — DRP core digital supply. This power domain is directly connected to the DBB core voltage VDD_CORE.

VR1 — Preregulated input to LDOOS, LDOA, and LDORF DRP embedded LDOs. This power domain is isolated from the VCORE, thus allowing the VR1 to be switched off while keeping VCORE active or in low-consumption mode.

VR2 — Preregulated input to LDOX DRP-embedded LDO. This power domain is isolated from the VCORE, thus allowing the VR2 to be switched off while keeping VCORE active or in low-consumption mode.

VRAPC — Preregulated supply to APC DAC analog output stage. This power domain is isolated from the VCORE, thus allowing the VRAPC domain to be switched off while keeping VCORE active or in low-consumption mode.

VDD_APC — APC output amplifier power supply. This power domain is isolated from the VCORE, thus allowing the VDD_APC domain to be switched off while keeping VCORE active or in low-consumption mode.

Table 6-32 compares the characteristics of the TWL3031 power domains and the LOCOSTO power domains.

Table 6-32. Power Domain Characteristics

LOCOSTO Power Domains	TWL3031 Power Domains	Triton Lite Voltage Output Level	Triton Lite Current Output Level	LOCOSTO Power Domain Level
VDD_PLL	VRPLL	1.3 V/1.4 V/1.05 V	10 mA	1.3 V
VDD_MEM	VRMEM	1.8 V	200 mA	1.8 V
VDD_IO	VRIO	1.8 V	200 mA	1.8 V
VDD_USIM	VRSIM	1.8 V/2.85 V	15 mA	1.8 V/2.8 V
VR1 VDD_APC VRAPC (indirectly)	VRMMC	1.8 V/2.85 V	100 mA	2.8 V
VR2	VREXTH	1.8 V/2.8 V	200 mA @ 1.8 V 100 mA @ 2.8 V	2.8 V
VDD_CORE VCORE	VREXTL	1.05 V/1.3 V 1.8 V/2.8 V	200 mA @ 1.05_UnicodeEncodeError_1.3 V 100 mA @ 1.8_UnicodeEncodeError_2.8 V	1.05 V/1.3 V

6.3.2 Power-Saving Requirements

As described in Section 6.2.3, *Power-Saving Modes*, the LOCOSTO device offers three power-saving modes: running, big sleep, and deep sleep (sleeping and wake-up modes are transition states).

- Running mode: All clocks are running except those cut off by software (see Table 6-33).
- Big sleep mode: Requested by setting the CNTL_ARM_CLK[0] BIG_SLEEP bit to 0. The ClkARM104 clock is cut off. The ClkDRP13 and ClkDRP13r clocks still run. To cut off the ClkDPLL104 and ClkDPLL104r clocks, put the DPLL in idle mode (see Section 6.2.5.1, *DPLL*).
- Deep sleep mode: Only the Clk32K clock runs. The CNTL_ARM_CLK[12] DEEP_SLEEP bit must be set to 0. The DPLL module must be in idle mode, as is the DSP.

Table 6-33 summarizes cut-off requirements of the clock domains based on the power-saving modes (Section 6.2.5, *Clock Controls*, provides details of the clock domains).

Table 6-33. Clock Domains Cut-Off Requirements

	Requirements to Cut Off Clock Domain in Running and Wake- Up Modes	Requirements to Cut Off Clock Domain in Big Sleep Mode
ARM104MHz	Not cut	None
DSP104MHz	Not cut	DPLL in idle mode
ClkBridge104MHz	Not cut	CNTL_CLK[1]=0x1 and DMA, NAND Flash IF, EMIF, USB, and UART have no pending operations.
ClkBridge52MHz	Not cut	ClkBridge104MHz is cut off.
UART_UnicodeEncodeError_SPI	CNTL_CLK_PROG_FREE_RUNNING[1:0]=0x3	CNTL_CLK_PROG_FREE_ RUNNING[1:0]=0x3
		CNTL_CLK_PROG_FREE_ RUNNING[1:0]=0x1 and CNTL_APLL_DIV_CLK[0]=0x0
		CNTL_CLK_PROG_FREE_ RUNNING[1:0]=0x2 and DPLL in idle mode
Watchdog	Not cut	CNTL_CLK[2]=0x1 and time mode
Timer	Not cut	Not cut

Table 6-33. Clock Domains Cut-Off Requirements (continued)

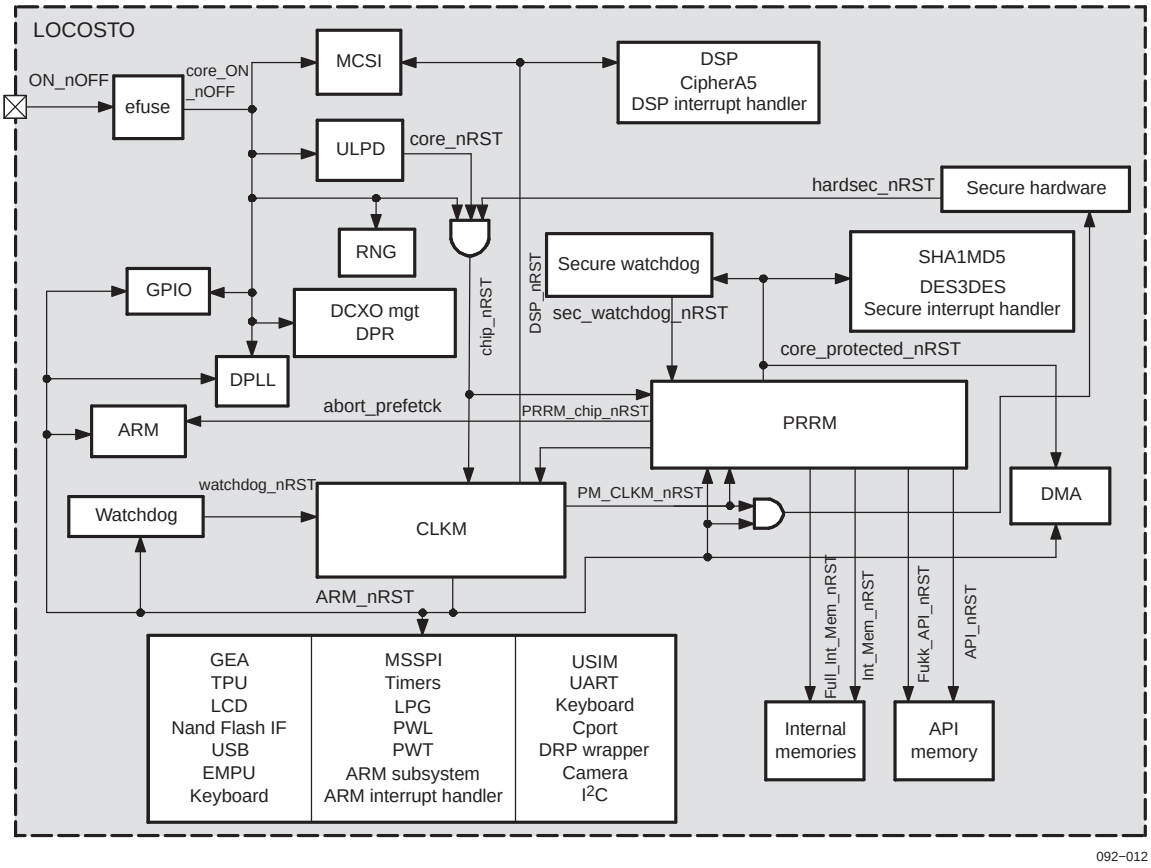
	Requirements to Cut Off Clock Domain in Running and Wake- Up Modes	Requirements to Cut Off Clock Domain in Big Sleep Mode
ARM interrupt handler Secure interrupt handler	Not cut	CNTL_CLK[0]=0x1
		CNTL_ARM_CLK[1]=0x0 and CNTL_CLK[0]=0x0 and DPLL in idle mode
Internal camera	CNTL_CAMERA_ DIV_CLK[0]=0x0	CNTL_CAMERA_DIV_CLK[0]=0x0
	CNTL_CAMERA_ DIV_CLK[0]=0x1 and CNTL_CAMERA DIV CLK[2 1] =0x0 or 0x3	CNTL_CAMERA_DIV_CLK[0]=0x1 and CNTL_CAMERA_DIV_ CLK[2:1]=0x0 or 0x3
		CNTL_CAMERA_DIV_CLK[0]=0x1 and CNTL_CAMERA_DIV_ CLK[2:1]=0x2 and ClkBridge52MHz is cut off.
USB	CNTL_CLK_USB[1:0]=0x3	CNTL_CLK_USB[1:0]=0x3
		CNTL_CLK_USB[1:0]=0x2 and ClkBridge52MHz is cut off.
USB_EXT master camera	CNTL_APLL_DIV_CLK[0]=0x0 and no request from USB or camera	CNTL_APLL_DIV_CLK[0]=0x0 and no request from USB or camera
Debug unit	No request generated by ARM	None
Cport	CNTL_CLK[4]=0x0	CNTL_CLK[4]=0x0
13 MHz	Not cut	Not cut
ULPD gauging	Not cut	Not cut
32 kHz	Not cut	Not cut

6.4 Reset

6.4.1 Reset Distribution

Figure 6-12 shows the LOCOSTO device reset diagram. The following modules generate hardware and software reset signals: effuse, ULPD, CLKM, watchdogs, PRRM, and secure hardware. Section 6.4.2, *Reset Generation*, presents the conditions that activate each reset.

Figure 6-12. Reset Diagram



NOTE: The MCSI module can be reset by DSP_nRST only when this module is controlled by the DSP. The rest of the shared peripherals get only the ARM_nRST reset.

6.4.2 Reset Generation

Table 6-34 lists the conditions that activate each reset signal. The conditions are a combination of the following events or actions:

- Active protected mode (APM) is enabled/disabled (for more information, see Chapter 22, *Security Features*).
- Opcode pre-fetch abort signal is enabled/disabled (the PRRM_CNTL_ REG[3] ABORT_PREFETCH bit).
- Secure watchdog clock is enabled/disabled (the PRRM_CNTL_REG[4] SEC_WDG_CLK_EN bit).
- Protected resources are reset by software (the PRRM_CNTL_REG[0] SOFT_PROT_RST bit, the PRRM_CNTL_REG[1] SOFT_INT_RST bit, and the PRRM_CNTL_REG[2] SOFT_API_RST bit).
- Full reset of the memory is enabled/disabled (the PRRM_CNTL_REG[5] FULL_RST_EN bit).
- The watchdog modules activate their reset (for more information, see Chapter 14, *Timers and Watchdogs*).
- DSP subsystem is reset by software (the CNTL_RST[2] DSP_RESET bit).
- Reset signals are deactivated/activated.

Table 6-34. Conditions Generating Reset Signals in the LOCOSTO Device

Source Module	Reset	Description	Reset Condition(s)
PRRM	abort_prefetch	Opcode prefetch abort sent to MPU	APM enabled and PRRM_CNTL_REG[3]=0x1 and sec_watchdog_nRST activated and PRRM_CNTL_REG[4]=0x1
PRRM	PRRM_chip_nRST	Reset of the LOCOSTO core	APM enabled and PRRM_CNTL_REG[3]=0x0 and sec_watchdog_nRST activated and PRRM_CNTL_REG[4]=0x1
PRRM	core_protected_nRST	Reset of the protected re- (RNG sources RNG, SHA1MD5, DES3DES, DMA secure channel register secure interrupt handler, and secure watchdog) dog)	chip_nRST activated
			ARM_nRST activated
			PM_CLKM_nRST activated
			sec_watchdog_nRST activated and PRRM_CNTL_REG[4]=0x1
			PRRM_CNTL_REG[0]=0x1
PRRM	Int_Mem_nRST	Internal memory space reset from start address to end address; end address must be greater than start address (see RESET_ INT_MEM_ START_ADDR and RESET_ INT_ MEM_END_ADDR registers)	ARM_nRST and PRRM_CNTL_REG[5]=0x0
			PM_CLKM_nRST activated and PRRM_CNTL_REG[4]=0x1 and PRRM_CNTL_REG[5]=0x0
			sec_watchdog_nRST activated and PRRM_CNTL_REG[4]=0x1 and PRRM_CNTL_REG[5]=0x0
			PRRM_CNTL_REG[1]=0x1 and PRRM_CNTL_REG[5]=0x0
PRRM	API_nRST	API memory space reset from start address to end address; end address must be greater than start address (see RESET_API_STAR T_ADDR and RESET_API_END_ ADDR registers)	ARM_nRST and PRRM_CNTL_REG[5]=0x0
			PM_CLKM_nRST activated and PRRM_CNTL_REG[4]=0x1 and PRRM_CNTL_REG[5]=0x0
			sec_watchdog_nRST activated and PRRM_CNTL_REG[4]=0x1 and PRRM_CNTL_REG[5]=0x0
			PRRM_CNTL_REG[2]=0x1 and PRRM_CNTL_REG[5]=0x0
PRRM	Full_Int_Mem_nRST	Reset of the whole internal memory space	chip_nRST activated
			ARM_nRST and PRRM_CNTL_REG[5]=0x1
			PM_CLKM_nRST activated and PRRM_CNTL_REG[5]=0x1
			sec_watchdog_nRST activated and PRRM_CNTL_REG[4]=0x1 and PRRM_CNTL_REG[5]=0x1
			PRRM_CNTL_REG[1]=0x1 and PRRM_CNTL_REG[5]=0x1

Table 6-34. Conditions Generating Reset Signals in the LOCOSTO Device (continued)

Source Module	Reset	Description	Reset Condition(s)
PRRM	Full_API_nRST	Reset of the whole API memory space	chip_nRST activated
			ARM_nRST and PRRM_CNTL_REG[5]=0x1
			PM_CLKM_nRST activated and PRRM_CNTL_REG[4]=0x1 and PRRM_CNTL_REG[5]=0x1
			sec_watchdog_nRST activated and PRRM_CNTL_REG[4]=0x1 and PRRM_CNTL_REG[5]=0x1
			PRRM_CNTL_REG[2]=0x1 and PRRM_CNTL_REG[5]=0x1
ULPD	core_nRST	Power_on reset	Activated when core_ON_nOFF signal activated and released when the Clk13_en signal is enabled (see Section 6.2.2, *Power-Up*)
Watchdogs	watchdog_nRST	Nonsecure watchdog reset	Activated when the watchdog counter reaches 0
	sec_watchdog_nRST	Secure watchdog reset	Activated when the watchdog counter reaches 0
Efuse	core_ON_nOFF	Power-on reset	Released 744 x 32 kHz clock periods after the release of the ON_nOFF signal
Secure hardware	hardsec_nRST	Secure hardware reset	An attempt to access external memory while the protection mode is locked.
CLKM	DSP_nRST	Reset to the DSP subsystem, including MCSI and CipherA5 modules	CNTL_RST[2]=0x1
			watchdog_nRST activated
			chip_nRST activated
			PRRM_chip_nRST activated
	ARM_nRST	Reset to LOCOSTO core, except the DSP subsystem and MCSI and tem CipherA5 modules	watchdog_nRST activated
			chip_nRST activated
			PRRM_chip_nRST activated
	PM_CLKM_nRST	Protected resource reset	Sequence of writings: 1. When APM enabled, write CNTL_PM[0]=0x1. 2. When HSPM enabled and APM disabled, write CNTL_PM[0]=0x0

6.4.3 Memory Resets

The PRRM module is implemented in the LOCOSTO device to control 512K bytes of internal RAM and 32K bytes of API memory space. The lower 256K bytes of the internal RAM are protected with the PRRM module.

A 32-bit memory location is erased every clock cycle in normal mode. Access to the memory space being erased is locked during the entire reset sequence. The internal RAM cannot be accessed during the sequence, but the MPU can access the internal memory interface when the current access does not concern the internal RAM (that is, the wait signal is received only when the internal RAM is accessed).

The RESET_INT_MEM_START_ADDR and RESET_INT_MEM_END_ ADDR registers provide the start and end addresses. The end address must be greater than the start address for the reset sequence to start. Because the granularity of the access to the internal protected RAM is 2x32 bits (on the PRRM module side), both addresses must be in the 0x0000_UnicodeEncodeError_0x7FFF range.

The API memory space cannot be accessed during the reset sequence. As for the internal memory, the RESET_API_START_ADDR and RESET_API_END_ADDR registers provide the start and end addresses. The end address must be greater than the start address for the reset sequence to start. Because the granularity of the access to the API memory is 16 bits (on the PRRM module side), both addresses must be in the 0x0000_UnicodeEncodeError_0x3FFF range.

To provide a fast memory reset procedure on power-up, the ten 32KB internal RAM memory blocks are reset in parallel. This feature is also available during normal operation by enabling the PRRM_CNTL_REG[5] FULL_RST_EN bit. This feature does not concern the API memory.

Notes:

- The nonprotected 64K bytes of the internal RAM cannot be reset.
- During a protected memory reset, the nonprotected memory can be used as an alternative to store temporary data.

6.5 Register Manual

Table 6-35 shows the base addresses and accesses for the PRCM modules.

Table 6-35. Instance Summary

Module Name	Base Address	Size	Access
ULPD	0xFFFE 2000	2K bytes	MPU
CONFIG	0xFFFE F100	256 bytes	MPU
DPLL	0xFFFF 9800	2K bytes	MPU
PRRM	0xFFFF FC00	256 bytes	MPU
CLKM	0xFFFF FD00	256 bytes	MPU

6.5.1 Module Register Mapping Summary

Table 6-36 through Table 6-40 describe the register bits.

Table 6-36. ULPD Register Addresses

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
INC_FRAC_REG	RW	16	0x00	0xFFFE 2000
INC_SIXTEENTH_REG	RW	16	0x02	0xFFFE 2002
SIXTEENTH_START_REG	R	16	0x04	0xFFFE 2004
SIXTEENTH_STOP_REG	R	16	0x06	0xFFFE 2006
COUNTER_32_LSB_REG	R	16	0x08	0xFFFE 2008
COUNTER_32_MSB_REG	R	16	0x0A	0xFFFE 200A
COUNTER_HI_FREQ_ LSB_REG	R	16	0x0C	0xFFFE 200C
COUNTER_HI_FREQ_ MSB_REG	R	16	0x0E	0xFFFE 200C
GAUGING_CTRL_REG	RW	16	0x10	0xFFFE 2010
GAUGING_STATUS_REG	R	16	0x12	0xFFFE 2012
GSM_TIMER_CNTL_REG	RW	16	0x14	0xFFFE 2014
GSM_TIMER_INIT_REG	RW	16	0x16	0xFFFE 2016
GSM_TIMER_VALUE_REG	R	16	0x18	0xFFFE 2018
GSM_TIMER_IT_REG	R	16	0x1A	0xFFFE 201A
SETUP_CLOCK_13MHZ_REG	RW	16	0x1C	0xFFFE 201C
SETUP_SLICER_REG	RW	16	0x1E	0xFFFE 201E
SETUP_VTCXO_REG	RW	16	0x20	0xFFFE 2020

Table 6-36. ULPD Register Addresses (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
SETUP_FRAME_REG	RW	16	0x22	0xFFFE 2022
SETUP_RF_REG	RW	16	0x24	0xFFFE 2024
ULPD_DCXO_SETUP_ SLEEPN	W	16	0x26	0xFFFE 2026
ULPD_DCXO_SETUP_ SYSCLKEN	W	16	0x28	0xFFFE 2028

Table 6-37. CONFIG Register Addresses

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
CONF_LOCOSTO_DEBUG	RW	16	0x20	0xFFFE F120

See Chapter 18, *Configuration*, for definitions of all CONFIG module registers.

Table 6-38. Register Addresses

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
DPLL_CNTL_REG	RW	16	0x00	0xFFFF 9800

Table 6-39. PRRM Register Addresses

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
RESET_ INT_MEM_START_ADDR	RW	16	0x00	0xFFFF FC00
Reset_Int_Mem_End_Addr	RW	16	0x02	0xFFFF FC02
Reset_API_Start_Addr	RW	16	0x04	0xFFFF FC04
Reset_API_End_Addr	RW	16	0x06	0xFFFF FC06
PRRM_CNTL_REG	RW	16	0x08	0xFFFF FC08
STATUS_REG	R	16	0x0A	0xFFFF FC0A
APPLICATION_ID	RW	16	0x0C	0xFFFF FC0C

Table 6-40. CLKM Register Addresses

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
CNTL_ARM_CLK	RW	16	0x00	0xFFFF FD00
CNTL_CLK	RW	16	0x02	0xFFFF FD02
CNTL_RST	RW	16	0x04	0xFFFF FD04
CNTL_CLK_10x	RW	16	0x06	0xFFFF FD06
CNTL_CAMERA_DIV_CLK	RW	16	0x08	0xFFFF FD08
RESERVED	R	16	0x0A	0xFFFF FD0A
CNTL_CLK_USB	RW	16	0x0C	0xFFFF FD0C
CNTL_CLK_PROG_FREE_ RUNNING	RW	16	0x0E	0xFFFF FD0E
CNTL_APLL_DIV_CLK	RW	16	0x10	0xFFFF FD10
CNTL_PM	RW	16	0x12	0xFFFF FD12

6.5.2 Register Description

6.5.2.1 ULPD

Table 6-41. INC_FRAC_REG

Address Offset	0x00		
Physical Address	0xFFFE 2000	**Instance**	ULPD
Description	GSM time base register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INC_FRAC															

Bits	Field Name	Description	Type	Reset
15:0	INC_FRAC	Fractional part of the duration of the 32 kHz clock period in 1/16 GSM bit unit	RW	0x0

Table 6-42. INC_SIXTEENTH_REG

Address Offset	0x02		
Physical Address	0xFFFE 2002	**Instance**	ULPD
Description	GSM time base register		
Type	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INC_SIXTEENTH															

Bits	Field Name	Description	Type	Reset
15:0	INC_SIXTEENTH	Integer part of the duration of the 32 kHz clock period in 1/16 GSM bit unit	RW	0x0

Table 6-43. SIXTEENTH_START_REG

Address Offset	0x04		
Physical Address	0xFFFE 2004	**Instance**	ULPD
Description	GSM time base at start of gauging		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIXTEENTH_START															

Bits	Field Name	Description	Type	Reset
15:0	SIXTEENTH_START	Value of GSM time base at the start of gauging	R	0x0

Table 6-44. SIXTEENTH_STOP_REG

Address Offset	0x06		
Physical Address	0xFFFE 2006	**Instance**	ULPD
Description	GSM time base at stop of gauging		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIXTEENTH_STOP															

Bits	Field Name	Description	Type	Reset
15:0	SIXTEENTH_STOP	Value of GSM time base at the stop of gauging	R	0x0

Table 6-45. COUNTER_32_LSB_REG

Address Offset	0x08		
Physical Address	0xFFFE 2008	**Instance**	ULPD
Description	Number of clock 32 KHz during gauging time		
Type	r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNTER_32_LSB															

Bits	Field Name	Description	Type	Reset
15:0	COUNTER_32_LSB	Lower value of the number of clock 32 kHz during gauging time	R	0x1

Table 6-46. COUNTER_32_MSB_REG

Address Offset	0x0A		
Physical Address	0xFFFE 200A	**Instance**	ULPD
Description	Number of clock 32 kHz during gauging time		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												COUNTER_32_MSB			

Bits	Field Name	Description	Type	Reset
15:4	RESERVED	Reserved	R	Undefined
3:0	COUNTER_32_MSB	Upper value of the number of clock 32 kHz during gauging time	R	0x0

Table 6-47. COUNTER_HI_FREQ_LSB_REG

Address Offset	0x0C		
Physical Address	0xFFFE 200C	**Instance**	ULPD
Description	Number of high frequency clock during gauging time		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNTER_HIGH_															

Bits	Field Name	Description	Type	Reset
15:0	COUNTER_HIGH_ FREQ_LSB	Lower value of the number of high frequency clock during gauging time	R	0x1

Table 6-48. COUNTER_HI_FREQ_MSB_REG

Address Offset	0x0E		
Physical Address	0xFFFE 200E	**Instance**	ULPD
Description	Number of high frequency clock during gauging time		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										COUNTER_HIGH_ FREQ_MSB					

Bits	Field Name	Description	Type	Reset
15:6	RESERVED	Reserved	R	Undefined
5:0	COUNTER_HIGH_ FREQ_MSB	Upper value of the number of high frequency clock during gauging time	R	0x0

Table 6-49. GAUGING_CTRL_REG

Address Offset	0x10		
Physical Address	0xFFFE 2010	**Instance**	ULPD
Description	Gauging control		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													SELECT_HI_FREQ_CLOCK	GAUGING_TYPE	GAUGING_EN

Bits	Field Name	Description	Type	Reset
15:3	RESERVED	Reserved	RW	Undefined
2	SELECT_HI_FREQ_CLOCK	Clock frequency selection for gauging 0: ClkDRP13 (13MHz) 1: ClkDPLL104 (typ. 104 MHz	RW	0x0
1	GAUGING_TYPE	0: Gauging versus GSM network timer 1: Gauging versus high frequency clock	RW	0x0
0	GAUGING_EN	0: Disable 1: Enable	RW	0x0

Table 6-50. GAUGING_STATUS_REG

Address Offset	0x12		
Physical Address	0xFFFE 2012	**Instance**	ULPD
Description	Gauging status		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													OVERFLOW_32	OVERFLOW_HIGH_FREQ	GAUGING_IT

Bits	Field Name	Description	Type	Reset
15:3	RESERVED	Reserved	RW	Undefined
2	OVERFLOW_32	0: No event 1: An overflow occurred on the counter_ 32 during gauging **Note:** This bit is cleared when a new gauging is started	R	0x0
1	OVERFLOW_HIGH_FREQ	0: No event 1: An overflow occurred on the counter_ high_freq during gauging versus high frequency clock	R	0x0

Bits	Field Name	Description	Type	Reset
		Note: This bit is cleared when a new gauging versus high frequency clock is started		
0	GAUGING_IT	0: No event	R	0x0
		1: Interrupt GAUGING_IT generated		
		Note: This bit is cleared on reading of this register		

Table 6-51. GSM_TIMER_CNTL_REG

Address Offset	0x14		
Physical Address	0xFFFE 2014	**Instance**	ULPD
Description	GSM timer control		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														FREEZE	LOAD

Bits	Field Name	Description	Type	Reset
15:2	RESERVED	Reserved	RW	Undefined
1	FREEZE	0: GSM timer is running	RW	0x0
		1: GSM timer is frozen		
0	LOAD	0: No action	RW	0x0
		1: Load the timer with TIMER_INIT value (toggle bit)		

Table 6-52. GSM_TIMER_INIT_REG

Address Offset	0x16		
Physical Address	0xFFFE 2016	**Instance**	ULPD
Description	Load value of the GSM timer		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMER_INIT															

Bits	Field Name	Description	Type	Reset
15:0	TIMER_INIT	Load value of the timer	RW	0x0

Table 6-53. GSM_TIMER_VALUE_REG

Address Offset	0x18		
Physical Address	0xFFFE 2018	**Instance**	ULPD
Description	Current GSM timer value		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMER_VALUE															

Bits	Field Name	Description	Type	Reset
15:0	TIMER_VALUE	Current timer value	R	0x1

Table 6-54. GSM_TIMER_IT_REG

Address Offset	0x1A		
Physical Address	0xFFFE 201A	**Instance**	ULPD
Description	GSM timer interrupt status		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															GSM_TIMER_IT_EVENT

Bits	Field Name	Description	Type	Reset
15:1	RESERVED	Reserved	R	Undefined
0	GSM_TIMER_IT_EVENT	0: No event	R	0x0
		1: GSM timer interrupt generated		

Table 6-55. SETUP_CLOCK_13MHZ_REG

Address Offset	0x1C		
Physical Address	0xFFFE 201C	**Instance**	ULPD
Description	FSM CLK13 counter		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								SETUP_CLK13							

Bits	Field Name	Description	Type	Reset
15:6	RESERVED	Reserved	RW	Undefined
5:0	SETUP_CLK13	Number of 32 kHz clock periods to enable the 13 MHz clock from slicer enable	RW	0x3F

Table 6-56. SETUP_SLICER_REG

Address Offset	0x1E		
Physical Address	0xFFFE 201E	**Instance**	ULPD
Description	FSM slicer counter		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				SETUP_SLICER											

Bits	Field Name	Description	Type	Reset
15:12	RESERVED	Reserved	RW	Undefined
11:0	SETUP_SLICER	Number of 32 kHz clock periods to start the CLK13 counter from VTCXO enable	RW	0xFFF

Table 6-57. SETUP_VTCXO_REG

Address Offset	0x20		
Physical Address	0xFFFE 2020	**Instance**	ULPD
Description	FSM VTCXO counter		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				SETUP_VTCXO											

Bits	Field Name	Description	Type	Reset
15:12	RESERVED	Reserved	RW	Undefined
11:0	SETUP_VTCXO	Number of 32 kHz clock periods to start the slicer counter	RW	0xFFF

Table 6-58. SETUP_FRAME_REG

Address Offset	0x22		
Physical Address	0xFFFE 2022	**Instance**	ULPD
Description	Number of frames to begin the wake up phase		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											SETUP_FRAME				

Bits	Field Name	Description	Type	Reset
15:5	RESERVED	Reserved	RW	Undefined
4:0	SETUP_FRAME	Number of frames to begin the wake up phase (in TDMA frame unit)	RW	0x0

Table 6-59. SETUP_RF_REG

Address Offset	0x24		
Physical Address	0xFFFE 2024	**Instance**	ULPD
Description	FSM RF counter		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								SETUP_RF							

Bits	Field Name	Description	Type	Reset
15:8	RESERVED	Reserved	RW	Undefined
7:0	SETUP_RF	Number of 32 kHz clock periods to start the vtcxo counter	RW	0x0

Table 6-60. ULPD_DCXO_SETUP_SLEEPN

Address Offset	0x26		
Physical Address	0xFFFE 2026	**Instance**	ULPD
Description	DCXO management setup		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								ULPD_DCXO_SETUP_SLEEPN							

Bits	Field Name	Description	Type	Reset
15:8	RESERVED	Reserved	W	undefined
7:0	ULPD_DCXO_SETUP_SLEEPN	Contains setup sleepz time	W	0xFF

Table 6-61. ULPD_DCXO_SETUP_SYSCLKEN

Address Offset	0x28		
Physical Address	0xFFFE 2028	**Instance**	ULPD
Description	DCXO management setup		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								ULPD_DCXO_SETUP_SYSCLKEN							

Bits	Field Name	Description	Type	Reset
15:8	RESERVED	Reserved	W	undefined
7:0	ULPD_DCXO_SETUP_SYSCLKEN	Contains setup_sys_clk_en time	W	0xFF

6.5.2.2 CONFIG

Table 6-62. CONF_LOCOSTO_DEBUG

Address Offset	0x20		
Physical Address	0xFFFE F120	**Instance**	CONFIG
Description	Locosto software debug override configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											CLK13M_OFF_VALUE	CUT_CLK13M	SECURE	ALL_ZERO_MODE	ALL_ONE_MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED	Reserved	RW	undefined
4	CLK13M_OFF_VALUE	When CUT_CLK13M is high after synchronization time the CLK13M_OUT follow the value stored in this bit. (then become usable as General purpose output)		
3	CUT_CLK13M	0: Enable Clock 13MHz 1: Disable to CLK13M_OFF_VALUE	RW	0
2	SECURE	This bit can be programmed to enable security feature, when no software debug signals inside the core can be monitored. 0: Normal debug values are available from debug module 1: All debug observation nodes are forced to zero.	RW	0
1	ALL_ZERO_MODE	This bit can be programmed to force to zero, all outputs from Locosto software debug module. 0: Normal debug values are available from debug module 1: All debug observation nodes are forced to one.	RW	0
0	ALL_ONE_MODE	This bit can be programmed to force to one, all outputs from Locosto software debug module. 0: Normal debug values are available from debug module 1: All debug observation nodes are forced to zero.	RW	0

6.5.2.3 PRM

Table 6-63. RESET_INT_MEM_START_ADDR

Address Offset	0x00		
Physical Address	0xFFFF FC00	**Instance**	PRRM
Description	This register defines the address from where the reset will start in the internal memory space		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	PRRM_INT_START														

Bits	Field Name	Description	Type	Power on Reset	Hardware and Software Resets
15	RESERVED	Reserved	RW	Undefined	Undefined
14:0	PRRM_INT_START	Start address	RW	0x0	Not Modified

Table 6-64. RESET_INT_MEM_END_ADDR

Address Offset	0x02		
Physical Address	0xFFFF FC02	**Instance**	PRRM
Description	This register defines the address where the reset will stop in the internal memory space		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	PRRM_INT_END														

Bits	Field Name	Description	Type	Power on Reset	Hardware and Software Resets
15	RESERVED	Reserved	RW	Undefined	Undefined
14:0	PRRM_INT_END	End address	RW	0x0	Not Modified

Table 6-65. RESET_API_START_ADDR

Address Offset	0x04		
Physical Address	0xFFFF FC04	**Instance**	PRRM
Description	This register defines the address from where the reset will start in the API memory space		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRRM_API_START															

Bits	Field Name	Description	Type	Power on Reset	Hardware and Software Resets
15:0	PRRM_API_START	Start address	RW	0x0	Not Modified

Table 6-66. RESET_API_END_ADDR

Address Offset	0x06														
Physical Address	0xFFFF FC06			**Instance**	PRRM										
Description	This register defines the address where the reset will stop in the API memory space														
Type	RW														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRRM_API_END															

Bits	Field Name	Description	Type	Power on Reset	Hardware and Software Resets
15:0	PRRM_API_END	End address	RW	0x0	Not Modified

Table 6-67. PRRM_CNTL_REG

Address Offset	0x08														
Physical Address	0xFFFF FC08			**Instance**	PRRM										
Description	Control register														
Type	RW														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								PROT_VIOL_RST	RNG_DOUT_VIOL	FULL_RST_EN	SEC_WDG_CLK_EN	ABORT_PREFETCH	SOFT_API_RST	SOFT_INT_RST	SOFT_PROT_RST

Bits	Field Name	Description	Type	Power on Reset	Hardware and Software Resets
15:8	RESERVED	Reserved	RW	Undefined	Undefined
7	PROT_VIOL_RST	Reset of the crypto modules (SHA1MD5, DES3DES, RNG) on a protected violation event 0: Disable 1: Enable	RW	0x0	Not Modified
6	RNG_DOUT_VIOL	0: Disable the RNG data out register watch mechanism 1: Enable the RNG data out register watch mechanism. If the random number is read while the RNG busy bit is active it will create a protected violation.	RW	0x0	0x0
5	FULL_RST_EN	0: Disable the full memory reset configuration 1: Enable the full memory reset configuration. Memory space start and end addresses are ignored. (Fast-up the reset sequence)	RW	0x1	0x0
4	SEC_WDG_CLK_EN	Secure watchdog timer clock 0: Disable 1: enable	RW	0x1	Not Modified
3	ABORT_PREFETCH	0: Disable watchdog abort prefetch signal 1: Enable watchdog abort prefetch signal on secure watchdog abort/ reset request	RW	0x0	Not Modified
2	SOFT_API_RST	0: None 1: Reset the API memory. Auto clear bit (always read as 0).	RW	0x0	Not Modified
1	SOFT_INT_RST	0: None	RW	0x0	Not Modified

Bits	Field Name	Description	Type	Power on Reset	Hardware and Software Resets
		1: Reset the internal memory. Auto clear bit (always read as 0).			
0	SOFT_PROT_RST	0: None 1: Reset the protected resources. Auto clear bit (always read as 0).	RW	0x0	Not Modified

Table 6-68. STATUS_REG

Address Offset	0x0A		
Physical Address	0xFFFF FC0A	**Instance**	PRRM
Description	Status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											API_RST	INT_RST	ILLEGAL_PM_RST	ILLEGAL_CHIP_RST	ILLEGAL_WDG_RST

Bits	Field Name	Description	Type	Power on Reset	Hardware and Software Resets
15:5	RESERVED	Reserved	R	Undefined	Undefined
4	API_RST	0: None 1: Indicates a reset on the API memory space. Active high.	R	0x0	Depends on Reset Source
3	INT_RST	0: None 1: Indicates a reset on the Internal memory space. Active high	R	0x0	Depends on Reset Source
2	ILLEGAL_PM_RST	0: None 1: Indicates a Protected Mode reset occurred when the system was in protected mode	R	0x0	NA
1	ILLEGAL_CHIP_RST	0: None 1: Indicates a chip reset occurred when the system was in protected mode	R	0x0	NA
0	ILLEGAL_WDG_RST	0: None 1: Indicates a watchdog abort/reset occurred when the system was in protected mode	R	0x0	NA

Table 6-69. APPLICATION_ID

Address Offset	0x0C		
Physical Address	0xFFFF FC0C	**Instance**	PRRM
Description	Application ID		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APPLICATION_ID															

Bits	Field Name	Description	Type	Power on Reset	Hardware and Software Resets
15:0	APPLICATION_ID	Application ID	RW	0x0	Not Modified

6.5.2.4 DPLL

Table 6-70. DPLL_CNTL_REG

Address Offset	0x00		
Physical Address	0xFFFF 9800	**Instance**	DPLL
Description	DPLL control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		IOB	Reserved	PLL_MULT				PLL_DIV			PLL_ENABLE	BYPASS_DIV		BREAKLN	LOCK

Bits	Field Name	Description	Type	Reset
15:14	Reserved	Reserved	RW	Undefined
13	IOB	Initialize On Break. 0: The DPLL module stays in lock mode even if the lock is lost. BREAKLN bit is then set to 0 1: The DPLL switches to Bypass mode and start a new locking sequence	RW	0x1
12	Reserved	Reserved	RW	Undefined
11:7	PLL_MULT	Lock mode, multiply value: 00000: ClkDPLL104 = ClkDRP13 / PLL_DIV Others: ClkDPLL104 = ClkDRP13 x PLL_MULT / PLL_DIV	RW	0x00
6:5	PLL_DIV	Lock mode, divide value. 00: No action on the ClkDRP13 clock 01: ClkDRP13 / 2 10: ClkDRP13 / 3 11: ClkDRP13 / 4	RW	0x0
4	PLL_ENABLE	0: DPLL in BYPASS mode 1: Requests the DPLL to enter the LOCK mode	RW	0x0
3:2	BYPASS_DIV	ClkDPLL104 clock frequency in BYPASS mode: 00: ClkDPLL104 = ClkDRP13 01: ClkDPLL104 = ClkDRP13 / 2 1X: ClkDPLL104 = ClkDRP13 / 4	RW	0x0
1	BREAKLN	0: The DPLL module has broken lock for some unknown reason 1: The lock condition is restored or a write to the DPLL_CNTL_REG register occurs	RW	Undefined
0	LOCK	0: The DPLL module is in the BYPASS mode 1: The DPLL module is locked	RW	Undefined

6.5.2.5 CLKM

Table 6-71. CNTL_ARM_CLK

Address Offset	0x00		
Physical Address	0xFFFF FD00	**Instance**	CLKM
Description	Source clock and mode selection		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		RESERVED	DEEP_SLEEP	DEEP_POWER				RESERVED					CLKIN_SEL	CLKIN_SEL0	BIG_SLEEP

Bits	Field Name	Description	Type	Reset
15:14	RESERVED	Reserved	RW	undefined
13	RESERVED	Must be set to 0x0 for proper behavior	RW	0x0
12	DEEP_SLEEP	Deep Sleep mode: 0: Enable 1: Disable	RW	0x1
11:8	DEEP_POWER	Defines the Tam period between RESPWR lifting and the 104-MHz clock starting. This is programmable in the number of 13-MHz clock cycles.	RW	0x0
7:3	RESERVED	Reserved	RW	0x1F
2	CLKIN_SEL	Clock-in selection: 0: 13 MHz (ClkDRP13r) 1: clkm_clk input (external up_UnicodeEncodeError_to 40MHz)	RW	0x0
1	CLKIN_SEL0	Source clock for MPU system: 0: ClkDPLL104r clock (typ. 104MHz) 1: DCXO or clkm_clk (see *CLKIN_SEL*)	RW	0x0
0	BIG_SLEEP	Big sleep mode: 0: Enable 1: Disable	RW	0x0

Table 6-72. CNTL_CLK

Address Offset	0x02		
Physical Address	0xFFFF FD02	**Instance**	CLKM
Description	Clock controls		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_CNTL	nIDLE3	RESERVED								EN_IDLE3_FLG	CPORT_CLK_EN	DPLL_DIS	TIMER_CLK_DIS	BRIDGE_CLK_DIS	IRQ_CLK_DIS

Bits	Field Name	Description	Type	Reset
15	FLASH_CNTL	External memory control: 0: Whenever the Bridge104MHz domain is cut off, the memory is also put in sleep mode 1: The memory is not allowed to go in sleep mode even if the Bridge104Mhz goes in sleep mode	RW	0x0
14	nIDLE3	0: DSP is in idle3 mode 1: DSP is not in idle3 mode	RW	0x1
13:6	RESERVED	Reserved	RW	undefined
5	EN_IDLE3_FLG	DSP idle flag control (to API): 0: Let API SAM / HOM modes to be set by software	RW	0x0

Bits	Field Name	Description	Type	Reset
		1: API is forcefully set to HOM mode when DSP enters in idle3 mode		
4	CPORT_CLK_EN	C-Port clock: 0: Disable 1: Enable	RW	0x1
3	DPLL_DIS	DPLL control: 0: DPLL is not stopped and continue providing an output clock (according to its control register content) when both DSP and ARM system are in IDLE3 and SLEEP mode respectively. 1: DPLL is set in IDLE mode when both DSP and ARM system are in IDLE3 and SLEEP mode respectively.	RW	0x0
2	TIMER_CLK_DIS	Timer clock control: 0: Left free running during Big sleep mode 1: Disabled and enabled according to the Big sleep mode	RW	0x0
1	BRIDGE_CLK_DIS	Bridge104MHz domain control: 0: Always running 1: Disabled and enabled according to the Big sleep mode	RW	0x0
0	IRQ_CLK_DIS	Interrupt handler clock control: 0: Always running 1: Disabled and enabled according to the Big sleep mode	RW	0x1

Table 6-73. CNTL_RST

Address Offset	0x04		
Physical Address	0xFFFF FD04	**Instance**	CLKM
Description	Reset Control		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												WATCHDOG_RESET	DSP_RESET	RESERVED	

Bits	Field Name	Description	Type	Reset
15:4	RESERVED	Reserved	RW	undefined
3	WATCHDOG_RESET	Watchdog reset (not applicable to the secure watchdog) 0: None 1: Occurred	RW	0x0
2	DSP_RESET	DSP reset command: 0: None 1: Reset DSP	RW	0x1
1:0	RESERVED	Reserved	RW	undefined

Table 6-74. CNTL_CLK_10x

Address Offset	0x06		
Physical Address	0xFFFF FD06	**Instance**	CLKM
Description	memory retention control for additional leakage current reduction		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						MCU_INTRAM_VDD	DRP_EXTRAM_VDD	DSP_EXTRAM_VDD	DSP_APIRAM_VDD	DSP_DARAM_VDD	MCU_INTRAM_VSS	DRP_EXTRAM_VSS	DSP_EXTRAM_VSS	DSP_APIRAM_VSS	DSP_DARAM_VSS

Bits	Field Name	Description	Type	Reset
15:10	Reserved	Reserved	RW	undefined
9	MCU_INTRAM_VDD	SLPZVDD control of MCU Internal memories 0: Disable 1: Enable	RW	0x0
8	DRP_EXTRAM_VDD	SLPZVDD control of DRP External RAM 0: Disable 1: Enable	RW	0x0
7	DSP_EXTRAM_VDD	SLPZVDD control of DSP External RAM 0: Disable 1: Enable	RW	0x0
6	DSP_APIRAM_VDD	SLPZVDD control of DSP API RAM memories 0: Disable 1: Enable	RW	0x0
5	DSP_DARAM_VDD	SLPZVDD control of DSP DARAM memories 0: Disable 1: Enable	RW	0x0
4	MCU_INTRAM_VSS	SLPZVSS control of MCU Internal memories 0: Disable 1: Enable	RW	0x1
3	DRP_EXTRAM_VSS	SLPZVSS control of DRP External RAM memories 0: Disable 1: Enable	RW	0x1
2	DSP_EXTRAM_VSS	SLPZVSS control of DSP External RAM memories 0: Disable 1: Enable	RW	0x1
1	DSP_APIRAM_VSS	SLPZVSS control of DSP API RAM memories 0: Disable 1: Enable	RW	0x1
0	DSP_DARAM_VSS	SLPZVSS control of DSP DARAM memories 0: Disable 1: Enable	RW	0x1

Table 6-75. CNTL_CAMERA_DIV_CLK

Address Offset	0x08		
Physical Address	0xFFFF FD08	**Instance**	CLKM
Description	Camera control clock		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													CAMERA_DIV_FACTOR		CAMERA_ENABLE

Bits	Field Name	Description	Type	Reset
15:3	RESERVED	Reserved	RW	undefined
2:1	CAMERA_DIV_FACTOR	Defines CAMERA clock 01: ClkAPLL (typ. 48MHz) 10: ClkBridge52 (typ. 52MHz) Others: No clock	RW	undefined
0	CAMERA_ENABLE	0: CAMERA disabled 1: CAMERA enabled	RW	0x0

Table 6-76. CNTL_CLK_USB

Address Offset	0x0C		
Physical Address	0xFFFF FD0C	**Instance**	CLKM
Description	USB control clock		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															CNTL_CLK_USB

Bits	Field Name	Description	Type	Reset
15:2	RESERVED	Reserved	RW	undefined
1:0	CNTL_CLK_USB	Defines the USB clock 00: 13 MHz 01: Reserved 10: ClkBridge52 (typ. 52 MHz) 11: No clock	RW	0x0

Table 6-77. CNTL_CLK_PROG _FREE_RUNNING

Address Offset	0x0E		
Physical Address	0xFFFF FD0E	**Instance**	CLKM
Description	Control clock for free running modules		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														CNTL_CLK_PROG_FREE_RUNNING	

Bits	Field Name	Description	Type	Reset
15:2	RESERVED	Reserved	RW	undefined
1:0	CNTL_CLK_PROG_FREE_RUNNING	Defines the free running clock 00: 13 MHz 01: ClkAPLL (typ. 48 MHz) 10: Clk52 (typ. 52 MHz) 11: No clock	RW	0x0

Table 6-78. CNTL_APLL_DIV_CLK

Address Offset	0x10		
Physical Address	0xFFFF FD10	**Instance**	CLKM
Description	APLL control		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															APLL_ENABLE

Bits	Field Name	Description	Type	Reset
15:1	RESERVED	Reserved	RW	undefined
0	APLL_ENABLE	0: APLL disabled 1: APLL enabled	RW	0x0

Table 6-79. CNTL_PM

Address Offset	0x12		
Physical Address	0xFFFF FD12	**Instance**	CLKM
Description	Protected resource reset and status register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														PM_STATUS	PM_RESET

Bits	Field Name	Description	Type	Reset
15:2	RESERVED	Reserved	RW	undefined
1	PM_STATUS	0: Protected mode halted or suspended 1: Protected mode activated in APM or HSPM mode (see Chapter 22, Security overview)	RW	0x0
0	PM_RESET	Protected mode reset control for protected resource 0: If this bit was set to 1, generate a reset – PM_CLKM_nRST – in HSPM mode, APM is deactivated 1: Can be set only when APM is activated	RW	0x0

On-Chip Memory

This chapter describes the on-chip memory of the LOCOSTO device.

Topic **Page**

7.1 Module Overview

The on-chip memory (OCM) consists of two on-chip memory controllers connected independently to an on-chip 64K-byte boot ROM (OCM BOOT ROM) and an on-chip 512K-byte SRAM/ROM (OCM SRAM and OCM ROM, respectively).

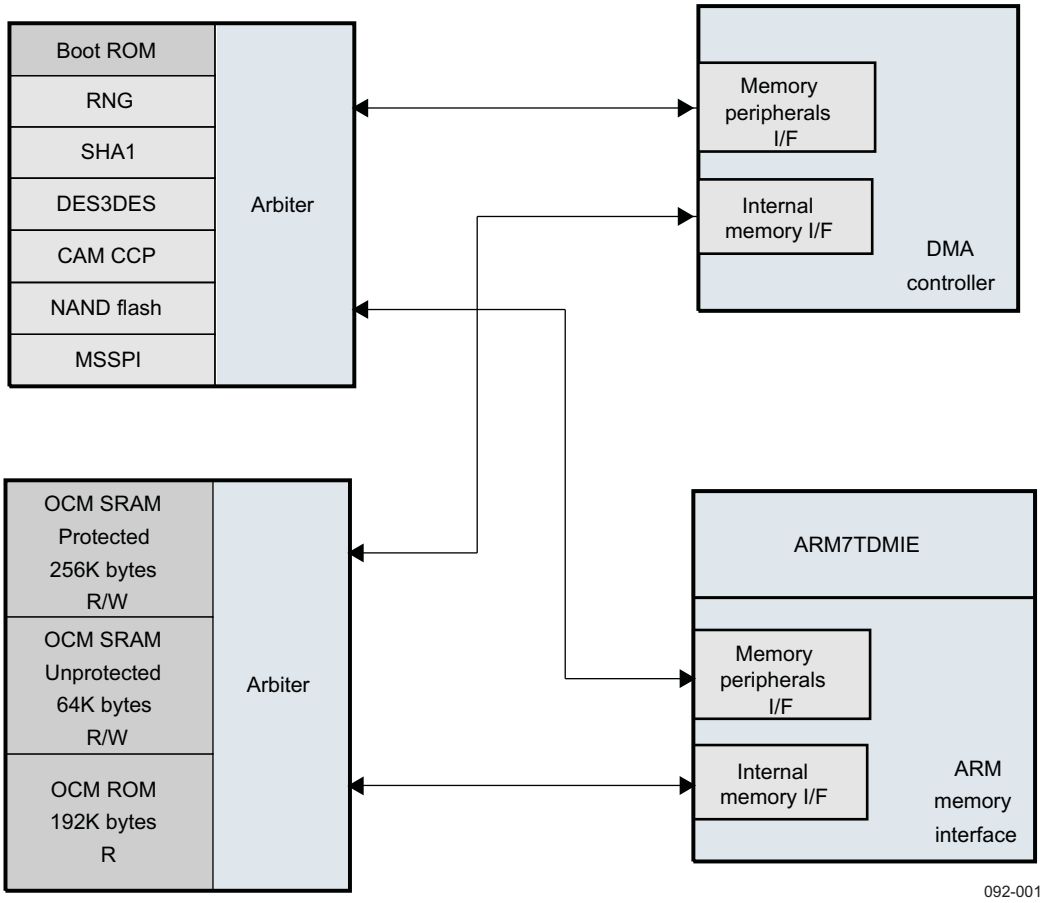
The 512K bytes of the SRAM/ROM are divided as follows:

- 256K bytes protected OCM SRAM
- 64K bytes nonprotected OCM SRAM
- 192K bytes OCM ROM

For more information about the protected static random access memory (SRAM), see Chapter 9, *Enhanced Memory Protection Unit*.

A reset mechanism is provided to reset the first 256K bytes of protected OCM SRAM with the help of the protected resource reset management (PRRM).

Figure 7-1. OCM Subsystem Overview



The direct memory access (DMA) controller and the microprocessor unit (MPU) have access to the SRAM/ROM through a DMA arbiter.

The OCM BOOT ROM is used only for the boot code.

7.2 OCM Subsystem Integration

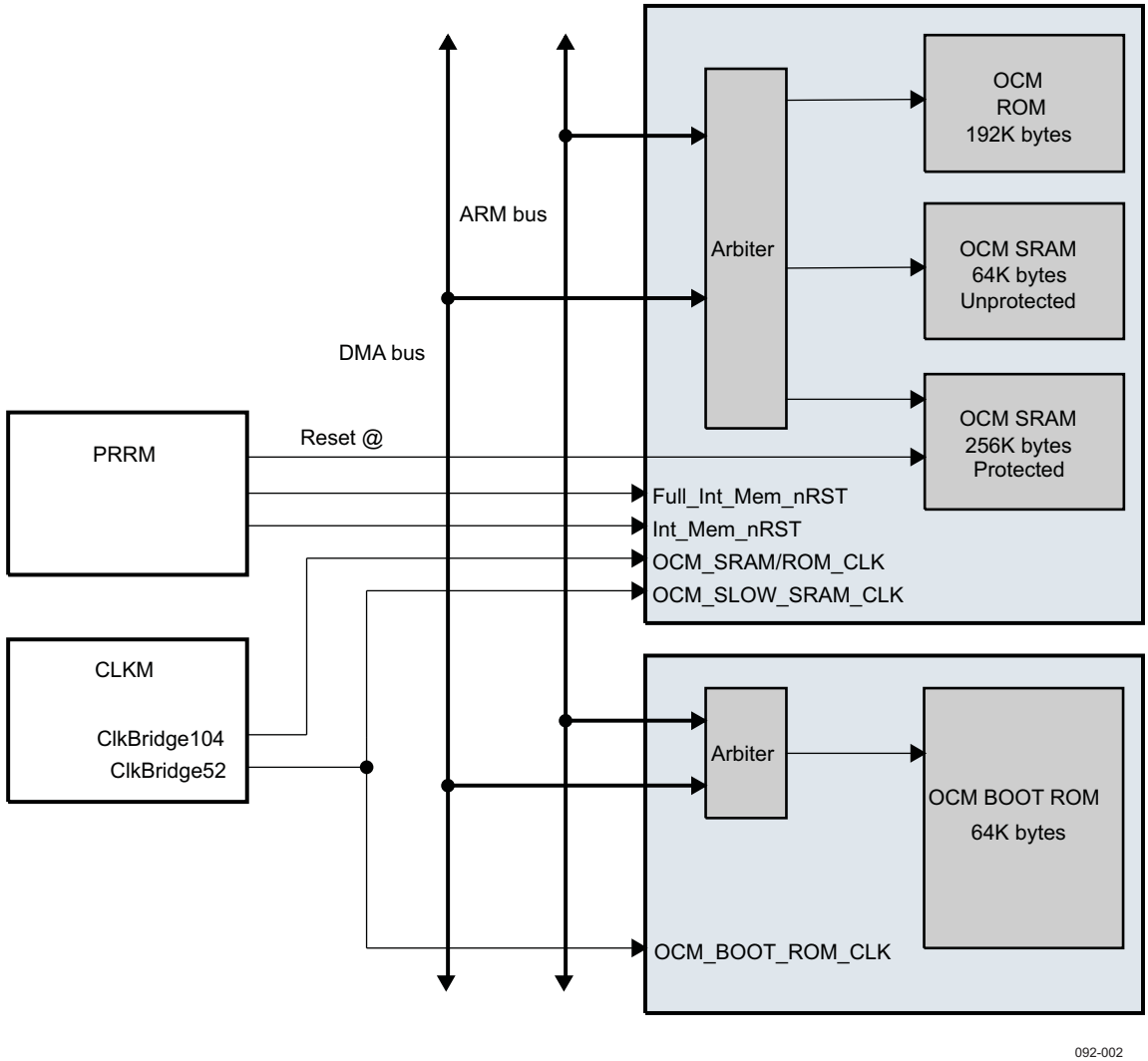
7.2.1 Description

The OCM SRAM allows data and program memory for the MPU and direct memory access through the DMA controller.

The OCM ROM is used primarily to contain a MIDI engine, down-loadable sound, JPEG, etc.

The OCM BOOT ROM is used only to store the boot code.

Figure 7-2. OCM Subsystem Integration to the LOCOSTO Device



Note: These signals are internal signals only; they are not accessible on pins.

7.2.2 Clocking, Reset, and Power-Management Scheme

7.2.2.1 Clocking

Table 7-1 describes the clocks used by the OCM.

Table 7-1. OCM Clock

Type	Name	Source	Typical Frequency	Description
Interface	OCM_SRAM/ROM_CLK	ClkBridge104	104 MHz	N/A
Functional	OCM_SRAM/ROM_CLK	ClkBridge104	104 MHz	N/A
Interface	OCM_SLOW_ SRAM_CLK	ClkBridge52	52 MHz	Used by first 256K bytes of protected SRAM during reset
Functional	OCM_SLOW_ SRAM_CLK	ClkBridge52	52 MHz	Used by first 256K bytes of protected SRAM during reset
Interface	OCM_BOOT_ ROM_CLK	ClkBridge52	52 MHz	N/A
Functional	OCM_BOOT_ ROM_CLK	ClkBridge52	52 MHz	N/A

For details, see Chapter 6, *Power, Reset, and Clock Management*.

7.2.2.1.1 OCM SRAM and OCM ROM

The interface clock OCM_SRAM/ROM_CLK comes from the ClkBridge104 clock, which is controlled by the clock manager (CLKM), and belongs to the bridge 104-MHz clock domain. The clock is also used as a functional clock for the OCM SRAM and OCM ROM modules.

A functional clock (OCM_SLOW_SRAM_CLK) is also provided for reset purposes. This clock comes from the ClkBridge52 clock, which is controlled by the CLKM, and belongs to the bridge 52-MHz clock domain. It is used only when the OCM SRAM is in reset mode.

The OCM SRAM and OCM ROM modules perform automatic internal clock gating when the memory is not accessed by the system. There is no extra latency in which the clock must be switched on following an idle state. The clock to the memory is dynamically gated.

7.2.2.1.2 OCM BOOT ROM

The interface clock OCM_BOOT_ROM_CLK comes from the ClkBridge52 clock, which is controlled by the CLKM, and belongs to the bridge 52-MHz clock domain; therefore, the OCM BOOT ROM is executed at 52 MHz. This clock is also used as a functional clock for the OCM BOOT ROM module.

The OCM BOOT ROM module performs automatic internal clock gating when the memory is not accessed by the system. There is no extra latency in which the clock must be switched on following an idle state. The clock to the memory is dynamically gated.

7.2.2.2 Reset

Table 7-2 lists the resets used by the OCM.

Table 7-2. OCM Reset

Type	Name	Source	Activation	Domain
Hardware/software	Full_Int_ Mem_nRST	PRRM_CNTL_REG (PRRM register)	1	First 256K bytes of SRAM
Software	Int_Mem_nRST	PRRM_CNTL_REG (PRRM register)	1	Address range within the first 256K bytes (defined into PRRM registers

7.2.2.2.1 OCM SRAM and OCM ROM

All resets are performed with a slow functional clock (bridge 52 MHz) running at half the speed of the normal clock.

Full reset of the module is done by activating the Full_Int_Mem_nRST signal in the PRRM module. A full reset fills the 256K bytes of the protected memory with 0. The full reset performs a 32-byte initialization per clock cycle (8K cycles for the full reset).

A partial reset of the module is done by activating the Int_Mem_nRST signal and by configuring the RESET_INT_MEM_START_ADDR and RESET_INT_ MEM_END_ADDR register in the PRRM module (see Chapter 6, *Power, Reset, and Clock Management*). RESET_INT_MEM_START_ADDR and RESET_INT_MEM_END_ADDR belong to the 256K-byte protected SRAM when set by users. The partial reset performs a 4-byte initialization per cycle.

A full reset is done on a hard reset.

Table 7-3. Comparison Between Full and Partial Reset for OCM SRAM

Reset	Full Reset	Partial Reset
Register source	PRRM_CNTL_REG FULL_RST_EN (bit 5)	PRRM_CNTL_REG SOFT_INT_RST (bit 1)
Signal source	Full_Int_Mem_nRST	Int_Mem_nRST
Reset range	First 256K bytes (protected memory)	From RESET_INT_MEM_START_ ADDR to RESET_INT_MEM_ END_ADDR (address included in protected memory)
Bytes reset per clock cycle	32	4
Reset frequency bridge	Bridge 52 MHz	Bridge 52 MHz
Hard/soft reset	Hard/soft reset	Soft reset only

During a reset (full or partial) access is still granted, but with the following limitations:

- 256K bytes of protected SRAM are not available (from ARM or DMA). On an access attempt, the PRRM_ NWAIT_INT signals are routed by the PRRM to the memory controller unit (MCU) or the DMA controller accordingly.
- 64K bytes of nonprotected SRAM are available with a normal clock (bridge 104 MHz).
- 192K bytes of ROM are fully available with a normal clock (bridge 104 MHz).

No reset is performed following a device deep sleep mode.

The OCM_ROM module is not affected by a reset.

7.2.2.2.2 OCM BOOT ROM

There is no reset for the OCM BOOT ROM, but the executed code at reset is read on the OCM BOOT ROM.

7.2.2.3 Power Domain

7.2.2.3.1 OCM SRAM and OCM ROM

Power for the OCM SRAM and OCM ROM is supplied by the CORE power domain (see Chapter 6, *Power, Reset, and Clock Management*).

When the device enters deep sleep mode, the on-chip memory goes into retention state. The contents of the memory are not lost.

7.2.2.3.2 OCM BOOT ROM

Power for the OCM BOOT ROM is supplied by the CORE power domain.

7.3 OCM Subsystem Functional Description

7.3.1 OCM Memory Mapping

Table 7-4 describes the mapping of the on-chip memory.

Table 7-4. OCM Memory Mapping

Device Name	Start Address	End Address	Size	Data Access	BUM Chapter
Boot Area 0x0000 0000 - 0x003F FFFF					
Boot ROM	0x0000 0000	0x0000 FFFF	64K bytes	8/16/32 bits R	7
Reserved	0x0001 0000	0x003F FFFF			
External Memory 0x000 0000 - 0x17FF FFFF					
External memory	0x0040 0000	0x07FF FFFF	124M bytes	8/16/32 bits R/W	11
Internal Memory 0x0800 0000 - 0x08FF FFFF					
Protected RAM	0x0800 0000	0x0803 FFFF	256K bytes	8/16/32 bits R/W	7
Nonprotected RAM	0x0804 0000	0x0804 FFFF	64K bytes	8/16/32 bits R/W	7
ROM	0x0805 0000	0x0807 FFFF	192K bytes	8/16/32 bits R	7
Reserved	0x0808 0000	0x08FF FFFF	15.5M bytes		
Internal Memory-like Peripherals 0x0900 0000 - 0x0FFF FFFF					
Boot ROM	0x0900 0000	0x0900 FFFF	64K bytes	8/16/32 bits R	7
Reserved	0x0901 0000	0x096F FFFF			
Camera	0x0970 0000	0x097F FFFF	1M byte	32 bits R/W	17
SHA1/MD5	0x0980 0000	0x098F FFFF	1M byte	32 bits R/W	15
DES/3DES	0x0990 0000	0x099F FFFF	1M byte	32 bits R/W	15
RNG	0x09A0 0000	0x09AF FFFF	1M byte	32 bits R/W	15
TI Reserved	0x09B0 0000	0x09CF FFFF			
NAND flash	0x09D0 0000	0x09DF FFFF	1M byte	8/16/32 bits R/W	10
MSSPI	0x09E0 0000	0x09EF FFFF	1M byte	32 bits R/W	26
Debug unit	0x09F0 0000	0x09FF FFFF	1M byte	32 bits R/W	8
Reserved	0x0A00 0000	0x0FFF FFFF			

7.3.2 OCM SRAM

The OCM SRAM contains 320K bytes of memory and has the following characteristics:

- The first 256K bytes can be protected by using the EMPU (see Chapter 9, *Enhanced Memory Protection Unit*). The EMPU can configure up to four protected regions with a maximum size of 64K bytes each and a granularity of 4 bytes.
- The last 64K bytes cannot be protected via the EMPU.
- Can be used as RAM by the MPU or the DMA controller
- Supports read and write mode in 1, 2, or 4 bytes
- Operates at full bus clock frequency
- Supports one cycle read and write

Note: The protected memory is not a secure device; therefore, it has nothing to do with the LOCOSTO device security features.

7.3.3 OCM ROM

The OCM ROM is used primarily for storing a MIDI engine, including down-loadable sound. The 192K-byte embedded ROM has the following characteristics:

- Is always accessible, even during reset mode
- Can be used by the MPU or the system DMA
- Supports read mode in 1, 2, or 4 bytes
- Operates at full bus clock frequency even in reset mode
- Supports one cycle read
- Generates an abort on a write

7.3.4 OCM BOOT ROM

The OCM BOOT ROM is used for boot purposes (see the ROM Code Reference Manual). The OCM BOOT ROM content cannot be read/dumped from the MPU or DMA. Any access to the secure-ROM subroutine area from outside the secure-ROM is denied, thereby returning data to 0 and generating a security violation to the EMPU for abort generation (the DMA violation is not flagged).

Debug Unit

This chapter describes the debug unit for the LOCOSTO device.

Topic **Page**

8.1 Debug Unit Overview

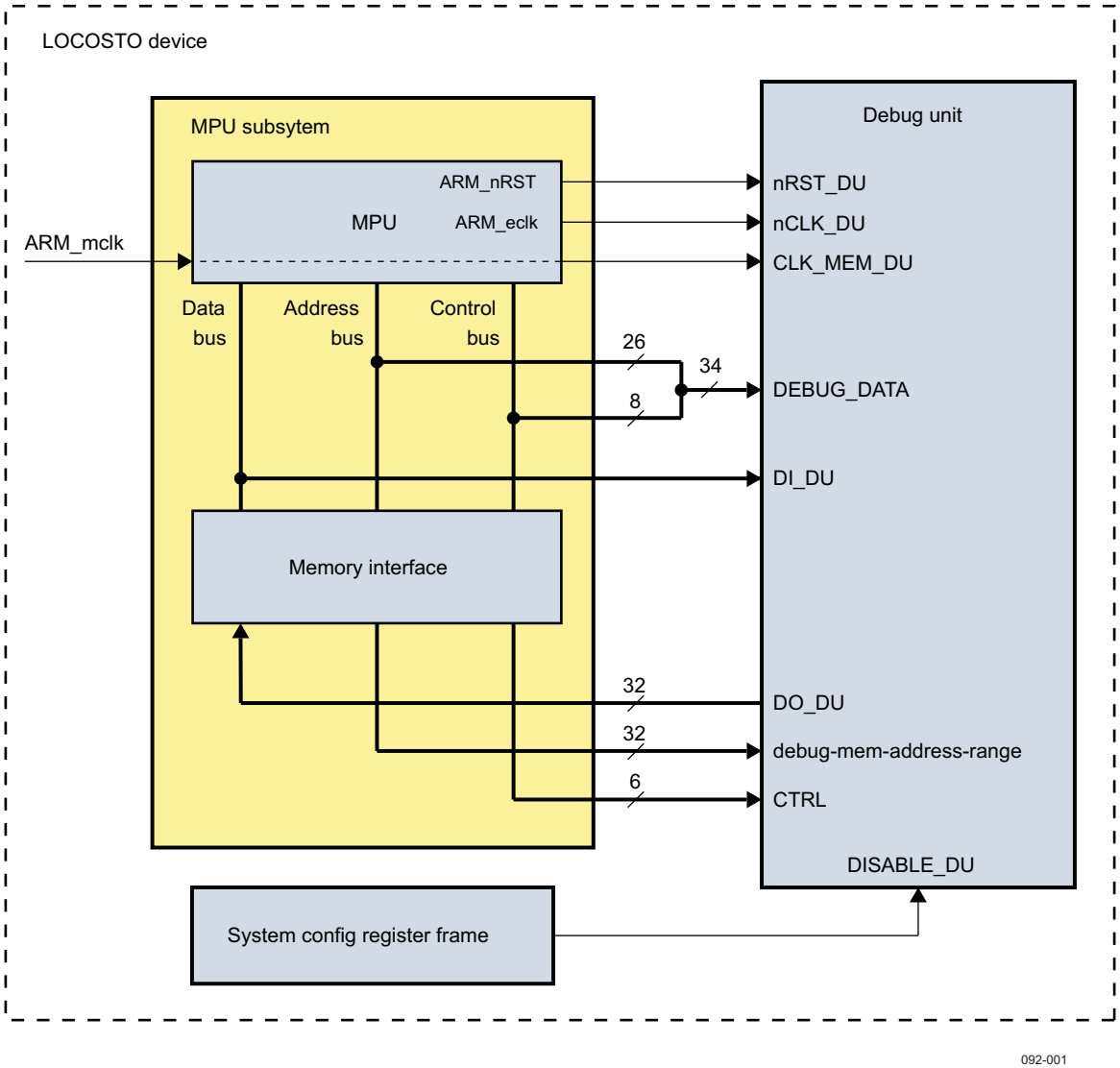
The debug unit is a hardware resource that provides additional support to a software abort-handler. The unit provides a 64-stage deep history table of the last memory accesses. The stored debug data containing the previous transaction can then be analyzed when the unit is disabled or in abort mode.

Note: In addition to the debug unit feature, the LOCOSTO device has a debug module that allows visibility on more than 270 LOCOSTO internal signals and is multiplexed to a debug bus (37 test ports) composed of 37 output pins.

For more information about the debug module, see the Debug Features application note (APN211), which also details use of the debug unit interpreting tool.

The debug unit is an autonomous function that needs no configuration. The unit is connected directly to the address and control buses of the processor from which it collects the data, and to the memory interface system where the saved history table can be read. Figure 8-1 shows a schematic overview of the debug unit.

Figure 8-1. Debug Unit Module Overview



8.1.1 Main Features

The debug unit has the following features:

- 64 32-bit words deep FIFO register file
- 26-bit processor address and 8 processor control signals recorded
- Continuous storage for every processor fetch (either instruction or data)
- Data record automatically frozen on the switch from normal to abort mode
- Enable/disable control from configuration register-bit
- General-purpose RAM when debug function is disabled or in abort mode

8.1.2 Signals and I/O Description

Table 8-1 lists the debug unit signals and I/O.

Table 8-1. Signals and I/O Description

Name	Function	Direction	Size
nCLK_DU	Clocking signal that controls the write of the debug data to the debug memory Rising edge: Address counter decrements Falling edge: Write memory	IN	1
nRST_DU	Resets the debug unit; clears the counter to 000000 Active low, release of reset state (nRST_DU rising edge) is expected synchronized to nCLK_DU falling edges	IN	1
DISABLE_DU	Disables the debug function and can be configured by the DU.BOOT_MODE_CONF[11] programmable register bit (DISABLE_DU bit) High: Debug function is disabled and the debug memory acts as a SRAM. Low: Debug function is enabled. This signal can be asynchronously asserted.	IN	1
DI_DU	Input SRAM data; valid only when the debug unit is disabled or frozen (debug/abort).	IN	32
DO_DU	Debug memory output data bus; valid only when the debug unit is disabled or frozen (debug/abort). When the debug unit is enabled, any access to the debug memory always returns the physical memory location 0x00, whatever the address pointed.	IN	32
CLK_MEM_DU	Clocking signal that controls the write of the DI_DU data bus (debug unit disabled or frozen) to the debug memory Rising edge: Write memory	IN	1
nWBE_DU	Selects which bytes (among the 32-bit word) are written. Active low and valid only when the debug unit is disabled or frozen (SRAM); expected active on CLK_MEM_DU falling edge.	IN	4
MAS_DU	Input debug data; connects the MPU signal MAS[1:0] (for more information, see Section 8.3.2, *Debug Data Organization*).	IN	2
nOPC_DU	Input debug data; connects the MPU signal nOPC (for more information, see Section 8.3.2, *Debug Data Organization*).	IN	1
nMREQ_DU	Input debug data; connects the MPU signal nMREQ (for more information, see Section 8.3.2, *Debug Data Organization*).	IN	1

Table 8-1. Signals and I/O Description (continued)

Name	Function	Direction	Size
nRW_DU	Input debug data; connects the MPU signal nRW (for more information, see Section 8.3.2, *Debug Data Organization*). This signal also controls the read/write function of the debug memory when set as a SRAM (debug unit disabled or frozen). A low level initiates a read sequence, and a high level sets a write. This signal is expected active during the high phase of the CLK_MEM_DU and is internally registered on the CLK_MEM_DU falling-edge.	IN	1
nEXEC_DU	Input debug data; connects the MPU signal nEXEC (for more information, see Section 8.3.2, *Debug Data Organization*).	IN	4
nM_DU	Input debug data; connects the MPU signal nM[3:0]] (for more information, see Section 8.3.2, *Debug Data Organization*).	IN	28
A_DU	Input debug data; connects the MPU signal A[27:0] (for more information, see Section 1.3.2, Debug Data Organization). The address bits from 7 down to 2 (A[7:2]) are also used as the debug memory address when the debug unit is disabled or frozen.	IN	28

8.2 Debug Unit Module Integration

The debug unit is implemented as a stand-alone module reading the MPU buses and not participating in the data processing. The information, which is saved in an internal SRAM of the debug unit, can be read when the DISABLE_DU bit DU.BOOT_MODE_CONF[11] is high. This is the only configuration bit.

For more information on the activation of the DU, see Section 8.4: *Debug Unit Programming Model*.

8.2.1 Clocking, Reset, and Power-Management Scheme

8.2.1.1 Clocks

The debug unit controller has two clocks, CLK_MEM_DU and nCLK_DU, driven by the MPU subsystem (see Table 8-2).

Table 8-2. Debug Unit Clocks

Type	Name	Source	Frequency	Description
Functional clock	CLK_MEM_DU	MPU subsystem	104 MHz	Clocking signal that controls the write of the DI_DU data bus (debug unit disabled or frozen) into the debug memory
Functional clock	nCLK_DU	MPU subsystem	104 MHz	Clocking signal that controls the write of the debug data (debug unit enabled) into the debug memory

8.2.2 Resets

The ARM_nRST signal of the MPU provides a hardware reset (see Table 8-3).

Table 8-3. Debug Unit Reset

Type	Name	Source	Activation	Description
Hardware	ARM_nRST	CLKM	0	Clears the address counter to 0

8.2.3 Power Management

If the application does not require such a debug facility or you want to save power consumption during critical application phases, the debug unit can be disabled by asserting the DISABLE_DU bit to 1 (for more information, see Chapter 6, *Power, Reset, and Clock Management*).

8.3 Debug Unit Functional Description

8.3.1 Description

The debug unit hardware is a 64-word dual-port (separate read/write bus) memory with additional control that makes the last written address the read base-address.

Each input word is 32 bits wide, which allows storage of 26 address bits and 6 additional bits that provide information about the processor operating state. The read-data bus is output in a 32-bit format.

- The only configuration to set up is the DISABLE_DU bit DU.BOOT_MODE_CONF[11], which allows the debug unit function to be enabled (DISABLE_DU bit = 0) or disabled (DISABLE_DU bit = 1).

Note: It is recommended that the debug unit be enabled at reset to record the first processing events.

- When the debug unit is enabled and for as long as the abort mode is not detected, the data input onto the debug data bus is partially encoded and then synchronously stacked into the 64-word register file .
- A 64-state rollover counter generates the write addresses for each access to the system memory, with the counter decrements pointing to the debug memory location where the bus transaction data is written.
- The rising-edge of the nCLK_DU input signal controls the counter decrement, and the falling edge of the nCLK_DU input signal controls the debug memory synchronous write.
- When the counter rolls over, debug memory locations sequentially update, thus overriding the debug memory content that causes the loss of previously stored data.

8.3.1.1 Collecting Debug Data

The debug unit continuously collects data until one of the following events occurs:

- The processor ABORT mode is entered (this is decoded when the MPU output signals nM[3] and nM[2] are at the value 1 and 0, respectively).
- The DISABLE_DU bit is asserted high (see Section 14.2.1.4.1, *Debug Unit Disable/Enable Control*).
- The MPU debug state is activated.

8.3.1.2 Abort Mode

Entering abort mode locks the automatic-write and freezes the debug memory content. The debug unit acts as a standard SRAM block where the data previously collected is available to the software abort-handler to work out the abort cause. The abort mode freezes the counter value, which points to the last write-addressed location.

- The counter value is used as offset. It is added to the address value supplied by the MPU to generate the read or write addresses in such a way that a read at the location 0x00 (address from MPU) returns the last data written (0x00 + write-add-pointer). A read to the location 0x01 returns the data before the last data written, and so on (see Table 8-8).

- The debug memory data-input bus is switched to the memory-interface output-data bus.
- The write function (write-enable, write-byte, and write-clock) is controlled from the MPU.
- The debug memory data-output bus is valid and can be read from the MPU.

CAUTION

The debug memory should be read only when the debug unit is disabled or abort mode is active. Reading the debug unit while collecting data returns an undefined value. The write from the processor is disabled and has no effect when the debug unit is collecting data.

When the abort mode is released (after the abort-handler is processed), the DU re-starts recording data; the 64-state rollover counter starts counting from where it stopped.

Note: The 64-state rollover counter value is initialized at 000000 on reset (nRST_DU low). This is the only way to set the counter value.

8.3.1.3 Internal Memory-Like Peripherals Mode

When the debug unit is disabled or in abort mode, it acts as an internal memory-like peripheral. Table 8-4 lists the memory mapping in this mode.

Table 8-4. Debug Unit Memory Mapping

Device Name	Start Address	Stop Address	Size	Unit	Data Access	Device Access
Debut unit	09F0:0000	09FF:FFFF	1.0	MB	32 RW	32

8.3.2 Debug Data Organization

The debug data is collected from the processor address and control buses. This debug data is 34 bits wide before being partially encoded to fit the debug memory (32 bits).

Table 8-5 lists the input signals and the coded debug data.

Table 8-5. Input Signals and Coded Debug Data Description

Name	Description
MAS_DU[1:0] (coded to a single bit named Byte)	MPU MAS_DU[1:0] definition: 00: Byte 10: Word 01: Half-word 11: Reserved Debug data byte definition: 0: Non-Byte 1: Byte
nOPC_DU, nRW_DU, nMREQ_DU (coded to Access-Type[1:0])	nOPC_DU:OP-Code fetch indicator, nRW_DU: not read/write, and nMREQ_DU: not memory request; these three signals are coded and give a 2-bit MPU access-type data named Access-Type MPU nOPC_DU definition: 0: Fetching instruction 1: Not fetching instruction MPU nRW_DU definition: 0: Read

Table 8-5. Input Signals and Coded Debug Data Description (continued)

Name	Description
	1: Write MPU nMREQ_DU definition: 0: Memory request 1: Not memory request Debug data access-type definition: 00: Read instruction 01: CPU internal cycle 10: Read data 11: Write data
nEXEC_DU	EXECuted instruction indicator; this signal is directly recorded. MPU nEXEC_DU definition: 0: Instruction being executed 1: Instruction not being executed
nM_DU[3:0]	MPU processor operating mode; nM_DU[1:0] are directly recorded; nM_DU[3:2] are not recorded but used to detect the processor ABORT. MPU nM_DU[1:0] definition: 00: Supervisor/abort/undefined/system 01: IRQ 10: FIQ 11: User
A_DU[27:0]	MPU processor address; among the 28-bit address bus, the 25-MSB signals (A[27:3]) are directly recorded. A[2] and A[1] are not recorded. When the Non-Byte is flagged (bit Byte = 0), the address LSBIT A_DU [0] is ignored, the MAS_DU [1] signal is recorded instead to indicate if the current access is an half-word (16-bit) or word (32-bit) type.

After the encoding, the debug data is a 32-bit word organized as described in Table 8-6.

Table 8-6. Debug Data Bit Organization

Bits	Name	Description
31	Byte	0: Non-byte 1: Byte
30:29	Acc-Typ	00: Read instruction 01: CPU internal cycle 10: Read data 11: Write data
28	nEXEC_DU	0: Instruction being executed 1:Instruction not being executed
27:26	nM_DU[1:0]	00: Supervisor/abort/undefined/system 01: IRQ 10: FIQ 11: User
25:1	A_DU[27-3]	25 MSB address bits
0	A_DU[0]/MAS[1]	If the Non-byte (bit byte = 0) MAS_DU [1] Else A_DU [0]

Note: A[2] and A[1] are not recorded and A[0] is recorded only if byte = 1, so the debug data contains only the 25-MSB address bits that give an address range.
If byte = 0, MAS[1] is recorded instead of A[0]. If MAS[1] = 1, it is a word access (32 bits); if MAS[1] = 0, it is a half-word access (16 bits). It is then possible to decrease the address range.

8.3.3 Examples of Recorded Data

Table 8-7 lists different examples of debug data words.

Table 8-7. Debug Unit Memory Mapping

Byte	Access Type	nEXEC	nM[1:0]	A[27-3]	A[0]/ MAS[1]	Description
1	11	0	11	0101 0101 0101 0101 0101 0101 0	1 (A[0])	BYTE DATA WRITE accesses address between 0x5555551 and 0x5555557 while the processor is in USER mode and effectively EXECUTES the instruction in the execute pipe stage.
0	00	0	10	1101 0101 0101 0101 0101 0101 0	1 (MAS[1])	WORD(32-bit) INSTRUCTION FETCH (32-BIS) accesses address between 0xD555550 and 0xD555554 while the processor is in FIQ mode and effectively EXECUTES the instruction in the execute pipe stage.
0	10	1	01	0001 0101 0101 0101 0101 0101 1	0 (MAS[1])	HALF-WORD(16-bit) DATA READ accesses address between 0x1555558 and 0x155555E while the processor is in IRQ mode and does NOT EXECUTE the instruction in the execute pipe stage.

Table 8-8 lists how the decrement counter points to the debug memory location where the bus transaction data is written and lists the read address of the corresponding debug data word.

Table 8-8. Debug Unit Memory Map

Generated Write Addresses	Bits						Read Addresses
	31 Byte	30:29 Acc-Type	28 nEXEC_DU	27:26 nM_DU[3:2]	25:1 A_DU[27-3]	0 A_DU[0]/ MAS[1]	
Counter							0x00
Counter + 1							0x04
Counter + 2	Transaction/time –2						0x08
Counter + 3	Transaction/time –3						0x0C
Counter + 4	Transaction/time –4						0x10
…	…						…
Counter + 62	Transaction/time –62						0xF8
Counter + 63	Transaction /time –63						0xFC

8.4 Debug Unit Programming Model

8.4.1 Debug Unit Enable/Disable Control

8.4.1.1 Enabling the Debug Unit

The debug unit must be enabled at reset by setting the DISABLE_DU bit to 0. The debug unit collects data until the abort mode is entered (see Section 8.5.2, *Register Description*).

If the application does not require such a debug facility or to save power consumption during critical application phases, the debug unit can be disabled by asserting the DISABLE_DU bit to 1. When the debug unit is disabled, the debug memory can be used as SRAM.

8.4.1.2 Disabling the Debug Unit

The debug unit can be disabled by setting the DISABLE_DU bit to 1.

Disabling the debug unit accomplishes the following:

- Locks the automatic-write mechanism
- Sets the debug memory write control to the memory interface (data, address, write-enable, write-byte, and write-clock signals)
- Freezes the 64-state rollover whenever the DISABLE_DU bit is 1. This nullifies the offset pointer-value effect and lets the MPU address bus directly control the read/write location.

Re-enabling the debug unit restarts the automatic-write mechanism and makes the debug unit start collecting data again.

Note: Using the debug unit with an interpreting tool is detailed in the *Debug Features* application note (APN211).

8.5 Debug Unit Registers

Table 8-9. Instance Summary

Module Name	Base Address	Size
BOOT_MODE_CONF	0xFFFF FB30	2 bytes

8.5.1 Module Register Mapping Summary

Table 8-10. Boot Mode Configuration Register Offset Address

Register Name	Type	Register Width (Bits	Offset
BOOT_MODE_CONF	RW	16	0x0

8.5.2 Register Description

Table 8-11. BOOT_MODE_CONF

Address Offset	0x0		
Physical address	0XFFFF FB30	**Instance**	BOOT_MODE_CONF
Description	Boot mode configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DISABLE_DU	Reserved										

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Write has no effect; read returns 0x0.	R	0x0
11	DISABLE_DU	0: Enable debug unit capture 1: Disable debug unit capture	RW	0x0
10:0	Reserved	Write has no effect; read returns 0x100.	R	0x100

Enhanced Memory Protection Unit

This chapter describes the enhanced memory protection unit (EMPU) of the LOCOSTO device.

Topic **Page**

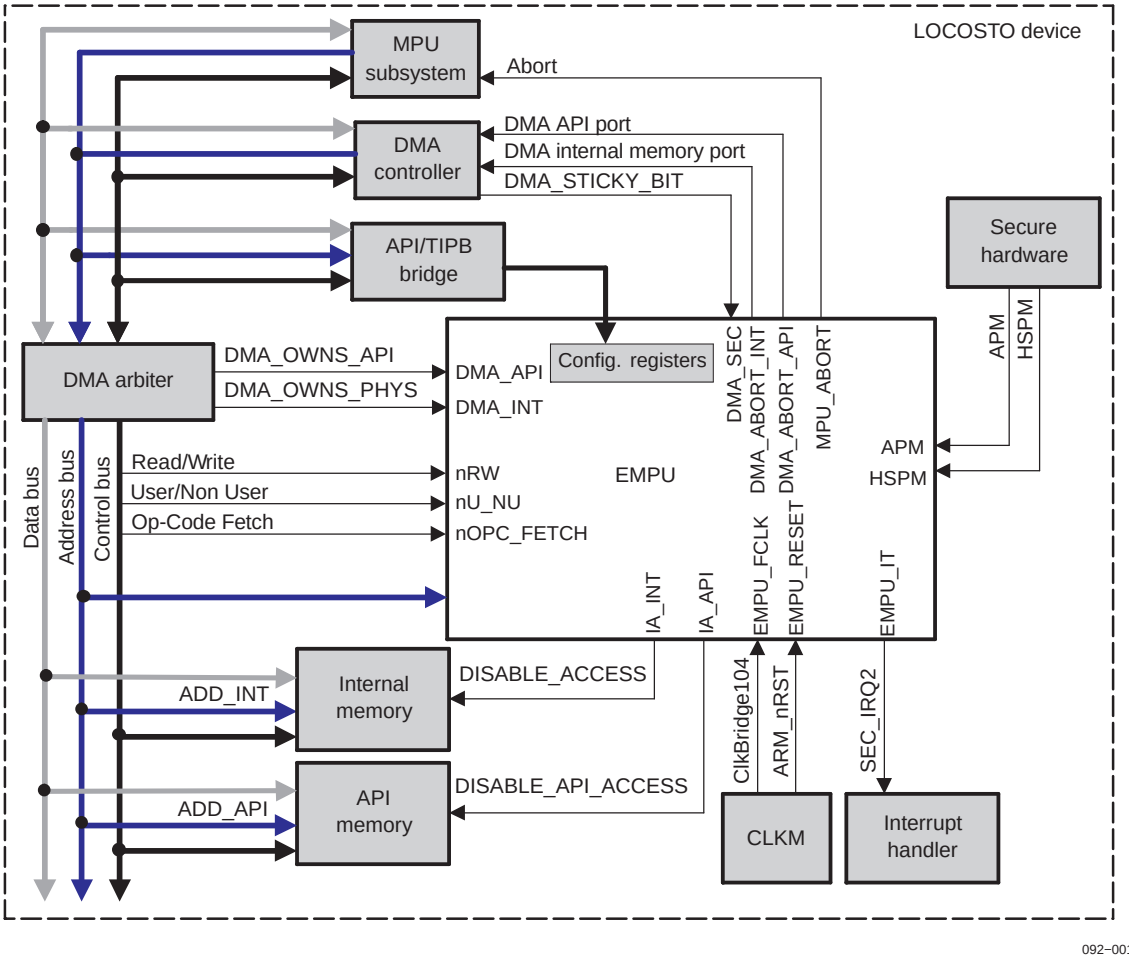
9.1 EMPU Overview

The enhanced memory protection unit (EMPU) of the LOCOSTO device protects against illegal access to the internal memory and the application programming interface (API). The external memory access protection is handled by the external memory interface (EMIF) module. For more information, see Chapter 11, *EMIF*.

The EMPU defines read/write access protection to memory sections within the global memory space. It validates access rights for all memory access requests. Any illegal access is blocked and the MPU is informed. This protection scheme allows definition of the program instructions, the system data, and the user data subsections in the memory space.

Figure 9-1 shows an overview of the EMPU.

Figure 9-1. EMPU Overview



The EMPU is configured by the application program via the TI peripheral bus (TIPB). The address bus from the processor is directly monitored by the EMPU. The access requests by the MPU and the direct memory access (DMA) are monitored by the EMPU; when a violation of the access rights is detected, the EMPU aborts the access and signals the source (that is, the MPU or the DMA).

Any attempt to breach the memory protection affects the illegal-access internal memory (IA_INT) and illegal-access API memory (IA_API) of the memory control signal, which block the write or read/write operation on the protected memory. The EMPU can generate an interrupt (EMPU_IT) to signal the interrupt handler. For more information, see Chapter 12, *Interrupt Handlers*.

9.1.1 Features

The EMPU module allows access protection on two different memory spaces shared between the MPU and the DMA. These memory spaces are the internal memory space and the API memory space.

The access protection control features for each of the two memory spaces are as follows:

- Internal memory
  - Up to four programmable protected regions within a maximum memory space of 256K bytes
  - Minimum granularity of 4 bytes on the internal protected memory region
  - 64K-byte maximum region size (256K bytes for the four protected regions). The last 64K bytes of internal memory are not under EMPU protection.
  - A privileged-code memory region giving privileged rights to the opcode of that region to access specific protected memory region for read/write operation
  - Configurable protection modes (nonuser read/write, user read only, ROM, privileged, etc.) for each protected memory subregion
  - Configurable base, start and end address for each protected region
- API
  - A configurable boundary between the API protected regions, splitting the memory space into two subregions
  - Minimum granularity of 1 byte on API protected memory region
  - Configurable protection modes (read protect, write protect, read/write protect) for each subregion
  - Protection enable/disable control through the active protection mode (APM) and halted/suspended protection mode (HSPM) control signals
  - The protection enable bit is “write once” after reset and is activated according to the external memory chip-select defined in the control register. This feature allows protecting an external memory in any chipselect configuration.

Note: In the LOCOSTO device, the external memory protection is handled by the EMIF module. For more information on external memory protection, see Chapter 11, *EMIF*.

9.1.2 Signal Description

- EMPU input signals
  - The nRW, nU_NU, and nOPC_FETCH signals indicate the type of current memory access request by the MPU. The nRW indicates read/write operation, the nU_NU indicates user/nonuser execution mode, and the nOPC_FETCH is an opcode fetch signal.
  - The APM and HSPM signals are controlled by the secure hardware and affect the protection scheme of the API memory.
  - The DMA_SEC, DMA_API and DMA_INT signals indicate the DMA memory access is using a secure channel to access the API or the internal memory space, respectively.
- EMPU-generated signals
  - The IA_INT and IA_API signals are asserted high when an illegal access is detected on the internal or API protected memory space. They block the current write or read/write access to the memory.
  - The MPU_ABORT signal is asserted high when an illegal memory access or an external source abort signal is detected. The MPU_ABORT signal informs the MPU of an illegal access error.
  - The EMPU_IT signal is asserted low when an illegal memory access or an external source abort signal is detected. EMPU_IT signals the illegal access to the secure interrupt controller.
  - The DMA_ABORT_INT and the DMA_ABORT_API signals are asserted high when an illegal memory access is made by the DMA on the internal memory or the API memory. They connect to the internal memory port and the API memory port of the DMA.

9.2 EMPU Integration

9.2.1 Clocking and Reset Management

9.2.1.1 Clock

The EMPU module is in the bridge104-MHz clock domain:

- The EMPU module operates from a fixed functional 104-MHz clock generated by DPLL through the CLKM module. For more information, see Chapter 6, *Power, Reset, and Clock Management*.
- The 104-MHz clock is fed into the module to provide the reference clocks for the register and timing setup (see Table 9-1).

Table 9-1. EMPU Clock

Type	Name	Source	Frequency	Description
Functional	EMPU_ FCLK	ClkBridge104	104 MHz	Functional clock for synchronization

9.2.1.2 Reset

The EMPU module receives one reset signal from the CLKM:

- Hardware reset of the EMPU module can be performed by activating the ARM_nRESET signal (for more information, see Chapter 6, *Power, Reset, and Clock Management*). Table 9-2 describes the global reset signal of the EMPU.

Table 9-2. EMPU Reset

Type	Name	Source	Activation	Domain
Hardware	EMPU_ RESET	ARM_nRESET	0	Global reset on configuration and status registers

At EMPU_RESET, the protection mode registers, control registers, and base, start, and end address registers are cleared to 0. The status register of the EMPU is also cleared to 0. Thus, all protection settings are disabled on reset.

9.2.2 Interrupt Request

Table 9-3 describes the EMPU interrupt.

Table 9-3. EMPU Interrupt

Type	Name	Source	Frequency	Description
Interrupt	EMPU_ IT	EMPU	Interrupt handler	Informs the MPU of a breach in the memory protection scheme as a result of an illegal access.

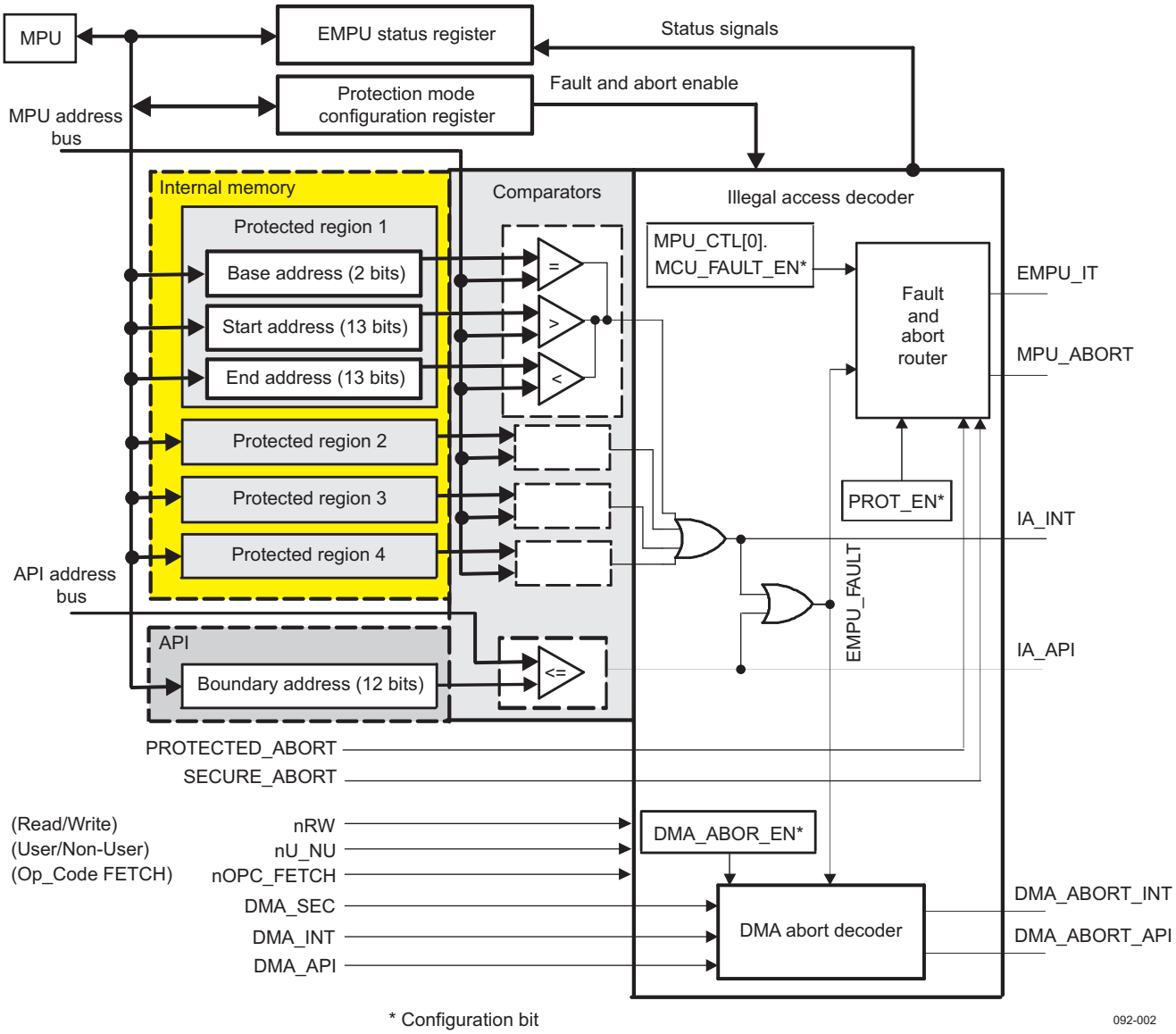
For more information, see Chapter 12, *Interrupt Handlers*.

9.3 EMPU Functional Description

9.3.1 Block Diagram

Figure 9-2 shows the functional block diagram of the EMPU module.

Figure 9-2. EMPU Block Diagram



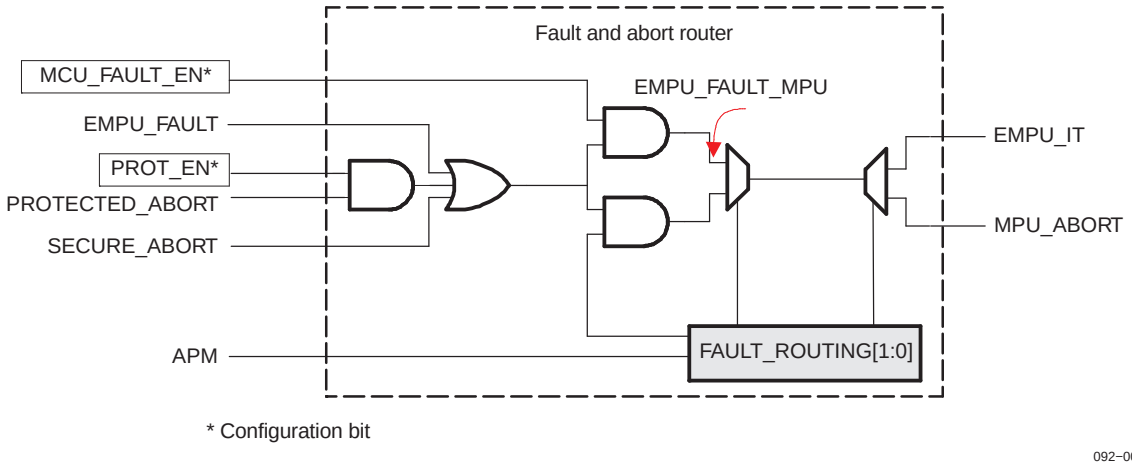
The EMPU module is essentially composed of bus comparators and a logic decoder to detect illegal access to the protected internal and API memory regions. The associated programmable register frames of the EMPU are configured to set the access protection scheme according to the application needs.

The EMPU provides real-time monitoring of the address bus and the control signals. It generates illegal memory access signals when a memory protection breach is detected. The error signal is sent to the internal status register, and the MPU is also informed of the breach. The illegal memory access is aborted, and the memory control signals disable write or read/write of memory according to the selected protection mode. The MPU can read the EMPU status register EMPU.MPU_ST to determine the cause of the latest memory protection breach and take the necessary action.

9.3.2 Fault and Abort Routing

Figure 9-3 shows the fault and abort router.

Figure 9-3. Fault and Abort Router



The fault and abort routing network gives user-configurable routing control over the fault signal (see Figure 9-3). The fault signal can be routed either to the MPU subsystem or to the secure interrupt handler.

The EMPU_FAULT generated by the illegal access decoder network is gated together with the external PROTECTED_ABORT (abort signal from protected resources) and the SECURE_ABORT (abort signal from secure resources). The EMPU.MPU_CTL[6] PROT_EN bit allows gating of the PROTECTED ABORT signal. The combined abort signal is also gateable through the EMPU.MPU_CTL[0] MCU_FAULT_EN bit.

The APM signal allows to bypass the fault enable control (EMPU.MPU_CTL[0] MCU_FAULT_EN bit) and generate the fault signal. The fault signal is routed according to the EMPU.MPU_CTL[8:7] FAULT_ROUTING field to the MPU_ABORT or EMPU_IT (secure interrupt handler). Table 9-4 lists the configurable routing schemes.

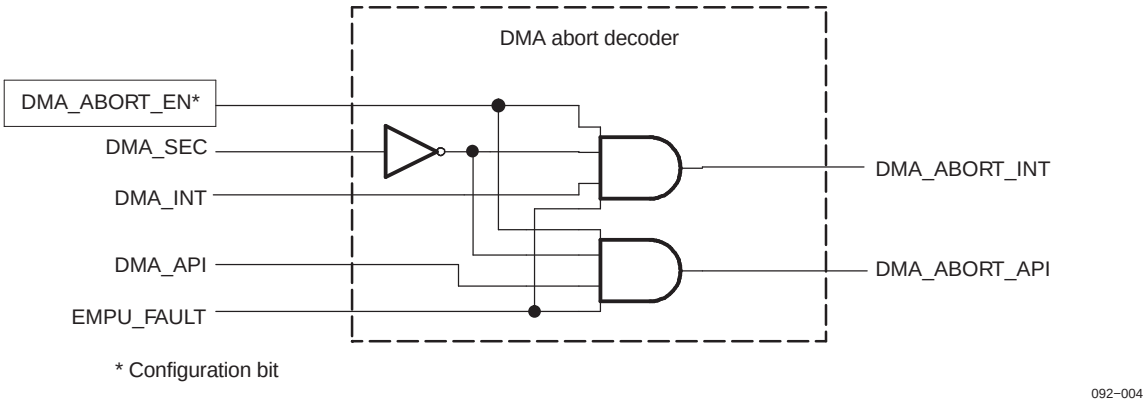
Table 9-4. Fault Routing

FAULT_ROUTING 1	FAULT_ROUTING 0	Fault Routing
0	0	Disabled
0	1	MPU_ABORT output
1	0	EMPU_IT output
1	1	Reserved

9.3.3 DMA and Secure DMA Channel

Figure 9-4 shows the DMA abort decoder.

Figure 9-4. DMA Abort Decoder



The EMPU detects any illegal access operation (EMPU_FAULT) by the DMA over the internal and API memory space by monitoring the DMA_INT and DMA_API signals (see Figure 9-4). It generates the corresponding DMA_ABORT_INT or DMA_ABORT_API signal to indicate the protection breach by the DMA. The DMA abort signals are gated by the EMPU.MPU_ CTL[5] DMA_ABORT_EN bit.

If the DMA_SEC signal (which indicates that the DMA channel accessing the memory is secure) is active, the EMPU disables its protection on the memory space accessed by the DMA. When the channel is insecure the memory protection remains active.

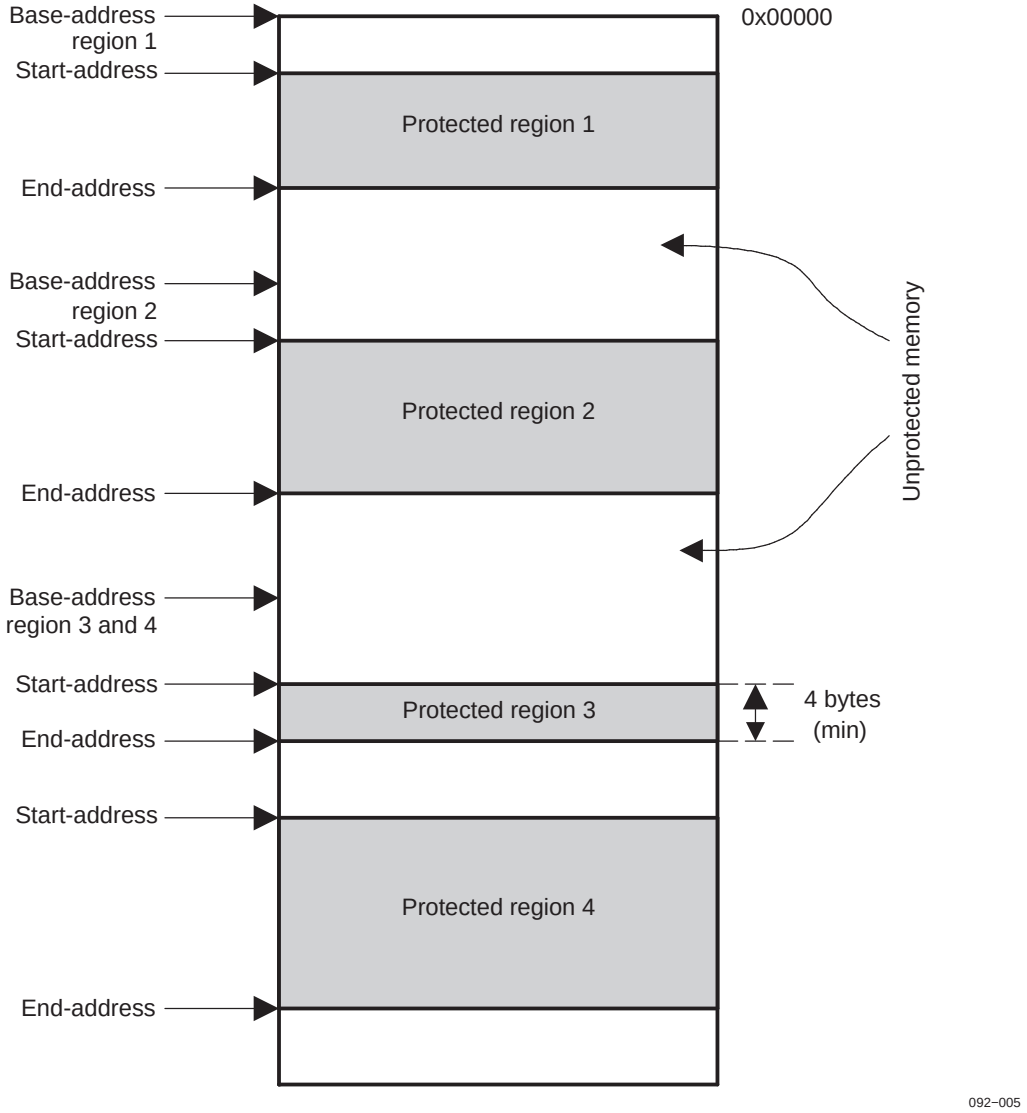
9.3.4 Memory Protection Schemes

9.3.4.1 Internal Memory Protected Regions

The EMPU can divide the internal memory space into four protected regions. Each internal protected memory region is defined by three bit fields (the base, start, and end address bit fields). The base and the start address are configured in the EMPU.MPU_BSTn[14:13] BASEn field and the EMPU.MPU_ BSTn[12:0] STARTn field, respectively. The end address is configured in the EMPU.MPU_ENDn[12:0] ENDn field, where n represents an internal protected memory region instance and varies from 1 to 4.

Any memory section outside the start and the end address of the four protected regions is the unprotected memory area. Figure 9-5 shows an example of the internal memory protection scheme.

Figure 9-5. Internal Memory Protection



The base address of two or more protected regions can be the same. The boundaries of each protected region are defined by its start and end address (offset from its base address). The size of a protected region in the internal memory can vary from 4 bytes to 64K bytes.

9.3.4.2 Internal Memory Protection Modes

Each of the four possible protected memory regions can have a different protection mode. The protection mode is configured by setting the corresponding EMPU.MPU_PM[11:0] PMn bit field, where n corresponds to one of the four protected regions and varies from 1 to 4.

The EMPU recognizes the application execution modes and uses this identification to build the protection scheme. The following are some of the modes that can be detected:

- User mode: Most application programs run in user mode.
- Nonuser mode: The interrupts and exceptions are serviced in nonuser mode. The protected resources are also accessed in this mode.
- Privileged-region mode: This operation mode is specific to the EMPU architecture. The EMPU allows to define the privileged memory regions so that all operational codes in that region have the right to access (read/write) a given memory region.

Table 9-5 summarizes the configurable internal memory protection modes.

Table 9-5. Protection Mode Settings for Internal Memory

	PMn2	PMn1	PMn0	Protection Mode
	0	0	0	Protection disabled: Nonuser/user read/write
	0	0	1	Nonuser read/write, user read only
	0	1	0	Reserved
	0	1	1	ROM: Nonuser/user read only
Privileged modes	1	0	0	Writes authorized only when performed from opcode fetched from protected region 1.
	1	0	1	Writes authorized only when performed from opcode fetched from protected region 1 or region 2.
	1	1	0	Reads and writes authorized only when performed from opcode fetched from protected region 1.
	1	1	1	Reads and writes authorized only when performed from opcode fetched from protected region 1 or region 2.

Note: *n* in PMn stands for an instance of the protected internal memory region and varies from 1 to 4.

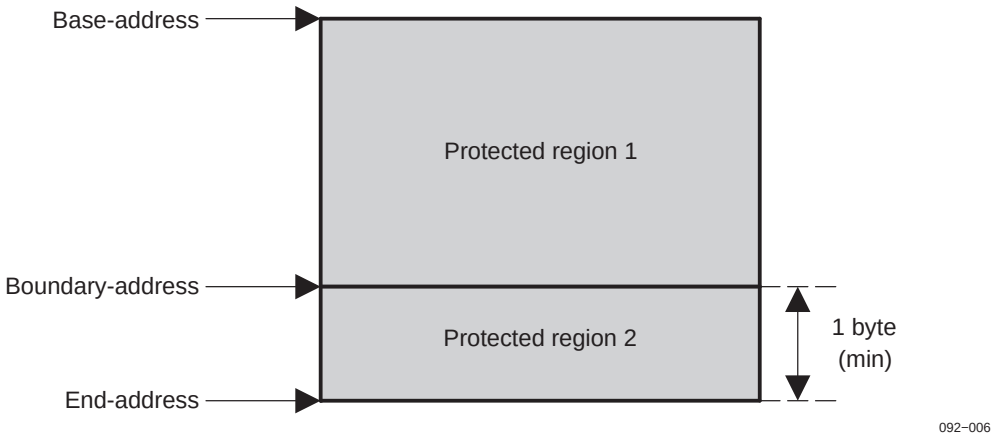
Note: Protected area bounding limits must be set before enabling the protection violation fault signal EMPU_FAULT_MPU to the MPU by setting the EMPU.MCU_CTL[0].MCU_FAULT_EN bit.

9.3.4.3 API Protected Regions

The EMPU can split the API memory space into two independent protected memory regions, each with its own protection mode (see Figure 9-6). The address of the boundary separating the two protected regions is configured in the EMPU.MPU_API_BOUND register. The first region starts from the base address of the API memory up to the boundary address, and the second region starts at the boundary address +1 and goes up the end of the API memory. The API memory protection is active only when the HSPM signal is active.

The minimum granularity of the protected region in the API memory is 1 byte.

Figure 9-6. API Memory Protection



When the APM signal is low, the protection mode of region 1 is enabled and that of region 2 is disabled. Similarly, when the APM signal is high, the protection mode of region 1 is disabled and that of region 2 is enabled.

9.3.4.4 API Protection Modes

Different protection modes can be configured for each protected region of the API memory space. The protection mode for protected region 1 and region 2 are configured in the EMPU.MPU_PM_API[3:2] PM_API2 and EMPU.MPU_ PM_API[1:0] PM_API1 fields.

Table 9-6 summarizes the configurable API protection modes.

Table 9-6. Protection Mode Settings for API

PM_APin[1]	PM_APin[0]	Protection Mode
0	0	Protection disabled: Read/write
0	1	Write only. Read access is not authorized.
1	0	Read only. Write access is not authorized.
1	1	Read and write access are not authorized.

Note: *n* in PM_APIn stands for an instance of the API protected memory region and varies between 1 and 2.

Note: Protection is enabled in region 1 and disabled in region 2 when APM = 0.

Protection is disabled in region 1 and enabled in region 2 when APM = 1

For information on external memory protection, see Chapter 11, *EMIF*.

9.4 EMPU Programming Model

To configure and enable the internal and API memory protection, the corresponding register bits must be set. The following sections describe the configuration and enabling of the memory protection mode for the internal and the API memory. The configuration of the abort signal routing is also discussed.

9.4.1 Internal Memory Protection Configuration and Enabling

To configure and enable internal memory protection, follow this procedure:

1. Set the bounding limits:
   a. The base and start address of the corresponding internal memory protected region are assigned to the EMPU.MPU_BSTn register.
   b. The end address of the corresponding internal memory protected region is assigned to EMPU.MPU_ENDn register.
2. Set the protection mode: The protection mode selection bits EMPU.MPU_PM[11:0] PMn bit field are set according to the mode settings in Table 9-6.
3. Reset the status register bits: Write to clear the EMPU.MPU_ST[3:0] IA_Rn bits.
4. Enable abort signal to MPU and DMA:
   a. The abort signals to the DMA are enabled by setting the EMPU.MPU_CTL [5] DMA_ABORT_EN bit.
   b. The abort signal to the MPU is enabled by setting the EMPU.MPU_CTL [0] MCU_FAULT_EN bit.

Where *n* corresponds to one of the four protected regions of internal memory and varies from 1 to 4.

9.4.2 API Memory Protection Configuration and Enabling

To configure and enable the API memory protection, follow this procedure:

1. Set the bounding limit: The boundary address, which separates protected region 1 from protected region 2, is assigned to the EMPU.MPU_API_BOUND register.
2. Set the protection mode: The protection mode selection bits in the EMPU.MPU_PM_API[3:0] PM_APIn bit field are set according to the mode settings in Table 9-6.

3. Reset the status register bits: Write to clear the EMPU.MPU_ST[10:9] IA_APIn bits.
4. Enable the abort signals to the MPU and DMA:
   a. The abort signals to the DMA are enabled by setting the EMPU.MPU_CTL[5] DMA_ABORT_EN bit.
   b. The abort signal to the MPU is enabled by setting the EMPU.MPU_CTL[0] MCU_FAULT_EN bit.

Where *n* corresponds to one of the two protected regions of the API memory and varies from 1 to 2.

9.4.3 Abort Signal Routing Configuration and Enabling

To configure and enable the abort signal routing, follow this procedure:

1. Route the abort signals to the MPU or to the secure interrupt controller by configuring the EMPU.MPU_CTL[8:7] FAULT_ROUTING bit field (see Table 9-4).
2. Enable abort signals from external protected sources by setting the EMPU.MPU_CTL[6] PROT_EN bit.

9.4.4 Status Read Operation

The status register of the EMPU saves the condition that generated the last illegal access abort operation. It can thus provide useful information to the application program regarding the source that initiated the exception.

- The EMPU sets the EMPU.MPU_ST[13] PROT bit/ the EMPU.MPU_ ST[11] NPROT bit to indicate an abort signal from external protected/unprotected resources.
- The EMPU.MPU_ST[12] SEC bit is set to indicate an abort signal from external secure resources.
- The EMPU.MPU_ST[10] IA_API1 bit or the EMPU.MPU_ST[9] IA_API2 bit indicates an illegal access on the API protected memory region 1 or region 2.
- The EMPU.MPU_ST[3] IA_R4 bit/ EMPU.MPU_ST[2] IA_R3 bit/ EMPU.MPU_ST[1] IA_R2 bit/ EMPU.MPU_ST[0] IA_R1 bit indicates an illegal access on the protected internal memory region 4/3/2/1.

9.5 EMPU Register Manual

Table 9-7 summarizes the EMPU instance.

Table 9-7. Instance Summary

Module Name	Base Address	Size
EMPU	0xFFFF FF00	256 bytes

9.5.1 Module Register Mapping Summary

Table 9-8 lists the EMPU register offset addresses.

Table 9-8. EMPU Register Offset Addresses

Register Name	Type	Register Width (Bits)	Offset
MPU_ST	R/C(1)	16	0x00
MPU_PM	R/W(2)	16	0x02
MPU_CTL	R/W(2)	16	0x08
MPU_BST1	R/W(2)	16	0x0A
MPU_END1	R/W(2)	16	0x0C
MPU_BST2	R/W(2)	16	0x0E
MPU_END2	R/W(2)	16	0x10
MPU_BST3	R/W(2)	16	0x12

(1) R = Read, C = Clear (write 0 only). Cleared only in nonuser mode if enabled.
(2) Write disabled only when HSPM is enabled.

Table 9-8. EMPU Register Offset Addresses (continued)

Register Name	Type	Register Width (Bits)	Offset
MPU_END3	R/W(2)	16	0x14
MPU_BST4	R/W(2)	16	0x16
MPU_END4	R/W(2)	16	0x18
MPU_PM_API	R/W(2)	16	0x20
MPU_API_BOUND	R/W(2)	16	0x22

9.5.2 Register Description

Table 9-9 through Table 9-21 describe the individual EMPU registers.

Table 9-9. MPU_ST

Address Offset	0x00		
Physical Address	0xFFFFFF00	**Instance**	EMPU
Description	This register contains the status bits.		
Type	R/C		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		PROT	SEC	NPROT	IA_API1	IA_API2		Reserved				IA_R4	IA_R3	IA_R2	IA_R1

Bits	Field Name	Description	Type	Reset
15:14	Reserved	Reading these bits gives undefined values. Writing to them has no effect.	R/C	0x0
13	PROT	This read/clear-only bit set to 1 indicates an abort signal from a protected resource.	R/C	0x0
12	SEC	This read/clear-only bit set to 1 indicates an abort signal from a secure resource.	R/C	0x0
11	NPROT	This read/clear-only bit set to 1 indicates an abort signal from a nonprotected resource.	R/C	0x0
10	IA_API1	This read/clear-only bit set to 1 indicates an illegal access to the protected API region 1.	R/C	0x0
9	IA_API2	This read/clear-only bit set to 1 indicates an illegal access to the protected API region 2.	R/C	0x0
8:4	Reserved	Reading this bit gives undefined values. Writing to them has no effect.	R/C	0x00
3	IA_R4	This read/clear-only bit set to 1 indicates an illegal access to the internal memory protected region 4	R/C	0x0
2	IA_R3	This read/clear-only bit set to 1 indicates an illegal access to the internal memory protected region 3	R/C	0x0
1	IA_R2	This read/clear-only bit set to 1 indicates an illegal access to the internal memory protected region 2	R/C	0x0
0	IA_R1	This read/clear-only bit set to 1 indicates an illegal access to the internal memory protected region 1	R/C	0x0

Table 9-10. MPU_PM

Address Offset	0x02		
Physical Address	0xFFFFFF02	**Instance**	EMPU
Description	This register contains the protection mode register bits for the internal memory.		
Type	R/W		

Table 9-10. MPU_PM (continued)

Write Latency	Not relevant

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				PM4				PM3				PM2			
														PM1	

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0
11:9	PM4	Protection mode of region 4	R/W	0x0
8:6	PM3	Protection mode of region 3	R/W	0x0
5:3	PM2	Protection mode of region 2	R/W	0x0
2:0	PM1	Protection mode of region 1	R/W	0x0

See Table 9-5 for a detailed description of the protection modes and their associated EMPU.MPU_PM[11:0] PMn bit values (where n in PMn stands for an instance of the protected internal memory region and varies between 1 and 4).

Note: Protected area bounding limits must be set before defining the protection mode in the EMPU.MPU_PM[11:0] PMn field.

Table 9-11. MPU_CTL

Address Offset	0x08		
Physical Address	0xFFFFFF08	**Instance**	EMPU
Description	This register contains the fault routing, protected resources abort, DMA abort, and the MCU fault abort enable bits.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FAULT_ROUTING		PROT_EN	DMA_ABORT_EN	Reserved			MCU_FAULT_EN

Bits	Field Name	Description	Type	Reset
15:9	Reserved	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x00
8:7	FAULT_ROUTING	Routes the fault signal to the MPU abort output, to the secure interrupt handler, or neither. 00: Disabled 01: MPU_ABORT output 10: EMPU_IT output 11: Reserved **Note:** This routing function is valid only when the protected mode is set. Otherwise, the fault signal is routed to the MPU_ABORT output. At reset, the default value of the FAULT_ROUTING register is 01 (MPU_ABORT).	R/W	0x1
6	PROT_EN	Enables an abort signal to the MPU when an abort signal is received from protected resource	R/W	0x0
5	DMA_ABORT_EN	Enables the DMA abort signals to the DMA. (ABORT_DMA_INT and DMA_ABORT_API).	R/W	0x0

Bits	Field Name	Description	Type	Reset
		0: Memory protection is active. The status register is recording the EMPU events and remains available for reading. However, the fault signal is not signaled to the DMA (that is, an abort is not generated). 1: Any fault flagged in the status register signals the DMA, indicating the fault occurrence (that is, an abort is generated).		
4:1	Reserved	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x00
0	MCU_FAULT_EN	Enables the EMPU_FAULT_MPU signal to the MPU. 0: Memory protection is active. The status register is recording the EMPU events and remains available for reading. However, the signal EMPU_FAULT_MPU is locked to a low level not passing the fault indication to the processor (that is, an abort is not generated). 1: Any fault flagged into the status register initiates an EMPU_FAULT_MPU signal transition (high level) indicating the fault occurrence to the processor (that is, an abort is generated). **Note:** Before setting the MCU_FAULT_EN register bit the EMPU status register must be cleared. A fault, which did not generate an abort signal (while MCU_FAULT_EN was disabled), can be flagged in this register.	R/W	0x0

Note: Protected area bounding limits must be set before enabling the EMPU_FAULT signal to the MPU.

Table 9-12. MPU_BST1

Address Offset	0x0A		
Physical Address	0xFFFFFF0A	**Instance**	EMPU
Description	This register contains the base and the start address of protected internal memory region 1.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	BASE							START							

Bits	Field Name	Description	Type	Reset
15	Reserved	Reading this bit gives undefined values. Writing to it has no effect.	R/W	0x0
14:13	BASE	These read/write bits define the base address for the protected memory region 1.	R/W	0x0
12:0	START	These read/write bits define the start address for the protected memory region 1.	R/W	0x0000

Table 9-13. MPU_END1

Address Offset	0x0C		
Physical Address	0xFFFFFF0C	**Instance**	EMPU
Description	This register contains the end address of protected internal memory region 1.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			END1												

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reading these bit gives undefined values. Writing to them has no effect.	R/W	0x0
12:0	END1	These read/write bits define the end address for the protected memory region 1.	R/W	0x0000

Note: The end address must be greater than or equal to the start address.

Table 9-14. MPU_BST2

Address Offset	0x0E		
Physical Address	0xFFFFFF0E	**Instance**	EMPU
Description	This register contains the base and the start address of protected internal memory region 2.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	BASE2		START2												

Bits	Field Name	Description	Type	Reset
15	Reserved	Reading this bit gives undefined values. Writing to it has no effect.	R/W	0x0
14:13	BASE2	These read/write bits define the base address for the protected memory region 2.	R/W	0x0
12:0	START2	These read/write bits define the start address for the protected memory region 2.	R/W	0x0000

Table 9-15. MPU_END2

Address Offset	0x10		
Physical Address	0xFFFFFF10	**Instance**	EMPU
Description	This register contains the end address of protected internal memory region 2.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			END2												

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0
12:0	END2	These read/write bits define the end address for the protected memory region 2.	R/W	0x0000

Table 9-16. MPU_BST3

Address Offset	0x12		
Physical Address	0xFFFFFF12	**Instance**	EMPU
Description	This register contains the base and the start address of protected internal memory region 3.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	BASE3							START3							

Bits	Field Name	Description	Type	Reset
15	Reserved	Reading this bit gives undefined values. Writing to it has no effect.	R/W	0x0
14:13	BASE3	These read/write bits define the base address for the protected memory region 3.	R/W	0x0
12:0	START3	These read/write bits define the start address for the protected memory region 3.	R/W	0x0000

Table 9-17. MPU_END3

Address Offset	0x14		
Physical Address	0xFFFFFF14	**Instance**	EMPU
Description	This register contains the end address of protected internal memory region 3.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								END3							

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0
12:0	END3	These read/write bits define the end address for the protected memory region 3.	R/W	0x0000

Table 9-18. MPU_BST4

Address Offset	0x16		
Physical Address	0xFFFFFF16	**Instance**	EMPU
Description	This register contains the base and the start address of protected internal memory region 4.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	BASE4							START4							

Bits	Field Name	Description	Type	Reset
15	Reserved	Reading this bit gives undefined values. Writing to it has no effect.	R/W	0x0
14:13	BASE4	These read/write bits define the base address for the protected memory region 4.	R/W	0x0
12:0	START4	These read/write bits define the start address for the protected memory region 4.	R/W	0x0000

Table 9-19. MPU_END4

Address Offset	0x18		
Physical Address	0xFFFFFF18	**Instance**	EMPU
Description	This register contains the end address of protected internal memory region 4.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			END4												

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0
12:0	END4	These read/write bits define the end address for the protected memory region 4.	R/W	0x0000

Table 9-20. MPU_PM_API

Address Offset	0x20		
Physical Address	0xFFFFFF20	**Instance**	EMPU
Description	This register contains the protection mode register bits for the API.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PM_API2		PM_API1	

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x000
3:2	PM_API2	Protection mode of API region 2	R/W	0x0
1:0	PM_API1	Protection mode of API region 1	R/W	0x0

See Table 9-6 for a detailed description of the protection modes and their associated EMPU.MPU_PM_API[3:0] PM_APIn bit values (where n in PM_APIn stands for an instance of the protected API memory region and varies between 1 and 2).

Table 9-21. MPU_API_BOUND

Address Offset	0x22		
Physical Address	0xFFFFFF22	**Instance**	EMPU
Description	This register contains the API boundary address.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOUND															

Bits	Field Name	Description	Type	Reset
15:0	BOUND	Defines the boundary address between the two API protected regions	R/W	0x0000

NAND Flash Controller

This chapter introduces the NAND flash controller of the LOCOSTO device.

Topic **Page**

10.1 NAND Flash Controller Overview

The NAND flash controller interfaces the host processor and the NAND flash memory and allows the NAND flash EEPROM to be connected as an external mass storage facility.

The aim of this controller is to have a fully automatic transfer process from/to the NAND flash port. The interface implements an 8-bit parallel data bus (commands, addresses, and data are multiplexed) in addition to the control signals for selecting chip, writing/reading, command and address latching, and ready/busy status.

Figure 10-1 highlights the NAND flash controller.

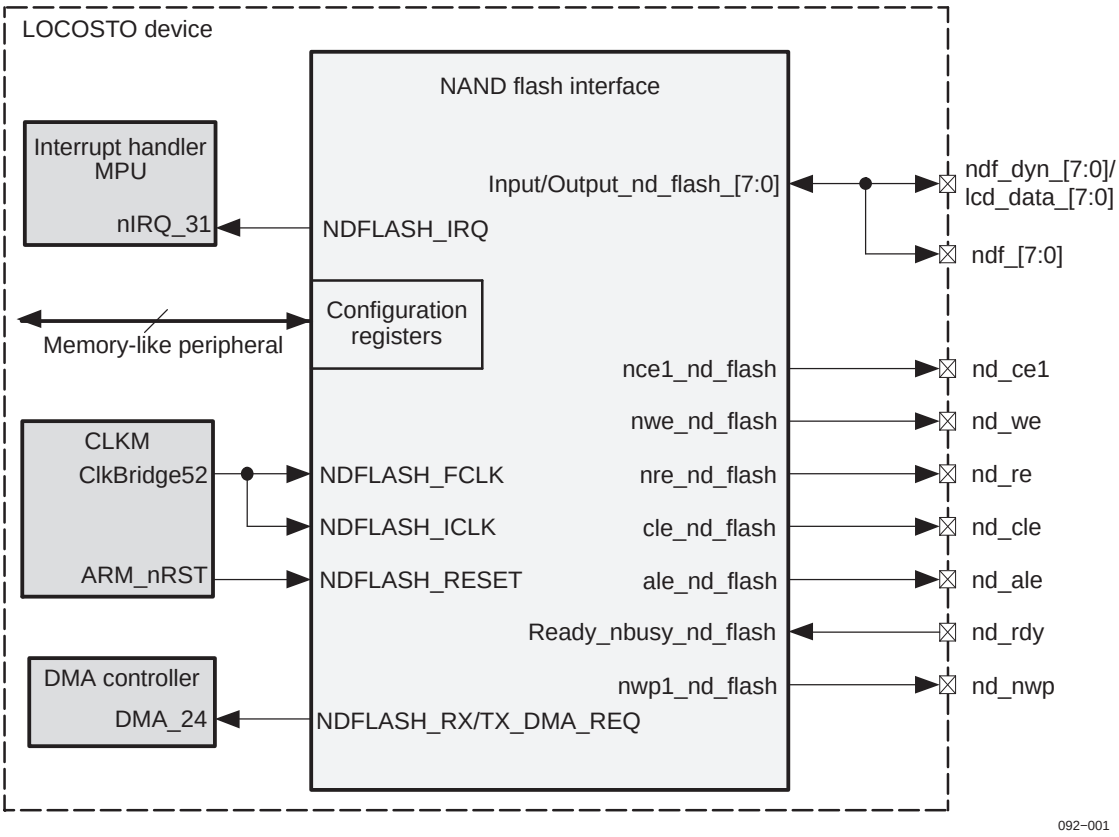


Figure 10-1. NAND Flash Controller Overview

Note: The NAND flash controller is multiplexed with the LCD interface in the default configuration. The NAND data bus (see ndf_[0:7] in Figure 10-1) is also accessible on other balls (the GPIO optional configuration) (see Chapter 18, *Configuration*, for detailed information about this optional NAND data bus access).

The NAND flash controller includes the following main features:

- 8-, 16-, and 32-bit interface from the host processor (mapped on the internal memory interface)
- Double page FIFO implementation (2*128 bytes)
- Programmable access time
- Interrupt and flag (polling operations) for end page or end transfer
- Fully automatic transfer process
- Programmable number of bytes for command, address and data
- Hardware computation support of error code correction (ECC) in the read and program mode
- Support of DMA transfer with DMA request

10.2 NAND Flash Controller Environment

Figure 10-2 shows the connection between the NAND flash controller module and an 8-bit external NAND memory device.

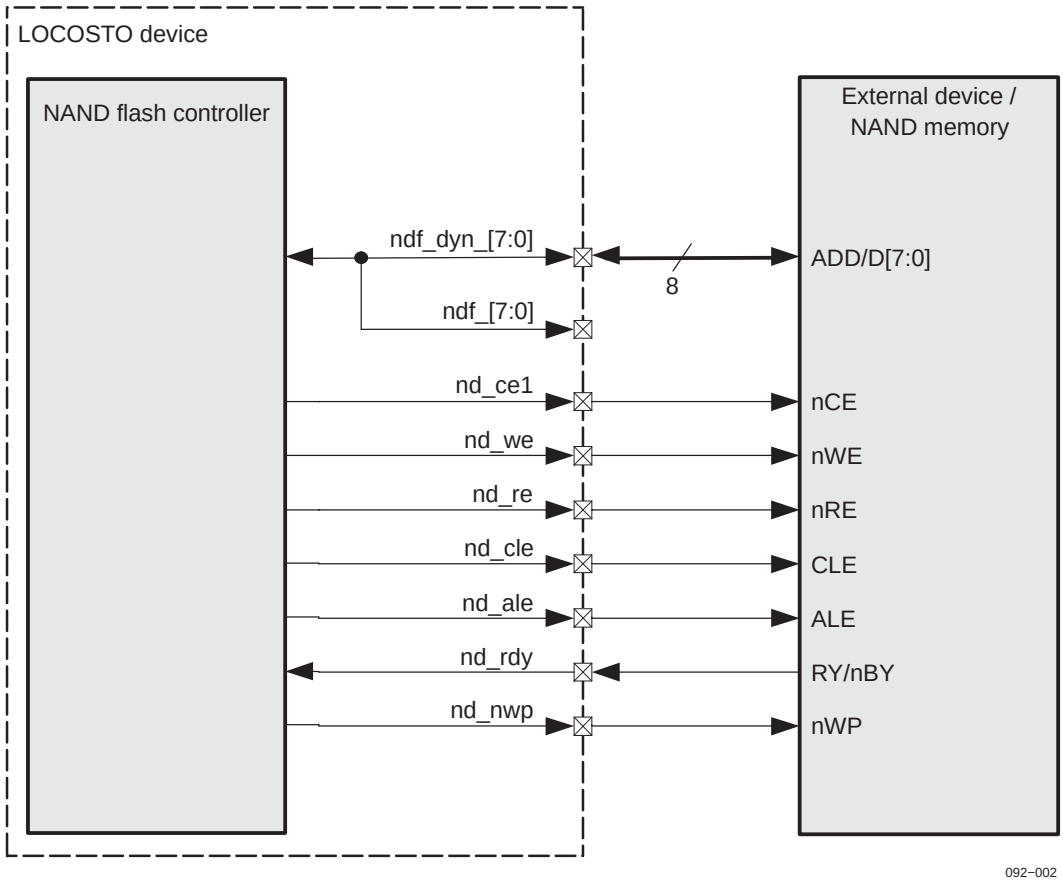


Figure 10-2. NAND Flash Controller to 8-bit Command/Address/Data Multiplexed Memory

Table 10-1 lists the NAND flash controller I/O pins.

Table 10-1. NAND Flash Controller I/O Description

Pin Name	I/O [1]	Description	Reset Value
ndf_dyn_[7:0]	I/O	NAND flash address/data bus	Output:0x00
ndf_[7:0]	I/O	NAND flash address/data bus (also accessible on other balls)	Output:0x00
nd_ce1	O	NAND flash chip enable	1
nd_we	O	NAND flash write enable	0
nd_re	O	NAND flash read enable	0
nd_cle	O	NAND flash command latch enable	0
nd_ale	O	NAND flash address latch enable	0
nd_rdy	I	NAND flash ready/busy signal	-
nd_nwp	O	NAND flash write protect signal	0

(1) I = input; O = output

10.3 NAND Flash Controller Integration

Figure 10-3 shows how the NAND flash module interacts with other modules embedded in the LOCOSTO device.

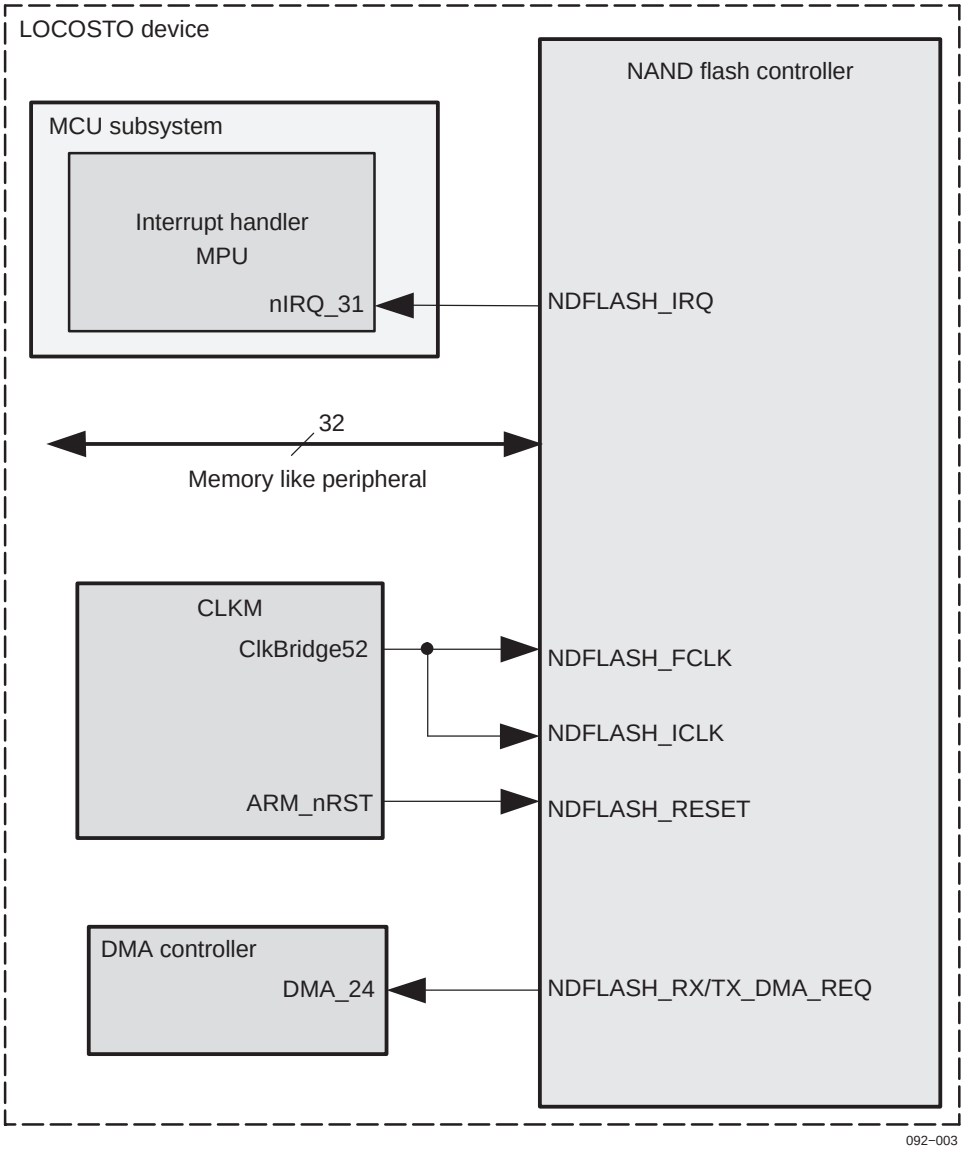


Figure 10-3. NAND Flash Controller Integration With the LOCOSTO Device

10.3.1 Clocking, Reset, and Power Management Scheme

10.3.1.1 Clocks

NDFLASH_FCLK comes internally from the CLKM module. This clock is both the functional and interface clock for the NAND flash controller module.

Table 10-2 lists the characteristics of the clocks used inside the module.

Table 10-2. Clocks

Type	Name	Source	Frequency	Description
Functional	NDFLASH_FCLK	ClkBridge52	52 MHz	n/a
Interface	NDFLASH_ICLK	ClkBridge52	52 MHz	n/a

10.3.1.2 Power Management

The NAND flash controller is part of the I/O power domain. To reduce power consumption, the NDFLASH_FCLK functional clock can be cut when entering the big sleep power mode. Thus, cutting the ClkBridge52 clock domain requires the following conditions:

- Set the CLKM.CNTL_CLK [1] BRIDGE_CLK_DIS bit to 0x1.
- Ensure that there are no pending operations on the bus.

10.3.1.3 Hardware and Software Reset

Table 10-3 describes the following NAND flash controller hardware and software resets.

- Global reset of the NAND flash controller is activated by the ARM_nRST signal (see Chapter 6, *Power, Reset, and Clock Management*, for more information).
- The NAND flash controller can be reset under software control using the NDFLASH.CONTROL_REG[0] SOFTWARE_RESET bit.

Table 10-3. Hardware and Software Resets

Type	Name	Source	Activation	Domain
Hardware	NDFLASH_RESET	ARM_nRST signal	0	Global reset
Software	SOFTRESET	CONTROL_REG (NDFLASH register)	1	Module reset

10.3.2 Hardware Requests

One DMA request goes from the NAND flash controller (NDFLASH_RX/ TX_DMA_REQ) to the DMA controller (DMA_24).

One interrupt request goes from the NAND flash controller (NDFLASH_IRQ) to the microcontroller unit (MCU) interrupt handler (nIRQ31).

10.4 NAND Flash Controller Functional Description

10.4.1 Block Diagram Description

The NAND flash controller provides a fully automatic transfer process, supporting various NAND flash memory sizes with different page sizes. The buffer can be accessed by either the CPU or the DMA.

The NAND flash controller is divided into the following eight main blocks (see also Figure 10-4):

- CPU port interface
- Prescaler
- Address generator
- Sequencer
- Double page buffer
- Interrupt generator
- ECC
- External NAND flash memory port interface

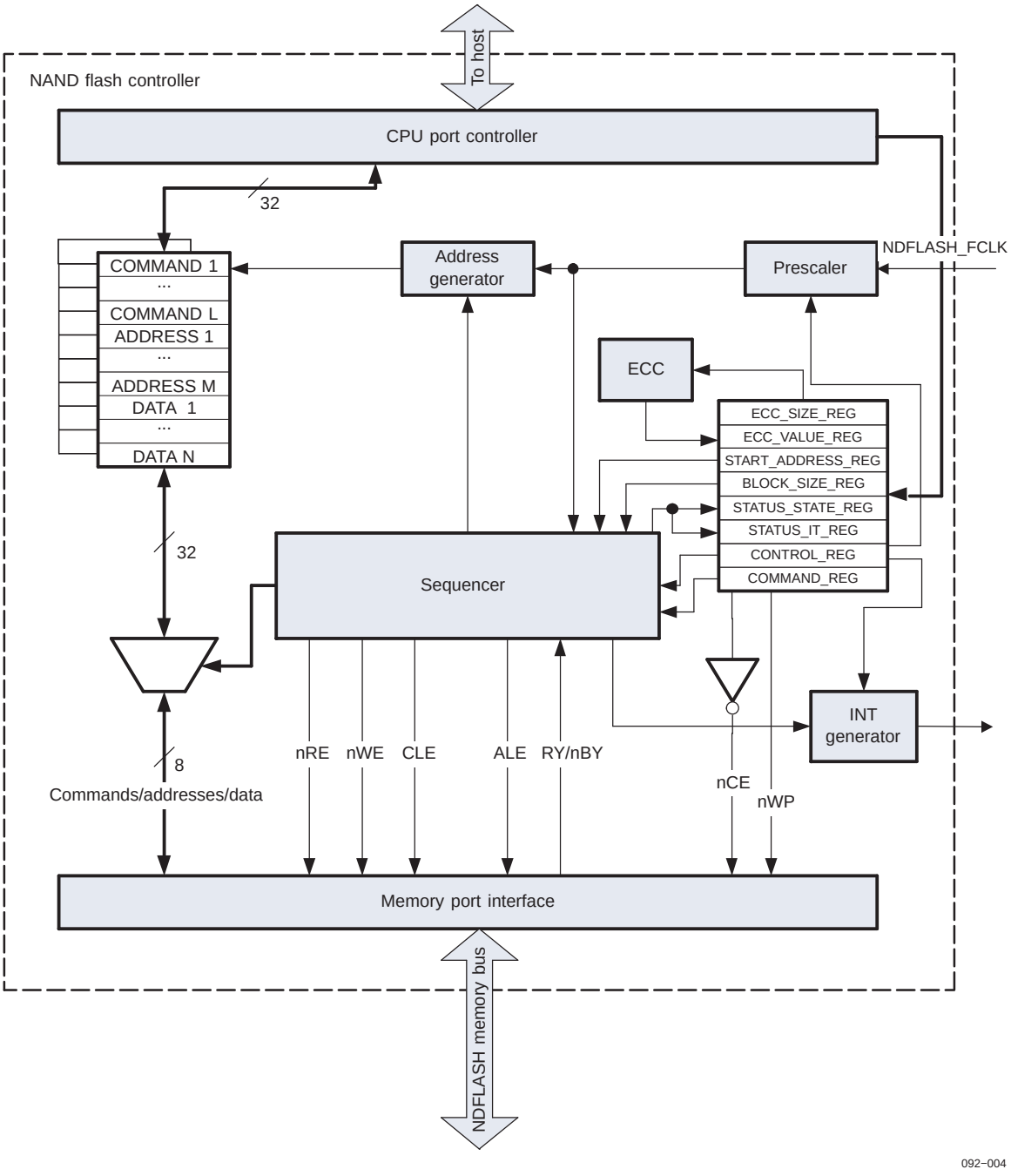


Figure 10-4. NAND Flash Controller Block Diagram

10.4.1.1 CPU Port Interface

The NAND flash controller is accessed by the CPU through an interface bus, as shown in Figure 10-4.

10.4.1.2 Prescaler

The prescaler has the functional 52-MHz clock as input and gives the clock to the sequencer and the address generator. The prescaler is programmed by the NDFLASH.CONTROL_REG [9:6] CLK_DIV field and proposes different division factors from 1 to 15.

The prescaler defines the access time to the NAND flash memory and is active only during a transfer.

10.4.1.3 Address Generator

The address generator is a 128-state counter managed by the sequencer to increment the address on the current page during a transfer. The address generator is cleared by sequencer each time the page is full and each time a new transfer starts.

On a read access to the NAND flash memory, the sequencer clears the address generator after first command(s) and address(es) are sent in case there is no second command. Otherwise, the sequencer clears the address generator after the first command(s), address(es) and second command(s) are sent. This is done to start recording data at the beginning of a page.

10.4.1.4 Sequencer

The sequencer is a state-machine clocked by the prescaler module, which provides control signals for the following modules.

- NAND flash port
- Address generator
- Page switch
- Interrupts generation

The sequencer starts when the processor sets the NDFLASH.COMMAND_ REG [0] ENABLE_TRANSFER bit to 0x1.

The sequencer controls all of the NAND flash port signals to read or write in the external NAND flash memory (except chip-enables, which are controlled by the software). The sequencer also controls the switching of the buffer page to give one for the processor and one for itself.

For example, the CPU fills in the first buffer page. When empty, the first buffer page is handled by the sequencer and the CPU continues to store data using the second buffer page.

Each time the sequencer completes a buffer page in read or in write and the CPU completes its transfer to another page, the sequencer updates the NDFLASH.STATUS_IT_REG status interrupt register and sends an interrupt request (if enabled) or a DMA request (if the DMA is used).

Note: The sequencer contains states with dummy cycles because the busy signal sent by the NAND flash memory can arrive after a long time. To avoid accessing the NAND flash memory during this wait time, the dummy cycles can be programmed in the NDFLASH.CONTROL_REG [13:10] DUMMY_CYCLE field.

10.4.1.5 Double Page Buffer

The double page buffer can be accessed simultaneously in read and write. When there is a write transfer to the NAND flash memory, the CPU writes to the buffer and the sequencer reads the buffer. When there is a read transfer to the NAND flash memory, the sequencer writes to the buffer and the CPU reads the buffer. This operation allows the CPU to be used at full speed and the sequencer to be used continuously.

The buffer is always accessed in bytes by the sequencer and is accessed in 8, 16, or 32 bits by the CPU.

Note: The buffer page must be read or written from low to high addresses because an access in read or write to the last byte address (128) of the buffer page indicates to the sequencer to switch the page.

10.4.1.6 Interrupt Management

The controller can interrupt the CPU and initiate a DMA transfer. Interrupts and the DMA can be enabled or disabled using the control register.

Table 10-4 describes the four events that can generate the interrupts.

Table 10-4. Interrupts Description

Interrupt Name	Interrupt Source	Interrupt Control (enabled/disabled)
it_end_page	End of page	NDFLASH.CONTROL_REG [1] EN_IT_PAGE bit
it_end_transfer	End of transfer	NDFLASH.CONTROL_REG [2] EN_IT_TRANSFER bit
it_end_ecc	End of ECC operation	NDFLASH.CONTROL_REG [15] EN_IT_ECC bit
it_end_busy	End of busy (only in program or erase; generated only if main clock is active)	NDFLASH.CONTROL_REG [16] EN_IT_BUSY bit

When the CPU receives an interrupt, it must read the NDFLASH.STATUS_ IT_REG status register to determine which event has occurred. This status interrupt register has four bits that indicate the end of a page, the end of a transfer, the end of ECC, and the end of busy.

One cycle after the read, this register is cleared along with the active interrupt. The interrupt is level.

It is possible to read two or three interrupt sources:

- End of page and end of ECC
- End of transfer and end of ECC
- End of transfer and end of busy
- End of page and end of transfer
- End of page, end of transfer, and end of ECC
- End of page, end of transfer, and end of busy

When using the DMA, the NDFLASH.CONTROL_REG [3] DMA_ENABLE bit must be set to 1 to send a DMA request at the end of a page (whether a read or a write operation) and at the end of a transfer in read. The DMA request is active at the low level during one prescaler period.

Note: The status interrupt register must always be read before enabling a new transfer to clear pending interrupts.

10.4.1.7 ECC Module

To protect data, the NAND flash controller provides ECC circuitry. The algorithm for the ECC, based on the Hamming code, allows error detection so that software can correct 1-bit errors.

To correct more than one bit per NAND flash memory page, set the NDFLASH.ECC_SIZE_REG register to indicate that the ECC must be calculated on the corresponding value.

For example, if a 512 data byte page NAND flash memory is used, and if value 128 is set in this register, the ECC will calculate each 128 bytes. This allows four bits to be corrected instead of one bit. The ECC calculated is available in the NDFLASH.ECC_VALUE_REG register.

The NDFLASH.ECC_VALUE_REG is a 32-bit register that allows an ECC to be calculated on an 8K-byte NAND flash memory page maximum. All bits are calculated when the ECC is enabled; thus, when the CPU reads the ECC value, it must mask bits based on the size of the ECC.

For example, if the ECC is calculated on 512 bytes, the CPU masks bits 12 to 15 and 28 to 31 when reading the ECC value.

When using the ECC, an ECC counter starts as soon as data are sent or received to or from NAND flash memory. The ECC is accumulated until the counter reaches the value stored in the NDFLASH.ECC_SIZE_REG[15:0] ECC_SIZE field. An ECC interrupt is then sent to the CPU (if enabled) and the status interrupt register is updated.

The sequencer stops as long as the CPU reads the NDFLASH.ECC_VALUE_ REG register. One CPU cycle after this read, the NDFLASH.ECC_VALUE_ REG register is auto-cleared, the ECC counter is reset, and the sequencer restarts. This process is repeated until the end of the transfer.

The ECC can be used in read or program operations.

To enable or disable the ECC in the control register, use the NDFLASH.CONTROL_ REG [14] ECC_ENABLE bit.

Note: The ECC value register is cleared each time a new transfer is started.

Note: When using the ECC for a sequential read with correctly set registers, an ECC interrupt is sent on the last byte of the data received (before receive spare). Then there are two choices:

- Disable the ECC before reading the ECC value so that no ECC is calculated on the spare, then poll the busy bit in the status state register and when busy, again enable the ECC so that the ECC is calculated on the next data. Repeat this process until the end of the transfer.
- Modify the ECC size register before reading the ECC value and set it to the spare size so that an ECC is calculated on the spare. When all data in the spare are received, an ECC interrupt is generated and the transfer is frozen. The ECC size register is then modified again to calculate the ECC on the data. Repeat this process until the end of the transfer.

10.4.1.8 External Memory Port Interface

The external port interface controls all related commands, addresses, data, and control signals required to interface the NAND flash controller-supported memories.

10.4.1.9 Ready/Busy Management

The RY/nBY ready/busy signal from the NAND flash memory (the nd_rdy pin on the LOCOSTO device side) indicates the status of the device. When low, the signal indicates that a program, erase, or random read operation is in process.

As the timing response of the signal can be superior to one cycle time, programmable dummy cycles (in the NDFLASH.CONTROL_REG [13:10] DUMMY_CYCLE register) are applied between the command and the data (when there is no address byte), between the addresses and the data (if there is no secondary command byte), after a secondary command, and when there is a switch of a page in NAND flash memory.

This last case depends on the change page frequency in bytes and the start address. Thus, for a read sequential transfer, the start address and block size of the registers must be programmed before the transfer.

10.5 Programming Model

10.5.1 Possible Sequences Sent by the NAND Flash Controller

The NAND flash controller can send the following transfers:

- Command(s)
- Command(s) and data
- Command(s), addresses and data
- Command(s), addresses and command(s)
- Command(s), addresses, command(s) and data
- Command(s), addresses, data and command(s) (in write only)
- Command(s), addresses, command(s), data and command(s) (in write only)
- Data
- Data, command(s) (in write only)

The controller allows one to three commands to be sent at the beginning, following the addresses and the data.

10.5.2 Operation Mode

Either the CPU or the DMA can transfer to/from the NAND flash memory.

To use the DMA, the CPU must set the NDFLASH.CONTROL_REG [3] DMA_ENABLE bit to 0x1 so that the DMA request is sent (DMA requests are not masked).

Section 10.5.3, *Write Transfer*, describes a write transfer and a read transfer without ECC and without a second and a third command.

10.5.3 Write Transfer

Either the CPU or the DMA can write in the NAND flash memory.

1. The CPU must configure the NAND flash controller using the NDFLASH.CONTROL_REG register:
   - Set interrupts in the NDFLASH.CONTROL_REG register.
   - If required, allow the DMA to transfer data to/from the buffer: set the NDFLASH.CONTROL_REG [3] DMA_ENABLE bit to 0x1.
   - Define the division factor applied to the clock source to generate the system clock: set the NDFLASH.CONTROL_REG [9:6] CLK_DIV field.
   - Program the time to wait for the busy signal (very slow signal coming from the memory): set the NDFLASH.CONTROL_REG [13:10] DUMMY_CYCLE field to match busy timing.
2. The CPU must fill the first buffer page (even when the DMA is used) with command byte(s), address bytes, and data bytes. Before starting the transfer, the CPU must also write in the NDFLASH.COMMAND_REG register:
   - Select the external NAND flash memory: set the NDFLASH.COMMAND_ REG [1] nCE1 bit to 0X0.
   - Enable data to be written to the NAND flash memory: set the NDFLASH.COMMAND_REG [5] RnW_FLASH bit to 0x0.
   - To indicate the number of command bytes to write first in the NAND flash memory, set the NDFLASH.COMMAND_REG [7:6] FIRST_COMMAND_BYTE field.
   - To indicate the number of address bytes to write in the NAND flash memory, set the NDFLASH.COMMAND_REG [10:8] ADDRESS_BYTE field.
   - To indicate the number of command bytes to write after addresses in the NAND flash memory, set the NDFLASH.COMMAD_REG [12:11] SECOND_COMMAND_BYTE (here 0x0).
   - Set the number of data bytes to read or write from/to the NAND flash memory in the NDFLASH.COMMAND_REG [29:13] DATA_BYTE field. It is possible to read or write 128K bytes in one transfer.
   - Set the number of command bytes to write after data in the NAND flash memory in the NDFLASH.COMMAND_REG [31:30] THIRD_COMMAND_BYTE field (here 0x0).
3. To start the transfer, set the NDFLASH.COMMAND_REG [0] ENABLE_TRANSFER bit to 0x1.

4. Transfer management
   - Using the CPU
     - One CPU cycle after setting the ENABLE_TRANSFER bit, the sequencer switches the page so that the CPU can write additional data to the other page.
     - One prescaler cycle after the page switch, the CPU starts writing to the NAND flash port.
     - Commands are sent first, followed by the address bytes. The data read from the buffer page are then written to the NAND flash port.
     - When the sequencer completes its page, it waits until the other page is fully filled by the CPU before switching the page, updates the NDFLASH.STATUS_IT_REG status interrupt register, and, if enabled, sends an interrupt to the CPU.
     - The CPU uses interrupts or polling to determine that an empty page is available for filling. This process is repeated until the last page.
     - When the last page is not fully filled, to indicate to the sequencer that the page is ready, the CPU either accesses the last byte address (in read or write) of the buffer or sets the NDFLASH.CONTROL_ REG [5] PAGE_READY bit to 0x1.
     - The PAGE_READY bit is auto_UnicodeEncodeError_cleared when the sequencer completes its transfer.
     - The CPU uses interrupts or polling to determine the end of the transfer.
   - Using the DMA
     - One CPU cycle after setting the ENABLE_TRANSFER bit, the sequencer switches the page and sends a DMA request to inform the DMA that it can write to the other page.
     - One prescaler cycle after the page switch, the DMA starts writing to the NAND flash port.
     - Commands are sent first, followed by the address bytes. The data read from the buffer page are then written to the NAND flash port.
     - When the sequencer completes its page, it waits until the other page is fully filled by the DMA before switching the page, updates the NDFLASH.STATUS_IT_REG status interrupt register, and sends a DMA request.
     - The DMA knows through the DMA request that an empty page is available for filling. This process is repeated until the last page.
     - If the last page is not filled, the DMA cannot access the PAGE_READY bit. The DMA fills the page with dummy data or informs the CPU when it completes the transfer so that the CPU accesses the buffer last byte address (in read or write) or sets the PAGE_READY bit to 0x1.
     - The PAGE_READY bit is auto_UnicodeEncodeError_cleared when the sequencer completes its transfer.
     - The CPU uses interrupts or polling to determine the end of the transfer.

Note: All registers except the NDFLASH.COMMAND_REG register can be modified during a transfer only if the sequencer stops.

The sequencer is stopped during a busy signal and during an ECC interrupt as long as the CPU has not read the ECC_VALUE_REG register.

10.5.4 Read Transfer

Either the CPU or the DMA can read the NAND flash memory.

1. The CPU must configure the NAND flash controller using the NDFLASH.CONTROL_REG register:
   - Set interrupts in the NDFLASH.CONTROL_REG register.
   - If needed, let the DMA transfer data to/from the buffer: set the NDFLASH.CONTROL_REG [3] DMA_ENABLE bit to 0x1.
   - Define the division factor applied to the clock source to generate the system clock: set the NDFLASH.CONTROL_REG [9:6] CLK_DIV field.
   - Program the time to wait for the busy signal (very slow signal from the memory): set the NDFLASH.CONTROL_REG [13:10] DUMMY_CYCLE field to match the busy timing.
2. In sequential read, indicate to the sequencer when there is a switch of a page in the NAND flash memory:

- Indicates to the sequencer the page change frequency in bytes to set the NDFLASH.BLOCK_SIZE register. BLOCK_SIZE indicates to the sequencer the page change frequency in bytes. For example, if there is a sequential read transfer with a change of page each 32 bytes, BLOCK_SIZE must be programmed at 32.
- To know how many bytes until the first page change: indicates the sequencer the start address in setting the NDFLASH.START_ ADDRESS register. This register is directly associated with the BLOCK_SIZE register. The sequencer must have the start address to determine the number of bytes until the first page change. With this information and the BLOCK_SIZE, the sequencer can correctly apply the dummy cycle time required to wait for the busy signal.

3. The CPU fills the first buffer page (even when the DMA is used) with the command byte(s), address bytes, and data bytes. Before starting the transfer, the CPU also writes to the NDFLASH.COMMAND_REG register:
   - Select the external NAND flash memory: set the NDFLASH.COMMAND_ REG [1] nCE1 bit to 0x0.
   - Enable the reading operation from the NAND flash memory: set the NDFLASH.COMMAND_REG [5] RnW_FLASH bit to 0x1.
   - To indicate the number of command bytes to write first in the NAND flash memory, set the NDFLASH.COMMAND_REG [7:6] FIRST_COMMAND_BYTE field.
   - To indicate the number of address byte to write in the NAND flash memory, set the NDFLASH.COMMAND_REG [10:8] ADDRESS_BYTE field.
   - To indicate the number of command bytes to write after addresses in the NAND flash memory, set the NDFLASH.COMMAD_REG [12:11] SECOND_COMMAND_BYTE field (here 0x0).
   - Set the number of data bytes to read or write from/to the NAND flash memory in the NDFLASH.COMMAND_REG [29:13] DATA_BYTE field. It is possible to read or write 128K bytes in one transfer.
   - Set the number of command byte to write after data in the NAND flash memory in the NDFLASH.COMMAND_REG [31:30] THIRD_COMMAND_ BYTE field (here 0x0).
4. To start the transfer, set the NDFLASH.COMMAND_REG [0] ENABLE_TRANSFER bit to 0x1.
5. Transfer management
   - Using the CPU
     - One CPU cycle after setting the ENABLE_TRANSFER bit, the sequencer switches the page.
     - Commands are sent first, followed by the address bytes. The address generator is cleared before starting the data transfer.
     - Data are read from the NAND flash port and written to the buffer page.
     - When the sequencer completes its page, it updates the NDFLASH.STATUS_IT_REG status interrupt register and, if enabled, sends an interrupt to the CPU.
     - The CPU uses interrupts or polling to determine that it can read the page. This process is repeated until the last page.
     - The CPU uses interrupts or polling to determine the end of the transfer.
   - Using the DMA
     - One CPU cycle after setting the ENABLE_TRANSFER bit, the sequencer switches the page.
     - Commands are sent first, followed by the address bytes. The address generator is cleared before starting the data transfer.
     - Data are read from the NAND flash port and written to the buffer page.
     - When the sequencer completes its page, it updates the NDFLASH.STATUS_IT_REG status interrupt register and sends a DMA request.
     - The DMA uses the DMA request to determine that it can read the page. This process is repeated until the last page.
     - The CPU uses interrupt or polling to determine the end of the transfer.

10.6 NAND Flash Controller Register Manual

This section provides information on the NAND flash controller module within this product. Table 10-6 provides a summary of the NAND flash controller registers. Each register within the module is described separately in the remaining parts of this section.

CAUTION

The NAND flash controller registers are limited to 32-bit access. Data access of 16/8 bits is not allowed and can corrupt register contents.

Table 10-5. Instance Summary

Module Name	Base Address	Size
NAND flash controller	0x09D0 0000	1M bytes

10.6.1 NAND Flash Controller Register Mapping Summary

Table 10-6. NAND Flash Controller Register Offset Address

Register Name	Type	Register Width (Bits)	Offset
COMMAND_REG	R/W	32	0x0
CONTROL_REG	R/W	32	0x4
STATUS_IT_REG	R	32	0x8
STATUS_STATE_REG	R	32	0xC
BLOCK_SIZE_REG	R/W	32	0x10
START_ADDRESS_REG	R/W	32	0x14
ECC_SIZE_REG	R/W	32	0x18
ECC_VALUE_REG	R	32	0x1C

10.6.2 Register Description

Table 10-7. COMMAND_REG

Address Offset	0x0		
Physical address	0x9D0 0000	**Instance**	Unknown
Description	This register contains the operation protocol used to transfer data to/from the NAND flash memory.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
THIRD_COMMAND_BYTE								DATA_BYTE												SECOND_COMMAND_BYTE				ADDRESS_BYTE				FIRST_COMMAND_BYTE		RnW_FLASH	Reserved	nWP1	Reserved	nCE1	ENABLE_TRANSFER

Bits	Field Name	Description	Type	Reset
31:30	THIRD_ COMMAND_BYTE	Number of command byte sent after data to the NAND flash memory	R/W	0x0
29:13	DATA_BYTE	Number of data byte to transfer from or to the NAND flash memory	R/W	0x0
12:11	SECOND_ COMMAND_BYTE	Number of command byte sent after addresses to the NAND flash memory	R/W	0x0
10:8	ADDRESS_BYTE	Number of address byte to sent to the NAND flash memory	R/W	0x0
7:6	FIRST_ COMMAND_BYTE	Number of command byte to sent first to the NAND flash memory	R/W	0x0
5	RnW_FLASH	0 => Write the NAND flash memory 1 => Read the NAND flash memory	R/W	0x0
4	Reserved	Do not change the reset value, read returns reset value.	R/W	0x0
3	nWP1	Write protect for NF_UnicodeEncodeError_Memory 1	R/W	0x0
2	Reserved	Do not change the reset value, read returns reset value.	R/W	0x1
1	nCE1	Chip enable for NF_UnicodeEncodeError_Memory 1	R/W	0x1
0	ENABLE_TRANSFER	0 => No transfer 1 => Transfer is enabled	R/W	0x0

Table 10-8. CONTROL_REG

Address Offset	0x4		
Physical address	0x9D0 0004	**Instance**	Unknown
Description	This register contains controls for reset, interrupts, prescaler, dummy-cycle, DMA enable, ECC.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EN_IT_BUSY	EN_IT_ECC	ECC_ENABLE	DUMMY_CYCLE				CLK_DIV			PAGE_READY	TEST_RAM	DMA_ENABLE	EN_IT_TRANSFER	EN_IT_PAGE	SOFTWARE_RESET

Bits	Field Name	Description	Type	Reset
31:17	Reserved	Write has no functional effect.	R/W	0x0
16	EN_IT_BUSY	0 => Interrupt end busy is disabled 1 => Interrupt end busy is enabled	R/W	0x0
15	EN_IT_ECC	0 => Interrupt end ECC is disabled 1 => Interrupt end ECC is enabled	R/W	0x0
14	ECC_ENABLE	0 => ECC is disabled 1 => ECC is enabled	R/W	0x0
13:10	DUMMY_CYCLE	Programmable dummy to respect busy timing Time = programmed value * prescaler clock	R/W	0x0
9:6	CLK_DIV	Define the division factor applied to clock source to generate the system clock. Division factor = CPU clock / program value	R/W	0x0
5	PAGE_READY	When on indicates sequencer a page is ready	R/W	0x0

Bits	Field Name	Description	Type	Reset
4	TEST_RAM	0 => Functional mode 1 => Test mode	R/W	0x0
3	DMA_ENABLE	0 => Use of the CPU 1 => Use of the DMA	R/W	0x0
2	EN_IT_TRANSFER	0 => Interrupt end transfer is disabled 1 => Interrupt end transfer is enabled	R/W	0x0
1	EN_IT_PAGE	0 => Interrupt end page is disabled 1 => Interrupt end page is enabled	R/W	0x0
0	SOFTWARE_RESET	0 => No reset 1 => NF controller is reset	R/W	0x0

Table 10-9. STATUS_IT_REG

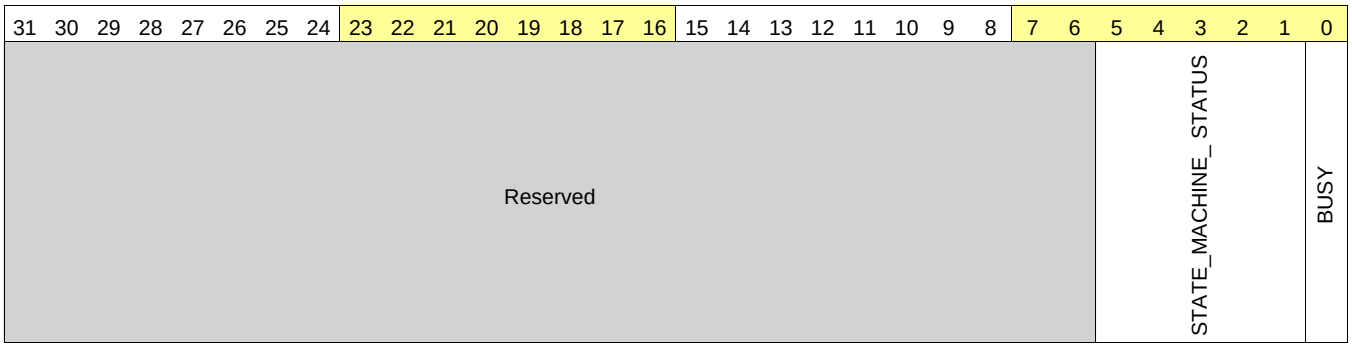
Address Offset	0x8		
Physical address	0x9D0 0008	**Instance**	Unknown
Description	This register indicates the status of the transfer.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								OVERRUN	Reserved	Reserved	END_BUSY	END_ECC	END_TRANSFER	END_PAGE	

Bits	Field Name	Description	Type	Reset
31:7	Reserved	Read returns reset value	R	0x0
6	OVERRUN	When setting to one, indicates CPU that a page has been overwritten.	R	0x0
5	Reserved	Read returns reset value.	R	0x0
4	Reserved	Read returns reset value.	R	0x0
3	END_BUSY	1 => End of busy (in write only)	R	0x0
2	END_ECC	0 => ECC on going or not active 1 => ECC ends its operation	R	0x0
1	END_TRANSFER	0 => No transfer or transfer on going 1 => End of transfer	R	0x0
0	END_PAGE	0 => Sequencer and CPU keep their current page 1 => Sequencer switches the page	R	0x0

Table 10-10. STATUS_STATE_REG

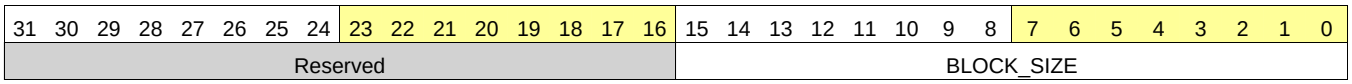
Address Offset	0xC		
Physical address	0x9D0 000C	**Instance**	Unknown
Description	This register indicates the current state of the state-machine and the busy state of the NAND flash memory.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:6	Reserved	Read returns reset value.	R	0x0
5:1	STATE_MACHINE_ STATUS	Indicates the current state of the sequencer state-machine	R	0x0
0	BUSY	0 => NAND flash is busy 1 => NAND flash is ready	R	0x1

Table 10-11. BLOCK_SIZE_REG

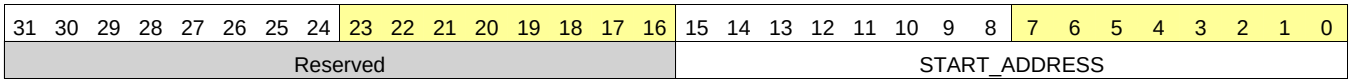
Address Offset	0x10		
Physical address	0x9D0 0010	**Instance**	Unknown
Description	This register indicates to sequencer the page change frequency in term of bytes.		
Type	R/W		



Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write has no functional effect.	R/W	0x0
15:0	BLOCK_SIZE	Indicates page change frequency in term of bytes	R/W	0x0

Table 10-12. START_ADDRESS_REG

Address Offset	0x14		
Physical address	0x9D0 0014	**Instance**	Unknown
Description	This register indicates to sequencer where the first read will be done.		
Type	R/W		



Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write has no functional effect.	R/W	0x0
15:0	START_ADDRESS	Indicates in read sequential mode where the first read will be done.	R/W	0x0

Table 10-13. ECC_SIZE_REG

Address Offset	0x18		
Physical address	0x9D0 0018	**Instance**	Unknown
Description	None		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ECC_SIZE															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write has no functional effect.	R/W	0x0
15:0	ECC_SIZE	Indicates ECC, the number of data bytes to take, before sending interrupt and stop the transfer.	R/W	0x0

Table 10-14. ECC_VALUE_REG

Address Offset	0x1C		
Physical address	0x9D0 001C	**Instance**	Unknown
Description	This register contains the calculated ECC value.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINE_PARITY_PRIME															COLUMN_PARITY_ PRIME	LINE_PARITY														COLUMN_PARITY	

Bits	Field Name	Description	Type	Reset
31:19	LINE_PARITY_PRIME	P8', P16', P32', P64', to P32768' value	R	0x0
18:16	COLUMN_PARITY_ PRIME	P1', P2' and P4' value	R	0x0
15:3	LINE_PARITY	P8, P16, P32, P64, to P32768 value	R	0x0
2:0	COLUMN_PARITY	P1, P2 and P4 value	R	0x0

EMIF

This chapter describes the external memory interface (EMIF) of the LOCOSTO device.

Topic **Page**

11.1 EMIF Module Overview

The EMIF manages synchronous/asynchronous 16-bit data bus read/write accesses between external memories, the MPU, and the DMA controller. Figure 11-1 shows an overview of the EMIF.

11.1.1 Main Features

The EMIF incldues the following main features:

- 8-,16-, and 32-bit read/write from the MPU or the DMA
- 16-bit word data bus
- Burst read/write from the DMA is supported and functional. DMA bursts are always aligned on the 8 x 16 memory address.
- The burst read (instruction/data) with the prefetch buffer enable from the MPU is supported and functional.
- Enhanced Least Recently Used (ELRU) priority between the MPU/DMA.
- 4 chip-selects (CS0, CS1, CS2, and CS3). With 32M bytes for each chipselect (except for CS0 with 28M bytes).
- Maximum of 124M bytes of external memory. Each chip-select is separately configurable for:
  - Asynchronous read/write (default)
    - Programmable wait-states
    - Supports dynamic access time extension through the external nRDY signal
  - Synchronous burst read/write
    - 8 x 16-bit-word burst is supported by the EMIF external interface. Internally, the prefetch buffer and the DMA launch a 4 x 32-bit-word burst.
    - 52-MHz clock for burst accesses
    - Supports dynamic access time extension through the external nRDY signal
- Insertions of wait-state for read and write access
- Multiplexing of address/data only
- The EMIF supports the following common external memory signals:
  - Output enable (nMOE)
  - Write enable (RnW)
  - Address valid (ADV)
  - Byte enable (nBE[1:0])
  - Ready (nRDY)
  - Device clock (CKM)
  - Write protect (nFWP)
- External memory frequency configuration
- Protect mode support (read only):
  - Entire chip-select support
  - Range of addresses from base address

Note: The burst read without prefetch buffer enable is not supported by design (the LDM opcode cannot lead to any burst; data buffering is handled by the prefetch buffer only). The burst write is not supported by design (the STM opcode cannot lead to any burst; the prefetch buffer is used only for read access).

11.1.2 Signals and I/O Description

Figure 11-1 shows an overview of the EMIF. Table 11-1 describes the EMIF module I/Os and the interaction with other modules inside the LOCOSTO device.

Figure 11-1. EMIF Overview

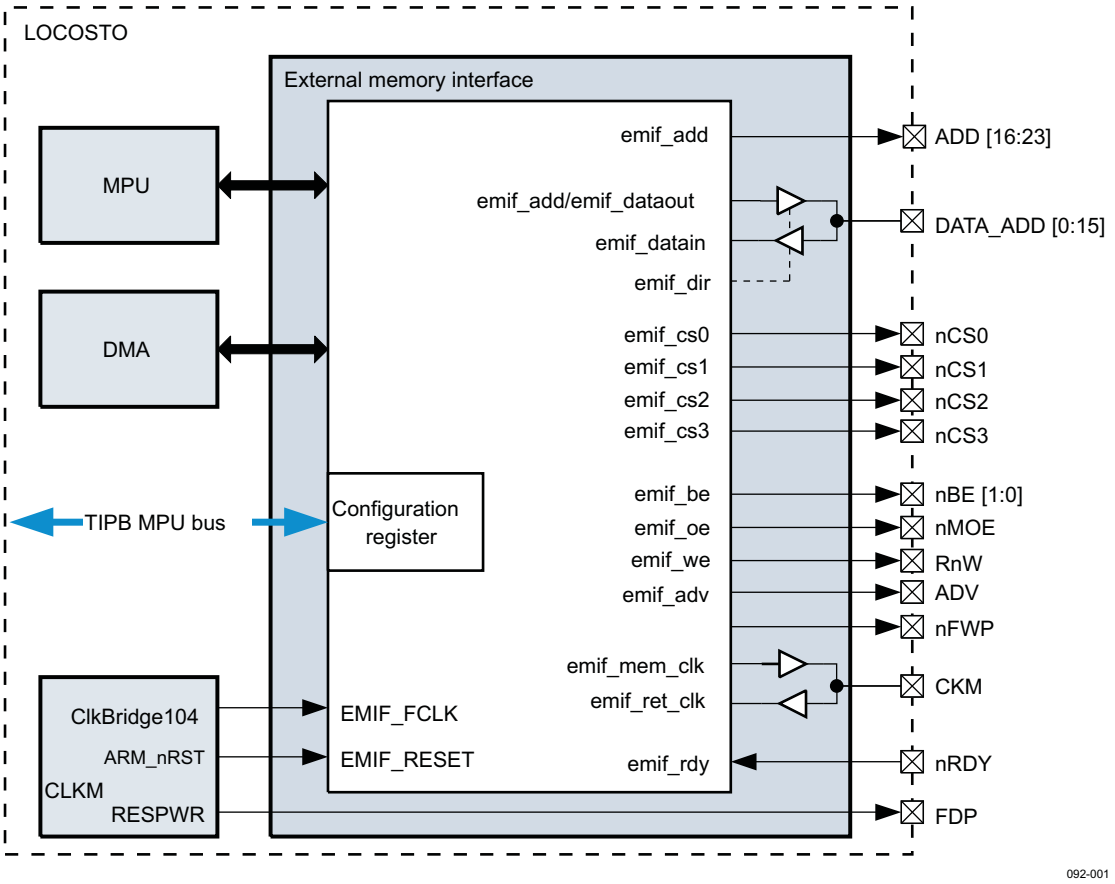


Table 11-1. I/O Description

Signal Name	I/O[1]	Description	Reset Value
DATA_ADD[15:0]	I/O	Address and data inputs/outputs multiplexed	Unknown
ADD[23:16]	O	Most-significant bit (MSB) of address	Unknown
nCS[0:3]	O	Activates the device memory: 0: Device memory activate 1: Device memory deactivate	1
RnW	O	Writes an enable signal to the memory device: 0: Write 1: Read	1
nMOE	O	Controls data outputs during the Bus Read operation of the device memory: 0: Output enable 1: Output disable	1
nBE[0:1]	O	High and low byte enable for byte write	1
FDP	O	Usually connected to memory reset, thus reducing the current consumption	1

(1) I=Input, O=Output

Table 11-1. I/O Description (continued)

Signal Name	I/O[1]	Description	Reset Value
ADV	O	Indicates to memory device that a valid address is present on the address inputs (address bits A15–A0 are multiplexed, address bits A23–A16 are address only).	1
nFWP	O	Gives an additional hardware protection for write action on memory device: 0: Disable write on memory 1: Enable write	1
CKM	O	Synchronizes the memory to the frequency of the EMIF during synchronous operations such as burst mode.	0
nRDY	I	Provides data-valid feedback during read and write in full-handshaking mode.	1

Note: The internal signal emif_dir multiplexes mem_datain and emif_ memadd/emif_memdat:

When emif_dir = 1, then DATA_ADD = emif_datain

When emif_dir = 0, then emif_add/emif_dataout = DATA_ADD

11.1.3 EMIF Environment

The synchronous/asynchronous EMIF supports the most common memory interface protocols through a flexible programming and timing signal control. A number of different muxed/nonmuxed memory types connect with the EMIF. These memory types share pins on the device, but their functionalities and controlling logic may differ.

The LOCOSTO EMIF module supports the following protocols: PSRAM, NOR, and ADD/DATA multiplexed. The EMIF can control up to four memory devices, with four chip-select signals, for a total address range up to 124M bytes (1 * 28M bytes + 3 * 32M bytes). Table 11-2 lists the memories that are compatible with the EMIF. An interconnection example is also shown in Table 11-3.

Table 11-2. EMIF-Compatible Memories

Reference:	NOR: S29NS016J/S29NS064J/S29NS128J
Type:	Read/write burst mode flash memory
ADD/DAT:	Muxed
Size	16 / 64 / 128M bits
Data bus:	16 bits
Manufacturer:	Spansion
Power supply:	1V8
Reference:	NOR: M58WR016FU
Type:	Read/write burst mode flash memory
ADD/DAT:	Muxed
Size	16M bits
Data bus:	16 bits
Manufacturer:	ST Microelectronics
Power supply:	1V8
Reference:	NOR: RD38F3050L0YBQ1S
Type:	Read/write burst mode flash memory

Table 11-2. EMIF-Compatible Memories (continued)

ADD/DAT:	Muxed
Size	64M bits
Data bus:	16 bits
Manufacturer:	Intel
Power supply:	1V8
Reference:	PSRAM: MT45W4MW16BFB
Type:	Read/write burst mode flash memory
ADD/DAT:	Nonmuxed
Size	64M bits
Data bus:	16 bits
Manufacturer:	Micron
Power supply:	1V8
Reference:	MCP: S71NS128JA0 (128Mb Spansion NOR / 64Mb Micron PSRAM)
Type:	Read/write burst mode flash memory
ADD/DAT:	Muxed
Size	128M bits Spansion NOR / 64M bits Micron PSRAM
Data bus:	16 bits
Manufacturer:	Spansion (Micron PSRAM)
Power supply:	1V8

Table 11-3. EMIF Environment Example

LOCOSTO					**DIR**	**S29NS064J**			
Signal	**Ball**	**Mode**	**I/O(1)**	**Power**		**Signal**	**Ball**	**I/O(1)**	**Power**
Control Signals									
ADV	J7	0	O	VDD_MIF	→	nAVD	B4	I	VRMEM
nCS0	E3	0	O		→				
nCS1	D2	1	O		→				
nCS2	F5	1	O		→				
nCS3	L5	0	O		→	nCE	B9	I	
CKM	L6	1	IO		↔	CLK	A4	IO	
nMOE	M1	0	O		→	nOE	C10	I	
nRDY	M3	0	I		←	RDY	A1	O	
nBHE	L3	0	O		→				
nBLE	M2	0	O		→				
nFWP	G6	1	O		→	nWP	B7	I	
FDP	K6	0	O		→	nRESET	B6	I	
RnW	L1	0	O		→	nWE	A6	I	
Address Signals									
ADD16	G3	0	O	VDD_MIF	→	A16	B2	I	VRMEM
ADD17	F1	0	O		→	A17	A9	I	
ADD18	F2	0	O		→	A18	B8	I	
ADD19	H6	0	O		→	A19	A8	I	
ADD20	J8	0	O		→	A20	B3	I	
ADD21	F3	0	O		→	A21	A2	I	
ADD22	G5	1	O		→	A22	A10	I	
ADD23	H7	1	O		→				
Data/Address Signals									
ADD/DAT0	K4	0	IO	VDD_MIF	↔	A/DQ0	D10	IO	VRMEM

(1) I = Input 0 = Output

Table 11-3. EMIF Environment Example (continued)

LOCOSTO					DIR	S29NS064J			
Signal	Ball	Mode	I/O[1]	Power		Signal	Ball	I/O[1]	Power
ADD/DAT1	L2	0	IO		↔	A/DQ1	D9	IO	
ADD/DAT2	K5	0	IO		↔	A/DQ2	C7	IO	
ADD/DAT3	K3	0	IO		↔	A/DQ3	C6	IO	
ADD/DAT4	K2	0	IO		↔	A/DQ4	D5	IO	
ADD/DAT5	J5	0	IO		↔	A/DQ5	D4	IO	
ADD/DAT6	J3	0	IO		↔	A/DQ6	C3	IO	
ADD/DAT7	J2	0	IO		↔	A/DQ7	C2	IO	
ADD/DAT8	J4	0	IO		↔	A/DQ8	C9	IO	
ADD/DAT9	J6	0	IO		↔	A/DQ9	C8	IO	
ADD/DAT10	H2	0	IO		↔	A/DQ10	D7	IO	
ADD/DAT11	H3	0	IO		↔	A/DQ11	D6	IO	
ADD/DAT12	H5	0	IO		↔	A/DQ12	C5	IO	
ADD/DAT13	G2	0	IO		↔	A/DQ13	C4	IO	
ADD/DAT14	G1	0	IO		↔	A/DQ14	D2	IO	
ADD/DAT15	H4	0	IO		↔	A/DQ15	D1	IO	

11.1.4 Clocking, Reset, and Power-Management Scheme

11.1.4.1 Clocks

The EMIF module has one functional domain clock (see Table 11-4). The single and fixed functional clock is derived from the 104-MHz clock generated by a digital phase-locked loop (DPLL) through the CLKM module (for more information, see Chapter 6, *Power, Reset, and Clock Management*). The 104-MHz clock is fed to the module and then divided (by programming) to produce the reference clocks for the register and timing setup. For more information, see Section 11.2.1.3, *Modes and Control Signal Configuration*.

Table 11-4. EMIF Clock

Type	Name	Source	Frequency	Description
Functional	EMIF_FCLK	ClkBridge 104	104 MHz	Functional clock, divided for synchronization

11.1.4.2 Power Management

The EMIF supports two power-saving modes:

- Power-down acknowledgement
- Auto-idle mode

The EMIF supports auto-idle mode (clock-gating) to dynamically reduce power consumption when no requests are pending and no accesses are ongoing. The auto-idle mode is enabled by setting the PWD_ENA bit EMIF. EMIF_CONF[2] to 1. When this bit is set, the EMIF cuts its clock when there are no requests from the hosts through auto-gating clocks. The EMIF emerges from the idle mode when there is a request from any initiator (that is, the MPU or the DMA).

At the system level, power reduction techniques can be applied by shutting down certain internal clock and power domains of the device. The EMIF supports an idle request protocol. This protocol is used at system level to shut down EMIF clocks in a clean and controlled manner. The EMIF can send an idle request acknowledge when all ongoing transactions and accesses are completed. This signal allows the CLKM module to cut the EMIF source clock properly. The idle request acknowledge feature is enabled by setting the PDE bit EMIF.EMIF_CONF[3] to 1. Table 11-5 shows the global and dynamic power-down modes available for power-saving and their configurations.

Table 11-5. Power-Saving Mode

PDE[1]	DWD_ENA[1]	Mode of Operation
0	0	Power-down mode is disabled.
1	0	Global power-down mode is enabled.
0	1	Dynamic power-down mode is enabled.
1	1	Both modes are enabled.

(1) In boot configuration and normal use, it is recommended to set the PDE and PWD_ENA bits to 1.

11.1.4.3 Hardware and Software Resets

The EMIF module has one reset:

Global reset or hardware reset of the EMIF module is performed by activating the ARM_nRST signal (for more information, see Chapter 6, *Power, Reset, and Clock-Management*).

External common memory supports the RESET input pin to allow the device state-machine to be properly reset on power-up and also to minimize power consumption in idle mode. The EMIF interface includes a FDP (active low) output pin (for more information, see Chapter 6, *Power, Reset and Clock-Management*). Table 11-6 describes the EMIF reset.

Table 11-6. EMIF Reset

Type	Name	Source	Activation	Domain
Hardware	EMIF_RESET	ARM_nRST	0	Global reset on state-machine and registers

After reset, each EMIF chip-select configuration register is configured to interface in an asynchronous mode. The number of wait-states for an access is set to 15 (0x0F) cycles from REF_CLK. FCLKDIV is set at 1 at reset; therefore, EMIF_FCLK is divided by 2. This latency ensures a maximum compatibility with various existing devices. The EMIF configuration should be done when no initiator (other than the MPU) is accessing the EMIF.

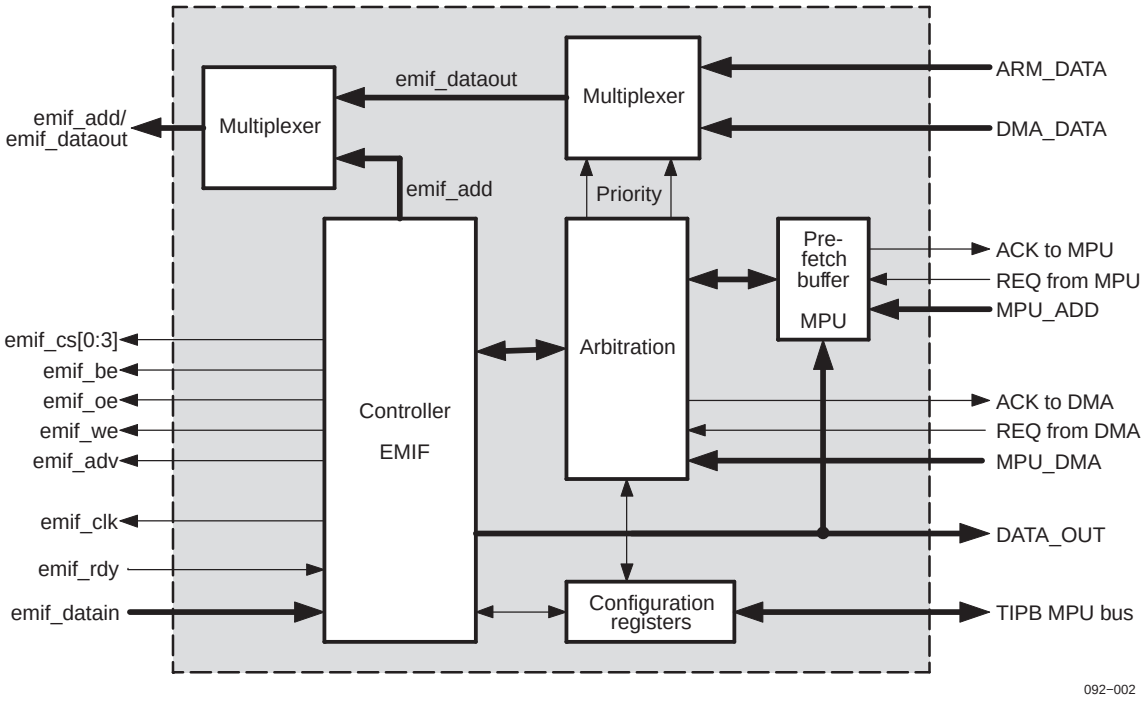
11.2 EMIF Functional Description

11.2.1 Block Diagram

Figure 11-2 shows the block diagram of the EMIF module. The EMIF module consists of the following five submodules:

- Arbitration: EMIF locally arbitrates simultaneous access between the MPU and the DMA using an ELRU algorithm
- Multiplexer: EMIF supports most common multiplexed address and data memory.
- Controller: Generates external memory signals.
- Configuration registers: Allows the setup of different timing constraints on the external control signal for each chip-select.
- Prefetch buffer: Improves performance of the MPU access using a 2-line prefetch buffer.

Figure 11-2. EMIF Block Diagram

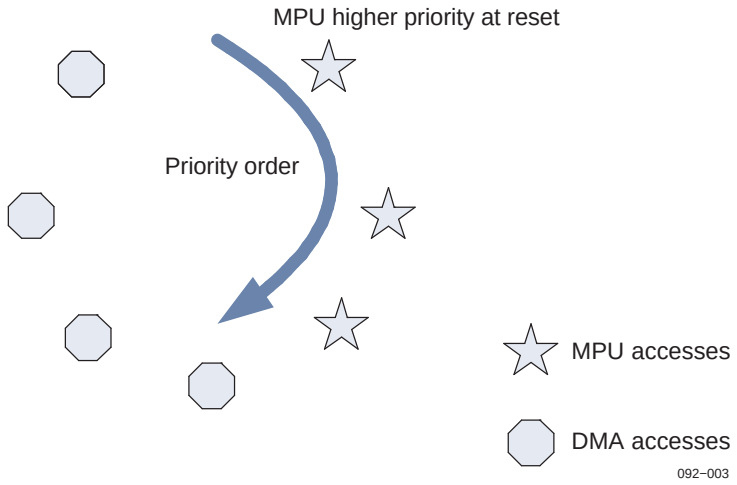


11.2.1.1 Arbitration

Once priority algorithms for resolving simultaneous accesses to the EMIF interface are implemented:

- ELRU
  - The requester with the highest priority is the least-recently used. A round-robin scheme is used.
  - The active requester can have consecutive accesses that are software programmable and controlled by the EMIF priority register EMIF.EMIF_LRU_PRIO.
  - The active requester will loose the priority if there is a dead cycle between requests.
  - All initiators have an ensured minimum bandwidth that is defined during the setup phase using the EMIF priority register EMIF. EMIF_LRU_PRIO. Figure 11-3 shows an example of an enhanced round-robin scheme where the EMIF priority register is programmed to allow three consecutive accesses to MPU and four consecutive accesses to DMA.

Figure 11-3. ELRU Priority Example



11.2.1.2 EMIF Address Mapping and Data Control in Multiplexed Mode

Four chip selects (CS0, CS1, CS2, and CS3) are provided for external memories. Each of them has an address range of 32M bytes, as shown in Table 11-7 and Figure 11-4.

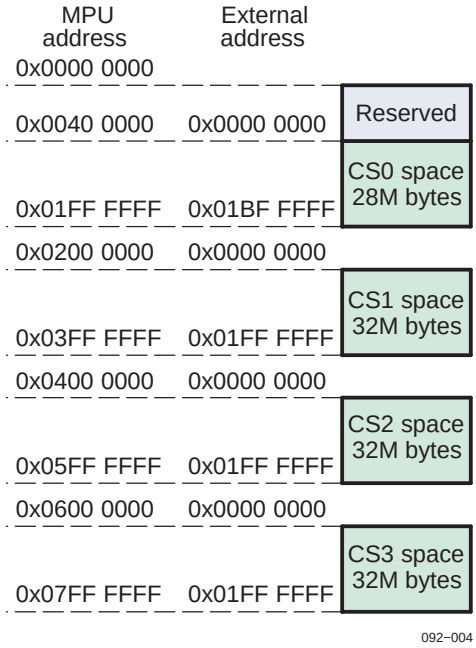
Table 11-7. External Memory Mapping

Chip Select	Start Address	Stop Address	Size	Data Access
nCS0	0x0040 0000	0x01FF FFFF	28m BYTES	8/16 RW
nCS1	0x0200 0000	0x03FF FFFF	32M bytes	8/16 RW
nCS2	0x0400 0000	0x53FF FFFF	32M bytes	8/16 RW
nCS3	0x0600 0000	0x07FF FFFF	32M bytes	8/16 RW

CAUTION

The address of the external memory allocated on CS0 is shifted by 4M bytes to prevent overlay with the reserved MPU space address. See Chapter 2, *Memory Mapping*, for more information.

Figure 11-4. External Memory Mapping with MPU Address



11.2.1.2.1 Endianism

The EMIF follows the little-endian format. Table 11-8 shows the little-endian format for a 32-bit word.

Table 11-8. Little-Endian Format for 32-bit Word

Address A1 _UnicodeEncodeError_ A0	11	10	01	00
Significant	MSB			LSB

Table 11-9 shows the little-endian format for a 16-bit word.

Table 11-9. Little-Endian Format for 16-bit Word

Address A0	1	0
Significant	MSB	LSB

11.2.1.2.2 Byte Enables

The EMIF has a 2-byte enable signal, which reflects the byte that the EMIF is currently accessing during a read or a write. Table 11-10 shows the byte-enable assertion depending on the external memory data bus width and initiator address. The example concerns a 16-bit width data bus memory. A read or write access size is determined by the data size.

Table 11-10. Byte Enable

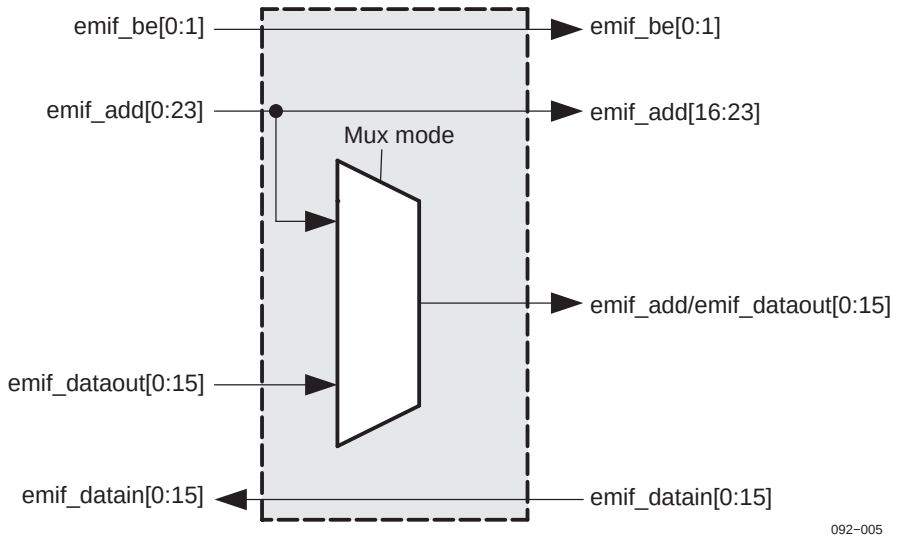
A1	A0	be(1)	be(0)	Data size
X	0	1	0	8 bits
X	1	0	1	8 bits
X	X	0	0	16 bits

The multiplexed address/data bus is connected to the memory as listed in Table 11-11 and as shown in Figure 11-5.

Table 11-11. Multiplexing of Address and Data

16-Bit Device		Pin Names
MPU Address	MPU Data	
A0		
A1	D0	A0-D0
A2	D1	A1-D1
A3	D2	A2-D2
A4	D3	A3-D3
A5	D4	A4-D4
A6	D5	A5-D5
A7	D6	A6-D6
A8	D7	A7-D7
A9	D8	A8-D8
A10	D9	A9-D9
A11	D10	A10-D10
A12	D11	A11-D11
A13	D12	A12-D12
A14	D12	A13-D13
A15	D14	A14-D14
A16	D15	A15-D15
A17	-	A16
A18	-	A17
A19	-	A18
A20	-	A19
A21	-	A20
A22	-	A21
A23	-	A22
A24	-	A23

Figure 11-5. Multiplexing of Address/Data Bus



11.2.1.3 Modes and Control Signal Configuration

Memory control signal nCSi, nMOE, RnW, ADV, and RDY setup and hold timing are controlled by a set of programmable configuration registers.

- The operation mode of the EMIF for a given chip-select region is selected by the bit field MEM_MODE EMIF.EMIF_CSi [18:16] registers, with i = [3:0], as shown in Table 11-12. The following operations are supported:
  - Asynchronous read: Used for any asynchronous memory, including flash and CellularRAM_UnicodeEncodeError_ devices.
  - Asynchronous write: Used for any asynchronous memory, including flash and CellularRAM devices.
  - Synchronous burst read protocol 1: Used for flash devices supporting synchronous mode.
  - Synchronous burst read protocol 2: Used for CellularRAM memory.
  - Synchronous burst write protocol 2: Used for CellularRAM memory.

Table 11-12. Configuration Mode with MEM_MOD

MEM_MOD	Read Operation	Write Operation
000	Asynchronous read	Asynchronous write
001	Sync burst read protocol1	Asynchronous write
010	Sync burst read protocol2	Sync burst write protocol2
011	Reserved	Reserved
100	Reserved	Reserved
101	Reserved	Reserved
110	Reserved	Reserved
111	Reserved	Reserved

CAUTION

When MEM_MOD = 001, the read operation is synchronous and the write operation is asynchronous.

Synchronous burst write for flash is not supported.

In both asynchronous and synchronous modes, all EMIF-to-memory control signals are controlled with an EMIF internal reference clock, REF_CLK, which is the EMIF_FCLK divided down.

- The REF_CLK is divided from EMIF_FCLK (see Table 11-13) by a programmable value contained in the FCLKDIV field EMIF.EMIF_CSi [1:0]. This accommodates the timing constraints of slow devices, even with a high system clock rate.

Table 11-13. Clock Divider for REF_CLK

FCLKDIV	REF_CLK
00	Reserved
01	EMIF_FCLK/2
10	EMIF_FCLK/4
11	EMIF_FCLK/6

- Depending on the chip-select configuration, this internal clock REF_CLK can be available outside through the CKM output pin.
  - In asynchronous mode, output CKM is kept low.
  - In synchronous mode, REF_CLK is available through output CKM via emif_mem_clk.

CAUTION

The REF_CLK frequency must be set to comply with the timing constraints of the attached device in combination with the EMIF timing controls.

- nMOE (emif_oe) valid timing from chip-select and address valid/invalid is programmable through the OESETUP bit field EMIF.MIF_ADV_CSi[3:0].
- RnW (emif_we) valid timing from chip-select, address, and ADV is programmable through WRWST and the WELEN bit field EMIF.EMIF_ CSi[15:8]. RnW hold time is fixed to one REF_CLK.
  - In synchronous and asynchronous mode, the address set up time ADV valid to ADV invalid is controlled by the ADVHOLD bit field EMIF.EMIF_ADV_CSi[8] with respect to the EMIF_FCLK rising edge.

    The ADV valid to invalid setup time is defined by:
    - (ADVHOLD + 1) REF_CLK cycle in asynchronous mode
    - (ADVHOLD + 1) REF_CLK + 1 EMIF_CLK cycle in synchronous mode
  - In write and read burst mode, the address hold time is extended to one EMIF_FCLK from ADV invalid time.
- Read and write access time is controlled by programmable internal waitstate generation (non-full-handshaking mode). An external ready input pin, nRDY, can be used in combination with internal wait-state (fullhandshaking mode).
  - In non-full-handshaking mode, the RDWST bit EMIF.EMIF_CSi[7:4] controls the internal read wait-state generation.
  - In non-full-handshaking mode, the WELEN bit field EMIF.EMIF_ CSi[15:12] controls the internal write wait-state generation, in addition with WRWST bit field EMIF.EMIF_CSi[11:8].
  - In non-full-handshaking protocol, nRDY is never monitored.
  - In full-handshaking mode, nRDY is monitored by the EMIF to control read and write access time. Access is complete when both the internal wait-state expires and nRDY is deasserted by the external device (the wait state must be smaller than or equal to the memories' minimum wait state to avoid incomplete access or data corruption). The nRDY timing constraint depends on synchronous or asynchronous access mode (for more information, see Section 11.2.1.4, *Extending Wait-States Using nRDY*).

11.2.1.4 Extending Wait-States Using nRDY

The EMIF supports the nRDY (emif_rdy) signal coming from an external memory. This feature allows the EMIF to extend wait-states when requested by the external memory.

- Non-full-handshaking mode: nRDY is not considered.
- Full-handshaking mode: The assertion of nRDY by the external memory extends the access time until nRDY is deasserted.

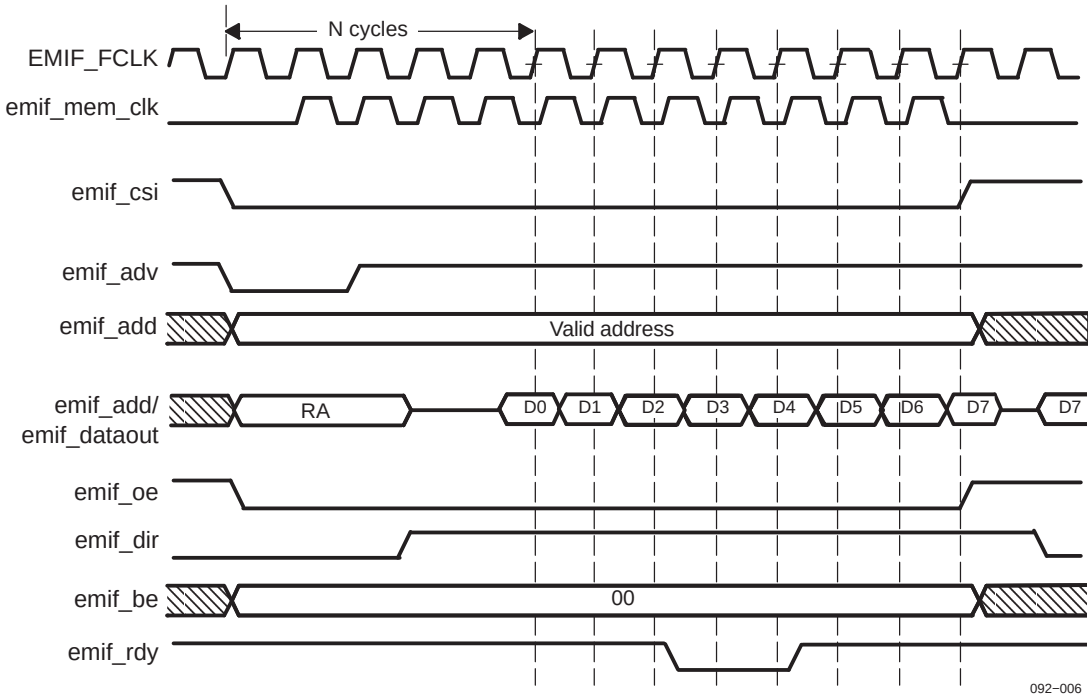
Note: This feature is available for each chip-select.

11.2.1.4.1 Non-Full-Handshaking Mode

In this mode of operation, the nRDY (emif_rdy) signal from the external memory is ignored. nRDY is not considered by the EMIF regardless of read or write operation.

Figure 11-6 shows synchronous read access when the EMIF is in non-fullhandshaking mode.

Figure 11-6. Synchronous Read Access When EMIF Is In Non-Full-Handshaking Mode



11.2.1.4.2 Full-Handshaking Mode

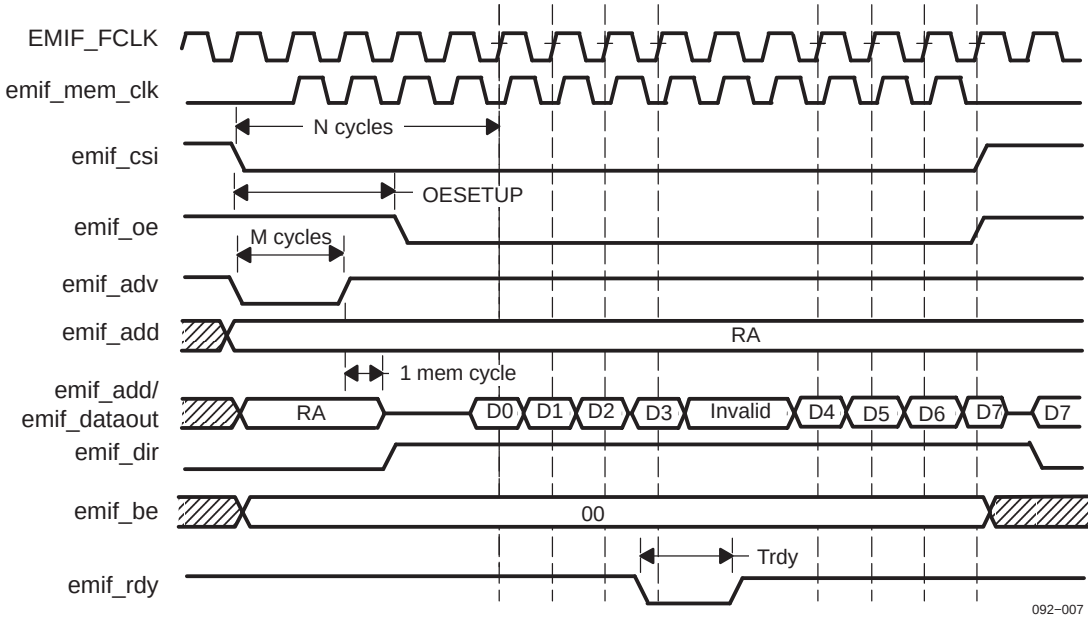
In full-handshaking mode, nRDY is monitored by the EMIF to control access. The access is completed when both the internal wait-state expires and nRDY is deasserted.

In this mode of operation, the nRDY signal from an external memory extends the access time. nRDY can be asserted using the following two timing constraints:

- nRDY is asserted at the same time of data.
- nRDY is asserted one cycle before the data is available.

These modes can be configured through HDK_CSi bit field EMIF.EMIF_ DWS[3:0]. Full-handshaking mode is supported in both synchronous and asynchronous mode of operation. If the EMIF hangs up due to a continuously asserted low, the state of nRDY pin can be checked internally by software by reading the FR bit EMIF.EMIF_CONF[2]. Figure 11-7 shows the nRDY from external memory asserted one cycle before the data is available.

Figure 11-7. Synchronous Read Access When nRDY Is Asserted By One Memory Clock Before Data



CAUTION

The assertion timing constraint of nRDY depends on the type of external memories used.

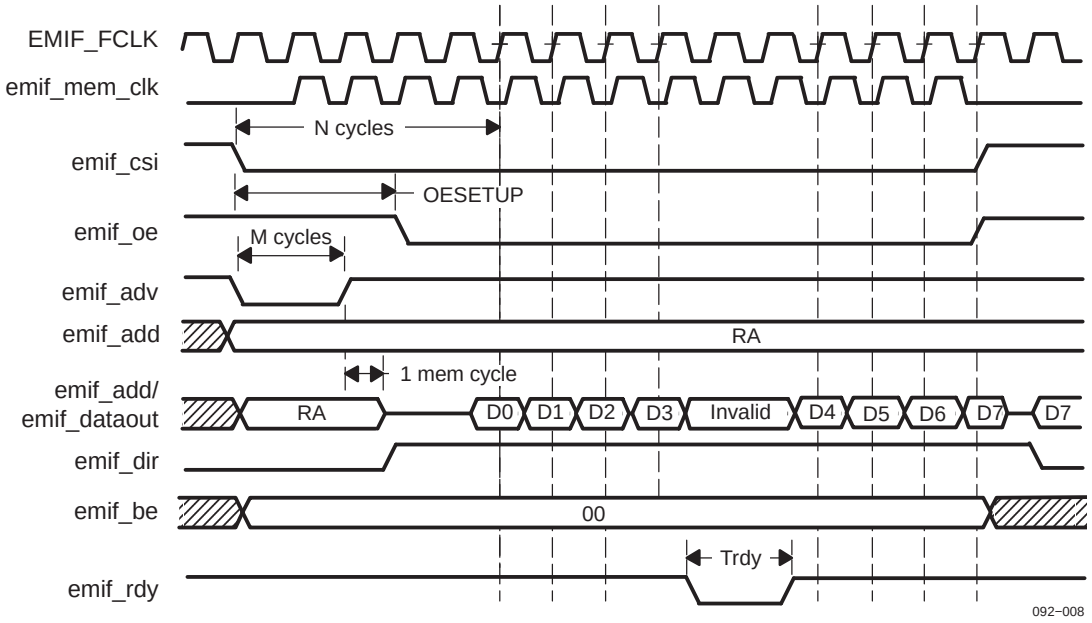
A submode to the full-handshaking mode allows the masking of the nRDY signal during write access by using the WRRDYMASK_CSi bit EMIF.EMIF_ DWS[7:4]. Table 11-14 shows the bit field programming for each modes.

Table 11-14. Full-handshaking Submode

HDK_CSi	WRRDYMASK_CSi	Handshaking Mode
0	0	nRDY is monitored for both read and write accesses (fullhandshaking).
0	1	nRDY is monitored for read access and masked for write access.
1	X	nRDY is ignored for both read and write accesses (non-fullhandshaking).

Figure 11-8 shows the synchronous read access when the ready asserted when the data is available.

Figure 11-8. Synchronous Read Access When nRDY Is Asserted by Memory When Data Is Available



11.2.1.5 Bus Turn-Around for Read to Read/Write Transition

The latency between read-to-write operation (same or different chip-select), or between read-to-read operation (same or different chip-select), is programmable from 0 to 15 cycles by the BTWST bit field EMIF.EMIF_CSi[25:22]. This feature avoids bus contention when using a slow device in the read mode. Figure 11-9 and Figure 11-10 show the read-to-read transition and read-to-write with bus turn-around. This bus transition wait-state inserts wait cycles after every read operation.

Figure 11-9. Bus Turn Cycles for Read-to-Read Operation BTWST (CSX) = 2 and BTWST (CSY) = 1

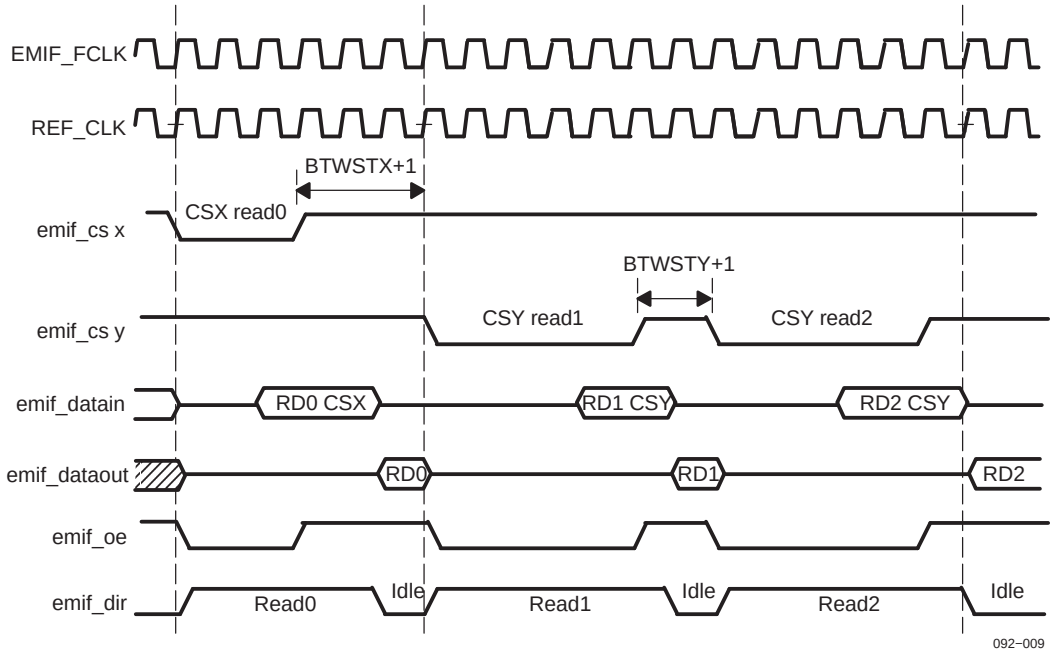
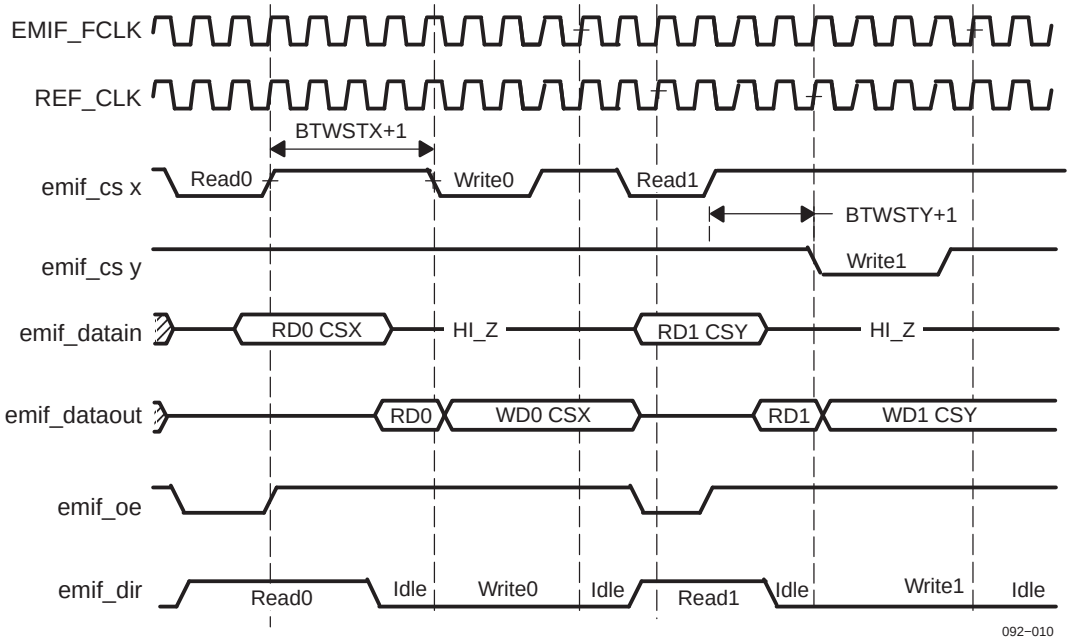


Figure 11-10. Bus Turn Cycles for Read-to-Write Transition with BTWST CSX=2



A read operation is completed immediately after BTWST clock cycles. The EMIF drives the data bus during idle state (when there are no transactions) and write operation. During read operation, the external memory drives the data bus.

The BTWST can also be used to introduce bus turn-around cycles during write-to-read and write-to-write on the same chip-select. For each chip-select CSi, the BTMODE bit EMIF_ADV_CSi[9] selects the type of transaction depending on where the cycles are inserted.

- If the BTMODE bit is set to 0, the bus turn-around cycles are introduced only between read to any transaction.
- If the BTMODE bit is set to 1, the bus turn-around cycles are introduced also between write to any transaction of the same chip-select apart from the read to any transaction.

The Figure 11-11 shows the write-to-write and write-to-read transitions with BTMODE set to 1.

Figure 11-11. Bus Turn Cycles for Write-to-Write and Write-to-Read Transition with BTMODE = 1 and BTWST = 3

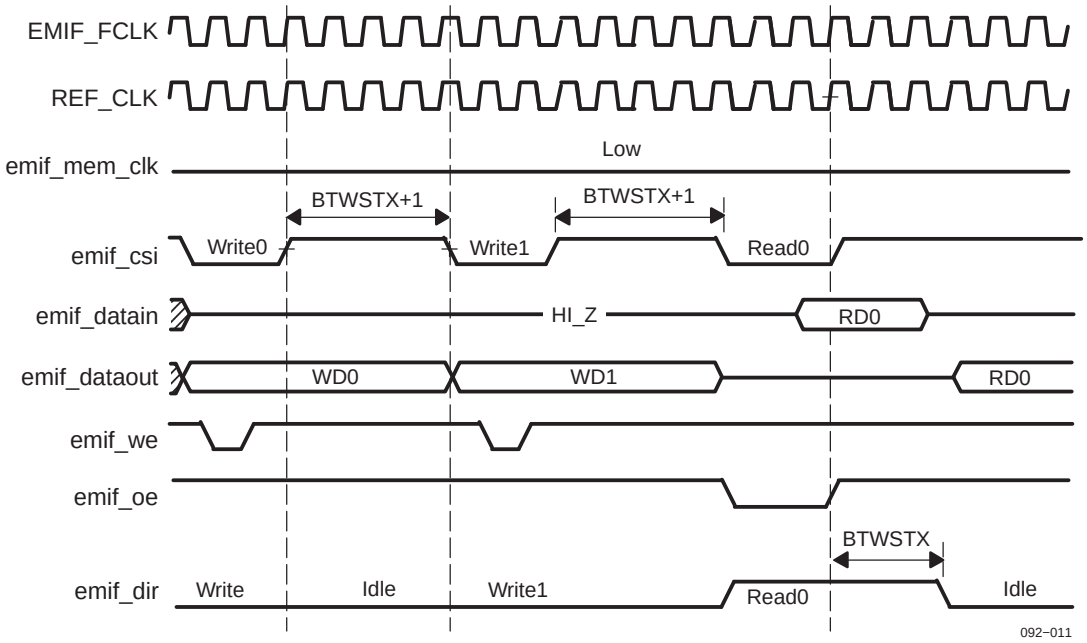


Table 11-15 shows the latency configuration for turn-around cycles during various transitions.

Table 11-15. Turn-around Latency Between Different Bus Access Depending on BTMODE

BTMODE	Access(n)	Access(n+1)	Chip Select	Latency
0	Read	Read	Same	Required BTWST ≠ 0
	Read	Write	Same	Required BTWST ≠ 0
	Write	Read	Same	Not required
	Write	Write	Same	Not required
	Read	Read	Different	Required BTWST ≠ 0
	Read	Write	Different	Required BTWST ≠ 0
	Write	Read	Different	Not required
	Write	Write	Different	Not required
1	Read	Read	Same	Required BTWST ≠ 0
	Read	Write	Same	Required BTWST ≠ 0
	Write	Read	Same	Required BTWST ≠ 0
	Write	Write	Same	Required BTWST ≠ 0
	Read	Read	Different	Required BTWST ≠ 0
	Read	Write	Different	Required BTWST ≠ 0
	Write	Read	Different	Not required
	Write	Write	Different	Not required

11.2.1.6 Retimed Protocol

Because of IC I/O and board delays, the theoretical external memory maximum frequency might not be usable for the REF_CLK value without the retiming function. Thus, enabling the retiming protocol is always safe when working at the 52-MHz external memory. In the synchronous mode, the retiming mode allows read data to be latched by a delayed emif_ret_clk (obtained through the IC I/O feedback of emif_mem_clk). This offers optimum data and sampling clock alignment.

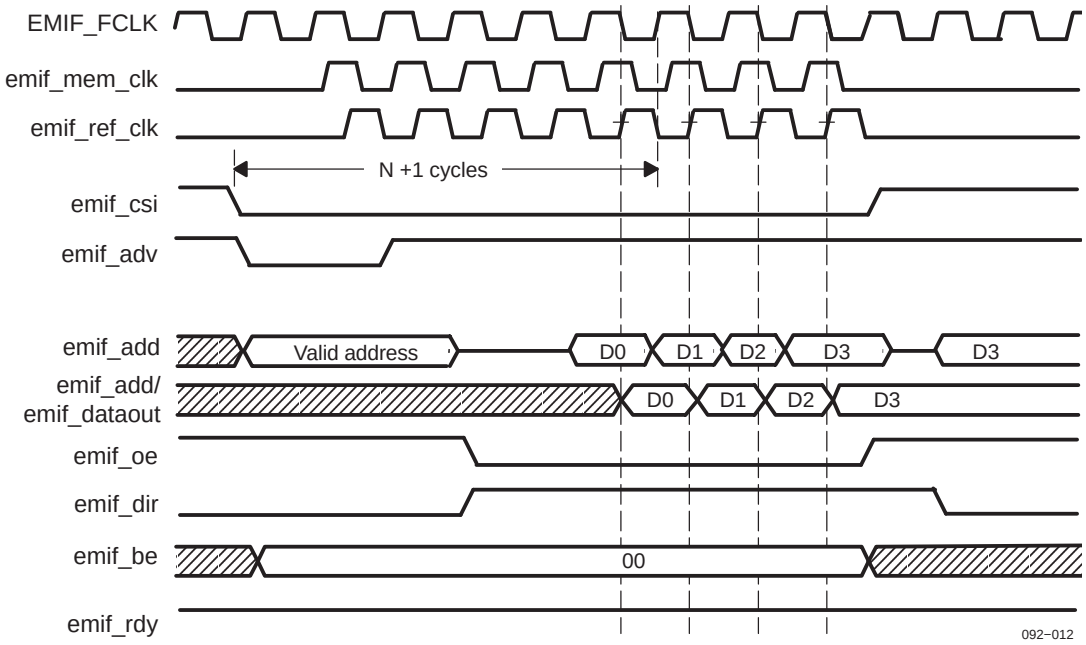
- Retiming mode enables a pipelined read access. Compared to nonretimed access, the first access takes one extra REF_CLK cycle and the following accesses in a burst take one REF_CLK cycle each.
- The retiming mode is enabled through the RT bit EMIF.EMIF_CSi[2].
- Retiming mode is only allowed in synchronous read modes and has no effect on write accesses.
- As in non-retimed mode, chip-select, ADV, address, BE, and OE are driven with respect to REF_CLK.
- Synchronous read programming model and protocol behavior remain the same as in non-retimed mode.
- In the retiming mode, the RDWST is still referenced to REF_CLK. The retiming relaxed timing (extra delay for data valid) is included in the IC timing parameters.
- emif_rdy is also retimed with the emif_ret_clk. The retiming relaxed timing (extra delay for ready valid) is included in the IC timing parameters.

See Figure 11-12 for synchronous burst read operation with retiming.

CAUTION

The RT bit EMIF_CSi[2] can be set only in the synchronous read mode. The system enters into an unrecoverable state (hangs), if the retiming bit is set in other modes (asynchronous modes), because the retiming logic depends on the returned flash clock emif_ret_clk. In asynchronous mode, if there is no flash clock and the system hangs, then a reset is required.

Figure 11-12. Synchronous Burst Read Operation with Retiming on RDWST=3, ADVHOLD=0, OESETUP = 3



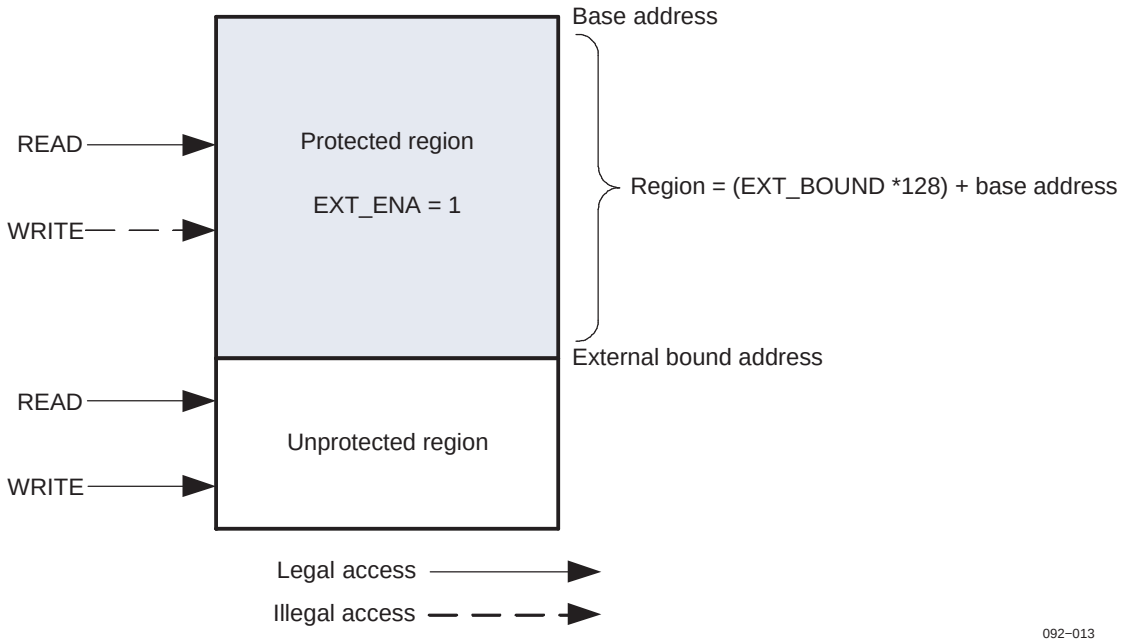
11.2.1.7 Memory Protection

Common flash memory supports the write protection, WP, and input pin (active low) to prevent an erroneous write access sequence from corrupting their content. The EMIF interface includes the nFWP output pin and provides full software control of the nFWP output pin. The WP bit EMIF.EMIF_CONF[6] enables the write protection on the four chip-selects.

In the EMIF, the enhanced memory protection unit (EMPU) controls the write protection attribute of a

particular memory region. This mode protects only one full chip-select address space programmed by CS_EXT EMIF.EMIF_ MASK[9:8]. The granularity available for the external memory is 128 bytes giving a 32 M-byte protection for one CS. The protected memory region is defined by the BOUND_ADD bit field EMIF.EMIF_BOUND[17:0]. The bound address register defines the upper bound of the chip-select protected space from the external memory base address, as shown in Figure 11-13.

Figure 11-13. Protected Memory Example for One Chip-Select



At power-on reset, all control and status register bits are cleared to 0, thus disabling protection for all regions (read/write access allowed). Therefore, after the power-on reset is released, the external bound register frame can be set according to the application-protection scheme. Protection mode on external memory is enabled or disabled by EXT_ENA bit EMIF.EMIF_PRT_MOD[0].

During the process, the incoming external memory address is compared to the bound address, which defines the boundary of the protected region. The memory protection mechanism does not impact on performances because the incoming addresses are compared in parallel while the memory selection is decoded.

When an access tries to write (or write/read) a protected area, an illegal-access signal is asserted to indicate the access has been aborted due to protection. The current write access is disabled meaning no write happens to memory.

The PRO_MOD_ERR bit EMIF.EMIF_ABORT_STA[3] indicates when an error has occurred on a protected mode. In the same manner, the HOST_ID bit field EMIF.EMIF_ABORT_STA[2:1] provides useful information to the user regarding the source that initiates the exception/interruption.

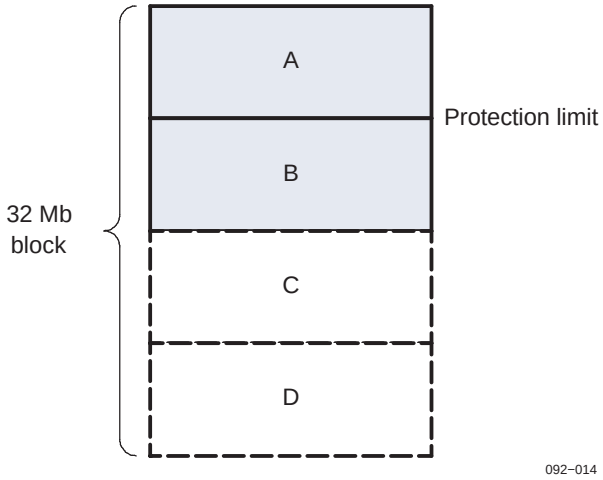
Note: EMPU status registers indicate errors and sources of errors. See Chapter 9, *Enhanced Memory Protection Unit*, for more information.

11.2.1.8 External Memory Protection Address Mask

This feature is used when the memory device is less than 32M bytes of size as defined in the external memory mapping of the LOCOSTO device. To avoid a protection bypass, the MPU can be configured to compare the least-significant bit (LSB) address to only whatever the unused MSBs are.

The PRO_MASK field EMIF.EMIF_MASK[7:0] determines the address bit to be masked. That is, a 16-Mb flash memory is mapped onto a 32M byte external memory chip-select (see Figure 11-14). The address memory bit is connected to the external memory bit address. In this configuration, the LOCOSTO external memory address MSB is not connected to the 32M byte flash memory.

Figure 11-14. Memory Address Mask Example



When the memory protection unit compares all the address bits, it will not protect region C, which is the image of region A, when the address MSB is ignored by the flash memory. To avoid this security opening, the address MSB can be masked internally to enable the protection on the virtual region C. The EMIF will not compare the whole address length and then protects both regions A and C.

- Up to eight address MSBs can be masked, giving a maximum external memory size chip-select ratio of 1:256.
- The memory protection is available for the maximum external memory size of 32MB.
- The external memory minimum size is 256K bytes.
- The protected region minimum size is 256 bytes.

Note: When the external memory size matches the chip-select size, no specific programming is required. External memory address connections (32Mb 25 address bits).

11.2.1.9 Prefetch Buffer

To improve the performance of the MPU to external memory accesses, a 2- line prefetch buffer is in the EMIF. The prefetch unit is placed between the MPU interface and the arbitration logic of the EMIF, as shown in Figure 11-2. The function of the prefetch buffer is similar to a 2-line cache. The prefetch buffer includes the following features:

- Prefetch buffer length is two lines of 4 x 32-bit words.
- Allocation in the prefetch buffer happens in round-robin fashion.
- MPU transactions (instruction and/or data) to external memory cause a wrap 8 burst of the size 8 x 16 on the EMIF interface. The requested word is prefetched first. The consecutive hits in the prefetch buffer are returned to the ARM even as the rest of the line is filled.
- When the prefetch is enabled, all burst transactions are wrap 8 burst.
- Prefetch buffer invalidates the tag bit if any write happens to the same location by the MPU or the DMA.
- Locked accesses are not prefetched; read is followed by write.
- If a read access of the locked accesses hits the data in the prefetch buffer, the data is forwarded to the MPU.
- Invalidated data is prefetched only if the MPU renews its request.
- A prefetch line is invalidated if a write on a locked access hits the tag address.

During no transition, the prefetch buffer can be flushed for proper use by setting the FLUSH_PREFETCH bit EMIF.EMIF_CONF[5] (this bit should be asserted while disabling the prefetch buffer). To re-enable the prefetch mode, set the FLUSH_PREFETCH bit to 0.

The prefetch buffer supports three modes, programmable through the PREFETCH_MODE bit field EMIF.EMIF_CONF[4:3]:

- Prefetch buffer off (All accesses are bypassed through the prefetch buffer.)
- Prefetch buffer support for instruction and data
- Prefetch buffer support for instruction fetches only (NOPC signal from ARM qualifies the instruction or data fetch).

Note: Use the prefetch mode for instruction only to produce the best performance (only in the synchronous mode).

11.2.2 Errors Reporting

The EMIF can generate an abort in two cases:

- Restricted access mode violation

When protection mode is enabled, all the write access within that region is aborted by the EMIF, and an abort signal is sent to the MPU.

- Time-out issue

The time-out feature prevents the system from hanging, in case of extended wait-states. A time-out error is generated when the counter is timed out. An 8-bit value in the TIMEOUT_VAL bit field EMIF.EMIF_ATOR[7:0] is used to initialize the time-out counter. When there is a request from an initiator and it is served by the arbitrator, the counter starts counting down. If the counter reaches 0 before the memory responds, the EMIF generates a time-out error. The TIME_OUT_ERR bit EMIF.EMIF_ATYPER[0] is set. The decrementing of the counter is clocked on the internal clock, REF_CLK. This time-out mode is enabled by the TIMEOUT_ENA bit EMIF.EMIF_ATOR[8].

When an abort occurs, the following occurs:

- The EMIF writes the host ID of the initiator, which caused the abort, in the HOST_ID bit field EMIF.EMIF_ATYPER[2:1]. When there are multiple aborts, only the first abort is registered.
- In case of an abort in any one of the initiators, the EMIF sends the address of the abort access, which is registered in the EMIF.EMIF_AADDR register.

Note: At reset, EMIF_ATYPER and the EMIF_AADDR registers are set to 0 and the TIMEOUT_ENA bit is set with TIMEOUT_VAL set to 0xFF. EMIF_ATYPER and EMIF_AADDR are updated whenever there is an abort for the first time. The ABORT bit EMIF_ATYPER[0] is clear on MPU reads.

11.3 EMIF Programming Model

11.3.1 Setup for Typical Configuration(s)

Depending on the flash/RAM/etc. device associated with each chip-select, the EMIF configuration registers must be initialized. If the device used is a flash, then the flash may have to be initialized according to the correct protocol for maximum performance.

Note: For more information on the configuration and use of EMIF, see the *Locosto Memory Interface Software Programming Guidelines* (APN213).

The following sequence shows a typical flow. The important point to remember is to configure an external flash device in synchronous burst protocol:

1. Read mode
2. Frequency configuration
3. Data output configuration
4. Burst order
5. Burst length (must be greater than 8)
6. CLK configuration
7. Flash mode operation (asynchronous or synchronous)
8. nRDY(Wait) configuration
9. ADV signal for nonmuxed memory

11.3.2 Operational Mode

The following presents some basic programming model for various modes of operations.

11.3.2.1 Asynchronous Read Operation for Flash and CellularRAM (Protocol1/Protocol 2)

The asynchronous read mode is selected by setting MEM_MODE to 000 in the corresponding EMIF chip-select configuration EMIF_CSi. Figure 11-15 shows an asynchronous read operation with multiplexed address/data bus. Table 11-16 shows the timing for ADV and chip-select pulses.

- The REF_CLK is divided from EMIF_FCLK by a programmable value contained in the FCLKDIV bit field EMIF.EMIF_CSi[1:0].
- The chip-select pulse width depends on the RDWST bit field EMIF.EMIF_ CSi[7:4].
  The chip-select pulse is represented by N in Figure 11-15.
- The ADV pulse width depends on the ADVHOLD bit field EMIF.EMIFS_ADV_CSi[8].
  The ADV pulse is represented by M in Figure 11-15.
- The address drive time follows the chip-select activation (no setup time guarantee).
  - Address setup time to ADV rising edge is controlled by ADVHOLD.
  - Address latch occurs on rising edge of ADV.
  - Address deasserted one REF_CLK cycle after ADV is deasserted.
- Read data is latched on the same REF_CLK rising edge that deactivates CSi.
- The OE activation (high to low) delay time from chip-select and address valid is programmable through the OESETUP field EMIF.EMIF_ ADV_CSi[3:0]. Activation delay timing is equals to OESETUP REF_CLK cycles (represented) by OESETUP in Figure 11-15.

CAUTION

The chip-select minimum pulse width is four EMIF_FCLK cycles. Therefore, the OE delay and advance timing value must be set with the following condition: OESETUP is inferior or equal to RDWST.

Noncompliant programming ends up with bad access completion.

- After a read completion, if no other access (RD, WR) is pending, the data bus is driven with the previous read value. The bus turn-around time (OE going high to direction going out) is a minimum of one REF_CLK cycle and can be extended through BTWST.
- In asynchronous mode, REF_CLK is not provided outside the EMIF, and emif_mem_clk is kept low.

Figure 11-15. Asynchronous Read with RDWST = 2, OESETUP = 2; ADVHOLD = 0

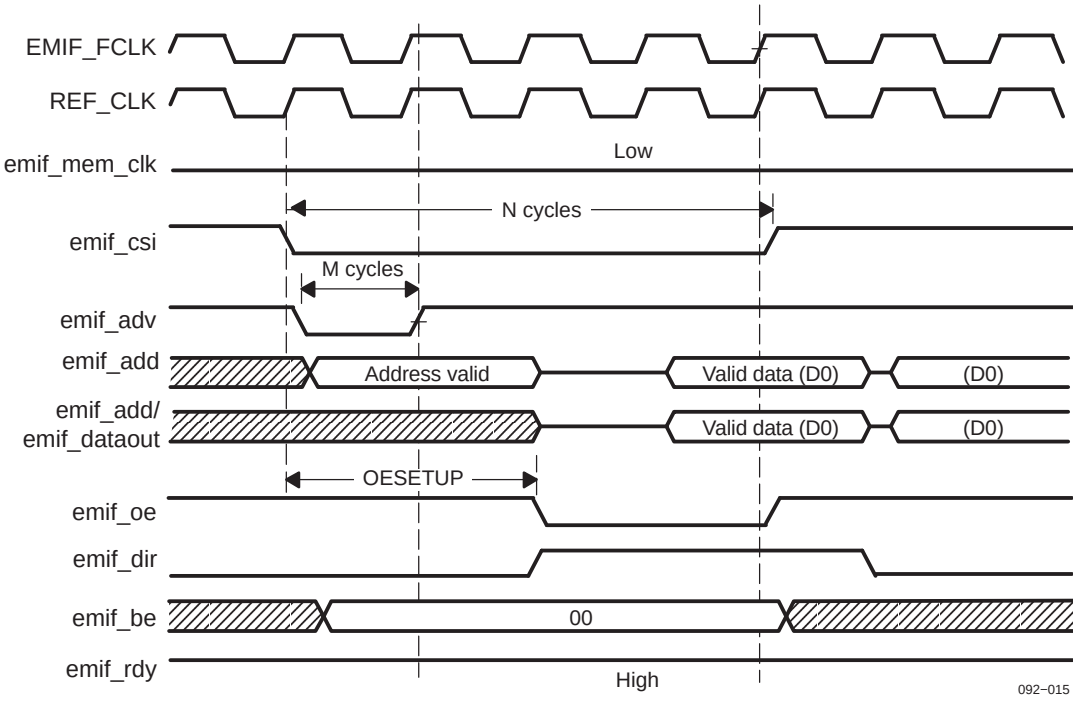


Table 11-16. Asynchronous Read Configuration Timing

FCLKDIV	N (CHi_select pulse)	M (ADV pulse width)	Unit
00	Reserved	Reserved	Reserved
01	2 _UnicodeEncodeError_ (RDWST + 1) + 2	(ADVHOLD + 1) _UnicodeEncodeError_ 2	EMIF_FCLK
10	4 _UnicodeEncodeError_ (RDWST + 1) + 4	(ADVHOLD + 1) _UnicodeEncodeError_ 4	EMIF_FCLK
11	6 _UnicodeEncodeError_ (RDWST + 1) + 6	(ADVHOLD + 1) _UnicodeEncodeError_ 6	EMIF_FCLK

11.3.2.2 Asynchronous Write Operation for Flash Device and CellularRAM (Protocol1 and Protocol2)

The asynchronous write mode is selected by setting MEM_MOD to 000 in the corresponding EMIF chip-select configuration EMIF_CSi. Figure 11-16 and Figure 11-17 show an asynchronous write operation with multiplexed address/data bus for a flash device and CellularRAM, respectively. Table 11-17 shows the timing for ADV and RnW delay and pulse.

- The ADV pulse width depends on ADVHOLD bit field EMIF.EMIFS_ ADV_CSi[8].
  The ADV pulse is represented by M in Figure 11-16 and Figure 11-17.
- RnW (emif_we) is asserted low after a delay equals: WRWST REF_CLK cycles (represented by L in Figure 11-16).

CAUTION

For CellularRAM, WRWST must equal 0 as CellularRAM sample RnW at addressing time to fix on a write or read operation.

- RnW length is determined by:
  - (WELEN + 2) REF_CLK cycles (represented by P in Figure 11-16 and Figure 11-17).
  - The address drive time follows the chip-select activation (no setup time guarantee).
  - Address setup time to ADV rising edge is controlled by ADVHOLD.
  - Address latch occurs on rising edge of ADV.
  - Address deasserted one REF_CLK cycle after ADV is deasserted
- The device latches data on rising edge of emif_we.
- The device latches data on rising edge of emif_we.
- In asynchronous mode, EMIF_FCLK is not provided outside the EMIF, and emif_mem_clk is kept low.

Figure 11-16. Asynchronous Write for Flash Device with WRWST = 2, WELEN = 2, ADVHOLD = 0, Q = 1

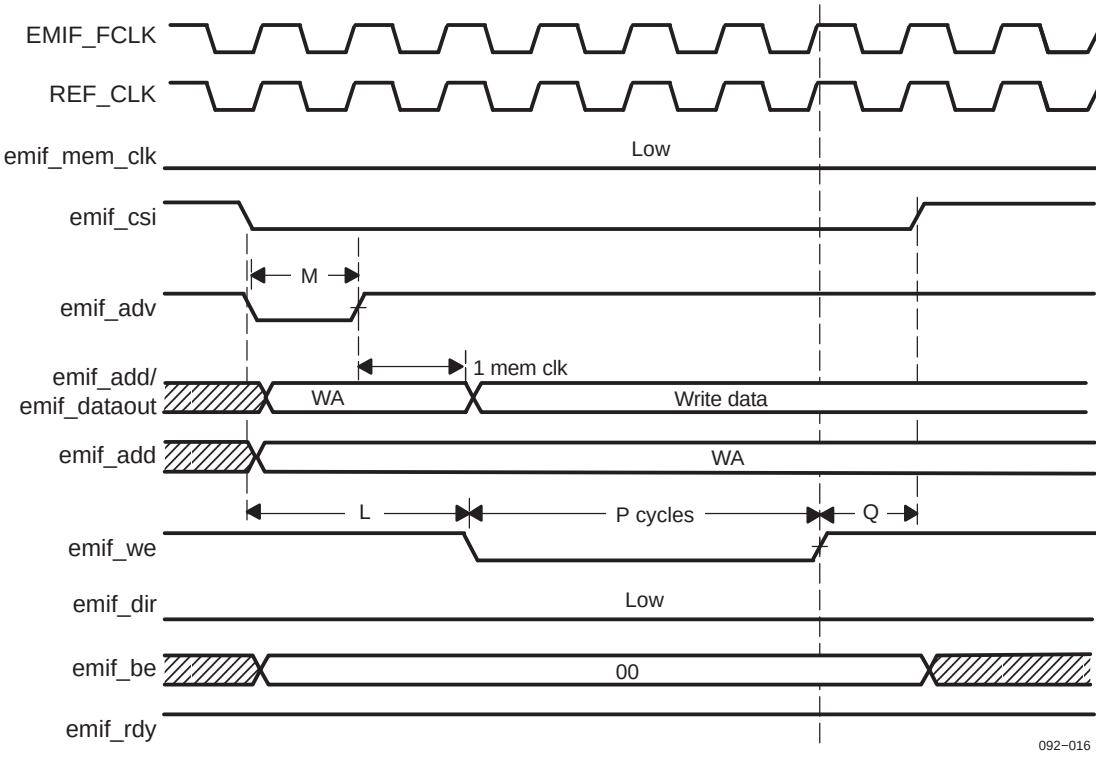
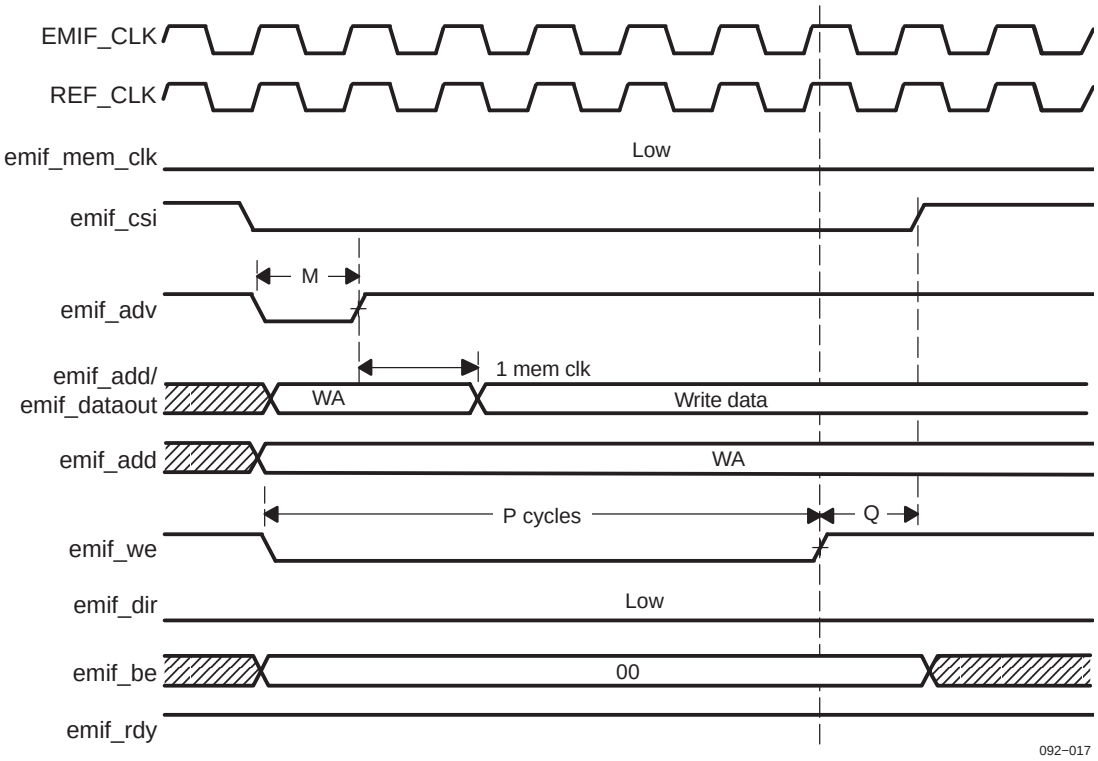


Table 11-17. Asynchronous Write Configuration Timing for Flash and CellularRAM Device

FCLKDIV	L	P	M (ADV pulse width)	Q	Unit
00	Reserved	Reserved	Reserved	Reserved	Reserved
01	2 * (WRWST)	2 * (WELEN + 2)	(ADVHOLD + 1) _UnicodeEncodeError_ 2	2	EMIF_FCLK
10	4 * (WRWST)	4 * (WELEN + 2)	(ADVHOLD + 1) _UnicodeEncodeError_ 4	4	EMIF_FCLK
11	6 * (WRWST)	6 * (WELEN + 2)	(ADVHOLD + 1) _UnicodeEncodeError_ 6	6	EMIF_FCLK

Figure 11-17. Asynchronous Write for CellularRAM Device with WRWST = 0, WELEN = 4, ADVHOLD = 0, Q = 1



11.3.2.3 Synchronous Burst Read Operation for Flash Device and CellularRAM (Protocol1 and Protocol2)

The synchronous read mode is selected by setting MEM_MODE to 001 or 010 for flash and CellularRAM device, respectively, in the corresponding EMIF chip-select configuration EMIF_CSi. Figure 11-18 shows a synchronous read operation with multiplexed address/data bus. Table 11-18 shows the timing for ADV and chip-select pulses.

Note: During burst read, the timings are the same when using a flash or a Cellular- RAM device.

Figure 11-18. Synchronous Burst Read with RDWST = 2 , FCLKDIV =1, ADVHOLD = 0, OESETUP = 3.

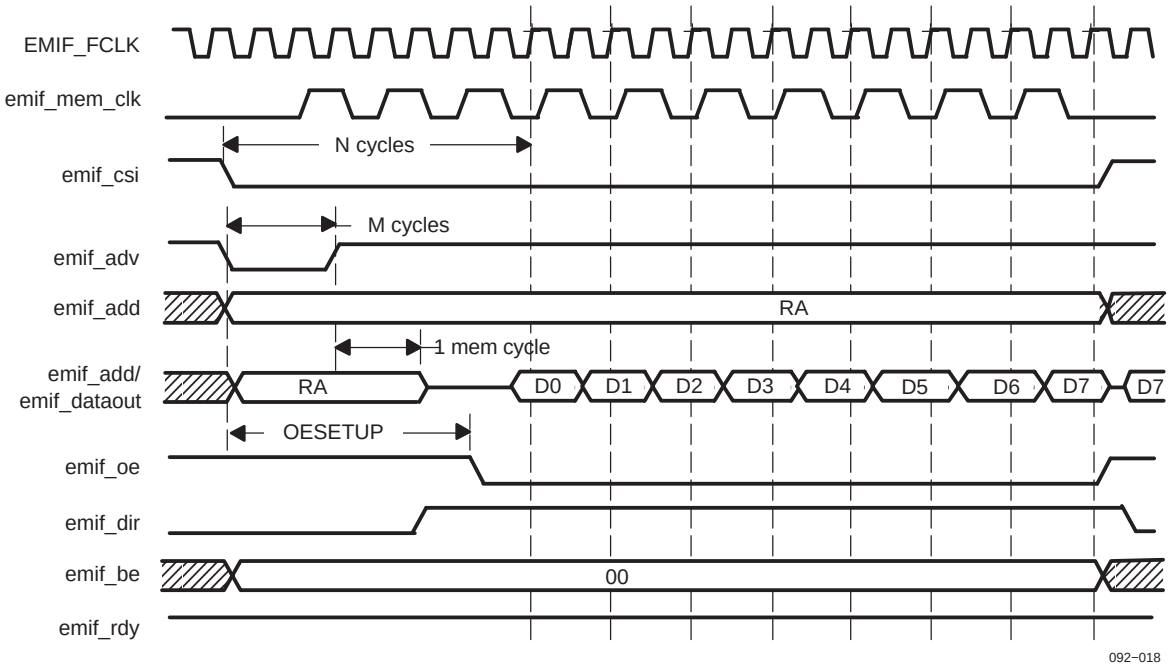


Table 11-18. Synchronous Read Configuration Timing

FCLKDIV	N (Chip-select Pulse)	M (ADV Pulse Width)	Unit
00	Reserved	Reserved	Reserved
01	2 _UnicodeEncodeError_ (RDWST + 1) + 2	1 + (ADVHOLD + 1) _UnicodeEncodeError_ 2	EMIF_FCLK
10	4 _UnicodeEncodeError_ (RDWST + 1) + 4	1 + (ADVHOLD + 1) _UnicodeEncodeError_ 4	EMIF_FCLK
11	6 _UnicodeEncodeError_ (RDWST + 1) + 6	1 + (ADVHOLD + 1) _UnicodeEncodeError_ 6	EMIF_FCLK

- The chip-select pulse width depends on RDWST bit field EMIF_CSi[7:4]. The chip-select pulse width equals:
  (2 * (RDWST + 1) + 2) EMIF_FCLK cycles (represented by N in Figure 11-18).
- The ADV pulse width depends on the ADVHOLD bit field EMIFS_ADV_CSi[8]. The ADV pulse width equals:
  ((ADVHOLD + 1) * 2 + 1) EMIF_FCLK cycles (represented by M in Figure 11-18).
- Address latch occurs on the rising edge of emif_mem_clk while ADV is low.
- Address is deasserted one REF_CLK cycle after ADV is deasserted.
- The OE activation (high to low) delay time from chip-select and address valid is programmable through OESETUP bit field EMIF_ADV_CSi[3:0]. Activation delay timing equals OESETUP REF_CLK cycles (represented by OESETUP in Figure 11-18).
- Read data is latched after a delay equal to
  – 2 * (RDWST + 1) + 2) EMIF_FCLK cycles (represented by N in Figure 11-18
- After the first data is available, the next data are available at each rising edge of emif_mem_clk.
- Synchronous mode, REF_CLK is provided outside the EMIF via emif_mem_clk.

CAUTION

If ADVHOLD is set to 0, then OESETUP should be at least 3 to satisfy the hold time requirement for the address

If ADVHOLD is set to 1, then OESETUP should be at least 4 to satisfy the hold time requirement for the address.

The data enable signal (emif_dir) is asserted high (out to in) only after the address phase.

11.3.2.4 Synchronous Burst Write Operation for CellularRAM (Protocol2)

The synchronous write mode is selected by setting MEM_MODE to 010 in the corresponding EMIF chip-select configuration EMIF_CSi for a CellularRAM device. Figure 11-19 shows a synchronous write operation with multiplexed address/data bus. Table 11-19 shows the timing for ADV and the delay to write the first data.

Figure 11-19. Synchronous Burst Write CellularRAM with FCLKDIV = 01, ADVHOLD = 1, WELEN = 7.

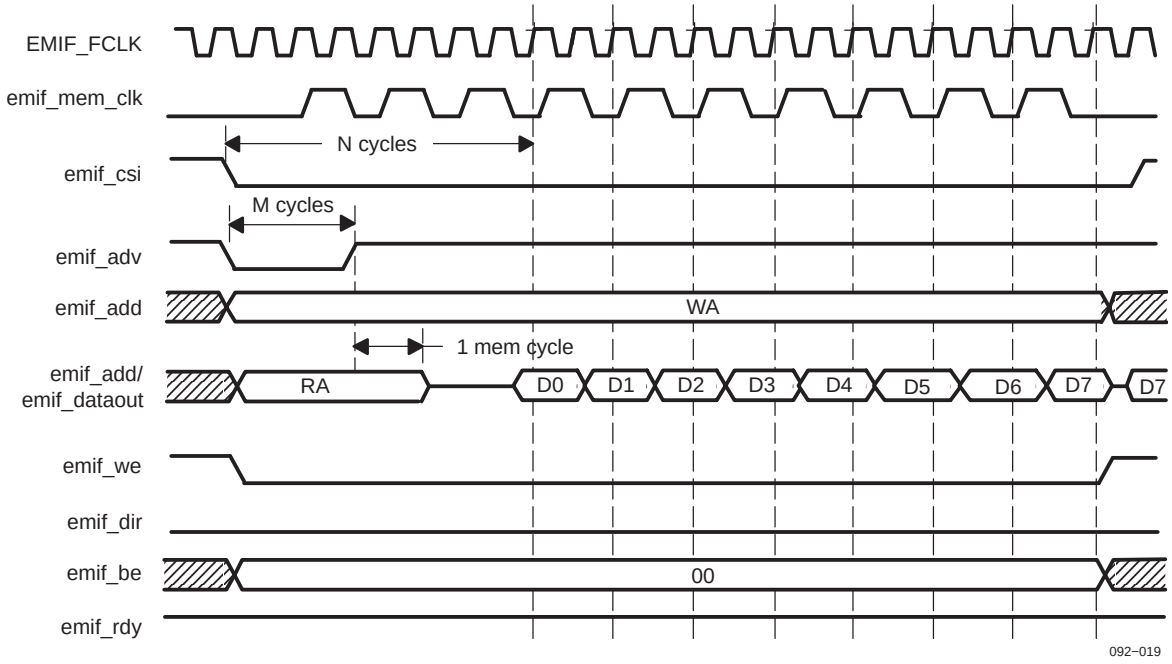


Table 11-19. Synchronous Write Configuration Timing for CellularRAM Devices

FCLKDIV	N (	M (ADV Pulse Width)	Unit
00	Reserved	Reserved	Reserved
01	2 _UnicodeEncodeError_ (WELEN + 1)	(ADVHOLD + 1) _UnicodeEncodeError_ 2 + 1	EMIF_FCLK
10	4 _UnicodeEncodeError_ (WELEN + 1)	(ADVHOLD + 1) _UnicodeEncodeError_ 4 + 1	EMIF_FCLK
11	6 _UnicodeEncodeError_ (WELEN + 1)	(ADVHOLD + 1) _UnicodeEncodeError_ 6 + 1	EMIF_FCLK

- The ADV pulse width depends on the ADVHOLD bit field EMIF_ADV_CSi[8]. The ADV pulse width equals:
  ((ADVHOLD + 1) * 2 + 1) EMIF_FCLK cycles (represented by M in Figure 11-19).

The address latch occurs on the rising edge of emif_mem_clk while ADV is low.

Address is deasserted one REF_CLK cycle after ADV is deasserted.

- First, data is sent according to
  Reserved (WELEN + 1) REF_CLK cycles (represented by N in Figure 11-19).
- After the first data is sent, the remaining data are sent at each rising edge of emif_mem_clk.
- In synchronous mode, the REF_CLK is provided outside the EMIF by emif_mem_clk.

11.4 EMIF Register Manual

Table 11-20. Instance Summary

Module Name	Base Address
External memory interface	0xFFFF FB00

11.4.1 Module Register Mapping Summary

Table 11-21. External Memory Interface Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
EMIF_LRU_PRIO	R/W	16	0x00	0xFFFF FB00
EMIF_CONF	R/W	16	0x02	0xFFFF FB02
EMIF_CS0	R/W	32	0x04	0xFFFF FB04
EMIF_CS1	R/W	32	0x08	0xFFFF FB08
EMIF_CS2	R/W	32	0x0C	0xFFFF FB0C
EMIF_CS3	R/W	32	0x10	0xFFFF FB10
EMIF_ADV_CS0	R/W	16	0x14	0xFFFF FB10
EMIF_ADV_CS1	R/W	16	0x16	0xFFFF FB16
EMIF_ADV_CS2	R/W	16	0x18	0xFFFF FB16
EMIF_ADV_CS3	R/W	16	0x1A	0xFFFF FB16
EMIF_DYN_WS	R/W	16	0x1C	0xFFFF FB16
EMIF_ATOR	R/W	16	0x1E	0xFFFF FB16
EMIF_PRT_MOD	R/W	16	0x20	0xFFFF FB16
EMIF_LOWER_BOUND	R/W	16	0x22	0xFFFF FB16
EMIF_UPPER_BOUND	R/W	16	0x24	0xFFFF FB24
EMIF_MASK	R/W	16	0x26	0xFFFF FB26
EMIF_AADD	R	32	0x28	0xFFFF FB26
EMIF_ATYPER	R	16	0x2C	0xFFFF FB2C

11.4.2 Register Description

Table 11-22. EMIF_LRU_PRIO

Address Offset	0x00		
Physical Address	0xFFFF FB00	**Instance**	External Memory Interface
Description	Set up consecutive access to MPU and DMA for LRU priority arbitration		
Type	RW		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DMA_ACCESS				Reserved	MPU_ACCESS		

Bits	Field Name	Description	Type	Reset
15:8	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.	R	0x00
7:4	DMA_ACCESS	Set up DMA consecutive access from 1 to 15 for LRU priority	RW	0x0
3	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.	R	Undefined
2:0	MPU_ACCESS	Set up MPU consecutive access from 1 to 8 for LRU priority	RW	0x0

Table 11-23. EMIF_CONF

Address Offset	0x02		
Physical Address	0xFFFF FB02	**Instance**	External Memory Interface
Description			
Type	RW		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WP	FLUSH_PREFETCH	PREFETCH_MODE		FR	PDE	PWD_ENA	

Bits	Field Name	Description	Type	Reset
15:7	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.	R	Undefined
6	WP	Write protect output pin control: 1: WP output pin is set high, all sectors are writable. 0: WP output pin is set low, disable write in the external memory.	RW	0x00
5	FLUSH_PREFETCH	Flushes the prefetched buffer.	RW	0
4:3	PREFETCH_MODE	Prefetch mode: 00: Prefetch off. 01: Prefetch for instruction and data. 10: Prefetch for instruction only. 11: Reserved.	RW	0x0
2	FR	Sampled from the nRDY pin: 1: nRDY pin is high. 0: nRDY pin is low.	R	1
1	PDE	System power-down acknowledge: 1: Acknowledge enabled. 0: Acknowledge disabled.	RW	1
0	PWD_ENA	Auto-Idle enable signal: 1: Enable EMIF power-down mode. 0: Disable EMIF power-down mode	RW	1

Table 11-24. EMIF_CS0

Address Offset	0x04		
Physical Address	0xFFFF FB04	**Instance**	External Memory Interface
Description	Set the handshaking protocol to be used with the device connected to the chip-select.		
Type	RW		
Write Latency	Not relevant		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BTWST				Reserved		MEM_MOD		WELEN				WRWST				RDWST				Reserved	RT	FCLKDIV	

Bits	Field Name	Description			Type	Reset
31:26	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined			R	Undefined
25:22	BTWST	Number of wait-states inserted during read-to-write transition			RW	0x0
21:19	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.			R	Undefined
18:16	MEM_MOD		**Read Operation**	**Write Operation**	RW	0x0
		000:	Asynchronous read	Asynchronous write		
		001	Sync burst read protocol1	Asynchronous write		
		010:	Sync burst read protocol2	Sync burst write protocol2		
		011:	Reserved	Reserved		
		100:	Reserved	Reserved		
		101:	Reserved	Reserved		
		110:	Reserved	Reserved		
		111	Reserved	Reserved		
15:12	WELEN	For write access, length of WE pulse duration			RW	0xF
11:8	WRWST	Number of wait-states for write operation			RW	0xF
7:4	RDWST	Number of wait-states for read operation			RW	0xF
3	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.			R	Undefined
2	RT	1: Data are retimed with REF_CLK			RW	0
		0: Data are not retimed				
1:0	FCLKDIV	Flash Clock Divider:			RW	0x01
		00: Reserved				
		01: Divide by 2				
		10: Divide by 4				
		11: Divide by 6				

Table 11-25. EMIF_CS1

Address Offset	0x08		
Physical Address	0xFFFF FB08	**Instance**	External Memory Interface
Description	Set the handshaking protocol to be used with the device connected to the chip-select.		
Type	RW		
Write Latency	Not relevant		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						BTWST				Reserved			MEM_MOD		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WELEN				WRWST				RDWST				Reserved	RT	FCLKDIV	

Bits	Field Name	Description	Type	Reset
31:26	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined	R	Undefined
25:22	BTWST	Number of wait-states inserted during read-to-write transition	RW	0x0
21:19	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.	R	Undefined

Bits	Field Name		Description		Type	Reset
18:16	MEM_MOD		**Read Operation**	**Write Operation**	RW	0x0
		000:	Asynchronous read	Asynchronous write		
		001	Sync burst read protocol1	Asynchronous write		
		010:	Sync burst read protocol2	Sync burst write protocol2		
		011:	Reserved	Reserved		
		100:	Reserved	Reserved		
		101:	Reserved	Reserved		
		110:	Reserved	Reserved		
		111	Reserved	Reserved		
15:12	WELEN	For write access, length of WE pulse duration			RW	0xF
11"8	WRWST	Number of wait-states for write operation			RW	0xF
7:4	RDWST	Number of wait-states for read operation			RW	0xF
3	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.			R	Undefined
2	RT	1: Data are retimed with REF_CLK			RW	0
		0: Data are not retimed				
1:0	FCLKDIV	Flash Clock Divider:			RW	0x01
		00: Reserved				
		01: Divide by 2				
		10: Divide by 4				
		11: Divide by 6				

Table 11-26. EMIF_CS2

Address Offset	0x0C		
Physical Address	0xFFFF FB0C	**Instance**	External Memory Interface
Description	Set the handshaking protocol to be used with the device connected to the chip-select.		
Type	RW		
Write Latency	Not relevant		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						BTWST				Reserved			MEM_MOD		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WELEN				WRWST				RDWST				Reserved	RT	FCLKDIV	

Bits	Field Name		Description		Type	Reset
31:26	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined			R	Undefined
25:22	BTWST	Number of wait-states inserted during read-to-write transition			RW	0x0
21:19	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.			R	Undefined
18:16	MEM_MOD		**Read Operation**	**Write Operation**	RW	0x0
		000:	Asynchronous read	Asynchronous write		
		001	Sync burst read protocol1	Asynchronous write		
		010:	Sync burst read protocol2	Sync burst write protocol2		
		011:	Reserved	Reserved		
		100:	Reserved	Reserved		
		101:	Reserved	Reserved		
		110:	Reserved	Reserved		
		111	Reserved	Reserved		
15:12	WELEN	For write access, length of WE pulse duration			RW	0xF
11"8	WRWST	Number of wait-states for write operation			RW	0xF

Bits	Field Name	Description	Type	Reset
7:4	RDWST	Number of wait-states for read operation	RW	0xF
3	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.	R	Undefined
2	RT	1: Data are retimed with REF_CLK 0: Data are not retimed	RW	0
1:0	FCLKDIV	Flash Clock Divider: 00: Reserved 01: Divide by 2 10: Divide by 4 11: Divide by 6	RW	0x01

Table 11-27. EMIF_CS3

Address Offset	0x10		
Physical Address	0xFFFF FB10	**Instance**	External Memory Interface
Description	Set the handshaking protocol to be used with the device connected to the chip-select.		
Type	RW		
Write Latency	Not relevant		

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4	3	2	1 0
Reserved	BTWST / Reserved / MEM_MOD	WELEN	WRWST	RDWST	Reserved	RT	FCLKDIV

Bits	Field Name	Description			Type	Reset
31:26	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined			R	Undefined
25:22	BTWST	Number of wait-states inserted during read-to-write transition			RW	0x0
21:19	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.			R	Undefined
18:16	MEM_MOD		**Read Operation**	**Write Operation**	RW	0x0
		000:	Asynchronous read	Asynchronous write		
		001	Sync burst read protocol1	Asynchronous write		
		010:	Sync burst read protocol2	Sync burst write protocol2		
		011:	Reserved	Reserved		
		100:	Reserved	Reserved		
		101:	Reserved	Reserved		
		110:	Reserved	Reserved		
		111	Reserved	Reserved		
15:12	WELEN	For write access, length of WE pulse duration			RW	0xF
11:8	WRWST	Number of wait-states for write operation			RW	0xF
7:4	RDWST	Number of wait-states for read operation			RW	0xF
3	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.			R	Undefined
2	RT	1: Data are retimed with REF_CLK 0: Data are not retimed			RW	0
1:0	FCLKDIV	Flash Clock Divider: 00: Reserved 01: Divide by 2 10: Divide by 4 11: Divide by 6			RW	0x01

Table 11-28. EMIF_ADV_CS0

Address Offset	0x14		
Physical Address	0xFFFF FB14	**Instance**	External Memory Interface
Description	Extension of the configuration EMIF_CS0		
Type	RW		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RC	CLKMASK	BTMODE	ADVHOLD	Reserved				OESETUP			

Bits	Field Name	Description	Type	Reset
15:12	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.	R	0x00
11	RC	Facilitates use of the nRDY signal for different types of flash devices. This must be 0 to maintain compatibility with legacy software. 0: nRDY is asserted one EMIF_MEM_CLK cycle before the data phase. 1: nRDY is asserted in the same EMIF_MEM_CLK cycle as the data phase.	RW	0
10	CLKMASK	Clock masking for synchronous mode writes 0: EMIF_MEM_CLK is sent to the device during writes in synchronous modes. 1: EMIF_MEM_CLK is masked and is held low on the device during writes.	RW	0
9	BTMODE	Enables extended BTWST usage: 0: Bus turn-around cycles are inserted between each read operation to avoid data contention. 1: Bus turn-around cycles are inserted between WR RD and WR→ WR of the same chip-select apart from the RD to any transaction.	RW	0
8	ADVHOLD	Hold cycle for address valid signal in async and sync protocol1 and protocol2: 0: Hold cycle, ADV length is one flash clock cycle. 1: One flash clock cycle hold, ADV length is two flash clock cycles.	RW	0
7:4	Reserved	Reserved	R	Undefined
3:0	OESETUP	Controls the number of cycles inserted from chip -select low to OE low.	RW	0x2

Table 11-29. EMIF_ADV_CS1

Address Offset	0x16		
Physical Address	0xFFFF FB16	**Instance**	External Memory Interface
Description	Extension of the configuration EMIF_CS1		
Type	RW		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RC	CLKMASK	BTMODE	ADVHOLD	Reserved				OESETUP			

Bits	Field Name	Description	Type	Reset
15:12	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.	R	0x00
11	RC	Facilitates use of the nRDY signal for different types of flash devices. This must be 0 to maintain compatibility with legacy software. 0: nRDY is asserted one EMIF_MEM_CLK cycle before the data phase. 1: nRDY is asserted in the same EMIF_MEM_CLK cycle as the data phase.	RW	0
10	CLKMASK	Clock masking for synchronous mode writes 0: EMIF_MEM_CLK is sent to the device during writes in synchronous modes. 1: EMIF_MEM_CLK is masked and is held low on the device during writes.	RW	0
9	BTMODE	Enables extended BTWST usage: 0: Bus turn-around cycles are inserted between each read operation to avoid data contention. 1: Bus turn-around cycles are inserted between WR RD and WR→ WR of the same chip-select apart from the RD to any transaction.	RW	0
8	ADVHOLD	Hold cycle for address valid signal in async and sync protocol1 and protocol2: 0: Hold cycle, ADV length is one flash clock cycle. 1: One flash clock cycle hold, ADV length is two flash clock cycles.	RW	0
7:4	Reserved	Reserved	R	Undefined
3:0	OESETUP	Controls the number of cycles inserted from chip -select low to OE low.	RW	0x2

Table 11-30. EMIF_ADV_CS2

Address Offset	0x18		
Physical Address	0xFFFF FB18	**Instance**	External Memory Interface
Description	Extension of the configuration EMIF_CS2		
Type	W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RC	CLKMASK	BTMODE	ADVHOLD	Reserved				OESETUP			

Bits	Field Name	Description	Type	Reset
15:12	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.	R	0x00
11	RC	Facilitates use of the nRDY signal for different types of flash devices. This must be 0 to maintain compatibility with legacy software. 0: nRDY is asserted one EMIF_MEM_CLK cycle before the data phase. 1: nRDY is asserted in the same EMIF_MEM_CLK cycle as the data phase.	RW	0
10	CLKMASK	Clock masking for synchronous mode writes 0: EMIF_MEM_CLK is sent to the device during writes in synchronous modes. 1: EMIF_MEM_CLK is masked and is held low on the device during writes.	RW	0
9	BTMODE	Enables extended BTWST usage: 0: Bus turn-around cycles are inserted between each read operation to avoid data contention. 1: Bus turn-around cycles are inserted between WR RD and WR→ WR of the same chip-select apart from the RD to any transaction.	RW	0
8	ADVHOLD	Hold cycle for address valid signal in async and sync protocol1 and protocol2:	RW	0

Bits	Field Name	Description	Type	Reset
		0: Hold cycle, ADV length is one flash clock cycle.		
		1: One flash clock cycle hold, ADV length is two flash clock cycles.		
7:4	Reserved	Reserved	R	Undefined
3:0	OESETUP	Controls the number of cycles inserted from chip -select low to OE low.	RW	0x2

Table 11-31. EMIF_ADV_CS3

Address Offset	0x1A		
Physical Address	0xFFFF FB1A	**Instance**	External Memory Interface
Description	Extension of the configuration EMIF_CS3		
Type	RW		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RC	CLKMASK	BTMODE	ADVHOLD	Reserved				OESETUP			

Bits	Field Name	Description	Type	Reset
15:12	Reserved	To ensure software compatibility, the reserved bit should be written to 0s and the read value considered undefined.	R	0x00
11	RC	Facilitates use of the nRDY signal for different types of flash devices. This must be 0 to maintain compatibility with legacy software.	RW	0
		0: nRDY is asserted one EMIF_MEM_CLK cycle before the data phase.		
		1: nRDY is asserted in the same EMIF_MEM_CLK cycle as the data phase.		
10	CLKMASK	Clock masking for synchronous mode writes	RW	0
		0: EMIF_MEM_CLK is sent to the device during writes in synchronous modes.		
		1: EMIF_MEM_CLK is masked and is held low on the device during writes.		
9	BTMODE	Enables extended BTWST usage:	RW	0
		0: Bus turn-around cycles are inserted between each read operation to avoid data contention.		
		1: Bus turn-around cycles are inserted between WR RD and WR→ WR of the same chip-select apart from the RD to any transaction.		
8	ADVHOLD	Hold cycle for address valid signal in async and sync protocol1 and protocol2:	RW	0
		0: Hold cycle, ADV length is one flash clock cycle.		
		1: One flash clock cycle hold, ADV length is two flash clock cycles.		
7:4	Reserved	Reserved	R	Undefined
3:0	OESETUP	Controls the number of cycles inserted from chip -select low to OE low.	RW	0x2

Table 11-32. EMIF_DWS

Address Offset	0x1C		
Physical Address	0xFFFF FB1C	**Instance**	External Memory Interface
Description	Enables EMIF full-handshaking or non-full-handshaking protocol with masks during write access		
Type	RW		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WRRDYMASK_CS3	WRRDYMASK_CS2	WRRDYMASK_CS1	WRRDYMASK_CS0	HDK_CS3	HDK_CS2	HDK_CS1	HDK_CS0

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	0x00
7	WRRDYMASK_CS3	Mask nRDY during a write operation for CS0: 0: nRDY signal is monitored during write access if full-handshaking mode is enabled. 1: Masks nRDY signal during a write access irrespective of full-handshaking mode.	RW	0
6	WRRDYMASK_CS2	Mask nRDY during a write operation for CS0: 0: nRDY signal is monitored during write access if full-handshaking mode is enabled. 1: Masks nRDY signal during a write access irrespective of full-handshaking mode.	RW	0
5	WRRDYMASK_CS1	Mask nRDY during a write operation for CS0: 0: nRDY signal is monitored during write access if full-handshaking mode is enabled. 1: Masks nRDY signal during a write access irrespective of full-handshaking mode.	RW	0
4	WRRDYMASK_CS0	Mask nRDY during a write operation for CS0 0: nRDY signal is monitored during write access if full-handshaking mode is enabled. 1: Masks nRDY signal during a write access irrespective of full-handshaking mode.	RW	0
3	HDK_CS3	Enables full-/non-full-handshaking mode for CS3: 0: Full-handshaking 1: Non-full-handshaking	RW	1
2	HDK_CS2	Enables full-/non-full-handshaking mode for CS2: 0: Full-handshaking 1: Non-full-handshaking	RW	1
1	HDK_CS1	Enables full-/non-full-handshaking mode for CS1: 0: Full-handshaking 1: Non-full-handshaking	RW	1
0	HDK_CS0	Enables full-/non-full-handshaking mode for CS0: 0: Full-handshaking 1: Non-full-handshaking	RW	1

Table 11-33. EMIF_ATOR

Address Offset	0x1E		
Physical Address	0xFFFF FB1E	**Instance**	External Memory Interface
Description			
Type	RW		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							TIMEOUT_ENA	TIMEOUT_VAL							

Bits	Field Name	Description	Type	Reset
15:9	Reserved	Write has no functional effect; read returns undefined.	R	0x00
8	TIMEOUT_ENA	Enable the time-out counter: 0: Counter is disable. 1: Counter is enabled and abort is generated if counter equals 0.	RW	1
7:0	TIMEOUT_VAL	Time-out counter value in cycles	RW	0xFF

Table 11-34. EMIF_PRT_MODE

Address Offset	0x20		
Physical Address	0xFFFF FB20	**Instance**	External Memory Interface
Description	Enable read-only external feature.		
Type	RW		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EXT_ENA

Bits	Field Name	Description	Type	Reset
15:1	Reserved	Write has not functional effect; read returns undefined.	R	Undefined
0	EXT_ENA	Enable the read-only external protection: 1: Enable read-only, EMIF_BOUND cannot be modified. 0: Disable read-only.	RW	0x00

Table 11-35. EMIF_LOWER_BOUND

Address Offset	0x22		
Physical Address	0xFFFF FB22	**Instance**	External Memory Interface
Description	Lower address of the protected space		
Type	RW		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOWER_BOUND															

Bits	Field Name	Description	Type	Reset
15:0	LOWER_BOUND	Lower bound address with a granularity of 128 bytes	RW	0x00

Table 11-36. EMIF_UPPER_BOUND

Address Offset	0x24		
Physical Address	0xFFFF FB24	**Instance**	External Memory Interface
Description	Upper bound of protected space		
Type	RW		

Table 11-36. EMIF_UPPER_BOUND (continued)

Write Latency	Not relevant

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														UPPER_BOUND	

Bits	Field Name	Description	Type	Reset
15:2	Reserved	Write has no functional effect; read returns undefined.	R	Undefined
1:0	UPPER_BOUND	Upper bound address	RW	0x00

Table 11-37. EMIF_MASK

Address Offset	0x26		
Physical Address	0xFFFF FB26	**Instance**	External Memory Interface
Description			
Type	RW		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CS_EXT		PRO_MASK							

Bits	Field Name	Description	Type	Reset
15:10	Reserved	For future compatibility, write 0s and read is undefined.	R	Undefined
9:8	CS_EXT	Enable external memory protection on CS: 00: Chip-select CS0 01: Chip-select CS1 10: Chip-select CS2 11: Chip-select CS3	RW	0x0
7:0	PRO_MASK	Define the external memory masked address	RW	0x00

Table 11-38. EMIF_AADD

Address Offset	0x28		
Physical Address	0xFFFF FB28	**Instance**	External Memory Interface
Description	Report the address access during an abort.		
Type	R		
Write Latency	Not relevant		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABORT_ADD																															

Bits	Field Name	Description	Type	Reset
31:0	ABORT_ADD	Address of the abort	RW	0x00

Table 11-39. EMIF_ATYPER

Address Offset	0x2C		
Physical Address	0xFFFF FB2C	**Instance**	External Memory Interface
Description	Show the status of the abort_UnicodeEncodeError_		
Type	R		

Table 11-39. EMIF_ATYPER (continued)

Write Latency Not relevant

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											TIME_OUT_ERR	PRO_MOD_ERR	HOST_ID		ABORT

Bits	Field Name	Description	Type	Reset
15:5	Reserved	Reserved	R	Undefined
4	TIME_OUT_ERR	1: Time-out error occurred 0: No time-out error	RW	0x00
3	PRO_MOD_ERR	1: Protected mode error occurred 0: No protected mode error	RW	0
2:1	HOST_ID	Source of error: 00: MPU 01: DMA 10: Reserved 11: Reserved	RW	0x0
0	ABORT	1: Abort occurred, cleared by read from MPU 0: No abort	RW	0

Interrupt Handlers

This chapter describes the interrupt handler subsystem, which provides interrupt service for the LOCOSTO device.

Topic **Page**

12.1 Interrupt Handlers Overview

The LOCOSTO device uses three modules to handle interrupts from different functional blocks, to prioritize and mask them as required, and to route them to the microprocessor unit (MPU) and the digital signal processor (DSP) core. The MPU has two levels of interrupt handlers:

- The lowest level is the secure interrupt handler, which handles five interrupts coming from secure modules.
- The higher level is the master interrupt handler, which services up to 31 interrupts, including the resulting interrupt from the secure interrupt handler.

Both MPU interrupt handlers are independently programmable for priority, masking, and interrupt signal-level sensitivity (low-level or falling-edge triggering).

- MPU secure interrupt handler
  - The MPU secure interrupt handler prioritizes, masks secure interrupt requests (IRQs) from secure modules internal to the LOCOSTO, and then sends them as standard IRQs to the master interrupt handler. For details about IRQs, see the ARM7TDMIE reference manual.
- MPU interrupt handler
  - The MPU interrupt handlers prioritize interrupts received from functional blocks in the LOCOSTO, or from external modules through GPIO0, 1, and 2. Interrupts are then routed as a standard IRQ or a fast interrupt request (FIQ) to the MPU core. For details about FIQs, see the ARM7TDMIE reference manual.
  - The FIQ permits fast servicing of interrupts and can interrupt an existing IRQ. IRQ and FIQ signals are active low-level interrupts, synchronous with the interrupt-handler functional clock.

The DSP has one interrupt handler, which services up to 12 interrupts coming from internal modules.

- The DSP interrupt handler allows selection of the level sensitivity and the routing of all direct interrupts to the DSP core.
- Contrary to the MPU interrupts, the DSP interrupts are cleared, masked, and prioritized at the DSP core level. See the *C54x cDSPReference Set Vol 1: CPU* for more details.

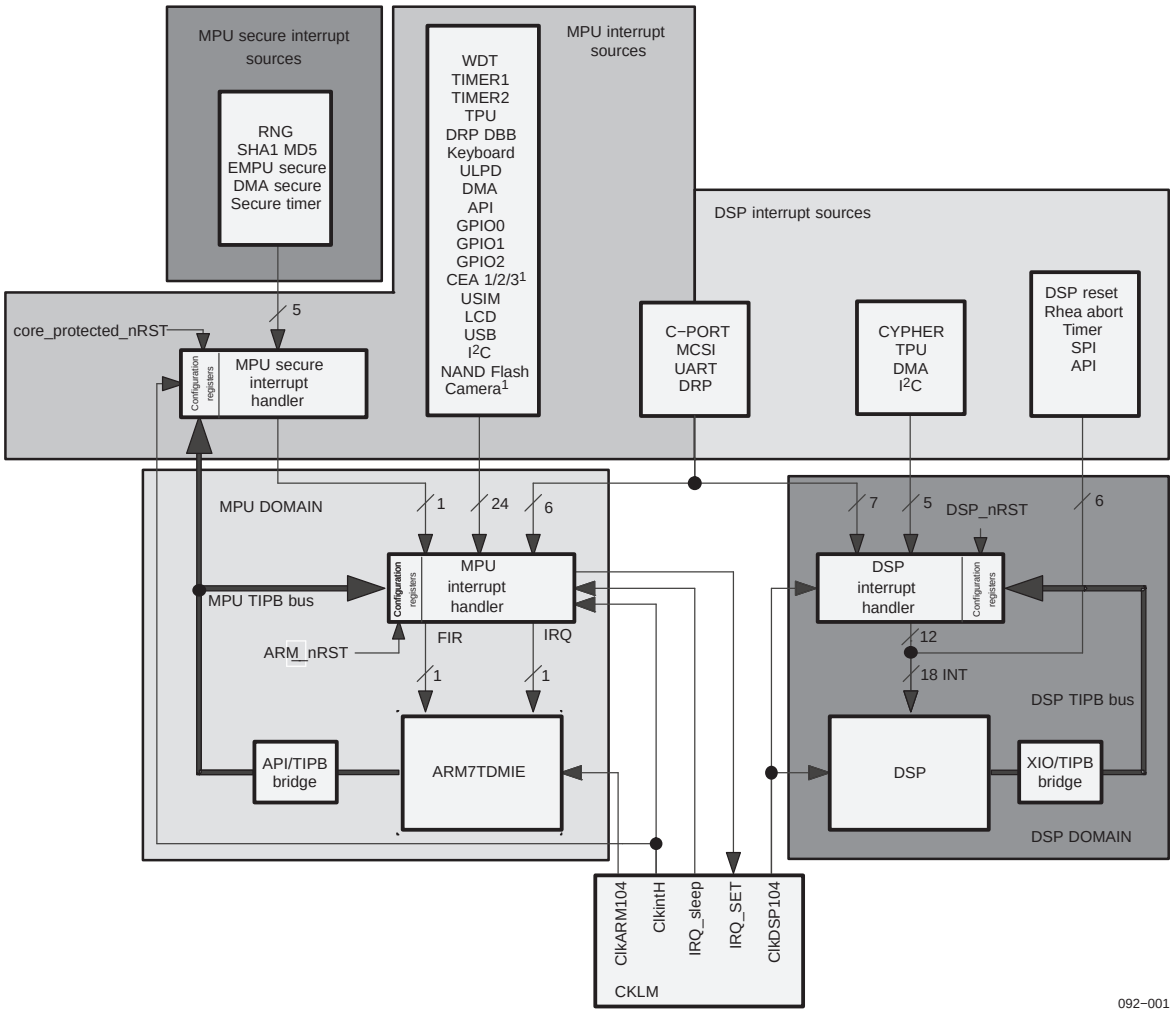
To wake up the system from an idle state with system clocks turned off, the MPU interrupt handlers provide the asynchronous signal CLKM_IRQ to the clock and reset management (CKLM) module. For more details on the wake procedure, see Chapter 6, *Power, Reset, and Clock Management*.

Figure 12-1 shows an overview of the LOCOSTO processors, peripherals interrupts, and interrupt handlers.

Table 12-4 through Table 12-6 detail the different events generated by the functional blocks, which will produce an interrupt IRQ, FIR, or INT on the processor core.

Note: Throughout this document, x denotes the channel number of the interrupt.

Figure 12-1. Interrupt Handler and Source Overview



NOTE: Camera interface and GPRS modules are not available on LOCOSTO Lite.

12.1.1 Main Features

The MPU secure interrupt handler includes the following features:

- 5 incoming interrupt lines from secure functional blocks
- 1 outgoing interrupt line, IRQ
- Each incoming interrupt line is individually programmable for:
  - Level sensitivity between falling-edge and low-level
  - Priority from HIGH (0) to LOW (5)
  - Masking of the interrupt

The MPU interrupt handler includes the following features:

- 30 incoming interrupt lines from the functional blocks
- 1 IRQ line coming from the secure interrupt handler
- 2 outgoing interrupt lines, IRQ and FIR
- Each incoming interrupt line is individually programmable for:
  - Level sensitivity between falling-edge and low-level
  - Priority from HIGH (0) to LOW (31)
  - Masking of the interrupt

  - IRQ or FIQ output
- Enables wakeup from big or deep sleep through the CLKM_IRQ signal to the CKLM module

The DSP interrupt handler includes the following features:

- 12 incoming interrupt lines from functional blocks
- 12 outgoing interrupt lines, INT
- Each incoming interrupt line is individually programmable for level sensitivity between falling-edge or low-level.

12.1.2 Signals and I/O Description

Table 12-1 through Table 12-3 and Figure 12-2 through Figure 12-4 briefly describe and illustrate the I/Os for the MPU interrupt handlers and the DSP handler.

Figure 12-2. MPU Interrupt Handler I/O

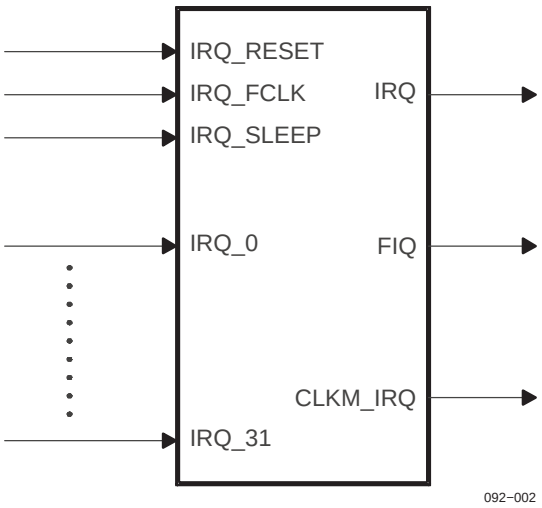


Table 12-1. MPU Interrupt Handler I/O Description

Signal Name	I/O[(1)]	Description	Reset Value
IRQ_RESET	I	Global reset of interrupt handler module. Active at low level.	-
IRQ_FCLK	I	Functional clock	-
IRQ_SLEEP	I	Mask global interrupt to the MPU core. Forces the state-machines to stay in wait-state. Active at high level.	-
IRQ_x	I	Incoming interrupts from 0 to 31. For details, see Table 12-4.	-
FIQ	O	Fast interrupt request to MPU core. Active at low level.	1
IRQ	O	Standard interrupt request to MPU core. Active at low level.	1
CLKM_IRQ	O	Interrupt to CLKM module. Active at low level.	1

(1) I = Input, O = Output

Figure 12-3. MPU Secure Interrupt Handler I/O

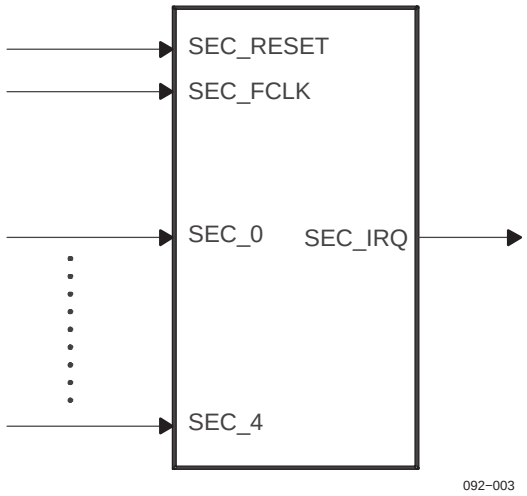


Table 12-2. MPU Secure Interrupt Handler I/O Description

Signal Name	I/O[1]	Description	Reset Value
SEC_RESET	I	Global reset of secure interrupt handler module. Active at low level.	-
SEC_FCLK	I	Functional clock	-
SEC_x	I	Incoming interrupts from 0 to 4. For details, see Table 12-5.	-
SEC_IRQ	O	Standard interrupt request to interrupt controller. Active at low level.	-

(1) I = Input, O = Output

Figure 12-4. DSP Interrupt Handler I/O

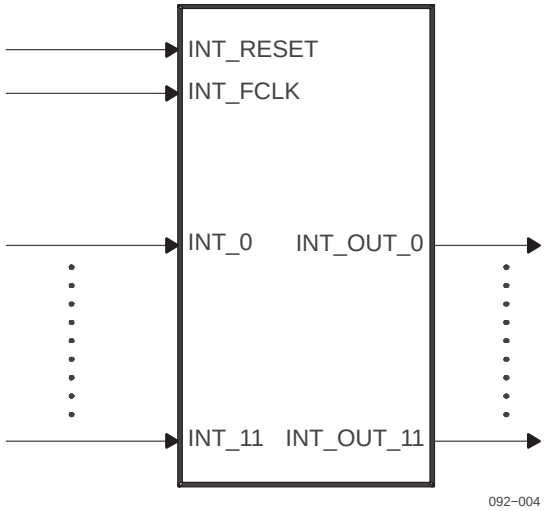


Table 12-3. DSP Interrupt Handler I/O Description

Signal Name	I/O[1]	Description	Reset Value
INT_RESET	I	Global reset of interrupt handler module. Active at low level.	-
INT_FCLK	I	Functional clock	-
INT_x	I	Incoming interrupts from 0 to 11. For details, see Table 12-6.	-
INT_OUT_x	O	Standard interrupt request to DSP. Active at low level.	-

(1) I = Input, O = Output

12.1.3 Interrupt Mapping

Table 12-4 through Table 12-6 list the different interrupts and trigger sources for each interrupt handler. For details about trigger methods and their programming, see the submodule or peripheral chapter.

CAUTION

To avoid missing an interrupt event, the sensibility level of each interrupt must be set up as described in Table 12-4 through Table 12-6.

12.1.3.1 MPU Interrupt Mapping

Table 12-4. MPU Interrupt Mapping

Interrupt Name	Compulsory Sensibility Level	Function
IRQ_0	Edge	Watchdog timer interrupt
IRQ_1	Edge	TIMER1 interrupt
IRQ_2	Edge	TIMER2 interrupt
IRQ_3	Level	MCSI interrupt resulting from: MCSI RX interrupt MCSI TX interrupt MCSI frame interrupt MCSI DAI interrupt
IRQ_4	Edge	TPU frame interrupt
IRQ_5	Edge	TPU page interrupt
IRQ_6	Level	DRP_DBB_SINTERRUPT
IRQ_7	Level	UART IrDA/modem interrupt resulting from: UART modem mode Receiver line status RX time-out RHR THR Modem status XOFF special character interrupt CTS, RTS, and DSR UART IrDA mode RHR THR Last byte in RX FIFO RX overrun Status FIFO interrupt TX status Receiver line status EOF
IRQ_8	Level	Keyboard interrupt
IRQ_9	Edge	DRP_DBB_RX_IRQ
IRQ_10(1)	Level	Camera interrupt resulting from: FIFO underflow

(1) Camera interface and GPRS modules are not available on LOCOSTO Lite.

Table 12-4. MPU Interrupt Mapping (continued)

Interrupt Name	Compulsory Sensibility Level	Function
		FIFO overflow
		FIFO threshold
		FIFO full
		FIFO not empty
		Shifted synchronization code
		False synchronization code
		Frame-width error
		Frame start
		Frame end
		Line start
		Line end
IRQ_11	Edge	ULPD end of gauging interrupt
IRQ_12	Level	ABB interrupt
IRQ_13	Level	MSSPI interrupt resulting from:
		Wakeup
		TX underflow
		RX overflow
		Write cycle
		Read/write cycle
IRQ_14	Level	DMA interrupt
IRQ_15	Edge	API interrupts
IRQ_16	Level/Edge	GPIO0 interrupt
IRQ_17	Edge	UART wakeup
IRQ_18	Edge	DRP_DBB_TX_IRQ
IRQ_19	Level	ULPD GSM timer
IRQ_20(1)	Level	GEA 1/2/3 interrupt
IRQ_21	Level/Edge	GPIO1 interrupt
IRQ_22	Level/Edge	GPIO2 interrupt
IRQ_23	Level	C-port interrupt resulting from:
		C-port (I2S) transmit interrupt
		C-port (I2S) receive interrupt
IRQ_24	Edge	USIM interrupt resulting from:
		No answer to reset
		Character underflow
		Character overflow
		Character to transmit
		Received character
IRQ_25	Level	LCD interrupt resulting from:
		TX FIFO empty
		Status register full
IRQ_26	Level	USB interrupt
		DMA transfer completed on transmit channel n
		DMA transfer completed from receive channel n
		End of transfer (EOT) packet detection on receive channel n
		Start of frame detected
		OUT transaction detected on endpoint n

Table 12-4. MPU Interrupt Mapping (continued)

Interrupt Name	Compulsory Sensibility Level	Function
		IN transaction detected on endpoint n Device status change Setup transaction completed on control endpoint 0 OUT transaction detected on control endpoint 0 IN transaction detected on control endpoint 0
IRQ_27	-	Unused
RQ_28	Level	I^2C interrupt resulting from: Data transfer error Data transfer completion (Triton)
IRQ_29	Level	Secure interrupt handler IRQ. See Table 12-5.
IRQ_30	Level	I^2C data transfer error/completion
IRQ_31	Level	NAND flash memory interrupt

CAUTION

IRQ_27 is an unused interrupt line; writing to an associated bit or field has no effect.

Note: To configure the switching level/edge in the interrupt level register (ILR_x) with the ILR_x.SENS_LEVEL[1] bit, write 0 to configure the corresponding interrupt as falling-edge sensitive; write 1 to configure the corresponding interrupt as low-level sensitive.

12.1.3.2 MPU Secure Interrupt Mapping

Table 12-5. MPU Secure Interrupt Mapping

Interrupt Name	Compulsory Sensibility Level	Function
SEC_IRQ_0	Edge	Random-number generator (RNG) interrupt
SEC_IRQ_1	Edge	SHA1 MD5 interrupt
SEC_IRQ_2	Edge	EMPU secure interrupt
SEC_IRQ_3	Edge	DMA secure interrupt
SEC_IRQ_4	Edge	Secure timer interrupt

12.1.3.3 DSP Interrupt Mapping

Table 12-6. DSP Interrupt Mapping

Interrupt Name	Compulsory Sensibility Level	Function
RSN	Level	DSP subsystem reset (hardware or software)
NMIN	-	Abort on TI peripheral bus (TIPB)
TINT	-	Timer interrupt
RINT	-	SPI receive interrupt
XINT	-	SPI transmit interrupt
AINT	-	API interrupt
INT_0	Edge	DRP_DBB_RX_INT
INT_1	Edge	DRP_DBB_TX_INT

Table 12-6. DSP Interrupt Mapping (continued)

Interrupt Name	Compulsory Sensibility Level	Function
INT_2	Level	DRP_DBB_SINTERRUPT UART IrDA/modem interrupt resulting from: UART modem mode Receiver line status RX time-out RHR THR Modem status XOFF special character interrupt CTS, RTS, and DSR UART IrDA mode RHR THR Last byte in RX FIFO RX overrun Status FIFO interrupt TX status Receiver line status EOF
INT_3	Level	MCSI-1 receive interrupt
INT_4	Level	MCSI-1 transmit interrupt
INT_5	Level	MCSI-1 frame duration error interrupt
INT_6	Level	MCSI-1 DAI interrupt C-port (I2S) transmit interrupt C-port (I2S) receive interrupt
INT_7	Edge	CYPHER interrupt resulting from: End of ciphering process Processing error
INT_8	Edge	TPU frame interrupt
INT_9	Edge	TPU programmable interrupt
INT_10	Level	DMA interrupt
INT_11	Level	I^2C error/completion interrupt

12.1.4 Clocking, Reset, and Power-Management Scheme

12.1.4.1 Clocks

The interrupt handler modules have three functional domain clocks: IRQ_FCLK, SEC_FCLK, and INT_FCLK. See Table 12-7.

Table 12-7. Interrupt Handler Clocks

Type	Name	Source	Frequency	Description
Functional	IRQ_FCLK	ClkIntH	104 MHz	Clock state-machines
Functional	SEC_FCLK	ClkIntH	104 MHz	Clock state-machines
Functional	INT_FCLK	clkDSP104	104 MHz	Use to identify falling edge

12.1.4.2 Power Management

At the system level, power-reduction techniques can be applied by shutting down certain internal clocks and power domains of the device. The clock manager can turn off the interrupt handler functional clocks. A handshaking protocol is defined for the clock and reset module to idle or wake up the interrupt handler. For details, see Chapter 6, *Power, Reset, and Clock Management*.

Even if an input clock is not running on IRQ_FCLK, the output signal CLKM_IRQ can generate an event when an unmasked interrupt is present on one of the IRQ_x signals.

12.1.4.3 Hardware and Software Resets

Each interrupt handler has a hardware reset driven by a different source (see Table 12-8). For details about the procedure, see Chapter 6, *Power, Reset, and Clock Management*.

Table 12-8. Interrupt Handler Reset

Type	Name	Source	Activation	Domain
Hardware	IRQ_RESET	ARM_nRST	0	Global reset
Hardware	SEC_RESET	core_protected_nRST	0	Global reset
Hardware	INT_RESET	DSP_nRST	0	Global reset

12.2 Interrupt Handler Functional Description

12.2.1 MPU Interrupt Handler and Secure Interrupt Handler

For the secure peripherals, a separate secure interrupt handler is present with:

- 5 incoming interrupt lines from RNG, SHA1, EMPU, DMA secure, and secure timer
- 1 outgoing interrupt line IRQ connected to IRQ_29 of the interrupt handler
- Each interrupt line is individually programmable for:
  - Level sensitivity for falling-edge or low-level
  - Priority
  - Masking

CAUTION

Programming an input secure line to generate an FIQ causes a loss of the interrupt because the output FIQ line is unused at the secure interrupt handler level.

After this first level of interrupts, the MPU interrupt handler permits up to 31 sources to generate requests to the MPU. Each source can be programmed to generate one of the following two MPU interrupts: FIQ and low-priority IRQ.

MPU interrupt handler features include:

- 30 incoming interrupt lines from functional blocks
- 1 incoming interrupt line, IRQ_29, from the MPU secure interrupt handler
- 2 outgoing interrupt lines (IRQ and FIQ) to the MPU core
- Each interrupt line is individually programmable for:
  - Level sensitivity for falling-edge or low-level
  - Priority
  - IRQ or FIQ output
  - Masking

CAUTION

IRQ_29 cannot be masked because it is the result of the secure interrupt handler.

12.2.1.1 Interrupt Control and Configuration

When an interrupt occurs, the interrupt input register (ITR) stores the incoming interrupt in the corresponding bit IRQ_x.

If there are several incoming interrupts, the MPU interrupt handler compares the priority level of the interrupts before sending an IRQ or FIQ to the MPU core.

Note: If all interrupts have the same priority level and are active at the same moment, the order of servicing is as follows:

IRQ_31, IRQ_30, IRQ_29,….. IRQ_1, IRQ_0. INTH groups master and secure interrupt handlers.

The number of the selected interrupt is stored in the INTH.SIR_IRQ[4:0] IRQ_NUM bit field or in the INTH.SIR_FIR[4:0] FIQ_NUM bit field for the MPU to determine which interrupt service routine (ISR) to execute.

The MPU can also clear INTH.ITR IRQ_x bits by writing 0 to the corresponding bits. Writing 1 keeps the previous value. This method is slower and is not recommended.

Each incoming interrupt can be masked individually by setting the corresponding INTH.MIR IRQ_x_MSK bit to 1.

One interrupt-level register (ILR) is associated with each incoming interrupt. ILR_x determines whether the interrupt is to be edge-triggered or level-sensitive, and assigns it a priority level: 0 (the highest priority), 1 ... 30, 31 (the lowest priority), and also permits routing of the interrupt to either the FIQ or the IRQ.

FIQ and IRQ outputs can be reset by writing 1 to the INTH.IRQ_CTRL_REG[1] NEW_FIQ_AGR bit for FIQ, or by writing 1 to the INTH.IRQ_CTRL_REG[0] NEW_IRQ_AGR bit for IRQ to enable a new FIQ or IRQ generation. This write also clears the SIR_IRQ or SIR_FIQ register. The corresponding bit in the ITR must be cleared before writing to the INTH.IRQ_CTRL_REG register.

Note: This method is applicable for secure and nonsecure interrupt handlers. The FIQ of the secure interrupt handler is not used.

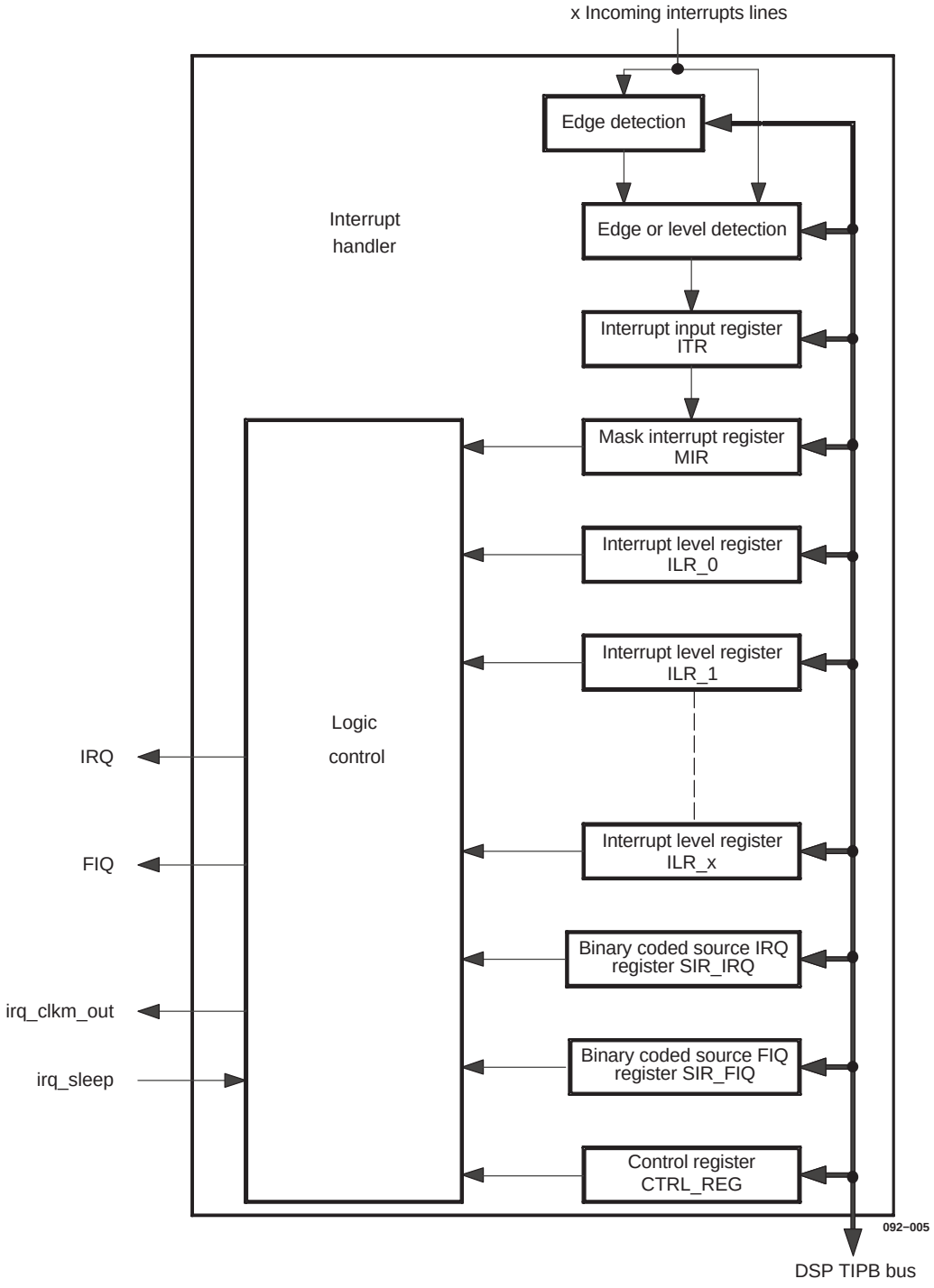
CAUTION

The latency from an incoming interrupt to the output interrupt generation depends on the number of interrupts arriving at the same time.

If there is only one incoming interrupt, the latency is five MPU clock cycles. If all interrupts are active at the same time and are routed to the same output interrupt, latency can reach 3 + N * 2 MPU cycles, where N is the number of incoming interrupts.

Figure 12-5 is the block diagram of the MPU interrupt handler for secure and nonsecure peripherals.

Figure 12-5. MPU Interrupt Handler Block Diagram



12.2.2 DSP Interrupt Handler

Two categories of interrupts are available on the DSP subsystem level:

- Maskable interrupts: Hardware or software interrupts that can be blocked (masked) or enabled (unmasked) using software. The DSP core supports up to 16 user-maskable interrupts:
  - INT_0 to INT_11
  - RINT, TINT (receive and transmit serial port interrupts)

  - TINT (timer interrupt)
  - AINT (API interrupt)
- Nonmaskable interrupts: Interrupts that cannot be blocked. The DSP core always acknowledges this type of interrupt and branches from the main program to an ISR. DSP nonmaskable interrupts include all software interrupts and two external hardware interrupts:
  - RSn (reset) affects all DSP operating modes.
  - nNMI does not affect any DSP modes.
  - RSn and nNMI can also be asserted using software.

The DSP interrupt handler manages up to 12 sources to generate interrupts to the DSP core. Each source, INT_0 to INT_11, is programmed through the INT_x bit DSP_INTH.EDGE_EN[11:0] to interrupt the DSP core on:

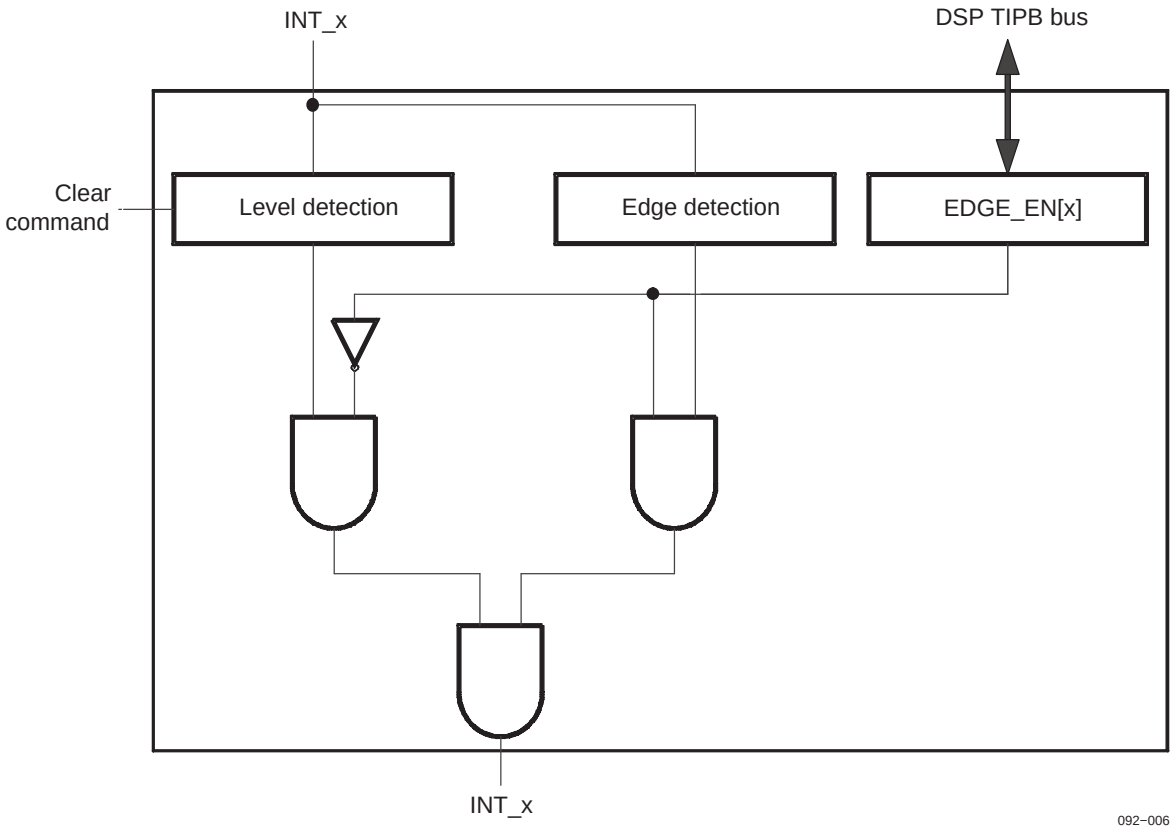
- Falling-edge
- Low-level (default value)

Figure 12-6 is the block diagram of the interrupt handler for INT_0 to INT_11. Edge and level detection are done in parallel, then depending on the value of the INT_x bit DSP_INTH.EDGE_EN[x], the interrupt is generated on the output.

When multiple hardware interrupts are triggered at the same time, the DSP core services them according to a set priority ranking in which 1 has the highest priority.

Note: For details about priorities, masks, and enable interrupts, see *C54X cDSP Reference Set Vol 1: CPU*.

Figure 12-6. DSP Interrupt Handler Block Diagram



12.2.2.1 Shared Level-Sensitive Interrupts

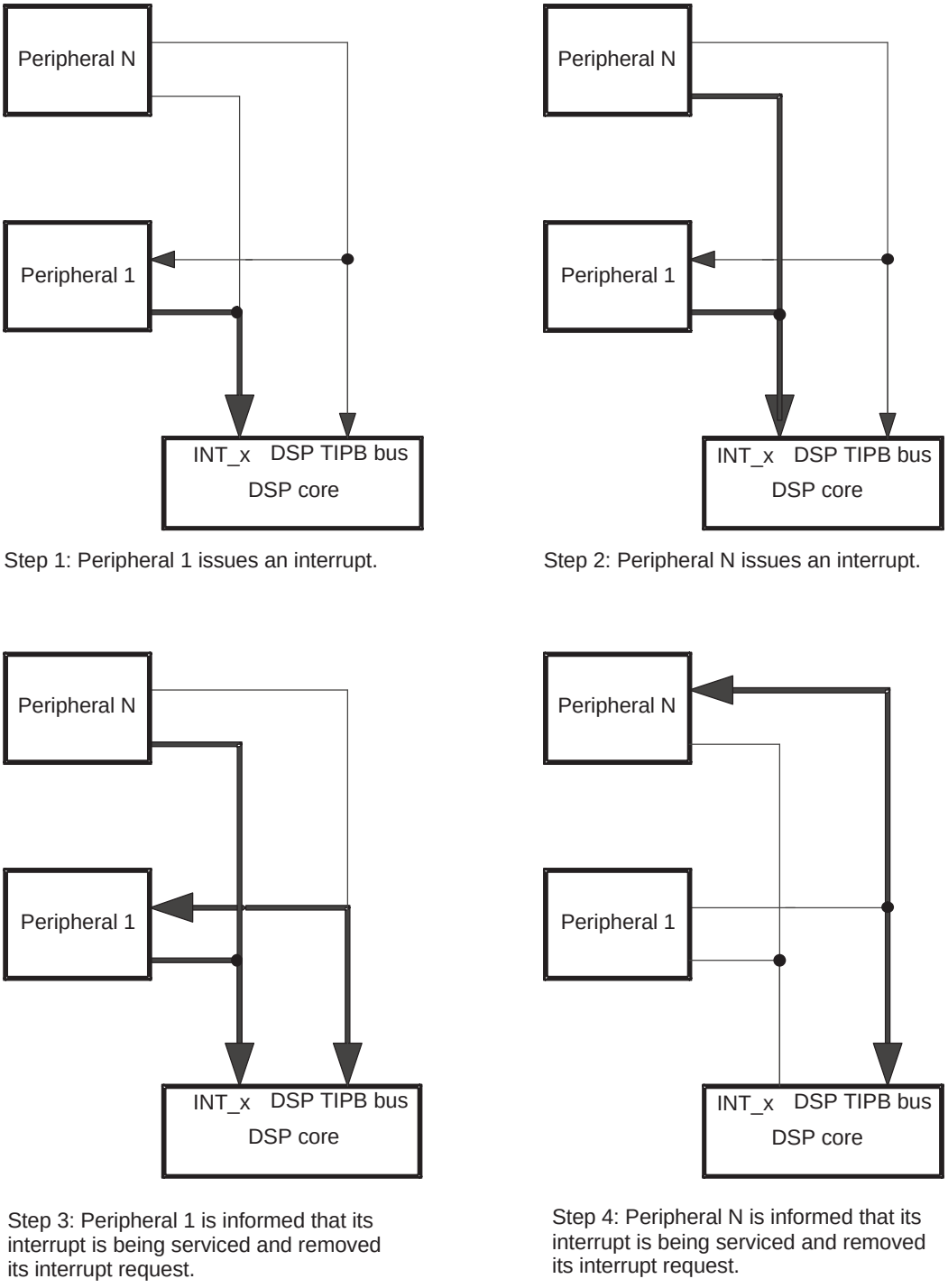
In the interrupt mapping of the LOCOSTO device, the following peripherals share the same interrupt line:

- INT_2 is shared between the DRP and the UART.
- INT_6 is shared between the MCSI and the C-port.

When a level-sensitive interrupt line is shared, the incoming INT_x line can remain active during the servicing of two or more peripherals sharing it. All peripherals sourcing the interrupt INT_x must keep the interrupt line active until notified by the DSP core that its request is being serviced. The interrupt handler can clear the interrupt by command, and then permit another interrupt to be issued as the input interrupt line INT_x remains active.

Figure 12-7 shows a typical flow of events in operating a shared level-sensitive interrupt line.

Figure 12-7. Shared Interrupt Line Procedure



In Step 1, peripheral 1 issues an interrupt request by pulling the INT_x line low. When the interrupt is registered in the DSP core and servicing of the interrupt begins, the DSP core interrupt service routine polls all peripherals sharing INT_x to determine which peripheral is making the request.

In Step 2, peripheral N issues an interrupt request before the DSP core has identified peripheral 1 as the original source of the interrupt request and has begun servicing this interrupt.

In Step 3, the DSP core informs peripheral 1, through the DSP TIPB, that its interrupt request is being serviced, and peripheral 1 responds by deactivating its request. However, the interrupt request remains active, because the request from peripheral N is still pending. When the DSP core finishes servicing the peripheral 1 request, it issues a command to clear the interrupt line to permit other interrupts to be issued, if any are pending.

In Step 4, peripheral N is notified that its request is being serviced, and peripheral N releases the interrupt request line INT_x. The process of servicing the two overlapping requests on the one shared interrupt line is complete.

12.3 Interrupt Handler Programming Model

12.3.1 MPU Secure and Nonsecure Interrupt Handler

To correctly process edge-triggered and level-sensitive interrupts, the following sequences must occur in the system. Only the IRQ treatment is described here. The FIQ treatment is identical for the interrupt controller.

Note: For DSP interrupt programming, see the appropriate *C54x cDSP Reference Set Vol 1* documentation.

CAUTION

To avoid missing an interrupt event, sensibility of each interrupt must be set up as described in Table 12-4 through Table 12-6.

12.3.1.1 Edge-Triggered Interrupts

1. The interrupt handler module receives one or more incoming interrupts, and then registers them in the ITR.
2. If there are several active incoming interrupts, the interrupt handler determines the highest-priority interrupt.
3. If the IRQ (interrupt from the interrupt handler to the MPU) is not active, the interrupt handler sends the interrupt to the MPU as an IRQ. The SIR_IRQ is then updated with the contents from the IRQ.
4. The MPU recognizes the interrupt and jumps to the ISR code.
5. Within the ISR code, the MPU reads the SIR_IRQ in the interrupt handler to determine which interrupt line caused the interrupt. The MPU then executes the appropriate code for this interrupt.
6. When the MPU reads the SIR_IRQ, the corresponding bit is reset in the ITR of the interrupt handler. The IRQ is still active.
7. When the MPU is ready to exit the ISR routine, it writes 1 to the NEW_IRQ_AGR bit. This deasserts the IRQ going to the MPU and enables the generation of a new IRQ.
8. The MPU exits the ISR and continues normal code execution.
9. When the NEW_IRQ_AGR bit is written, the process returns to Step 2.

12.3.1.2 Level-Sensitive Interrupts

1. The interrupt handler receives one or more incoming interrupts. Level-sensitive interrupts are not registered, but are used as is in the logic. The interrupt handler assumes that the peripheral will not deassert level-sensitive incoming interrupts until it is told to do so by the MPU.
2. The interrupt handler determines the highest-priority interrupt.
3. If the IRQ (interrupt from the interrupt handler to the MPU) is not active, the interrupt handler sends the highest-priority interrupt to the MPU as an IRQ. The SIR_IRQ is then updated with the new interrupt service. If the IRQ is active, which means the NEW_IRQ_AGR bit has not been set by the MPU, the incoming IRQ waits until the IRQ is not active.
4. The MPU recognizes the interrupt and jumps to the ISR code.
5. Within the ISR code, the MPU reads the SIR_IRQ in the interrupt handler to determine which interrupt line caused the interrupt.
1. The ISR code must do one of the following:
   - Inform the peripheral that the interrupt generated has been serviced so that the peripheral can deassert the interrupt request
   - Write to the interrupt handler mask interrupt register (MIR) to mask the level-sensitive interrupt. At this point, the peripheral must deassert the interrupt before the mask of the interrupt can be removed, so that the next interrupt can be recognized. If the peripheral deasserts the interrupt

before the code in the subroutine instructs it to do so, then the behavior is unpredictable and the interrupt may be lost.

7. When the MPU is ready to exit the ISR routine, it writes 1 to the NEW_IRQ_AGR bit. This deasserts the IRQ going to the MPU and enables the generation of a new IRQ.
8. The MPU exits the ISR and continues normal code execution.
9. When the NEW_IRQ_AGR bit is written to by the MPU, the process returns to Step 2.

12.4 Interrupt Handler Register Manual

Table 12-9 and Table 12-10 provide the base address and size for DSP and MPU accesses, respectively.

Table 12-9. DSP Access Instance Summary

Module Name	Base Address	Size
DSP interrupt handler	0xFA00	512 bytes

Table 12-10. MPU Access Instance Summary

Module Name	Base Address	Size
MPU interrupt handler	0xFFFF FA00	128 bytes
MPU secure interrupt handler	0xFFFF FA80	128 bytes

12.4.1 Module Register Mapping Summary

Table 12-11 and Table 12-12 list the offset address for the DSP and MPU interrupt handler registers, respectively. Table 12-13 lists the offset address for the MPU secure interrupt handler register.

Table 12-11. DSP Interrupt Handler Register Offset Address

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
EDGE_EN	R/W	16	0x00	0xFA00

Table 12-12. MPU Interrupt Handler Register Offset Address

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
ITR_0	R	16	0x00	0xFFFF FA00
ITR_1	R	16	0x02	0xFFFF FA02
MIR_0	R/W	16	0x08	0xFFFF FA08
MIR_1	R/W	16	0x0A	0xFFFF FA0A
SIR_IRQ	R	16	0x10	0xFFFF FA10
SIR_FIR	R	16	0x12	0xFFFF FA12
IRQ_CTRL_REG	R/W	16	0x14	0xFFFF FA14
ILR_0	R/W	16	0x20	0xFFFF FA20
ILR_1	R/W	16	0x22	0xFFFF FA22
ILR_2	R/W	16	0x24	0xFFFF FA24
ILR_3	R/W	16	0x26	0xFFFF FA26
ILR_4	R/W	16	0x28	0xFFFF FA28
ILR_5	R/W	16	0x2A	0xFFFF FA2A
ILR_6	R/W	16	0x2C	0xFFFF FA2C
ILR_7	R/W	16	0x2E	0xFFFF FA2E
ILR_8	R/W	16	0x30	0xFFFF FA30

Table 12-12. MPU Interrupt Handler Register Offset Address (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
ILR_9	R/W	16	0x32	0xFFFF FA32
ILR_10	R/W	16	0x34	0xFFFF FA34
ILR_11	R/W	16	0x36	0xFFFF FA36
ILR_12	R/W	16	0x38	0xFFFF FA38
ILR_13	R/W	16	0x3A	0xFFFF FA3A
ILR_14	R/W	16	0x3C	0xFFFF FA3C
ILR_15	R/W	16	0x3E	0xFFFF FA3E
ILR_16	R/W	16	0x40	0xFFFF FA40
ILR_17	R/W	16	0x42	0xFFFF FA42
ILR_18	R/W	16	0x44	0xFFFF FA44
ILR_19	R/W	16	0x46	0xFFFF FA46
ILR_20	R/W	16	0x48	0xFFFF FA48
ILR_21	R/W	16	0x4A	0xFFFF FA4A
ILR_22	R/W	16	0x4C	0xFFFF FA4C
ILR_23	R/W	16	0x4E	0xFFFF FA4E
ILR_24	R/W	16	0x50	0xFFFF FA50
ILR_25	R/W	16	0x52	0xFFFF FA52
ILR_26	R/W	16	0x54	0xFFFF FA54
ILR_27	R/W	16	0x56	0xFFFF FA56
ILR_28	R/W	16	0x58	0xFFFF FA58
ILR_29	R/W	16	0x5A	0xFFFF FA5A
ILR_30	R/W	16	0x5C	0xFFFF FA5C
ILR_31	R/W	16	0x5E	0xFFFF FA5E

Table 12-13. MPU Secure Interrupt Handler Register Offset Address

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
SEC_ITR	R	16	0x00	0xFFFF FA80
SEC_MIR	R/W	16	0x08	0xFFFF FA88
SEC_SIR_IRQ	R	16	0x10	0xFFFF FA90
SEC_SIR_FIQ	R	16	0x12	0xFFFF FA92
SEC_CTRL_REG	R/W	16	0x14	0xFFFF FA94
SEC_ILR_0	R/W	16	0x20	0xFFFF FAA0
SEC_ILR_1	R/W	16	0x22	0xFFFF FAA2
SEC_ILR_2	R/W	16	0x24	0xFFFF FAA4
SEC_ILR_3	R/W	16	0x26	0xFFFF FAA6
SEC_ILR_4	R/W	16	0x28	0xFFFF FAA8
EDGE_EN	R/W	16	0x00	0xFA00

12.4.2 Register Description

Table 12-14 through Table 12-28 describe the individual register bits.

Table 12-14. ITR_0

Address Offset	0x00		
Physical address	0xFFFF FA00	**Instance**	MPU interrupt handler
Description	Interrupt register		
Type	R		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQ_x															

Bits	Field Name	Description	Type	Reset
15:0	IRQ_x	Signals the occurrence of an edge-sensitive interrupt request When an edge-sensitive interrupt (or interrupts) occurs, the ITR bit corresponding to the calling interrupt is set to 1. The ITR bits are set, regardless of whether the interrupts are masked or unmasked. The same ITR bit(s) is reset when either the SIR_IRQ or SIR_FIQ register is read. Level-sensitive interrupts have no effect on the ITR.	R	0x00

Table 12-15. ITR_1

Address Offset	0x02		
Physical address	0xFFFF FA02	**Instance**	MPU interrupt handler
Description	Interrupt register		
Type	R		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQ_x															

Bits	Field Name	Description	Type	Reset
15:0	IRQ_x	Signals the occurrence of an edge-sensitive interrupt request When an edge-sensitive interrupt (or interrupts) occurs, the ITR bit corresponding to the calling interrupt is set to 1. The ITR bits are set, regardless of whether the interrupts are masked or unmasked. The same ITR bit(s) is reset when either the SIR_IRQ or SIR_FIQ register is read. Level-sensitive interrupts have no effect on the ITR.	R	0x00

Table 12-16. MIR_0

Address Offset	0x08		
Physical address	0xFFFF FA08	**Instance**	MPU interrupt handler
Description	Mask interrupt register		
Type	RW		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQ_x_MSK															

Bits	Field Name	Description	Type	Reset
15:0	IRQ_x_MSK	To mask an interrupt, set the corresponding register bit to 1. Interrupts continue to set the interrupt register (ITR) when masked.	RW	0xFF

Table 12-17. MIR_1

Address Offset	0x0A		
Physical address	0xFFFF FA0A	**Instance**	MPU interrupt handler
Description	Mask interrupt register		
Type	RW		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQ_x_MSK															

Bits	Field Name	Description	Type	Reset
15:0	IRQ_x_MSK	To mask an interrupt, set the corresponding register bit to 1. Interrupts continue to set the interrupt register (ITR) when masked.	RW	0xFF

Table 12-18. SIR_IRQ

Address Offset	0x10		
Physical address	0xFFFF FA10	**Instance**	MPU interrupt handler
Description	Source IRQ binary coded register		
Type	R		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												IRQ_NUM			

Bits	Field Name	Description	Type	Reset
15:5	Reserved	Reserved	R	0x3FF
4:0	IRQ_NUM	Indicates the active interrupt line (0_UnicodeEncodeError_31) making an IRQ request. This field is binary coded. For edge-sensitive interrupts, reading this register clears the corresponding ITR register bit.	R	0x0

Table 12-19. SIR_FIR

Address Offset	0x12		
Physical address	0xFFFF FA12	**Instance**	MPU interrupt handler
Description	Source FIQ binary coded register		
Type	R		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												FIQ_NUM			

Bits	Field Name	Description	Type	Reset
15:5	Reserved	Reserved	R	0x7FF
4:0	FIQ_NUM	Indicates the active interrupt line (0_UnicodeEncodeError_31) making an FIR request. This field is binary coded. For edge-sensitive interrupts, reading this register clears the corresponding ITR register bit.	R	0x0

Table 12-20. IRQ_CTRL_REG

Address Offset	0x14		
Physical address	0xFFFF FA14	**Instance**	MPU interrupt handler
Description	Control register		
Type	RW		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														NEW_FIQ_AGR	NEW_IRQ_AGR

Bits	Field Name	Description	Type	Reset
15:3	Reserved	Reserved	R	0x3FFF
1	NEW_FIQ_AGR	Indicates a new FIQ agreement To enable the generation of a new FIQ: 1. Clear the corresponding bit of the ITR. 2. Write 1 to this field. This in turn resets the FIQ output and clears the SIR_FIQ. Writing 0 to this field has no effect.	R/W	0x0
0	NEW_IRQ_AGR	Indicates a new IRQ agreement To enable the generation of a new IRQ: 1. Clear the corresponding bit of the ITR. 2. Write 1 to this field. This in turn resets the IRQ output and clears the SIR_IRQ. Writing 0 to this field has no effect.	R/W	0x0

Table 12-21. ILR_x

Address Offset	0x20 to 0x5E		
Physical address	0xFFFF FAxx	**Instance**	MPU interrupt handler
Description	Interrupt level register		
Type	RW		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PRIORITY						SENS_LEVEL	FIQ

Bits	Field Name	Description	Type	Reset
15:7	Reserved	Reserved	R	0x1FF
6:2	PRIORITY	Defines the priority level of the corresponding interrupt Priority values are binary-encoded and range from 0 to 31: 0 is the highest-priority level. 31 is the lowest priority level.	R/W	0x0
1	SENS_LEVEL	1: The corresponding interrupt is falling-edge sensitive. 0: The corresponding interrupt is low-level sensitive.	R/W	0x0
0	FIQ	0: The corresponding interrupt is routed to FIQ.	R/W	0x0

Bits	Field Name	Description	Type	Reset
		1: The corresponding interrupt is routed to IRQ.		

Table 12-22. SEC_ITR

Address Offset	0x00		
Physical address	0xFFFF FA80	**Instance**	MPU secure interrupt handler
Description	Interrupt register		
Type	R		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SEC_IRQ_x							

Bits	Field Name	Description	Type	Reset
15:5	Reserved	Reserved	R	0x7FF
4:0	SEC_IRQ_x	Signals the occurrence of an edge-sensitive secure interrupt request When an edge-sensitive interrupt (or interrupts) occurs, the ITR bit corresponding to the calling interrupt is set to 1. The ITR bits are set, regardless of whether the interrupts are masked or unmasked. The same ITR bit(s) is reset when either the SIR_IRQ or SIR_FIQ register is read. Level-sensitive interrupts have no effect on the ITR.	R	0x00

Table 12-23. SEC_MIR

Address Offset	0x08		
Physical address	0xFFFF FA88	**Instance**	MPU secure interrupt handler
Description	Mask interrupts register		
Type	RW		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SEC_IRQ_x_MSK							

Bits	Field Name	Description	Type	Reset
15:5	Reserved	Reserved	R	0x7FF
4:0	SEC_IRQ_x_MSK	To mask an interrupt, set the corresponding register bit to 1. Interrupts continue to set the interrupt register (ITR) when masked.	RW	0xFF

Table 12-24. SEC_SIR_IRQ

Address Offset	0x10		
Physical address	0xFFFF FA90	**Instance**	MPU secure interrupt handler
Description	Source IRQ binary coded register		
Type	R		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													SEC_IRQ_NUM		

Bits	Field Name	Description	Type	Reset
15:3	Reserved	Reserved	R	0x1FFF
2:0	SEC_IRQ_NUM	Indicates the active interrupt line (0_UnicodeEncodeError_4) making an IRQ request. This field is binary coded. For edge-sensitive interrupts, reading this register clears the corresponding ITR register bit.	R	0x0

Table 12-25. SEC_SIR_FIQ

Address Offset	0x12		
Physical address	0xFFFF FA92	**Instance**	MPU secure interrupt handler
Description	Indicates source of secure interrupts		
Type	R		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SEC_FIQ_NUM	

Bits	Field Name	Description	Type	Reset
15:3	Reserved	Reserved	R	0x1FFF
2:0	SEC_FIQ_NUM	Indicates the active interrupt line (0_UnicodeEncodeError_4) making an FIR request. This field is binary coded. For edge-sensitive interrupts, reading this register clears the corresponding ITR register bit.	R	0x0

CAUTION

The secure interrupt handler does not generate a FIQ; therefore, it is strongly suggested not to use any bits related to this functionality.

Table 12-26. SEC_CTRL_REG

Address Offset	0x14		
Physical address	0xFFFF FA94	**Instance**	MPU secure interrupt handler
Description	Control register		
Type	RW		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														NEW_SEC_FIQ_AGR	NEW_SEC_IRQ_AGR

Bits	Field Name	Description	Type	Reset
15:2	Reserved	Reserved	R	0x3FFF
1	NEW_SEC_FIQ_AGR	Indicates a new FIQ agreement To enable the generation of a new FIQ: Clear the corresponding bit of the ITR. Write 1 to this field. This in turn resets the FIQ output and clears the SIR_FIQ.	R/W	0x0

Bits	Field Name	Description	Type	Reset
		Writing 0 to this field has no effect.		
0	NEW_SEC_IRQ_AGR	Indicates a new IRQ agreement To enable the generation of a new IRQ: Clear the corresponding bit of the ITR. Write 1 to this field. This in turn resets the IRQ output and clears the SIR_IRQ. Writing 0 to this field has no effect.	R/W	0x0

CAUTION

The secure interrupt handler does not generate a FIQ; therefore, it is strongly suggested not to use any bits related to this functionality.

Table 12-27. SEC_ILR_x

Address Offset	0x20 to 0x28		
Physical address	0xFFFF FAxx	**Instance**	MPU secure interrupt handler
Description	Sets up priority level, type of trigger, and FIR/IRQ for each secure interrupt line		
Type	RW		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SEC_PRIORITY						SEC_SENS_LEVEL	SEC_FIQ

Bits	Field Name	Description	Type	Reset
15:5	Reserved	Reserved	R	0x7FF
4:2	SEC_PRIORITY	Defines the priority level of the corresponding interrupt Priority values are binary-encoded and range from 0 to 31: 0 is the highest-priority level. 31 is the lowest priority level.	R/W	0x0
1	SEC_SENS_LEVEL	0: The corresponding interrupt is falling-edge sensitive. 1: The corresponding interrupt is low-level sensitive.	R/W	0x0
0	SEC_FIQ	0: The corresponding interrupt is routed to FIQ. 1: The corresponding interrupt is routed to IRQ.	R/W	0x0

CAUTION

The secure interrupt handler does not generate a FIQ; therefore, it is strongly suggested not to use any bits related to this functionality.

Table 12-28. EDGE_EN

Address Offset	0x00		
Physical address	0xFA00	**Instance**	DSP Interrupt handler
Description	Defines the type of trigger for DSP interrupts		
Type	RW		
Write Latency	N/A		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			INT4_SW	INT_x											

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	R	0x0
12	INT4_SW	Connects INT_4 to DSP INT_4 or to DSP nNMI 0: Connected to INT_4 1: Connected to nNMI	R/W	0x0
11:0	INT_x	Sets type of trigger for INT_x: 0: Level-sensitive 1: Edge-sensitive	R/W	0x0

Direct Memory Access

This chapter introduces the direct memory access (DMA) subsystem of the LOCOSTO device.

Topic **Page**

13.1 DMA Module Review

The enhanced DMA controller provides six physical channels to sustain the increased data throughput required by multimedia peripherals (such as the liquid crystal display [LCD] or the camera interface). Each channel can transfer data from/to any of the five ports supported. The DMA can access secure resources through a secure channel mode setting.

Note: The camera I/F is not available on the LOCOSTO Lite device.

13.1.1 Main Features

The DMA subsystem of the LOCOSTO device supports the following main features:

- Six channels
- Five ports (RHEA, API, IMIF, EMIF, and IPERIPH)
- Concurrent DMA transfers. The DMA subsystem can access all ports at the same time when they are free.
- Two levels of priority on each channel. Each channel can be configured as a high- or low-priority channel. High-priority channels are served before low-priority channels.
- Hardware (HW) and software (SW) request capability. A DMA transfer can be initiated by a peripheral (hardware request) or by the software.
- Transfer data up to 32 bits
- Data pipelining capability
- Data packing/splitting. The DMA subsystem can pack data when the destination port supports larger words than the source port, or can split data when the source port is aligned on larger words than the destination port. This feature increases efficiency of bandwidth use.
- Data bursting
- Three addressing modes: Constant, post-increment, and frame-indexed
- Auto-initialization capability. A channel with this feature does not require software intervention to be reinitialized to serve the next DMA request.
- Generation of interrupts on various events

13.1.2 Signals and I/O Description

Figure 13-1 shows the overview and the environment of the DMA controller. This module can transfer data from/to internal memory, external memory, API memory, memory-like peripherals, and peripherals accessed through the Texas Instruments peripheral bus (TIPB).

Figure 13-1. DMA Controller Overview

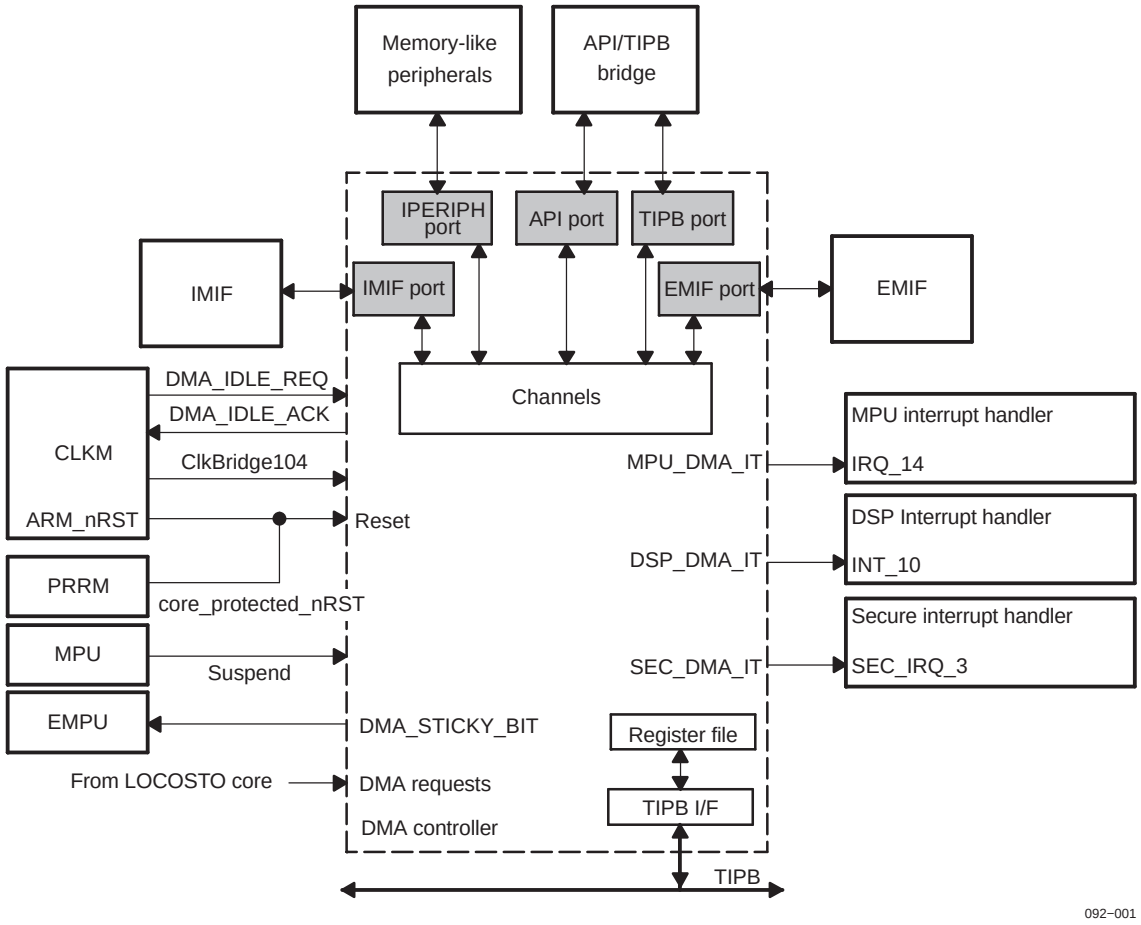


Table 13-1 lists the main I/O signals of the DMA controller. The following sections provide more information.

Table 13-1. I/O Descriptions

Signal Name	I/O(1)	Description	Reset Value
DMA_IDLE_REQ	I	DMA idle request	0
DMA_IDLE_ACK	O	DMA idle acknowledgement	0
Suspend	I	Suspension of the DMA operation	1
MPU_DMA_IT	O	Interrupt from a nonsecure channel allocated to the MPU	1
DSP_DMA_IT	O	Interrupt from a channel allocated to the DSP	1
SEC_DMA_IT	O	Interrupt from a secure channel	1
DMA_Req[30:0]	I	DMA request line	0x0
DMA_STICKY_BIT	O	Secured DMA transfer	0

(1) I = Input, O = Output

13.1.3 Clocking, Reset, and Power-Management Scheme

This section describes the clock, reset, and power-management scheme. Table 13-2 lists the functional and interface clocks. Table 13-3 lists the hardware resets.

13.1.3.1 Clocks

Table 13-2. Clocks

Type	Name	Source	Frequency	Description
Functional	ClkBridge104	CLKM	104 MHz	This clock is managed by the clock and reset management (CLKM) module; its frequency is typically 104 MHz.
Interface	TIPB_CLK	TIPB	52 MHz	Clock fed by the TIPB

For more information, see Chapter 6, *Power, Reset, and Clock Management*.

13.1.3.2 Resets

Table 13-3. Hardware Resets

Type	Name	Source	Activation	Domain
Hardware	Reset	ARM_nRST (CLKM) and Core_protected_nRST (PRRM)	See hapter 6, *Power, Reset, and Clock Management*.	Whole DMA subsystem

13.1.3.3 Power Management

If the AUTOGATING_ON bit DMA.DMA_GCR[3] is set to 1 and there is no activity in any channel while the ClkBridge104 clock is running, the functional clock is cut off inside the DMA subsystem. Alternately, if the AUTOGATING_ON bit DMA.DMA_GCR[3] is set to 0, the functional clock is cut off only when the ClkBridge104 clock is cut off.

If there is no activity in the DMA subsystem, the ClkBridge104 clock can be cut off by the CLKM module in big sleep mode. This clock is cut off when the LOCOSTO device is in deep sleep mode. In the latter case, a request acknowledgement system (through the DMA_IDLE_REQ and DMA_IDLE_ACK signals) is used to avoid the clock being cut off while a DMA transfer is running. For more information, see Chapter 6, *Power, Reset, and Clock Management*.

13.1.3.4 Power Domain

The DMA subsystem belongs to the VDD_core power domain. For more information, see Chapter 6, *Power, Reset, and Clock Management*.

13.1.4 Hardware Requests

Table 13-4 lists the hardware requests.

Table 13-4. Hardware Requests

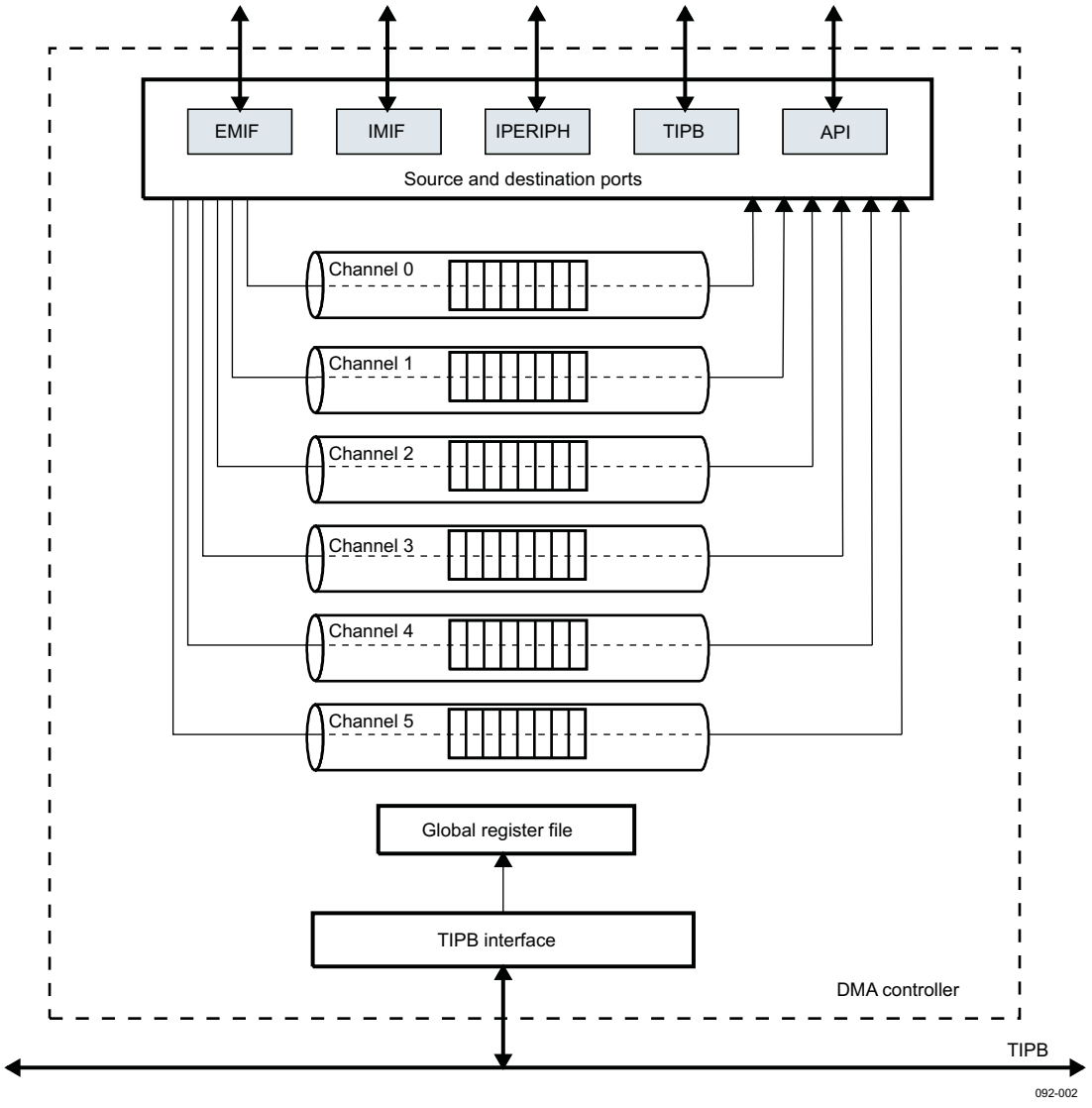
Type	Name	Source	Destination	Description
Interrupt	MPU_DMA_IT	DMA	MPU interrupt handler	Interrupt from a nonsecure channel allocated to the MPU
Interrupt	DSP_DMA_IT	DMA	DSP interrupt handler	Interrupt from a channel allocated to the DSP
Interrupt	SEC_DMA_IT	DMA	Secure interrupt handler	Interrupt from a secure channel
Reset	ARM_nRST	CLKM	DMA	Global chip reset

13.2 DMA Functional Description

13.2.1 DMA Controller Overview

This section is an overview of the DMA controller. The DMA transfers the data between the IMIFI, the EMIF (and thus external memories), the API/TIPB bridge (RHEA and API ports), and the memory-like peripherals (IPERIPH port). The DMA controller includes five ports and six independent channels that allow up to six DMA transfers to run simultaneously. Each channel has a set of dedicated registers and a global register. Figure 13-2 is the DMA controller block diagram.

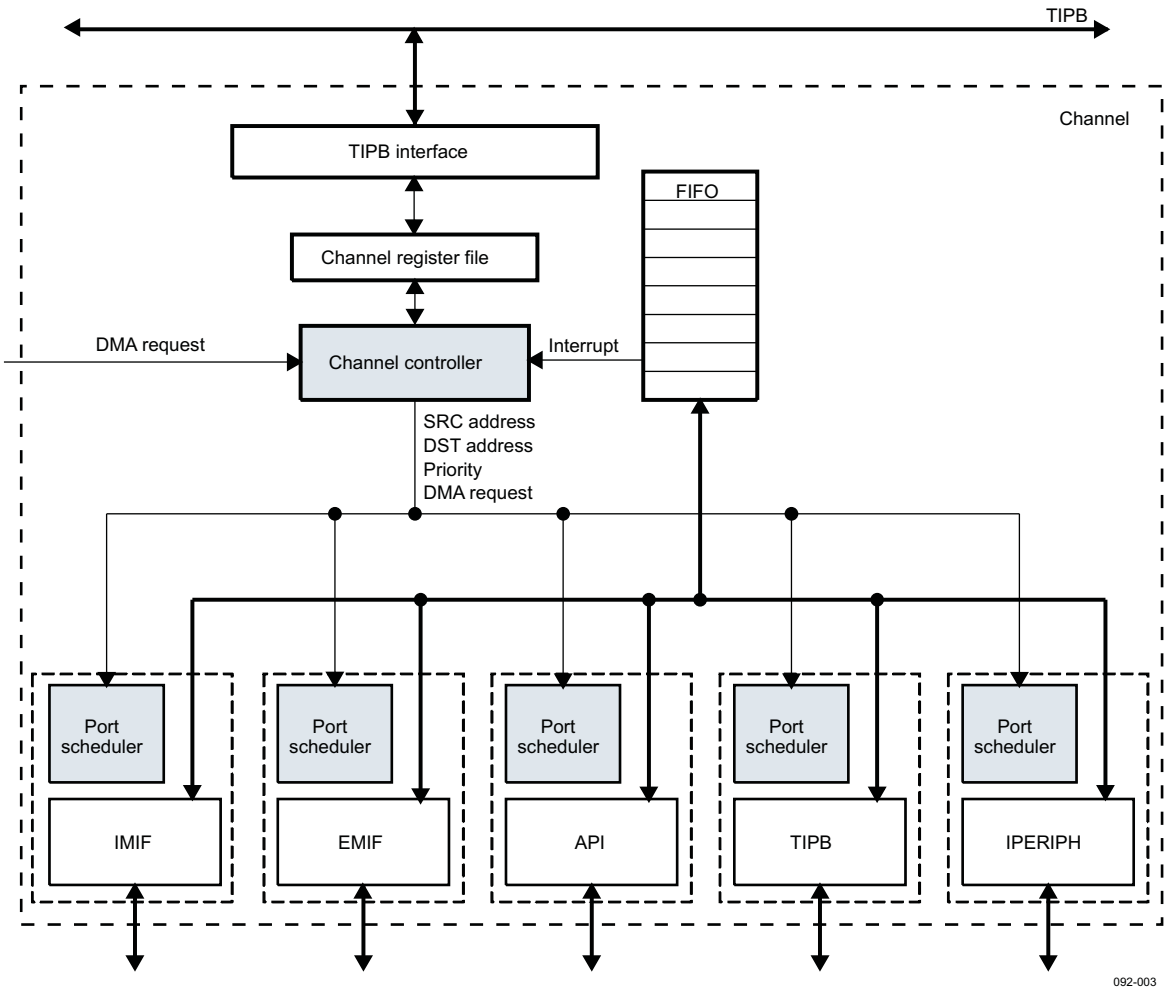
Figure 13-2. DMA Controller Block Diagram



13.2.2 Channel Block Diagram

The DMA subsystem transfers data between a source port and a destination port through one of the six possible channels. See Figure 13-3.

Figure 13-3. Channel Block Diagram



Each channel can be configured independently of other channels. A port can be shared by several channels.

Five ports are available:

- TIPB: Access to a peripheral through the TIPB
- API: Access to shared DSP-MPU memory
- IMIF: Access to internal memory
- EMIF: Access to external memory
- IPERIPH: Access to memory-like peripherals

Each channel is composed of a FIFO that stores the transferred data, a dedicated register file, and a controller. This controller stores information (such as the interrupt generated, the DMA requests received, and the channel configuration) and then generates commands for port schedulers, which manage the transfer of the data.

13.2.3 Channel Control Allocation

Either the MPU or the DSP can control a channel. The DMA.DMA_CAR register controls whether the dedicated channel registers are accessible to the MPU or the DSP. However, the DSP cannot access all global registers (the DMA.DMA_GCR and DMA.DMA_AR registers are exceptions). Global registers are accessible only in read mode by the DSP. The MPU can read or write to all global registers except the DMA.DMA_ISR register, which is accessible only in read mode.

Note: The DMA.DMA_CAR register can be set up only by the MPU in supervisor mode.

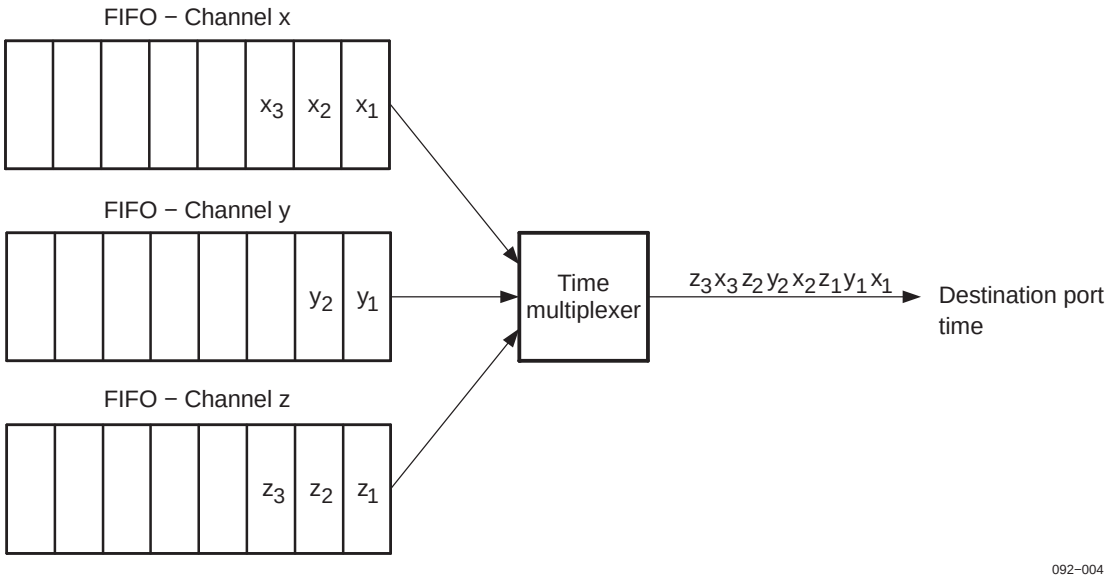
13.2.4 Channel Priority Management

Each channel can be given a low- or a high-priority level using the PRIORITY bit DMA.DMA_CCR[6]. When several channels request access to the same port, the priorities are established as follows:

- High-priority channels are served first.
- Low-priority channels are served only if all high-priority channels are served, stalled (by a slow source or destination), or waiting for a synchronization event.
- Channels at the same priority level are served in a round-robin manner. In this case, their requests are time-multiplexed.

Figure 13-4 shows an example of requests from three time-multiplexed channels. Channels x, y, and z can be read or write requests.

Figure 13-4. Time-Multiplex Policy on a Port



13.2.5 Secure Channel Management

The secure channel feature ensures the security of the data transferred between the protected memory area and the crypto modules. A channel is set as secure using the DMA.DMA_SCR register. This register can be accessed only when the MPU is runs a secure application.

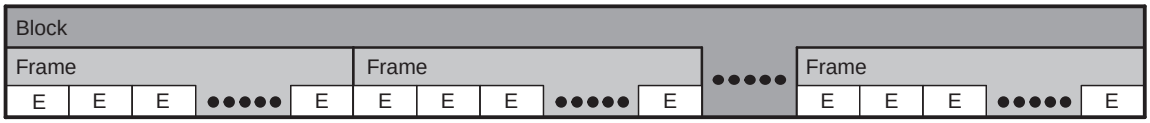
When a channel is set as secure, its dedicated registers can be accessed only by a secure application. Write accesses from a nonsecure application to the register set of a secure channel are discarded and read accesses are returned as a meaningless value (0x0000).

Note: When a channel is set as a secure channel, it is automatically allocated to the MPU, overriding the corresponding bit of the DMA.DMA_CAR register.

13.2.6 Transferred Data Block

The amount of data (block size) to transfer is programmed in bytes; this size can be odd or even. The data block to transfer is split in frames and elements, as shown in Figure 13-5.

Figure 13-5. Data Block Overview



The data block size in bytes is expressed as:

BS = FN x EN x ES

Where:

- BS: Block size
- FN: The number of frames in the block, 1 ≤ FN ≤ 65535 (unsigned), which is defined in the channel frame number register DMA.DMA_CFN
- EN: The number of elements per frame, 1 ≤ EN ≤ 65535 (unsigned), which is defined in the channel element number register DMA.DMA_CEN
- ES: The number of bytes per element, ES ∈ {1, 2, 4}, which is defined with the DATA_TYPE field DMA.DMA_CSDP[1:0]

CAUTION

Setting FN or EN equal to 0 is not allowed because it causes undefined results.

Note: FN, EN, and ES are the same for both source and destination.

13.2.7 Port Allocations

A source port and a destination port must be configured for every channel used. The choice is made by setting the SRC field DMA.DMA_CSDP[5:2] and the DST field DMA.DMA_CSDP[12:9]. Table 13-8 lists the devices for each port.

The following sections describe the sharing policy of each port between the MPU and the DMA. The main goal of the sharing policies is to avoid loss of performance on the MPU side and to maintain good throughput on the DMA side.

13.2.7.1 TIPB Port

The priority for accessing this bus can be given to either the MPU or the DMA. By default, the RHEA_PRIORITY bit DMA.DMA_AR[3] is set to 1, giving the priority to the MPU; the MPU has access to this bus for control and settings. This default setting avoids stalling the MPU with DMA accesses. If the RHEA_PRIORITY bit is set to 0, the MPU stalls when the MPU and the DMA perform concurrent accesses.

13.2.7.2 API Port

The priority for accessing this bus can be given to either the MPU or the DMA. By default, the API_PRIORITY bit DMA.DMA_AR[4] is set to 1, giving priority to the MPU. This default setting avoids stalling MPU accesses with DMA accesses, because the MPU has access to this memory to exchange data with the DSP.

13.2.7.3 IMIF Port

This memory handles traffic coming from the MPU and the DMA. On the one hand, the MPU uses this memory to run programs; therefore, the sharing policy must avoid the loss of MPU performance. On the other hand, the DMA must have enough bandwidth to perform data transfers.

Because this trade-off is highly dependent on the running application, a static priority mechanism cannot be used, unlike with the API port and the TIPB port. The mechanism implemented grants allocation of a minimum amount of bandwidth to the MPU and to the DMA.

The allocation of bandwidth is defined by the IMIF_PRIORITY field DMA.DMA_AR[2:0]. Depending on its value, MPU bandwidth varies from 0 percent to 87.5 percent, and DMA bandwidth varies from 100 percent to 12.5 percent (see Table 13-5). These are minimum values; for example, if no transfers are required by the MPU, 100% of the bandwidth is used by the DMA. Otherwise, a minimum bandwidth is allocated to the DMA based on the IMIF_PRIORITY field setting.

Table 13-5. Distribution of the IMIF Bandwidth Between the MPU and the DMA

IMIF_PRIORITY Field Value	Minimum MPU Bandwidth Granted	Minimum DMA Bandwidth Granted
0	0%	100%
1	50%	50%
2	66.6%	33.4%
3	75%	25%
4	80%	20%
5	83.3%	16.7%
6	85.7%	14.3%
7	87.5%	12.5%

13.2.7.4 EMIF Port

The EMIF port handles traffic coming from the MPU and the DMA to external memory. The priority is managed inside the EMIF module. For more information, see Chapter 11, *EMIF*.

13.2.7.5 IPERIPH Port

The IPERIPH port handles traffic coming from the MPU and the DMA and is also used to access internal memory, such as peripherals. The sharing policy is the same as for the IMIF port. Table 13-6 shows the allocation of the bandwidth between the MPU and the DMA according to the IPERIPH_PRIORITY field DMA.DMA_AR[2:0]. The given values are minimum values; that is, if no transfers are required by the MPU, 100 percent of the bandwidth is used by the DMA. Otherwise, a minimum bandwidth is allocated to the DMA based on the IPERIPH_PRIORITY field setting.

Table 13-6. Distribution of the IPERIPH Bandwidth Between the MPU and the DMA

IPERIPH_PRIORITY Field Value	Minimum MPU Bandwidth Granted	Minimum DMA Bandwidth Granted
0	0%	100%
1	50%	50%
2	66.6%	33.4%
3	75%	25%
4	80%	20%
5	83.3%	16.7%
6	85.7%	14.3%
7	87.5%	12.5%

13.2.7.6 Port Capability

The capability of a port is the widest bus that it can handle. Table 13-7 lists the capability of each port and the supported access widths.

Table 13-7. Port Capability

Port	Port Capability	Port Access Supported
TIPB	16 bits	8/16 bits
API	16 bits	8/16 bits
IMIF	32 bits	8/16/32 bits
EMIF	32 bits	8/16/32 bits
IPERIPH	32 bits	8/16/32 bits

13.2.8 Addressing Modes

The channel source and destination start addresses are configured in separate registers (DMA.DMA_CSSA_L/U and DMA.DMA_CDSA_L/U, respectively). The start addresses for a transfer are byte addresses and can be odd (not word-aligned). To track the progress of the transfer, check the DMA.DMA_CPC register, which gives the location of the 16 address least-significant bits (LSBs) of the last data written and the address space of each device accessed by DMA.

Table 13-8. Devices Accessed Through DMA

Device Name	Start Address	End Address	Size	Data Access
		Port: EMIF		
External memory	0x0040 0000	0x07FF FFFF	124M bytes	8/16/32 bits R/W
		Port: IMIF		
Protected RAM	0x0800 0000	0x0803 FFFF	256K bytes	8/16/32 bits R/W
Nonprotected RAM	0x0804 0000	0x0804 FFFF	64K bytes	8/16/32 bits R/W
ROM	0x0805 0000	0x0807 FFFF	192K bytes	8/16/32 bits R
Reserved	0x0808 0000	0x08FF FFFF		
		Port: IPERIPH		
Boot ROM	0x0900 0000	0x0900 FFFF	64K bytes	8/16/32 bits R
Reserved	0x0901 0000	0x096F FFFF		
Camera(1)	0x0970 0000	0x097F FFFF	1M bytes	32 bits R/W
SHA-1/MD5	0x0980 0000	0x098F FFFF	1M byte	32 bits R/W
DES/3DES	0x0990 0000	0x099F FFFF	1M byte	32 bits R/W
RNG	0x09A0 0000	0x09AF FFFF	1M byte	32 bits R/W
TI reserved(2)	0x09B0 0000	0x09CF FFFF		
NAND flash	0x09D0 0000	0x09DF FFFF	1M byte	8/16/32 bits R/W
MSSPI	0x09E0 0000	0x09EF FFFF	1M byte	32 bits R/W
Debug unit	0x09F0 0000	0x09FF FFFF	1M byte	32 bits R/W
Reserved	0x0A00 0000	0xFFCF FFFF		
		Port: API		
API RAM	0xFFD0 0000	0xFFD0 7FFF	32K bytes	16/32 bits R/W
Reserved	0xFFD0 8000	0xFFDF FFFF		
API control register	0xFFE0 0000	0xFFE0 0001	2 bytes	16 bits R/W
Reserved	0xFFE0 0002	0xFFFD FFFF		

(1) The camera interface is not available on the LOCOSTO Lite device.
(2) To avoid hazardous effects, do not write into this space.

Table 13-8. Devices Accessed Through DMA (continued)

Device Name	Start Address	End Address	Size	Data Access
		Port: TIPB		
DRP reserved(1)	0xFFFF 0000	0xFFFF 3FFF		
DRP external memory	0xFFFF 4000	0xFFFF 4FFF	4K bytes	16 bits R/W
APC(3)	0xFFFF 5000	0xFFFF 57FF	2K bytes	16 bits R/W
Reserved	0xFFFF 5800	0xFFFF 6FFF		
UART(3)	0xFFFF 7000	0xFFFF 727F	640 bytes	8 bits R/W
UART TIPB/OCP switch	0xFFFF 7280	0xFFFF 77FF	1.375K bytes	8 bits R/W
MCSI(3)	0xFFFF 7800	0xFFFF 7FFF	2K bytes	16 bits R/W
DRP dynamic switch	0xFFFF 8000	0xFFFF 87FF	2K bytes	16 bits R/W
APC TIPB static switch	0xFFFF 8800	0xFFFF 881F	32 bytes	16 bits R/W
MCSI TIPB static switch	0xFFFF 8820	0xFFFF 883F	32 bytes	16 bits R/W
C-port TIPB static switch	0xFFFF 8840	0xFFFF 885F	32 bytes	16 bits R/W
Reserved	0xFFFF 8860	0xFFFF 8FFF		
TPU RAM	0xFFFF 9000	0xFFFF 97FF	2K bytes	16 bits R/W
DPLL	0xFFFF 9800	0xFFFF 9FFF	2K bytes	16 bits R/W
LCD interface	0xFFFF A000	0xFFFF A7FF	2K bytes	16 bits R/W
USIM interface	0xFFFF A800	0xFFFF AFFF	2K bytes	16 bits R/W
USB	0xFFFF B000	0xFFFF B7FF	2K bytes	16 bits R/W
I^2C	0xFFFF B800	0xFFFF BFFF	2K bytes	16 bits R/W
GEA3	0xFFFF C000	0xFFFF C7FF	2K bytes	16 bits R/W
I^2C TWL3031	0xFFFF C800	0xFFFF CFFF	2K bytes	16 bits R/W
C-port (FIFO registers)(3)	0xFFFF D800	0xFFFF DFFF	2K bytes	16 bits R/W
C-port (control registers)(3)	0xFFFF D000	0xFFFF D7FF	2K bytes	16 bits R/W
TI reserved	0xFFFF E000	0xFFFF E7FF		
DMA controller	0xFFFF E800	0xFFFF EFFF	2K bytes	16 bits R/W
TPU	0xFFFF F000	0xFFFF F7FF	2K bytes	16 bits R/W
Watchdog	0xFFFF F800	0xFFFF F87F	128 bytes	16 bits R/W
Secure watchdog	0xFFFF F880	0xFFFF F8FF	128 bytes	16 bits R/W
API/TIPB bridge	0xFFFF F900	0xFFFF F9FF	256 bytes	16 bits R/W
Interrupt handler	0xFFFF FA00	0xFFFF FA7F	128 bytes	16 bits R/W
Secure interrupt handler	0xFFFF FA80	0xFFFF FAFF	128 bytes	16 bits R/W
EMIF	0xFFFF FB00	0xFFFF FB2D	46 bytes	16 bits R/W
API_RHEA_CNTL register	0xFFFF FB2E	0xFFFF FB2F	2 bytes	16 bits R/W
BOOT_MODE_CONF register	0xFFFF FB30	0xFFFF FB31	2 bytes	16 bits R/W
Reserved	0xFFFF FB32	0xFFFF FBFF		
PRRM	0xFFFF FC00	0xFFFF FCFF	256 bytes	16 bits R/W
CLKM	0xFFFF FD00	0xFFFF FDFF	256 bytes	16 bits R/W
JTAG IDCODE	0xFFFF FE00	0xFFFF FE03	4 bytes	16 bits R/W
EMPU	0xFFFF FF00	0xFFFF FFFF	256 bytes	16 bits R/W

(3) Module behind a configurable switch

An addressing mode is an address computation algorithm used by a DMA channel to determine the location of the data to be accessed. The DMA subsystem proposes to the software the following three addressing modes:

- Constant
- Post-incremented
- Frame-indexed

These modes are defined for both the source and the destination of the transfer by setting the SRC_ADD_MODE field DMA.DMA_CCR[13:12] and the DST_ADD_MODE bit DMA.DMA_CCR[15:14] field, respectively.

13.2.8.1 Constant Addressing Mode

For each element, the address remains constant. If Add(*n*) is the address of the byte at position *n* in the transfer, then Add(*n*) always equals the start address (SA) of the transfer.

13.2.8.2 Post-Incremented Addressing Mode

The address is always incremented by the size of the element (ES):

- The address of the first byte of the transfer gets the start address: Add(0) = SA.
- The addresses of the following elements are incremented by the ES: Add(*n*+1) = A(*n*) + ES.

13.2.8.3 Frame-Indexed Addressing Mode

This mode is a mix of the constant mode and the post-incremented mode. As long as the bytes belong to the same frame, their addresses are incremented by the ES: Add(0) = SA and Add(*n*+1) = A(*n*) + ES. At a new frame, the address restarts at the same start address as the previous frame.

13.2.9 Channel Reset

Several registers dedicated to a channel can be reset together by setting the corresponding bit of the DMA.DMA_SRR register to 1. The following registers are affected:

- The channel source start address: DMA.DMA_CSSA_L and DMA.DMA_CSSA_U
- The channel destination start address: DMA.DMA_CDSA_L and DMA.DMA_CDSA_U
- The number of frames in the block data: DMA.DMA_CFN
- The number of elements per frame: DMA.DMA_CEN
- The progress counter: DMA.DMA_CPC

13.2.10 Channel Synchronization and Transfer Start

Whether or not a channel is defined as a synchronized channel, it must be enabled by setting the EN bit DMA.DMA_CCR[7]. To synchronize the channel, set the SYNC_CNTL field DMA.DMA_CCR[4:0].

If the SYNC_CNTL field is 0x0, the transfer start is not synchronized on a DMA request. In this mode, a transfer begins immediately after setting the EN bit DMA.DMA_CCR[7]. The configuration registers must be set before this action.

If the value of the SYNC_CNTL field corresponds to one of the DMA requests listed in Table 13-9, the start of the transfer is triggered on a DMA request. The SYNC_STATUS bit DMA.DMA_CSR[6] indicates that a DMA request is made for the channel and the data transfer has not yet started.

Table 13-9. DMA Request Mapping

Peripheral Name	DMA Request Number	Request Type
DRP	1	TX
	2	RX
LCD interface	3	TX
UART IRDA/MODEM	4	TX
	5	RX
MSSPI	6	TX
	7	RX
Reserved	8	
	9	

Table 13-9. DMA Request Mapping (continued)

Peripheral Name	DMA Request Number	Request Type
USB	10	RX 1
	11	TX 1
	12	RX 2
	13	TX 2
	14	RX 3
	15	TX 3
I^2C (TWL3031)	16	RX
	17	TX
Reserved	18	
USIM	19	RX
	20	TX
Camera(1)	21	Req(2)
Reserved	22	
	23	
NAND flash	24	RX/TX(3)
I^2C	25	RX
	26	TX
SHA-1MD5	27	TX
DES3DES	28	RX
	29	TX
C-PORT (I2S)	30	RX
	31	TX

(1) The camera interface is not available on the LOCOSTO Lite device.
(2) Two time-multiplexed DMA requests coming from the camera module are triggered based on a FIFO fill threshold. For more information, see Chapter 17, *Camera Interface*.
(3) The requests are time-multiplexed inside the NAND flash interface. For more information, see Chapter 10, *NAND Flash Controller*.

13.2.11 Transfer Stop

13.2.11.1 Stop Manually

To stop a transfer manually, reset the EN bit DMA.DMA_CCR[7].

13.2.11.2 Auto-Initialization

A DMA channel (synchronized or not) can operate in two modes: single-transfer mode and auto-initialization mode.

- In single-transfer mode, the channel is disabled when the transfer of the current block is completed. The channel must be re-enabled through the EN bit DMA.DMA_CCR[7].
- In auto-initialization mode, the channel is not disabled when the transfer of the current block is completed. Therefore, it can service a new DMA request without any software overhead.

13.2.11.3 Suspended State

A synchronized channel enters a suspended state after a DMA request is serviced and then waits for a new DMA request. In this case, auto-initialization mode must be activated; otherwise, the channel is disabled.

All DMA channels go into a suspended state when the following occur:

- The CLKM module sends an idle request. The DMA completes all pending operations before giving an idle acknowledge; then the clock can be cut off externally to save power.
- The DMA receives a suspending request from the MPU (for emulation), and the FREE bit DMA.DMA_GCR[2] is set to 0. The processor asserts this input when it is halted by an emulator breakpoint. When this occurs, depending on the value of the FREE bit, the DMA suspends or continues its transfer. The DMA clock must be on when the DMA is suspended with the suspend input.

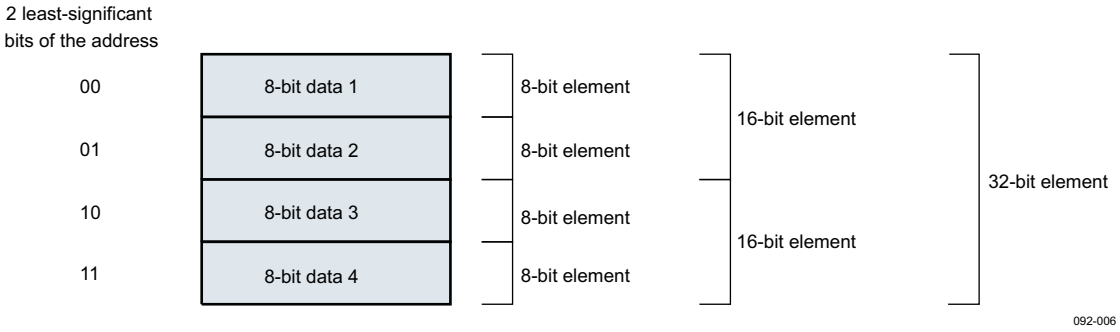
13.2.12 Data Management

13.2.12.1 Data Alignment

During a transfer, all addresses computed by the DMA are aligned on the transferred element size (see Section 13.2.6, *Transferred Data Block*) depending on the following conditions:

- If the element size is 8 bits, the addresses can have any value (see Figure 13-6).
- If the element size is 16 bits, the addresses are aligned on the 16-bit word boundary (the lowest bit of the address; always 0).
- If the element size is 32 bits, the addresses are aligned on the 32-bit word boundary (the 2 lowest bits of the address; always 00).

Figure 13-6. Data Alignment



13.2.12.2 Data Packing

A DMA channel can pack several consecutive accesses into a single access. The data packing feature is restricted by data alignment and port capability. The data cannot always be packed if the resulting packed data is not aligned. For example, in Figure 13-6, data 1 and data 2 can be packed, but data 2 and data 3 cannot be packed.

Note: It is unnecessary to pack data when the port capability is equal to the data type.

CAUTION

The software should enable data packing only when the device capability is at least equal to the port capability.

13.2.12.2.1 Source

The SRC_PACK bit DMA.DMA_CSDP[6] allows the DMA to request data from the source port at the maximum data width of the port (see Table 13-7 for the port capability).

13.2.12.2.2 Destination

The DST_PACK bit DMA.DMA_CSDP[13] allows the channel to pack the data stored in its FIFO based on data alignment and destination port capability (for port capability, see Table 13-7).

13.2.12.2.3 Burst Transactions

A burst transaction consists of four 32-bit single accesses that cannot be interrupted or interleaved with other accesses. The SRC_BURST field DMA.DMA_CSDP[8:7] and the DST_BURST field DMA.DMA_CSDP[15:14] allow the DMA to access, respectively, the source and the destination in burst mode. In both cases, the port capability must be 32 bits.

13.2.13 DMA Interrupts

Events on a DMA channel can generate an interrupt. Depending on the allocation and the secure mode of a channel, this interrupt is sent to the following:

- The MPU interrupt handler when the channel is allocated to the MPU and is defined as a nonsecure channel
- The secure interrupt handler when the channel is allocated to the MPU and is defined as a secure channel
- The DSP interrupt handler when the channel is allocated to the DSP

Because an interrupt can occur on any channel, the DMA.DMA_ISR register must be read to determine which channel is the source of the interrupt.

The DMA.DMA_CSR register must then be read to find the event that triggered the interrupt. Table 13-10 lists the events that can trigger an interrupt, the channel status register, and the interrupt control register.

Table 13-10. Events Generating an Input

Event	Register Bits that Enable the Interrupt	Corresponding Status Register Bits	Description
Time-out	IT_TIME_OUT_IE bit DMA.DMA_CICR[0]	IT_TIME_OUT bit DMA.DMA_CSR[0]	An access to the source or the destination timed out. To prevent a channel from definitively locking a memory or peripheral, all DMA ports support time-out information from the accessed resource. The channel is disabled when a time-out occurs.
Overload	IT_OVERLOAD_IE bit DMA.DMA_CICR[1]	IT_OVERLOAD bit DMA.DMA_CSR[1]	A new DMA request occurred before the end-of-service of the previous request.
Frame transferred	IT_FRAME_IE bit DMA.DMA_CICR[3]	IT_FRAME bit DMA.DMA_CSR[3]	The last byte of the current frame is written to its destination.
Block transferred	IT_BLOCK_IE bit DMA.DMA_CICR[5]	IT_BLOCK bit DMA.DMA_CSR[5]	The last byte of the transfer is written to its destination.
First half of the block transferred	IT_HALF_BLOCK_IE bit DMA.DMA_CICR[6]	IT_HALF_BLOCK bit DMA.DMA_CSR[6]	The last byte of the last frame of the first half block is written to its destination.

13.3 DMA Programming Model

13.3.1 Setup for Typical Configurations

Table 13-11 lists the global parameters that configure the DMA controller. Each parameter is associated with the register that defines it. Table 13-12 lists the parameters that configure each DMA channel.

Table 13-11. Global Parameters of the DMA Controller

Parameters	Register	Description
Allocation	DMA.DMA_CAR	Gives control of the channel to either the MPU or the DSP
Secure	DMA.DMA_SCR	Defines the channel as either a secure channel or a nonsecure channel
Priority DMA_UnicodeEncodeError_MPU	DMA.DMA_AR	Defines the bandwidth allocated to the MPU and to the DMA
Power saving	DMA.DMA_GCR[3]	Cuts off the clock when the DMA subsystem has no activity

Table 13-12. Parameters of a DMA Channel

Parameters	Register	Description
Source and destination ports	DMA.DMA_CSDP[5:2] and DMA.DMA_CSDP[12:9]	Selects the source port and the destination port among five ports
Source address	DMA.DMA_CSSA_L and DMA.DMA_CSSA_U	Start address of the transfer on the source device side
Destination address	DMA.DMA_CDSA_L and DMA.DMA_CDSA_U	Start address of the transfer on the destination device side
Pack	DMA.DMA_CSDP[6] and DMA.DMA_CSDP[13]	Allows the packing feature at the source level or at the destination level
Burst	DMA.DMA_CSDP[8:7] and DMA.DMA_CSDP[15:14]	Allows burst transactions at the source level or at the destination level
Addressing modes	DMA.DMA_CCR[15:14] and DMA.DMA_CCR[13:12]	Selects source and the destination addressing modes
Priority	DMA.DMA_CCR[6]	Allocates priority between channels
Synchronization on a DMA request	DMA.DMA_CCR[4:0]	Allows a transfer to start on a DMA request
Auto-initialization	DMA.DMA_CCR[8]	Keeps the channel enabled after the end of a transfer
Interrupt enable	DMA.DMA_CICR	Selects the events that generate an interrupt
Element number	DMA.DMA_CEN	Defines the number of elements per frame
Frame number	DMA.DMA_CFN	Defines the number of frames in the block to transfer
Data type	DMA.DMA_CSDP[1:0]	Defines the element size

13.4 DMA Register Manual

Table 13-13 lists the DMA instance summary. Table 13-14 provides an overview of the register mapping.

Table 13-13. Instance Summary

Module Name	MPU		DSP	
	Base Address	Size	Base Address	Size
DMA	0xFFFF E800	2K bytes	0xE800	4K bytes

Table 13-14. Register Mapping Overview

	MPU		DSP	
	Start Address	End Address	Start Address	End Address
Channel 0	0xFFFF E800	0xFFFF E83F	0xE800	0xE81F
Channel 1	0xFFFF E840	0xFFFF E87F	0xE820	0xE83F
Channel 2	0xFFFF E880	0xFFFF E8BF	0xE840	0xE85F
Channel 3	0xFFFF E8C0	0xFFFF E8FF	0xE860	0xE87F

Table 13-14. Register Mapping Overview (continued)

	MPU		DSP	
Channel 4	0xFFFF E900	0xFFFF E93F	0xE880	0xE89F
Channel 5	0xFFFF E940	0xFFFF E97F	0xE8A0	0xE8BF
Reserved	0xFFFF E980	0xFFFF EBFF	0xE8C0	0xE9FF
Global registers	0xFFFF EC00	0xFFFF EC0B	0xEA00	0xEA05
Reserved	0xFFFF EC0C	0xFFFF EFFF	0xE8A6	0xEFFF

13.4.1 Module Register Mapping Summary

Table 13-15 lists the DMA global addresses. Table 13-16 lists the channel register addresses through the MPU. Table 13-17 lists the channel register addresses through the DSP.

Table 13-15. Global Register Offset Address

Register Name	Register Width (bits)	MPU			DSP		
		Type	Address Offset	Physical Address	Type	Address Offset	Physical Address
DMA.DMA_GCR	16	R/W	0x0	0xFFFF EC00	Not accessible		
DMA.DMA_ISR	16	R	0x2	0xFFFF EC02	R	0x1	0xEA01
DMA.DMA_CAR	16	R/W[1]	0x4	0xFFFF EC04	R	0x2	0xEA02
DMA.DMA_SCR	16	R/W[2]	0x6	0xFFFF EC06	R	0x3	0xEA03
DMA.DMA_SRR	16	R/W	0x8	0xFFFF EC08	R	0x4	0xEA04
DMA.DMA_AR	16	R/W	0xA	0xFFFF EC0A	Not accessible		

1 Write only in supervisor mode
2 Write only in secure mode

092-T15

Table 13-16. Channel Register Addresses Through MPU

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address					
				Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
DMA.DMA_CSDP	R/W	16	0x00	0xFFFF E800	0xFFFF E840	0xFFFF E880	0xFFFF E8C0	0xFFFF E900	0xFFFF E940
DMA.DMA_CCR	R/W	16	0x02	0xFFFF E802	0xFFFF E842	0xFFFF E882	0xFFFF E8C2	0xFFFF E902	0xFFFF E942
DMA.DMA_CICR	R/W	16	0x04	0xFFFF E804	0xFFFF E844	0xFFFF E884	0xFFFF E8C4	0xFFFF E904	0xFFFF E944
DMA.DMA_CSR	R	16	0x06	0xFFFF E806	0xFFFF E846	0xFFFF E886	0xFFFF E8C6	0xFFFF E906	0xFFFF E946
DMA.DMA_CSSA_L	R/W	16	0x08	0xFFFF E808	0xFFFF E848	0xFFFF E888	0xFFFF E8C8	0xFFFF E908	0xFFFF E948
DMA.DMA_CSSA_U	R/W	16	0x0A	0xFFFF E80A	0xFFFF E84A	0xFFFF E88A	0xFFFF E8CA	0xFFFF E90A	0xFFFF E94A
DMA.DMA_CDSA_L	R/W	16	0x0C	0xFFFF E80C	0xFFFF E84C	0xFFFF E88C	0xFFFF E8CC	0xFFFF E90C	0xFFFF E94C
DMA.DMA_CDSA_U	R/W	16	0x0E	0xFFFF E80E	0xFFFF E84E	0xFFFF E88E	0xFFFF E8CE	0xFFFF E90E	0xFFFF E94E
DMA.DMA_CEN	R/W	16	0x10	0xFFFF E810	0xFFFF E850	0xFFFF E890	0xFFFF E8D0	0xFFFF E910	0xFFFF E950
DMA.DMA_CFN	R/W	16	0x12	0xFFFF E812	0xFFFF E852	0xFFFF E892	0xFFFF E8D2	0xFFFF E912	0xFFFF E952
RESERVED	R	16	0x14						
RESERVED	R	16	0x16						
DMA.DMA_CPC	R	16	0x18	0xFFFF E818	0xFFFF E858	0xFFFF E898	0xFFFF E8D8	0xFFFF E918	0xFFFF E958

092-T16

Table 13-17. Channel Register Addresses Through DSP

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address					
				Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
DMA.DMA_CSDP	R/W	16	0x0	0xE800	0xE820	0xE840	0xE860	0xE880	0xE8A0
DMA.DMA_CCR	R/W	16	0x1	0xE801	0xE821	0xE841	0xE861	0xE881	0xE8A1
DMA.DMA_CICR	R/W	16	0x2	0xE802	0xE822	0xE842	0xE862	0xE882	0xE8A2
DMA.DMA_CSR	R	16	0x3	0xE803	0xE823	0xE843	0xE863	0xE883	0xE8A3
DMA.DMA_CSSA_L	R/W	16	0x4	0xE804	0xE824	0xE844	0xE864	0xE884	0xE8A4
DMA.DMA_CSSA_U	R/W	16	0x5	0xE805	0xE825	0xE845	0xE865	0xE885	0xE8A5
DMA.DMA_CDSA_L	R/W	16	0x6	0xE806	0xE826	0xE846	0xE866	0xE886	0xE8A6
DMA.DMA_CDSA_U	R/W	16	0x7	0xE807	0xE827	0xE847	0xE867	0xE887	0xE8A7
DMA.DMA_CEN	R/W	16	0x8	0xE808	0xE828	0xE848	0xE868	0xE888	0xE8A8
DMA.DMA_CFN	R/W	16	0x9	0xE809	0xE829	0xE849	0xE869	0xE889	0xE8A9
RESERVED	R	16	0xA						
RESERVED	R	16	0xB						
DMA.DMA_CPC	R	16	0xC	0xE80C	0xE82C	0xE84C	0xE86C	0xE88C	0xE8AC

092-T17

13.4.2 Register Description

13.4.2.1 Global Registers

Table 13-18 through Table 13-23 describe the individual global registers.

Table 13-18. DMA.DMA_GCR

Address Offset	MPU: 0x0		
	DSP: NA		
Physical Address	MPU: 0xFFFF EC00	**Instance**	DMA global register
	DSP: NA		
Description	Global control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												AUTOGATING_ON	FREE	Reserved	

Bits	Field Name	Description		Type	Reset
15:4	Reserved	Write has no effect. Read returns 0x0.		R	0x0
3	AUTOGATING_ON	0:	The DMA clocks are always running.	RW	0X1
		1:	Allows the DMA to cut off its clocks according to its activity		
2	FREE	0:	Ongoing transfers are suspended when the DMA receives the suspend signal from the MPU.	RW	0x0
		1:	Ongoing transfers are not stopped when the DMA receives the suspend signal from the MPU.		

Bits	Field Name	Description	Type	Reset
1:0	Reserved	Write has no effect. Read returns 0x0.	R	0x0

Table 13-19. DMA.DMA_ISR

Address Offset	MPU: 0x2 DSP: 0x1		
Physical Address	MPU: 0xFFFF EC02 DSP: 0xEA01	**Instance**	DMA global register
Description	Interrupt source register; the status bits are automatically cleared after they are read.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										IT_CHANNEL_5	IT_CHANNEL_4	IT_CHANNEL_3	IT_CHANNEL_2	IT_CHANNEL_1	IT_CHANNEL_0

Bits	Field Name	Description		Type	Reset
15:6	Reserved	Write has no effect. Read returns 0x0.		R	0x0
5	IT_CHANNEL_5	0:	No event	R	0x0
		1:	Interrupt from channel 5		
4	IT_CHANNEL_4	0:	No event	R	0x0
		1:	Interrupt from channel 4		
3	IT_CHANNEL_3	0:	No event	R	0x0
		1:	Interrupt from channel 3		
2	IT_CHANNEL_2	0:	No event	R	0x0
		1:	Interrupt from channel 2		
1	IT_CHANNEL_1	0:	No event	R	0x0
		1:	Interrupt from channel 1		
0	IT_CHANNEL_0	0:	No event	R	0x0
		1:	Interrupt from channel 0		

Table 13-20. DMA.DMA_CAR

Address Offset	MPU: 0x4 DSP: 0x2		
Physical Address	MPU: 0xFFFF EC04 DSP: 0xEA02	**Instance**	DMA global register
Description	Channel allocation register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										ALLOC_CHANNEL_5	ALLOC_CHANNEL_4	ALLOC_CHANNEL_3	ALLOC_CHANNEL_2	ALLOC_CHANNEL_1	ALLOC_CHANNEL_0

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Write has no effect. Read returns 0x0.	R	0x0
5	ALLOC_CHANNEL_5	Channel 5 accessed by: 0: MPU 1: DSP	RW	0x0
4	ALLOC_CHANNEL_4	Channel 4 accessed by: 0: MPU 1: DSP	RW	0x0
3	ALLOC_CHANNEL_3	Channel 3 accessed by: 0: MPU 1: DSP	RW	0x0
2	ALLOC_CHANNEL_2	Channel 2 accessed by: 0: MPU 1: DSP	RW	0x0
1	ALLOC_CHANNEL_1	Channel 1 accessed by: 0: MPU 1: DSP	RW	0x0
0	ALLOC_CHANNEL_0	Channel 0 accessed by: 0: MPU 1: DSP	RW	0x0

Table 13-21. DMA.DMA_SCR

Address Offset	MPU: 0x6 DSP: 0x3		
Physical Address	MPU: 0xFFFF EC06 DSP: 0xEA03	**Instance**	DMA global register
Description	Secure channel register		
Type	Read/write only in secure mode		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										SEC_CHANNEL_5	SEC_CHANNEL_4	SEC_CHANNEL_3	SEC_CHANNEL_2	SEC_CHANNEL_1	SEC_CHANNEL_0

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Write has no effect. Read returns 0x0.	R	0x0
5	SEC_CHANNEL_5	0: Unsecured channel 5 1: Secured channel 5	RW	0x0
4	SEC_CHANNEL_4	0: Unsecured channel 4 1: Secured channel 4	RW	0x0
3	SEC_CHANNEL_3	0: Unsecured channel 3 1: Secured channel 3	RW	0x0
2	SEC_CHANNEL_2	0: Unsecured channel 2 1: Secured channel 2	RW	0x0
1	SEC_CHANNEL_1	0: Unsecured channel 1 1: Secured channel 1	RW	0x0
0	SEC_CHANNEL_0	0: Unsecured channel 0 1: Secured channel 0	RW	0x0

Table 13-22. DMA.DMA_SRR

Address Offset	MPU: 0x8		
	DSP: 0x4		
Physical Address	MPU: 0xFFFF EC08	**Instance**	DMA global register
	DSP: 0xEA04		
Description	Soft reset register		
	Reset the following registers: DMA_CSSA_L, DMA_CSSA_U, DMA_CDSA_L, DMA_CDSA_U, DMA_CEN, DMA_CFN, and DMA_CPC.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RST_CHANNEL_5	RST_CHANNEL_4	RST_CHANNEL_3	RST_CHANNEL_2	RST_CHANNEL_1	RST_CHANNEL_0

Bits	Field Name	Description		Type	Reset
15:6	Reserved	Write has no effect. Read returns 0x0.		R	0x0
5	RST_CHANNEL_5	0:	Reset complete	RW	0x0
		1:	Reset the register of channel 5		
4	RST_CHANNEL_4	0:	Reset complete	RW	0x0
		1:	Reset the register of channel 4		
3	RST_CHANNEL_3	0:	Reset complete	RW	0x0
		1:	Reset the register of channel 3		
2	RST_CHANNEL_2	0:	Reset complete	RW	0x0
		1:	Reset the register of channel 2		
1	RST_CHANNEL_1	0:	Reset complete	RW	0x0
		1:	Reset the register of channel 1		
0	RST_CHANNEL_0	0:	Reset complete	RW	0x0
		1:	Reset the register of channel 0		

Table 13-23. DMA.DMA_AR

Address Offset	MPU: 0xA		
	DSP: NA		
Physical Address	MPU: 0xFFFF EC0A	**Instance**	DMA global register
	DSP: NA		
Description	Arbiter register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								IPERIPH_PRIORITY			API_PRIORITY	RHEA_PRIORITY	IMIF_PRIORITY		

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write has no effect. Read returns 0x0.	R	0x0
7:5	IPERIPH_PRIORITY	Access priority on the memory-like peripherals between the MPU and the DMA. This field defines the number of consecutive cycles allocated to the MPU before DMA has one cycle allocated.	RW	0x4
4	API_PRIORITY	Access priority on the API memory between the MPU and the DMA. When the MPU and the DMA perform simultaneous accesses: 0: Access granted to DMA 1: Access granted to MPU	RW	0x1
3	RHEA_PRIORITY	Access priority on the TIPB between the MPU and the DMA. When the MPU and the DMA perform simultaneous accesses: 0: Access granted to DMA 1: Access granted to MPU	RW	0x1
2:0	IMIF_PRIORITY	Access priority on the internal memory interface between the MPU and the DMA. This field defines the number of consecutive cycles allocated to the MPU before the DMA has one cycle allocated.	RW	0x4

13.4.2.2 Dedicated Channel Registers

Table 13-24 through Table 13-34 describe the individual dedicated channel registers.

Table 13-24. DMA.DMA_CSDP

Address Offset	MPU: 0x00 DSP: 0x0		
Physical Address	MPU: See Table 13-16. DSP: See Table 13-17.	**Instance**	DMA channel
Description	Channel source destination parameters		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DST_BURST		DST_PACK	DST					SRC_BURST		SRC_PACK	SRC			DATA_TYPE	

Bits	Field Name	Description	Type	Reset
15:14	DST_BURST	Destination burst 0X: Disable 1X: Enable	RW	0x0
13	DST_PACK	Destination packing 0: Disable 1: Enable	RW	0x0
12:9	DST	Transfer destination 0000: IMIF 0001: TIPB 0010: API 0011: Memory-like peripherals 0100: EMIF Others: Illegal values	RW	0x0
8:7	SRC_BURST	Source burst 0X: Disable 1X: Enable	RW	0x0
6	SRC_PACK	Source packing	RW	0x0

Bits	Field Name	Description		Type	Reset
		0:	Disable		
		1:	Enable		
5:2	SRC	Transfer source		RW	0x0
		0000:	IMIF		
		0001:	TIPB		
		0010:	API		
		0011:	IPERIPH (memory-like peripherals)		
		0100:	EMIF		
		Others:	Illegal values		
1:0	DATA_TYPE	00:	8 bits	RW	0x0
		01:	16 bits		
		10:	32 bits		
		11:	Illegal value		

Table 13-25. DMA.DMA_CCR

Address Offset	MPU: 0x02 DSP: 0x1		
Physical Address	MPU: See Table 13-16. DSP: See Table 13-17.	**Instance**	DMA channel
Description	Channel control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DST_BURST_ADD_MODE		SRC_ADD_MODE		Reserved			AUTO_INIT	EN	PRIORITY	Reserved	SYNC_CNTL				

Bits	Field Name	Description		Type	Reset
15:14	DST_BURST_ADD_MODE	Destination addressing mode		RW	0x0
		00:	Constant address		
		01:	Post-incremented address		
		10:	Frame-indexed address		
		11:	Constant address		
13:12	SRC_ADD_MODE	Source addressing mode		RW	0x0
		00:	Constant address		
		01:	Post-incremented address		
		10:	Frame-indexed address		
		11:	Constant address		
11:9	Reserved	Write has no effect. Read returns 0x0.		R	0x0
8	AUTO_INIT	Auto-initialization at the end of the transfer		RW	0x0
		0:	The channel stops at the end of the current transfer.		
		1:	When the current transfer is complete, the channel automatically reinitializes itself and starts a new transfer.		
7	EN	0:	Stop and reset the transfer.	RW	0x0

Bits	Field Name	Description	Type	Reset
		1: Start the transfer.		
6	PRIORITY	Channel priority 0: Low priority 1: High priority	RW	0x0
5	Reserved	Write has no effect. Read returns 0x0.	R	0x0
4:0	SYNC_CNTL	Specifies which external DMA request can trigger a transfer in the channel. 0000: The transfer is not synchronized. Others: Transfer synchronized on the request n (value given by this field). See Table 13-9.	R	0x00

Table 13-26. DMA.DMA_CICR

Address Offset	MPU: 0x04 DSP: 0x2		
Physical Address	MPU: See Table 13-16. DSP: See Table 13-17.	**Instance**	DMA channel
Description	Channel interrupt control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									IT_HALF_BLOCK_IE	IT_BLOCK_IE	Reserved	IT_FRAME_IE	Reserved	IT_OVERLOAD_IE	IT_TIME_OUT_IE

Bits	Field Name	Description	Type	Reset
15:7	Reserved	Write has no effect. Read returns 0x0.	R	0x0
6	IT_HALF_BLOCK_IE	0: No interrupt is generated when half of the block has been transferred. 1: Interrupt is generated when half of the block has been transferred.	RW	0x0
5	IT_BLOCK_IE	0: No interrupt is generated when the transfer of the current block completes. 1: Interrupt is generated when the transfer of the current block completes.	RW	0x0
4	Reserved	Write has no effect. Read returns 0x0.	R	0x0
3	IT_FRAME_IE	0: No interrupt is generated when the transfer of the current frame completes. 1: Interrupt is generated when the transfer of the current frame completes.	RW	0x0
2	Reserved	Write has no effect. Read returns 0x0.	R	0x0
1	IT_OVERLOAD_IE	0: No interrupt is generated when a new request is made while the previous one is not finished. 1: Interrupt is generated when a new request is made while the previous one is not finished.	RW	0x1
0	IT_TIME_OUT_IE	0: No interrupt is generated when a time-out occurs. 1: Interrupt is generated when a time-out occurs.	RW	0x1

Table 13-27. DMA.DMA_CSR

Address Offset	MPU: 0x06		
	DSP: 0x3		
Physical Address	MPU: See Table 13-16.	**Instance**	DMA channel
	DSP: See Table 13-17.		
Description	Channel status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							IT_TIME_OUT_SRC_DST	IT_HALF_BLOCK	SYNC_STATUS	IT_BLOCK	Reserved	IT_FRAME	Reserved	IT_OVERLOAD	TIME_OUT

Bits	Field Name	Description		Type	Reset
15:9	Reserved	Write has no effect. Read returns 0x0.		R	0x0
8	IT_TIME_OUT_SRC_DST	Source or destination time-out status; linked to the IT_TIME_OUT bit		R	0x0
		0:	If IT_TIME_OUT = 1, a time-out occurred on the destination; otherwise, no time-out.		
		1:	If IT_TIME_OUT = 1, a time-out occurred on the source; otherwise, no time-out.		
7	IT_HALF_BLOCK	0:	No event	R	0x0
		1:	The first half block was transferred.		
6	SYNC_STATUS	0:	No DMA request is serviced.	R	0x0
		1:	A DMA request is made in a synchronized channel.		
5	IT_BLOCK	0:	No event	R	0x0
		1:	The current transfer is finished; another transfer may have started if the AUTO_INIT bit DMA.DMA_CCR[8] is set to 1.		
4	Reserved	Write has no effect. Read returns 0x0.		R	0x0
3	IT_FRAME	0:	No event	R	0x0
		1:	A complete frame was transferred.		
2	Reserved	Write has no effect. Read returns 0x0.		R	0x0
1	IT_OVERLOAD	0:	No event	R	0x0
		1:	An overload occurred.		
0	IT_TIME_OUT	0:	No event	R	0x0
		1:	A time-out occurred.		

Table 13-28. DMA.DMA_CSSA_L

Address Offset	MPU: 0x08		
	DSP: 0x4		
Physical Address	MPU: See Table 13-16.	**Instance**	DMA channel
	DSP: See Table 13-17.		
Description	Channel source start address: lower bits		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_START_ADDR_LSB															

Bits	Field Name	Description	Type	Reset
15:0	SRC_START_ADDR_LSB	Lower bits of the source start address	RW	Undefined

Table 13-29. DMA.DMA_CSSA_U

Address Offset	MPU: 0x0A DSP: 0x5		
Physical Address	MPU: See Table 13-16. DSP: See Table 13-17.	**Instance**	DMA channel
Description	Channel source start address: upper bits		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_START_ADDR_ MSB															

Bits	Field Name	Description	Type	Reset
15:0	SRC_START_ADDR_ MSB	Upper bits of the source start address	RW	Undefined

Table 13-30. DMA.DMA_CDSA_L

Address Offset	MPU: 0x0C DSP: 0x6		
Physical Address	MPU: See Table 13-16. DSP: See Table 13-17.	**Instance**	DMA channel
Description	Channel destination start address: lower bits		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DST_START_ADDR_LSB															

Bits	Field Name	Description	Type	Reset
15:0	DST_START_ADDR_LSB	Lower bits of the destination start address	RW	Undefined

Table 13-31. DMA.DMA_CDSA_U

Address Offset	MPU: 0x0E DSP: 0x7		
Physical Address	MPU: See Table 13-16. DSP: See Table 13-17.	**Instance**	DMA channel
Description	Channel destination start address: upper bits		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DST_START_ADDR_MSB															

Bits	Field Name	Description	Type	Reset
15:0	DST_START_ADDR_MSB	Upper bits of the destination start address	RW	Undefined

Table 13-32. DMA.DMA_CEN

Address Offset	MPU: 0x10 DSP: 0x8		
Physical Address	MPU: See Table 13-16. DSP: See Table 13-17.	**Instance**	DMA channel
Description	Channel element number		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEN															

Bits	Field Name	Description	Type	Reset
15:0	CEN	Defines the number of elements of a frame **Caution:** Must not be a 0, to avoid hazardous effects.	RW	Undefined

Table 13-33. DMA.DMA_CFN

Address Offset	MPU: 0x12 DSP: 0x9		
Physical Address	MPU: See Table 13-16. DSP: See Table 13-17.	**Instance**	DMA channel
Description	Channel frame number		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFN															

Bits	Field Name	Description	Type	Reset
15:0	CFN	Defines the number of frames within a block **Caution:** Must not be a 0, to avoid hazardous effects.	RW	Undefined

Table 13-34. DMA.DMA_CPC

Address Offset	MPU: 0x18 DSP: 0xC		
Physical Address	MPU: See Table 13-16. DSP: See Table 13-17.	**Instance**	DMA channel
Description	Channel progress counter		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPC_LSB															

Bits	Field Name	Description	Type	Reset
15:0	CPC_LSB	Lowest 16 bits of the address where the last data of a frame has been written	R	Undefined

Timers and Watchdogs

This chapter describes the timers and watchdogs subsystem for the LOCOSTO device.

Topic **Page**

14.1 Timers and Watchdogs Overview

The LOCOSTO chipset implements the following: CLKDRP13r

- Two 16-bit general-purpose timers
- One 16-bit watchdog, which can be configured as a general-purpose timer
- One 16-bit secure watchdog, which can be configured as a secure general-purpose timer or as a nonsecure general-purpose timer/watchdog

The timers are clocked from the 13-MHz source. The watchdogs are clocked from the 13-MHz source divided by 14 (that is, 928 KHz).

Note: By default, the secure watchdog is in the non-secure mode. To configure the secure watchdog in secure mode, see Chapter 22, *Security Features*.

14.1.1 Timers and Watchdogs Subsystem Main Features

The timers and watchdogs subsystem includes the following main features:

- Auto-reload timer functionality
- Configurable timer clock period
- On-the-fly read-and-write capabilities
- Switch in watchdog/timer modes (watchdog timer only)

14.1.2 Timers and Watchdogs Signals and I/O Descriptions

Figure 14-1 shows an integration overview of the LOCOSTO timers and watchdogs.

Figure 14-1. Timers and Watchdogs Overview

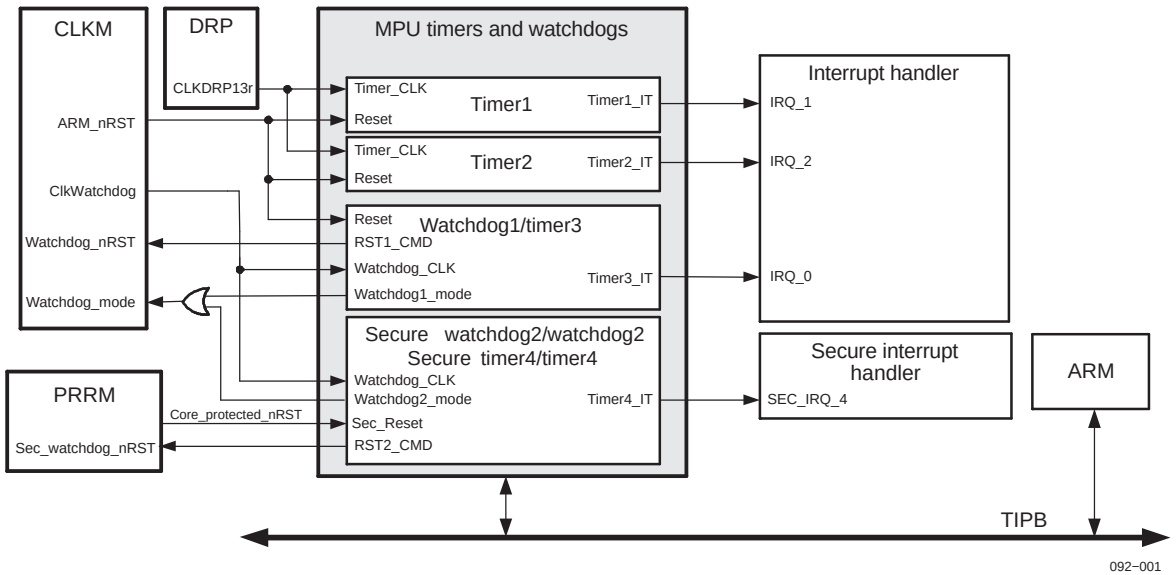


Table 14-1 describes the timers and watchdogs pins.

Table 14-1. I/O Description

Signal Name	I/O	Description	Reset Value
Timer1_IT	O	Interrupt signal from Timer1—active at low level	1
Timer2_IT	O	Interrupt signal from Timer2—active at low level	1

Table 14-1. I/O Description (continued)

Signal Name	I/O	Description	Reset Value
Timer3_IT	O	Interrupt signal from Watchdog1/Timer3—active at low level	1
Timer4_IT	O	Interrupt signal from Watchdog2/Timer4—active at low level	1
Timer_CLK	I	Timer functional clock	0
Watchdog_CLK	I	Watchdog functional clock	0
Reset	I	Reset the nonsecure timer and watchdog modules—active at low level	0
Sec_Reset	I	Reset the secure watchdog module—active at low level	0
RST1_CMD	O	Watchdog1 reset—active at low level—in watchdog mode	1
RST2_CMD	O	Watchdog2 reset—active at low level—in watchdog mode	1
Watchdog1_mode	O	Watchdog1 mode activated—active at high level	1
Watchdog2_mode	O	Watchdog2 mode activated—active at high level	1

14.1.3 Clocking, Reset and Power-Management Scheme

14.1.3.1 Clocks

Table 14-2 lists the clock domains of the timer and the watchdog modules.

Table 14-2. Clock Description

Type	Name	Source	Frequency	Description
Functional	Timer_CLK	ClkDRP13Clk (DRP2.0)	13 MHz	The 13-MHz clock comes directly from the DRP2.0 module, bypassing the CLKM module.
Functional	Watchdog_CLK	ClkWatchdog (CLKM)	928 kHz	Before being supplied to the watchdog, the 13-MHz clock is divided by 14 (that is, 928 kHz).
Interface	TIPB_CLK	TIPB	52 MHz	Clock fed by the TIPB

For more information, see Chapter 6, *Power, Reset, and Clock Management*.

14.1.3.2 Power Management and Power Domain

The timers belong to the VDD_CORE power domain. For more information, see Chapter 6, *Power, Reset, and Clock Management*.

14.1.3.3 Hardware and Software Resets

Table 14-3 lists the input resets. Table 14-4 lists the reset generated by the watchdog modules.

Table 14-3. Reset Domain

Type	Name	Source	Domain
Hardware	Reset	ARM_nRST (CLKM)	Timer1, Timer2, and Watchdog1
Hardware	Sec_Reset	Core_protected_nRST (PRRM)	Secure watchdog2

Table 14-4. Generated Resets

Type	Name	Source	Destination	Description
Reset	RST1_CMD	Watchdog1	Watchdog_nRST (CLKM)	Watchdog1 reset command
Reset	RST2_CMD	Watchdog2	Sec_watchdog_nRST (PRRM)	Watchdog2 reset command

For more information, see Chapter 6, *Power, Reset, and Clock Management*.

14.1.4 Hardware Requests

Table 14-5 lists the outputs and their destinations.

Table 14-5. Output Signals

Type	Name	Source	Destination	Description
IRQ	Timer1_IT	Timer1	IRQ_1 (interrupt handler)	Interrupt signal from Timer1
IRQ	Timer2_IT	Timer2	IRQ_2 (interrupt handler)	Interrupt signal from Timer2
IRQ	Timer3_IT	Watchdog1/ Timer3	IRQ_0 (interrupt handler)	Interrupt signal from Timer3
IRQ	Timer4_IT	Watchdog2/ Timer4	SEC_IRQ_4 (Secure interrupt handler)	Interrupt signal from Timer4
Signal	Watchdog1_mode	Watchdog1	Watchdog_mode (CLKM)	Active when Timer3 is configured as a watchdog timer
Signal	Watchdog2_mode	Watchdog2	Watchdog_mode (CLKM)	Active when Timer4 is configured as a watchdog timer

14.2 Functional Description

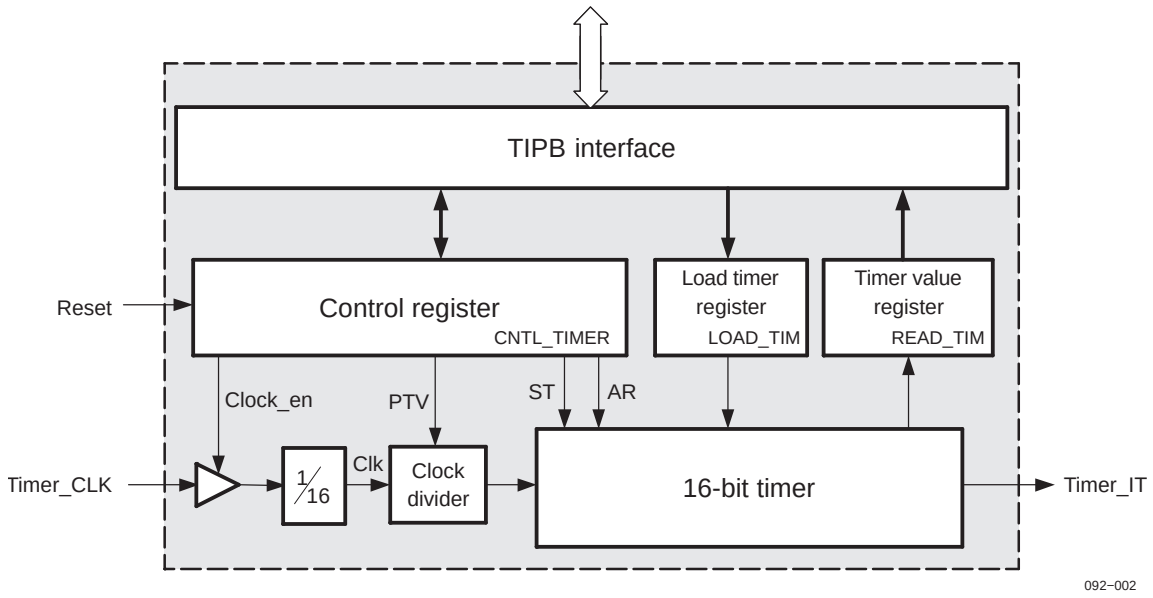
14.2.1 Timer

The following functional description pertains to Timer1 and Timer2.

14.2.1.1 Block Diagram

Figure 14-2 shows the internal block diagram of a timer.

Figure 14-2. Timer Block Diagram



14.2.1.2 Start and Stop

The timer is started by setting the CNTL_TIMER[0] ST bit and is stopped by resetting this bit. At start, the timer is loaded with the 16-bit value stored in the LOAD_TIM register. When the timer is stopped, its current value is frozen and can be read through the READ_TIM register.

CAUTION

After loading the timer value in the LOAD_TIM register, a new timer start requires a minimum delay of two Clk clock periods. The Clk clock is the Timer_CLK clock divided by 16.

14.2.1.3 Auto-Reload Failure

If the CNTL_TIMER[1] AR bit is set, the timer is loaded automatically with the value stored in the LOAD_TIM register when it passes 0. The timer then immediately starts to decrement again. Otherwise, if the AR bit is reset, the timer stops when it reaches 0.

CAUTION

To avoid hazardous effects, the AR bit must not be written while the timer is running.

14.2.1.4 Timer Delay

14.2.1.4.1 Timer Value

The timer delay is given by a timer value and the clock frequency. At start, the 16-bit timer is loaded with the value stored in the LOAD_TIM register. The timer 16-bit value can be read either on-the-fly or after the timer is stopped. Its current value is stored in the READ_TIM register.

CAUTION

To avoid hazardous effects, the LOAD_TIM register must not be updated while the timer is running.

14.2.1.4.2 Clock Frequency

The functional clock of the timer (TIMER_CLK) comes from the digital radio processor (DRP) module. The timer automatically divides this 13-MHz clock by 16 (that is, 812.5 kHz). A clock divider controlled by the CNTL_TIMER register performs another clock division.

The clock is enabled by the CLOCK_ENABLE bit of the CNTL_TIMER register.

The timer underflow delay is calculated as follows:

$$\text{Delay} = T_{Clk} \times (\text{LOAD\_TIM} + 1) \times 2^{PTV+1}$$

T_{Clk} is the period of the external clock divided by 16 (that is, 1.23 µs); LOAD_TIM corresponds to the 16-bit value stored in the LOAD_TIM register. The CNTL_TIMER[4:2] PTV field controls a clock divider. The minimum delay is then 2.46 µs, and the maximum delay is 20.63 s.

CAUTION

To avoid hazardous effects, the PTV field must not be written while the timer is running.

14.2.1.5 Interrupt

An interrupt (Timer_IT) is generated when the timer is running and reaches 0. The interrupt is a pulse-active at low level during one Timer_CLK clock period.

14.2.1.6 Additional Functionality for Debug Purposes

When set to 0, the CNTL_TIMER[6] FREE bit enables the counter content freeze feature. This feature freezes the count if the timer receives a suspend indication from the TI peripheral busTIPB. For example, a suspend indication can be received if execution is suspended for debug purposes.

When the FREE bit is set, the CNTL_TIMER[7] SOFT bit selects two timer behaviors:

- The SOFT bit is set to 0; if the timer receives a suspend indication, it freezes immediately, regardless of its state.
- The SOFT bit is set to 1; if the timer receives a suspend indication, it freezes after completing its current task.

  For example, if the auto-reload is enabled, the timer freezes after reaching 0.

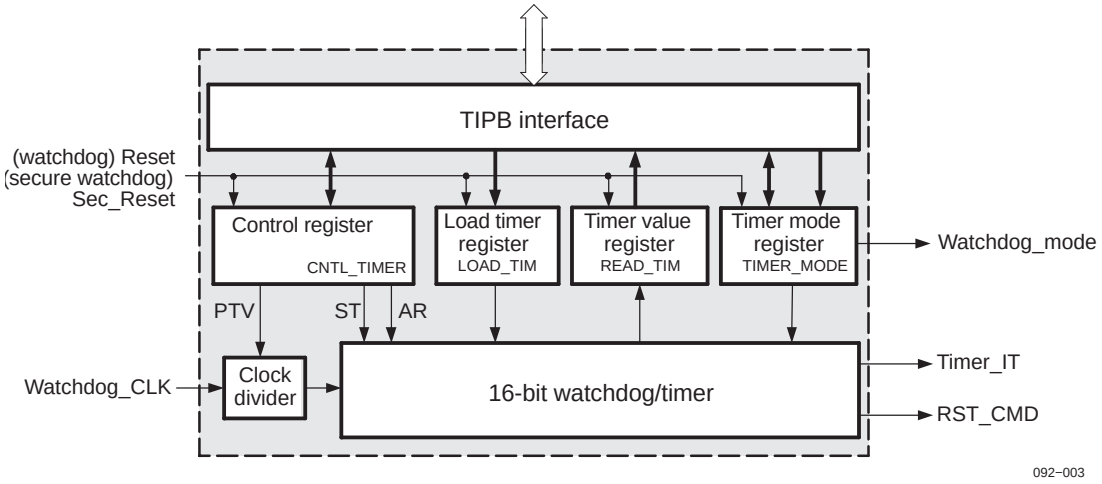
14.2.2 Watchdog Timers

The following functional description concerns Watchdog1/Timer3 and Watchdog2/Timer4.

14.2.2.1 Block Diagram

Figure 14-3 shows the internal block diagram of a secure/nonsecure watchdog.

Figure 14-3. Watchdog Block Diagram



14.2.2.2 Secure Watchdog

A secure timer is added in the secure solution and can be programmed as a secure watchdog timer when a protected space is defined in secure mode. The secure timer can be used as a second watchdog timer in normal mode.

For more information, see Chapter 22, *Security Features*.

14.2.2.3 Power-up

On power-up, both watchdog timers are enabled, the value of the pre-scalar field (CNTL_TIMER[11:9] PTV field) is fixed at 7, and the value loaded into the LOAD_TIM register is set to the maximum value (that is, 0xFFFF). This gives a time of 16,777,216 × T_{Timer_CLK} to switch to the timer mode or to write a new value (different from 0xFFFF) into LOAD_TIM register (T_{Timer_CLK} is the 13-MHz clock coming from the clock manager (CLKM) module, divided by 14). The reset signal then occurs approximately 18 s after power-up.

14.2.2.4 Timer/Watchdog Mode

The watchdog function can be disabled by software, if desired, and the watchdog can be used as a general-purpose timer.

To disable the watchdog function, the predefined sequence—0xF5 followed by 0xA0—must be written in the TIMER_MODE[7:0] WATCHDOG_DIS field. Receiving 0xF5 initializes a sequence decoder. Once in this state, if the next write is different from 0xA0, the watchdog timer resets the MPU system.

Note: Even if the timer is in the general-purpose timer mode, writing 0xF5 to the WATCHDOG_DIS field initializes the sequence, as it does in watchdog mode.

To return to the watchdog mode, the TIMER_MODE[15] WATCHDOG bit must be set. In this case, the LOAD_TIM register automatically is loaded with the value 0xFFFF as it is on power up.

Note: When switching from general-purpose mode to watchdog timer mode, the next write to the LOAD_TIM register must be delayed for three timer clock periods. This restriction does not exist when switching from watchdog mode to general-purpose timer mode.

14.2.2.5 Start and Stop

In watchdog mode:

The watchdog starts automatically; the ST bit has no functionality in this mode. At start, the watchdog value is the reset value of the LOAD_TIM register (that is, 0xFFFF). When the clock is running, the only way to stop the watchdog timer is to switch to timer mode by writing the defined sequence 0xF5–0xA0 into the TIMER_MODE[7:0] WATCHDOG_DIS field.

In timer mode:

The timer is started by setting the ST bit and is stopped by resetting this bit. At start, the timer is loaded with the 16-bit value stored in the LOAD_TIM register. When the timer is stopped, its current value is frozen and can be read through the READ_TIM register.

Note: The ST bit is a command, does not represent the timer state, and should not be interpreted as a status bit.

CAUTION

After loading the timer value in the LOAD_TIM register, a minimum delay of two clock periods is required before doing a new timer start; the clock is the Watchdog_CLK clock divided by 14.

14.2.2.6 Auto-Reload Feature

In watchdog mode:

The auto-reload functionality is not effective; the CNTL_TIMER[8] AR bit is not used.

In timer mode:

If the CNTL_TIMER[8] AR bit is set, the timer automatically is loaded with the value stored in the LOAD_TIM register when it passes 0. The timer then immediately starts to decrement again. Otherwise, if the AR bit is reset, the timer stops when it reaches 0.

CAUTION

To avoid hazardous effects, the AR bit must not be written while the timer is running.

14.2.2.7 Timer Delay

14.2.2.7.1 Timer Value

The timer delay is given by a timer value and the clock frequency.

In watchdog mode:

A program must write periodically to the LOAD_TIM register before the counter underflows. A write is considered valid only if the new loaded value is different from the previous one.

In timer mode:

At start, the 16-bit timer is loaded with the value stored in the LOAD_TIM register.

CAUTION

To avoid hazardous effects, the LOAD_TIM register must not be updated while the timer is running.

In both modes, the timer 16-bit value can be read either on-the-fly or after the timer is stopped. Its current value is stored in the READ_TIM register.

14.2.2.8 Clock Frequency

The functional clock of the timer (TIMER_CLK) comes from the CLKM module. Before being supplied to the timer, the 13-MHz clock is divided by 14. Another clock division is performed by a clock divider controlled by the CNTL_TIMER register.

In watchdog mode:

The timer underflow delay is calculated as follows:

$$\text{Delay} = T_{\text{Timer\_CLK}} \times (\text{LOAD\_TIM} + 1) \times 2^{8}$$

$T_{\text{Timer\_CLK}}$ is the period of the external clock divided by 16 (that is, 1.07 µs); LOAD_TIM corresponds to the 16-bit value stored in the LOAD_TIM register. The minimum delay is then 275.69 µs, and the maximum delay is 18 s.

Note: The CNTL_TIMER[11:9] PTV field is always set to 111 and cannot be changed.

In timer mode:

The timer underflow period is calculated as follows:

$$\text{Delay} = T_{\text{Timer\_CLK}} \times (\text{LOAD\_TIM} + 1) \times 2^{\text{PTV}+1}$$

$T_{\text{Timer\_CLK}}$ is the period of the external clock (that is, 1.07 µs); LOAD_TIM corresponds to the 16-bit value stored in the LOAD_TIM register; PTV is the CNTL_TIMER[11:9] PTV field, which controls a clock divider. The minimum delay is then 2.15 µs, and the maximum delay is 18 s.

CAUTION

To avoid hazardous effects, the PTV field must not be written while the timer is running.

14.2.2.9 Interrupt and Reset

In watchdog mode:

An underflow of the nonsecure watchdog generates a reset (RST1_CMD) to the MPU core through the CLKM module; the secure watchdog generates a reset (RST2_CMD) core through the PRRM module.

Both timer and watchdog modes:

An interrupt (Timer_IT) is generated when the timer is running and reaches 0. The interrupt is a pulse-active at low level during one Watchdog_CLK clock period.

Note: Even if the watchdog is used in nonsecure mode, the interrupt is sent to the secure handler.

14.3 Programming Model

14.3.1 Setup for Typical Configuration(s)

14.3.1.1 Power Up

Be aware that Watchdog1 and Watchdog2 are running at power-up and will reset the MPU subsystem after an 18-s delay if no action is performed.

14.3.1.2 Timer Delays

Table 14-6 lists the timer delay ranges, which are different for each timer.

Table 14-6. Timer Delay Ranges

Timer	Minimum	Maximum
Timer1 and Timer2	2.46 µs	20.63 s
Watchdog1 and Watchdog2	275.69 µs	18 s
Timer3 and Timer4	2.15 µs	18 s

14.3.2 Operational Mode

14.3.2.1 Timer (Timer1 and Timer2)

To configure and start a timer, perform the following steps:

1. Set the CNTL_TIMER[4:2] PVT field and write the start value of the timer in the LOAD_TIM register based on the desired timer delay (the formula is provided in Section 14.2.1.4, *Timer Delay*).
2. If desired, set the CNTL_TIMER[1] AR bit to activate the auto-reload feature.
3. Set the CNTL_TIMER[6] FREE bit and the CNTL_TIMER[7] SOFT bit to the desired debug options.
4. To enable the timer clock, set the CNTL_TIMER[5] Clock_ENABLE bit to 1.
5. To start the timer, set the CNTL_TIMER[0] ST bit to 1.
6. To stop the timer, reset the ST bit at any time.

14.3.2.2 Watchdog (Watchdog1/Timer3 and Watchdog2/Timer4)

Watchdog Mode

To start the watchdog, perform the following steps:

1. Set the FREE bit of the CONTL_TIMER register to the desired debug option.
2. (If necessary) To set the watchdog mode, set the TIMER_MODE[15] WATCHDOG bit to 1.
3. To avoid triggering a reset, write a different value from the previous one in the LOAD_TIM register. (The delay is calculated from the formula provided in Section 14.2.2.8, *Clock Frequency*.)
4. To stop the watchdog, switch to timer mode: write the predefined data (0xF5 followed by 0xA0) into the TIMER_MODE[7:0] WATCHDOG_DIS field.

Timer Mode

To configure and start a timer, perform the following steps:

1. (If necessary) To switch to timer mode, write the predefined data (0xF5 followed by 0xA0) to the TIMER_MODE[7:0] WATCHDOG_DIS field.
2. Set the CNTL-TIMER[11:9] PVT field and write the start value of the timer in the LOAD_TIM register to the desired timer delay. (The formula is provided in Section 14.2.2.8, *Clock Frequency*.)
3. To activate, or not, the auto-reload feature, set the CNTL_TIMER[8] AR bit.
4. Set the CNTL_TIMER[1] FREE bit to the desired debug option.
5. To start the timer, set the CNTL_TIMER[7] ST bit to 1.
6. To stop the timer, set the CNTL_TIMER[7] ST bit to 0.

14.4 Register Manual

Table 14-7 lists the base address space, block size, and access for the timers and watchdogs instances.

Table 14-7. Instance Summary

Module Name	Base Address	Size	Access
Timer1	0xFFFE 3800	2K bytes	MPU
Timer2	0xFFFE 6800	2K bytes	MPU
Watchdog1/Timer3	0xFFFF F800	128 bytes	MPU
Secure watchdog2/Timer4	0xFFFF F880	128 bytes	MPU

14.4.1 Module Register Mapping Summary

Table 14-8 through Table 14-11 list the registers for the timers and watchdogs.

Table 14-8. Timer1 Register Offset Address(1)

Register Name	Type	Register Width (Bits)	Address offset	Physical Address
CNTL_TIMER	R/W	16	0x00	0xFFFE 3800
LOAD_TIM	W	16	0x02	0xFFFE 3802
READ_TIM	R	16	0x02	0xFFFE 3802

(1) The LOAD_TIM register and the READ_TIM register share the same offset value.

Table 14-9. Timer2 Register Offset Address(1)

Register Name	Type	Register Width (Bits)	Address offset	Physical Address
CNTL_TIMER	R/W	16	0x00	0xFFFE 6800
LOAD_TIM	W	16	0x02	0xFFFE 6802
READ_TIM	R	16	0x02	0xFFFE 6802

(1) The LOAD_TIM register and the READ_TIM register share the same offset value.

Table 14-10. Watchdog1/Timer3 Register Offset Address(1)

Register Name	Type	Register Width (Bits)	Address offset	Physical Address
CNTL_TIMER	R/W	16	0x00	0xFFFF F800
LOAD_TIM	W	16	0x02	0xFFFF F802
READ_TIM	R	16	0x02	0xFFFF F802
TIMER_MODE	R/W	16	0x04	0xFFFF F804

(1) The LOAD_TIM register and the READ_TIM register share the same offset value.

Table 14-11. Secure Watchdog2/Timer4 Register Offset Address(1)

Register Name	Type	Register Width (Bits)	Address offset	Physical Address
CNTL_TIMER	R/W	16	0x00	0xFFFF F880
LOAD_TIM	W	16	0x02	0xFFFF F882
READ_TIM	R	16	0x02	0xFFFF F882
TIMER_MODE	R/W	16	0x04	0xFFFF F884

(1) The LOAD_TIM register and the READ_TIM register share the same offset value.

14.4.2 Register Description

14.4.2.1 Timer

Table 14-12 through Table 14-14 describe the timer register bits.

Table 14-12. CNTL_TIMER (Timer)

Address Offset	0x00		
Physical address	Timer1: 0xFFFE 3800	**Instance**	Timer
	Timer2: 0xFFFE 6800		
Description	Timer configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SOFT	FREE	CLOCK_ENABLE	PTV			AR	ST

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	RW	undefined
7	SOFT	This bit is used in conjunction with the FREE bit to determine the state of the peripheral when a breakpoint is encountered. This bit is used in the emulation mode. 0: The peripheral halts immediately, either retaining or discarding the current state. 1: The peripheral stops after completion of the current task.	RW	0
6	FREE	This bit is used in conjunction with the SOFT bit to determine the state of the peripheral when a breakpoint is encountered. This bit is used in the emulation mode. 0: The emulation mode is selected. 1: The peripheral clock runs free, regardless of the SOFT bit.	RW	0
5	CLOCK_ENABLE	Enables the timer clock 0: Disable 1: Enable	RW	0
4:2	PTV	Pre-scale timer clock 000: Divide timer clock by 2 001: Divide timer clock by 4 010: Divide timer clock by 8 011: Divide timer clock by 16 100: Divide timer clock by 32 101: Divide timer clock by 64 110: Divide timer clock by 128 111: Divide timer clock by 256	RW	000
1	AR	0: One-shot timer 1: Auto_UnicodeEncodeError_reload mode	RW	0
0	ST	0: Stop timer value decrement 1: Start timer value decrement	RW	0

Table 14-13. LOAD_TIM (Timer)

Address Offset	0x02		
Physical address	Timer1: 0xFFFE 3802	**Instance**	Timer
	Timer2: 0xFFFE 6802		
Description	Load timer value		

Table 14-13. LOAD_TIM (Timer) (continued)

Type	W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD_TIMER_VALUE															

Bits	Field Name	Description	Type	Reset
15:0	LOAD_TIMER_VALUE	This value is loaded when the timer passes 0, or when it starts.	W	Undefined

CAUTION

After loading the timer value in the LOAD_TIM register, a minimum delay of two clock periods is required before doing a new timer start (see Section 14.2.1.2, *Start and Stop*)

Table 14-14. READ_TIM (Timer)

Address Offset	0x02		
Physical address	Timer1: 0xFFFE 3802	Instance	Timer
	Timer2: 0xFFFE 6802		
Description	Read timer value		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
READ_TIMER_VALUE															

Bits	Field Name	Description	Type	Reset
15:0	READ_TIMER_VALUE	Current value of timer	R	Undefined

14.4.2.2 Watchdog

Table 14-15 through Table 14-18 describe the watchdog register bits.

Table 14-15. CNTL_TIMER (Watchdog)

Address Offset	0x00		
Physical address	Watchdog1/Timer3: 0xFFFF F800	Instance	Watchdog timer
	Secure watchdog2/Timer4: 0xFFFF F880		
Description	Watchdog timer configuration register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				PTV			AR	ST	Reserved					FREE	Reserved

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Reserved	R	Undefined
11:9	PTV	Pre-scale timer clock 000: Divide timer clock by 2 001: Divide timer clock by 4 010: Divide timer clock by 8 011: Divide timer clock by 16 100: Divide timer clock by 32 101: Divide timer clock by 64 110: Divide timer clock by 128 111: Divide timer clock by 256	RW	000
8	AR	0: One-shot timer 1: Auto_UnicodeEncodeError_reload mode	RW	0
7	ST	0: Stop timer value decrement 1: Start timer value decrement	RW	0
6:2	Reserved	Reserved	R	Undefined
1	FREE	0: Timer can be frozen while in debug mode. 1: Timer runs free.	RW	0
0	Reserved	Reserved	R	Undefined

Table 14-16. LOAD_TIM (Watchdog)

Address Offset	0x02		
Physical address	Watchdog1/Timer3: 0xFFFF F802 Secure watchdog2/Timer4: 0xFFFF F882	**Instance**	Watchdog
Description	Load Timer Value		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD_TIMER_VALUE															

Bits	Field Name	Description	Type	Reset
15:0	LOAD_TIMER_VALUE	**Timer:** This value is loaded when the timer passes 0, or when it starts. Watchdog: The watchdog value is reset at each start, but can be reloaded while running.	W	FFFF

CAUTION

After loading the timer value in the LOAD_TIM register, a new timer start requires a minimum delay of two clock periods (see Section 14.2.2.5, *Start and Stop*).

Table 14-17. READ_TIM (Watchdog)

Address Offset	0x02		
Physical address	Watchdog1/Timer3: 0xFFFF F802 Secure watchdog2/Timer4: 0xFFFF F882	**Instance**	Watchdog
Description	Read timer value		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
READ_TIMER_VALUE															

Bits	Field Name	Description	Type	Reset
15:0	READ_TIMER_VALUE	Current value of timer	R	FFFF

Table 14-18. TIMER_MODE (Watchdog)

Address Offset	0x04		
Physical address	Watchdog: 0xFFFF F804 Secure watchdog: 0xFFFF F884	**Instance**	Watchdog
Description	Read timer value		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WATCHDOG	Reserved							WATCHDOG_DIS							

Bits	Field Name	Description	Type	Reset
15	WATCHDOG	Write access: Writing a 0 in this bit has no effect. Writing 1 switches back to watchdog mode. Read access: Status of timer mode: 0: The timer is used as a general-purpose counter. 1: The timer is used as a watchdog timer.	RW	1
14:8	Reserved	Reserved	R	Undefined
7:0	WATCHDOG_DIS	Switch to timer mode 1. Writing a predefined sequence (0xF5 followed by 0xA0) in this field disables watchdog functionality. **Note:** After having received 0xF5, if the second write access is different from 0xA0, the ARM core is reset.	W	Undefined

GPRS-GSM Encryption

This chapter introduces GPRS-GSM encryption for the LOCOSTO device.

Topic **Page**

15.1 GPRS-GSM Encryption Overview

In the LOCOSTO device, the GPRS-GSM Encryption subsystem comprises two modules:

- The GEA3 module for GPRS and EGPRS encryption
- The Cipher_A5 module for GSM and ECSD encryption

The GEA3 module provides data encryption and the Cipher_A5 module offers voice security.

This chapter discusses the use of the GEA3 and the Cipher_A5 modules from the software point of view. No algorithm details are given.

Note: The GEA3 module is not available on the LOCOSTO Lite device.

15.2 GPRS Encryption Algorithm (GEA3) Module

In GPRS mode, data confidentiality is performed by a ciphering function called GEA (GPRS Encryption Algorithm). The ciphering is executed within the LLC upper layer. According to the option negotiated with the network, GEA mode 1 (GEA/1), mode 2 (GEA/2), or mode 3 (GEA/3) may be selected.

The purpose of the LLC is to convey information between the Mobile Station (MS) and the Serving GPRS Support Node (SGSN). The procedures used are modeled on the concept of high-level data-link control (HDLC). The LLC supports both acknowledge and unacknowledged modes and implements a Frame Check Sequence (FCS) according to the mode selected.

The GEA3 module of the LOCOSTO device supports the computation of the FCS according to the LLC and the ciphering/deciphering modes 1, 2, and 3.

The GEA3 module is mapped to the MPU Private TIPB. It is accessible only by the MPU.

15.2.1 GEA3 Integration

Figure 15-1 shows GEA3 module integration in the LOCOSTO device.

Figure 15-1. GEA3 Module Integration

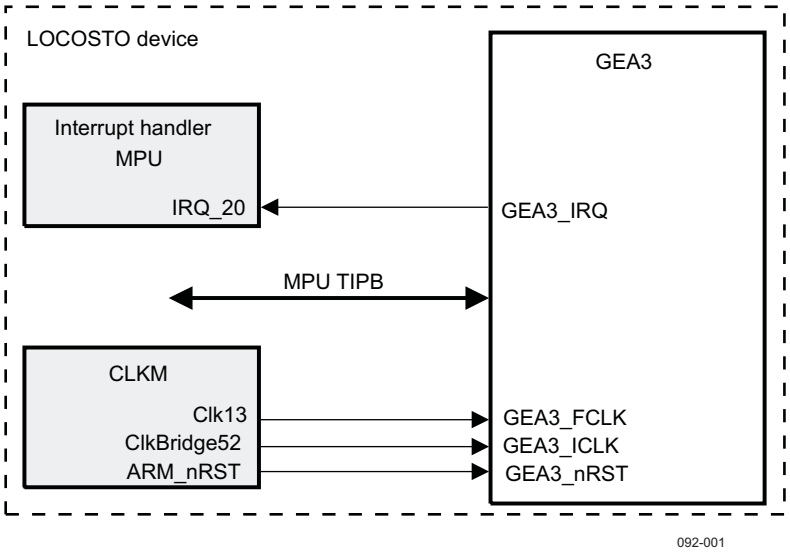


Table 15-1 summarizes the clock and reset signals of the GEA3 module.

Table 15-1. GEA3 Clocking and Resets

Module	Functional clock	Interface clock	Resets
GEA3	Clk13	ClkBridge52	**Hardware**: ARM_nRST **Software**: GEA3.GEA_CNTL_REG[1] NRESET_DL bit GEA3.GEA_CNTL_REG[0] NRESET_UL bit

15.2.1.1 Clocks

The GEA3 module has two clock domains: functional clock domain and interface clock domain.

15.2.1.1.1 Functional Clock Domain

GEA3_FCLK is a 13-MHz clock provided by the CLKM through the Clk13 clock signal. It is used inside the GEA3 module to generate the internal clock. The internal clock can be switched off by using the GEA3.GEA_CNTL_REG[3] CLOCK_EN bit . The clock is on when the CLOCK_EN bit field is set to 1.

15.2.1.1.2 Interface Clock Domain

GEA3_ICLK is the interface clock with the MPU Private TIPB. It is provided by the ClkBridge52 clock and is used to access the GEA3 module's registers.

For more information about these clocks, refer to Chapter 6, *Power, Reset, and Clock Management*.

15.2.1.2 Resets

15.2.1.2.1 Hardware Reset

A GEA3 module hardware reset is performed by the ARM_nRST reset signal.

For details about this signal, refer to Chapter 6, *Power, Reset, and Clock Management*.

15.2.1.2.2 Software Reset

The GEA3 module can have two internal software resets:

- GEA3.GEA_CNTL_REG[1] NRESET_DL bit: When this bit field is set to 0, it resets the downlink module (deciphering + FCS). The current processing is cancelled and the internal variables are reset. This bit is cleared by internal logic.
- GEA3.GEA_CNTL_REG[0] NRESET_UL bit: When this bit field is set to 0, it resets the uplink module (ciphering + FCS). The current processing is cancelled and the internal variables are reset. This bit is cleared by internal logic.

For more information, see Section 15.4, *GEA3 Module Registers*.

15.2.1.3 Interrupt Requests

The GEA3 module handles one interrupt line (GEA3_IRQ), which is linked to the IRQ_20 of the MPU interrupt handler.

The interrupt is controlled by software through the GEA3.GEA_CNTL_REG[5] IT_UL bit:

- When IT_EN is set to 0 (reset value), the interrupt is disabled.
- When IT_EN is set to 1, an interrupt is generated when processing is completed.

When an interrupt occurs (IT_EN is set to 1), software must read the GEA3.GEA_STATUS_REG[0] LLC_IT bit. The LLC_IT bit is automatically cleared after read; the interrupt is considered to be acknowledged.

15.2.2 GEA3 Functional Description

15.2.2.1 Link Layer Control (LLC) Overview

Link layer control (LLC) is treated as a sub-layer of layer 2 in the ISO 7-layer model. It conveys information between layer-3 entities in the Mobile Station (MS) and the Serving GPRS Support Node (SGSN).

The frame formats defined for LLC are based on those defined for link-access procedure (LAPD) on the D-channel, and on the radio-link protocol (RLP). The LLC procedures are modeled on the concepts of high-level data-link control (HDLC).

LLC supports variable-length information frames provided by layer-3 protocols.

The logical-link control layer service access points (SAPs) are the points at which the LLC layer provides services to the layer-3 protocols. In addition to the subnetwork-dependent convergence (SNDC) protocol, LLC provides service to the GPRS mobility management (GMM) protocol and to the short message service (SMS) protocol.

LLC supports both unacknowledged and acknowledged data transfers

15.2.2.2 Unacknowledged Operation

With unacknowledged operation, layer-3 information is transmitted in numbered Unconfirmed Information (UI) frames. These UI frames are not acknowledged at the LLC layer. Neither error recovery nor reordering mechanisms are defined, but transmission and format errors are detected. Duplicate UI frames are discarded.

There are two modes of unacknowledged operation:

- Protected mode, in which the frame check sequence (FCS) field protects the frame header and information field
- Unprotected mode, in which the FCS field protects the frame header and only the first N202 octets of the information field

15.2.2.3 Acknowledged Operation

With acknowledged operation, layer-3 information is transmitted sequentially in numbered Information (I) frames. These I-frames are acknowledged at the LLC layer. Error recovery and reordering procedures based on retransmission of unacknowledged I-frames are specified. Several I-frames may be unacknowledged at the same time. If errors occur that cannot be corrected by the logical-link control layer, a report is sent to GPRS mobility management.

15.2.2.4 Frame Format

Each LLC frame consists of a header, an information field, and a FCS.

- The header has a minimum size of 2 bytes and a maximum of 37 bytes. It is split into an address field (1 byte) and a control field (variable length, maximum 36 bytes). The control field typically consists of between one and three octets. The Selective ACKnowledgement (SACK) supervisor frame also includes a variable-length bitmap field of up to 32 octets.
- The information field of a frame, when present, follows the control field. The minimum value of N201 is 140 octets, and the maximum is 1520 octets.
- The FCS field has a fixed size of 3 bytes.

The MPU uses the frame header information to identify the type of the frame and the processing to be applied to it. Thus, the MPU is able to detect the FCS and ciphering configuration and how to split the header and information fields.

15.2.2.5 FCS

The FCS is a 24-bit cyclic redundancy check code that is used to detect bit errors in frame headers and information fields.

The FCS field contains the value of a CRC calculation performed over the entire contents of the header and information field, except for UI frames transmitted in unprotected mode. With these frames, the FCS field contains the value of a CRC calculation performed over the frame header and just the first N202 of the information field. The information over which the CRC is calculated is referred to as the “dividend.” The first bit of the dividend is the highest-order term in the calculation.

The CRC calculation is performed before ciphering at the transmitting side, and after deciphering at the receiving side.

The result of the CRC calculation is placed in the FCS field, with the highest order terms transmitted first.

15.2.2.6 Ciphering

LLC provides user data confidentiality by applying a ciphering function to the information field and the FCS, but not to the frame header. The GEA3 module supports all three ciphering algorithms currently defined by the ETSI.

The three algorithms require the following input variables:

- Kc
  - The ciphering key (Kc) is generated in the GPRS authentication and key management procedure. The length of the key is 64 bits (54 bits filled with zeros) for the GEA1 and GEA2 algorithms, and up to 128 bits for the GEA3 algorithm.
- INPUT
  - This is the LLC frame-dependent input parameter (32 bits) for the ciphering algorithm. It is also called the message key.
- DIRECTION
  - This defines the direction (1 bit) of the data transmission (uplink/downlink).
    Set to 0 if the direction of LLC frame transmission is from the mobile station (MS) to the serving GPRS support node (SGSN) (uplink).
    Set to 1 if the direction of LLC frame transmission is from the SGSN to the MS (downlink).

The Kc key is received once by the mobile station from GPRS mobility management (GMM) and is valid for all communication, whereas the ciphering input is regularly computed by the mobile station.

In the sender entity, the output string is bit-wise XORed with the plain text and the result is sent over the radio interface. In the receiving entity, the output string is bit-wise XORed with ciphered text and the original plain text is obtained. The ciphering module is in charge of applying this bit-wise XOR on the data. The three GPRS encryption algorithms use the same input variables; only their internal processing differs.

15.2.2.7 UI Frames

There are two modes for the FCS computation of the UI frames.

- Mode 1 is the classical mode whereby FCS is applied on the header plus all the information bits (protected mode).
- Mode 2 consists of applying the FCS only on the header and the N202 information bits (unprotected mode). If the length of the information field is less than N202 octets, the FCS covers the complete information field. This solution protects the LLC and the SNDCP headers, but not the information bits.
  To improve data transfers when non-protected mode and no ciphering are selected (second mode), the MPU has two possibilities:
  - The MPU writes the LLC-PDU header and the subsequent N202 information bytes into the input buffer. The LLC-PDU size given to the module is equal to LLC-PDU header + N202 size. The module computes the FCS only on these bytes and returns its result after the last N202 byte. This method is optimal for CPU load used for data transfer.
  - The MPU writes the LLC-PDU header and the full LLC-PDU information field to the input buffer. The LLC-PDU size given to the module is equal to LLC-PDU header + LLC-PDU information field.

However, the MPU specifies to the module that non-protected mode is used. Therefore, the module computes its FCS only on LLC-PDU header + N202 bytes. This method is optimal if the MPU uses the same data flow for both protected and non-protected modes.

In both cases, the same buffer formats are used and first byte shifting is still applied if specified.

15.2.2.8 Memory Management

All external accesses to the memory inside the GEA3 module are performed through the TIPB bus. From outside the module the memory is seen as a register; the address increment is managed by the GEA3 module.

The memory is managed using three address pointers. Each time a word is written to the data register, the write pointer is incremented. Once all the words are written and the START bit in the GEA3.GEA_CNTL_REG[2] register is activated, the data are processed by the GEA3 module and the processed pointer is incremented. When all the data are processed, the WORKING bit in the GEA3.GEA_STATUS_REG[0] register goes down and a maskable interrupt is sent (GEA3_IRQ). The data can then be read by the MPU and the read pointer is incremented each time a word is read. It is not mandatory to read the data. The MPU can start writing again and the read pointer takes the value of the processed value.

These pointers are directly managed by the GEA3 module.

8- and 16-bit accesses can be made to the memory using two different registers: GEA3.GEA_DATA16_REG and GEA3.GEA_DATA8_REG.

Up to 1560 bytes (37 bytes [MAX header] + 1520 bytes [MAX Info. Field] + 3 bytes [FCS]) can be written to memory. Therefore, a whole frame can be processed in one go.

15.2.3 GEA3 Programming Guide

Program the GEA3 module using the MPU in the following order:

15.2.3.1 Uplink Mode

Step 1. Set the GEA3.GEA_CNTL_REG[3] CLOCK_EN bit to 1.
→ Clock is ON.

Step 2. Set the GEA3.GEA_CNTL_REG[0] NRESET_UL bit to 0, then wait for it to return to 1 (the clock must not be cut during this reset).
→ Uplink GEA3 registers are reset.

Step 3. Fill the configuration registers:
GEA3.GEA_CONF_UL_REG1-3
(Uplink configuration + PDU size + header size,…)

Step 4. Fill the ciphering registers:
a. GEA3.GEA_KC_REG1-4

Note: It is not necessary to set the ciphering registers every frame; it can be done once at the beginning.

b. GEA3.GEA_KC_REG5-8
→ if the GEA3 algorithm has been chosen (long key: 128 bits)
c. GEA3.GEA_CONF_UL_REG4-5 (input/message key)

Step 5. Set the GEA3.GEA_SWITCH_REG[0] SWITCH_CLOCK bit to 1 (to write inside the GEA3 memory).

Step 6. Fill GEA3.GEA_DATAx_REG register n times with the data to be processed (if 6 words, n = 6).

Note: Accesses can be made in 8-bit (through GEA3.GEA_DATA8_REG) or 16-bit (through GEA3.GEA_DATA16_REG) word format.

Step 7. Set the GEA3.GEA_SWITCH_REG[0] SWITCH_CLOCK bit to 0 (end the write inside the GEA3 memory).

Step 8. Set in one shot: START bit to 1, UL_DL to 0, and IT_EN to 1 in the GEA3.GEA_CNTL_REG register (GEA3.GEA_CNTL_REG[2], GEA3.GEA_CNTL_REG[4], and GEA3.GEA_CNTL_REG[5], respectively).

Step 9. (Optional) To verify that the module is working, check the value of the GEA3.GEA_STATUS_REG[0] WORKING bit. If the value is 1, everything is fine; if the value is 0, check all previous steps.

Step 10. Wait for the GEA3 interrupt (GEA3_IRQ).

Step 11.

a. (Exclusive of 11b) For a driver using interrupts (for instance, when IT_EN is set): when an interrupt occurs, read the GEA3.GEA_STATUS_IR_REG[0] LLC_IT bit to clear the GEA3_IRQ interrupt.

b. (Exclusive of 11a) For a polling type of driver: poll the GEA3.GEA_STATUS_REG[0] WORKING bit until it equals 0.

Step 12. Set the GEA3.GEA_SWITCH_REG[0] SWITCH_CLOCK bit to 1 (to read inside the GEA3 memory).

Step 13. If yes, read n times the processed data through GEA3.GEA_DATAx_REG.

(GEA3.GEA_DATA16_REG for 16-bit access, or GEA3.GEA_DATA8_REG for 8-bit access).

Step 14. Set the GEA3.GEA_SWITCH_REG[0] SWITCH_CLOCK bit to 0 (end the read inside GEA3 memory).

Step 15. If this is not the last part of the LLC-PDU (not one shot, n frames), go to step 5.

Step 16. If this is the last part of the LLC-PDU and the GEA3.GEA_CONF_UL_REG1[5] F bit enabled, read the GEA3.GEA_FCS_UL_REG1-2 registers that contain the FCS result.

The clock can be disabled now; however, if another frame needs to be processed, the module is ready to restart from step 2.

15.2.3.2 Downlink Mode

Step 1. Set the GEA3.GEA_CNTL_REG[3] CLOCK_EN bit to 1.

→ Clock is ON

Step 2. Set the GEA3.GEA_CNTL_REG[1] NRESET_DL bit to 0, then wait for it to return to 1.

(the clock must not be cut during this reset).

→ Downlink GEA3 registers are reset.

Step 3. Fill the configuration registers:

GEA3.GEA_CONF_DL_REG1-3

(Uplink configuration + PDU size + Header size,…)

Step 4. Fill the ciphering registers:

a. GEA3.GEA_KC_REG1-4

Note: It is not necessary to set the ciphering registers every frame; it can be done once at the beginning.

b. GEA3.GEA_KC_REG5-8

→ if the GEA3 algorithm has been chosen (long key: 128 bits)

c. GEA3.GEA_CONF_DL_REG4-5 (input/message key)

Step 5. Set the GEA3.GEA_SWITCH_REG[0] SWITCH_CLOCK bit to 1 (to write inside the GEA3

memory)

Step 6. Fill GEA3.GEA_DATAx_REG n times with the data to be processed (if 6 words, n = 6)

Note: Accesses can be made in 8-bit (through GEA3.GEA_DATA8_REG) or 16-bit (through GEA3.GEA_DATA16_REG) word format.

Step 7. Set the GEA3.GEA_SWITCH_REG[0] SWITCH_CLOCK bit to 0 (end the write inside the GEA3 memory)

Step 8. If this is the last part of the LLC_PDU, GEA3.GEA_CONF_DL_REG2 must have been filled with the downlink PDU size.

Step 9. Set in one shot: START bit to 1, UL_DL to 1, and IT_EN to 1 in the GEA3.GEA_CNTL_REG register (GEA3.GEA_CNTL_REG[2], GEA3.GEA_CNTL_REG[4], and GEA3.GEA_CNTL_REG[5], respectively).

Step 10. (Optional) To verify that the module is working, check the value of the GEA3.GEA_STATUS_REG[0] WORKING bit. If the value is 1, everything is fine; if the value is 0, check all previous steps.

Step 11. Wait for the GEA3 interrupt (GEA3_IRQ)

Step 12.

a. (Exclusive of 12b) For a driver using interrupts (for instance, when IT_EN is set): when an interrupt occurs, read the GEA3.GEA_STATUS_IR_REG[0] LLC_IT bit to clear the GEA3_IRQ interrupt.

b. (Exclusive of 12a) For a polling type of driver: poll the GEA3.GEA_STATUS_REG[0] WORKING bit until it equals 0.

Step 13. Set the GEA3.GEA_STATUS_REG[0] SWITCH_CLOCK bit to 1 (to read inside the GEA3 memory).

Step 14. If yes, read n times the processed data through GEA3.GEA_DATAx_REG.

(GEA3.GEA_DATA16_REG for 16-bit access, or GEA3.GEA_DATA8_REG for 8-bit access).

Step 15. Set the GEA3.GEA_SWITCH_REG[0] SWITCH_CLOCK bit to 0 (end the read inside GEA3 memory).

Step 16. If this is not the last part of the LLC-PDU, go to step 5.

Step 17. If this is the last part of the LLC-PDU and the GEA3.GEA_CONF_UL_REG1[5] F bit is enabled, read the GEA3.GEA_FCS_DL_REG1-2 registers that contain the FCS result and check the GEA3.GEA_STATUS_REG[1] FCS_STATUS bit (good/bad).

The clock can be disabled now, but if another frame needs to be processed, the module is ready to restart from step 2.

15.3 Cipher_A5 Module

The Cipher_A5 module of the LOCOSTO device implements the A5/1, A5/2 (GSM) and the A5/3 (GSM/EDGE) algorithms. These algorithms realize the protection of both user data and signaling information elements at the physical layer on the dedicated channel. A5/3 algorithm implements additional registers to configure long key (up to 128 bits) and to generate long encipher/decipher data.

The Cipher_A5 module is mapped to the DSP Private TIPB. It is accessible only by the DSP.

15.3.1 Cipher_A5 Module Integration

Figure 15-2 highlights the Cipher_A5 module integration in the LOCOSTO device.

Figure 15-2. Cipher_A5 Module Integration

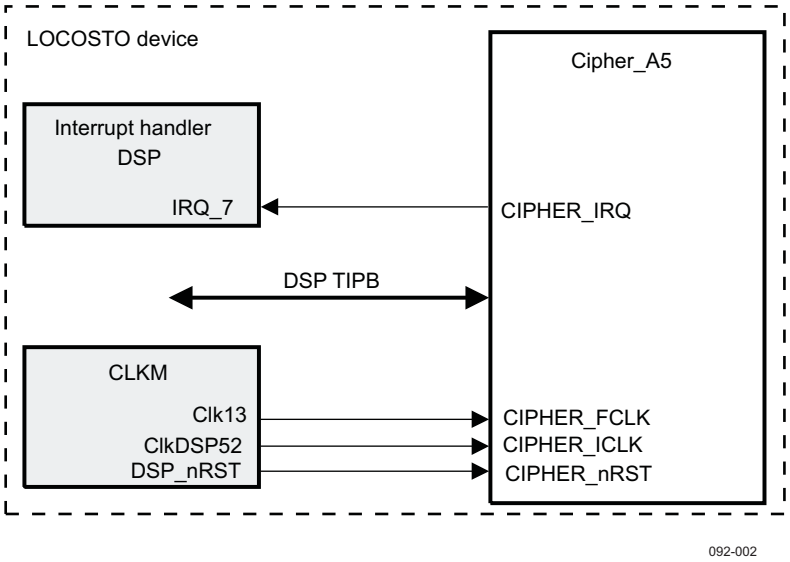


Table 15-2 summarizes the clocks and reset signals of the Cipher_A5 module.

Table 15-2. Cipher_A5 Clocking and Resets

Module	Functional clock	Interface clock	Resets
Cipher_A5	Clk13	ClkDSP52	**Hardware:** DSP_nRST **Software:** CIPHER.CNTL_REG[1] RESET_SW bit

15.3.1.1 Clocks

The Cipher_A5 module has two clock domains: functional clock domain and interface clock domain.

15.3.1.1.1 Functional Clock Domain

CIPHER_FCLK is a 13-MHz clock provided by the CLKM (through the Clk13 clock signal). It is used inside the Cipher_A5 module to generate the internal clock. The internal clock can be switched off by the CIPHER.CNTL_REG[4] CLK_EN bit. The clock is on when the CLK_EN bit field is set to 1.

15.3.1.1.2 Interface Clock Domain

CIPHER_ICLK is the interface clock with the DSP Private TIPB. It is provided by the ClkDSP52 clock to access the Cipher_A5 module registers.

For more information about these clocks, refer to Chapter 6, *Power, Reset, and Clock Management*.

15.3.1.2 Resets

15.3.1.2.1 Hardware Reset

A Cipher_A5 module hardware reset is performed by the DSP_nRST reset signal.

For more information about these clocks, refer to Chapter 6, *Power, Reset, and Clock Management*.

15.3.1.2.2 Software Reset

The Cipher_A5 module includes one internal software reset:

- The CIPHER.CNTL_REG[1] RESET_SW bit register resets the internal blocks of the Cipher_A5 module. It does not reset the registers (control, status, and data registers). The reset must be performed before and in between each processing. Refer to Section 15.3.3, *Cipher_A5 Module Programming Guide*.

15.3.1.3 Interrupt Requests

The Cipher_A5 module handles one interrupt line (CIPHER_IRQ), which is linked to the IRQ_7 of the DSP interrupt handler.

The interrupt is not maskable.

When an interrupt occurs, software must read the CIPHER.STATUS_IRQ_REG[0] IT_FIN bit. The IT_FIN bit is automatically cleared after read: the interrupt is considered as acknowledged.

15.3.2 Cipher_A5 Functional Description

The Cipher_A5 module implements the functionality of ciphering algorithms A5/1, A5/2 and A5/3.

These algorithms protect both user data and signaling information at the physical layer on the dedicated channel. They are stream ciphers used to encrypt/decrypt blocks of data under a confidentiality key Kc.

A5/3 algorithm integration needs to implement additional registers to configure the long key (128 bits) and to generate long encipher/decipher block data: 348-bit key stream strings for ECSD encryption.

15.3.2.1 General Purpose of the Ciphering Module

The module generates two binary sequences of 114 bits in GSM mode or 348 bits in ECSD mode. They are called BLOCK1 and BLOCK2. BLOCK1 (downlink block) is used to decipher data coming from the base station (BS), and BLOCK2 (uplink block) is used to cipher data going to the base station.

The ciphering takes place before modulation and after interleaving, while the deciphering takes place after demodulation symmetrically.

The module requires an explicit time variable: the TDMA frame numbering COUNT produced by the radio subsystem, which is expressed using 22 bits. The cipher key, called Kc, has a 64-bit format in GSM A5/1-A5/2 mode and 128-bit format in A5/3 mode (GSM or ECSD).

For the A5/1-A5/2 algorithms, key length is fixed at 54 bits and is associated with complementary non-significant bits forced to zero. It is assumed that the non-significant bits are the least-significant bits of the 64-bit word, and that the actual key constitutes the most-significant bits of the 64-bit word.

For the A5/3 algorithm, this length is fixed at 128 bits and should be filled by software programming if it is less than 128 bits.

Note: Bit ordering of the Kc key depends on algorithm used.

In the Locosto device, the Cipher_A5 module is configured by the DSP through the DSP private TIPB.

15.3.2.2 Implementation and Operational Considerations

GSM 03.20 provides the performance requirements for the A5 (A5/1/2/3) cipher algorithm for the GSM/ECSD.

15.3.2.2.1 GSM

For ciphering, each 4.615 ms the A5 Algorithm produces a sequence of 114 encipher/decipher bits (here called a BLOCK), which is combined by a bit-wise modulo 2 addition with the 114-bit plain text block (this step is not performed inside this module).

For each slot, deciphering is performed on the MS side with the first block (BLOCK1) of 114 bits produced by A5, and enciphering is performed with the second block (BLOCK2). As a consequence, on the network side BLOCK1 is used for enciphering and BLOCK2 for deciphering. Therefore, algorithm A5 must produce two 114-bit blocks each 4.615 ms.

15.3.2.2.2 ECSD

In ECSD mode, the block size is greater than 114 bits. With ECSD, a modification of the usage of the A5 algorithm is employed that produces BLOCK1 and BLOCK2, each containing 348 bits. The other parameters are not modified. The modified algorithm produces both blocks during a TDMA frame duration (that is, 4.615 ms). The blocks are combined by bit-wise modulo 2 addition with the plain text data (step not done; i.e., GSM).

In ECSD, the plain text data block for either uplink or downlink can be shorter than 348 bits. In this case, only the first part of the corresponding output parameter BLOCK is used in the bit-wise addition and the rest of the bits are discarded.

The Cipher_A5 module generates both data blocks that are combined with plain text (logic XOR operation) on another level/layer.

15.3.3 Cipher_A5 Module Programming Guide

This section gives the programming sequences for full ciphering in GSM/ECSD mode (A5/1-A5/2-A5/3).

15.3.3.1 A5/1-A5/2 Encryption/Decryption

1. Reset the module, if not already done:
   Write 0x0000 in CIPHER.CNTL_REG register
2. Set the count value:
   Write the count value to the CIPHER.COUNT_REG1 and CIPHER.COUNT_REG2 registers
3. Set the CK key value
   Write the appropriate CK key value to the CIPHER.KC_REGi registers, with i from 1 to 4.
4. Enable the functional clock module, release the SW reset, set the mode:
   Write in CIPHER.CNTL_REG register:
   - 0x0012 for no algorithm mode (In this case, outputs BLOCK1 and BLOCK2 are set to zero)
   - 0x0016 for A5/1 mode
   - 0x001A for A5/2 mode
5. Start the process:
   Write to CIPHER.CNTL_REG register:
   - 0x0013 for no algorithm mode
   - 0x0017 for A5/1 mode
   - 0x001B for A5/2 mode

Only GSM encryption is available for algorithms A5/1 and A5/2.

6. Wait for CIPHER_IRQ interrupt and check the interrupt status:
   Read the CIPHER.STATUS_IRQ_REG[0] IT_FIN bit to verify that it is set to 1.
7. Decipher/encipher data blocks are available:
   Read BLOCK1 (114 bits) in the CIPHER.DECI_REG_i registers, with i from 1 to 8
   Read BLOCK2 (114 bits) in the CIPHER.ENCI_REG_i registers, with i from 1 to 8
8. Jump to step 1 in case another data processing is required. If not, stop the clock and leave the module under reset:
   Write 0x0000 to the CIPHER.CNTL_REG register.

15.3.3.2 A5/3 Encryption/Decryption

1. Reset the module, if not already done:
   Write 0x0000 to CIPHER.CNTL_REG register
2. Set the count value:
   Write the count value to CIPHER.COUNT_REG1 and CIPHER.COUNT_REG2 registers
3. Set the 128 bits CK key value:
   Write the appropriate CK value to the CIPHER.KC_REGi registers, with i from 1 to 8
4. Enable the functional clock module, release the SW reset, set the mode:
   Write to CIPHER.CNTL_REG register:
   - 0x001E for A5/3 GSM processing
   - 0x003E for A5/3 ECSD processing
5. Start the process:
   Write to CIPHER.CNTL_REG_REG register:
   - 0x001F for A5/3 GSM processing
   - 0x003F for A5/3 ECSD processing
6. Wait for CIPHER_IRQ interrupt and check the interrupt status:
   Read the CIPHER.STATUS_IRQ_REG[0] IT_FIN bit to verify that it is set to 1.
7. Decipher/encipher data blocks are available.
   For A5/3 GSM processing:
   - Read BLOCK1 (114 bits) in the CIPHER.DECI_REG_i registers, with i from 1 to 8
   - Read BLOCK2 (114 bits) in the CIPHER.ENCI_REG_i registers, with i from 1 to 8

   For A5/3 ECSD processing:
   - Read BLOCK1 (348 bits) in the CIPHER.DECI_REG_i registers, with i from 1 to 22
   - Read BLOCK2 (348 bits) in the CIPHER.ENCI_REG_i registers, with i from 1 to 22
8. Jump to step 1 in case another data processing is required. If not, stop the clock and leave the module under reset:
   - Write 0x0000 in CIPHER.CNTL_REG register.

15.4 Registers

15.4.1 GEA3 Register Manual

All registers are controlled directly by the MPU Private TIPB. Only the MPU can access them. Table 15-3 shows the base address and address space for the GEA3 module.

Table 15-3. GEA3 Module Instance Summary

Module Name	MPU Base Address	Size
GEA3	0xFFFF C000	2K bytes

15.4.1.1 GEA3 Module Register Mapping

Table 15-3 lists the GEA3 module's registers.

Table 15-4. GEA3 Module Register Offsets

Register Name	Type	Register Width (Bits)	Offset
GEA_CNTL_REG	R/W	16	0x00
GEA_STATUS_REG	R	16	0x02
GEA_STATUS_IR_REG	R	16	0x04
GEA_CONF_UL_REG1	R/W	16	0x06
GEA_CONF_UL_REG2	R/W	16	0x08
GEA_CONF_UL_REG3	R/W	16	0x0A
GEA_CONF_UL_REG4	R/W	16	0x0C
frGEA_CONF_UL_REG5	R/W	16	0x0E
GEA_CONF_DL_REG1	R/W	16	0x10
GEA_CONF_DL_REG2	R/W	16	0x12
GEA_CONF_DL_REG3	R/W	16	0x14
GEA_CONF_DL_REG4	R/W	16	0x16
GEA_CONF_DL_REG5	R/W	16	0x18
GEA_KC_REG1	R/W	16	0x1A
GEA_KC_REG2	R/W	16	0x1C
GEA_KC_REG3	R/W	16	0x1E
GEA_KC_REG4	R/W	16	0x20
GEA_KC_REG5	R/W	16	0x22
GEA_KC_REG6	R/W	16	0x24
GEA_KC_REG7	R/W	16	0x26
GEA_KC_REG8	R/W	16	0x28
GEA_FCS_UL_REG1	R	16	0x2A
GEA_FCS_UL_REG2	R	16	0x2C
GEA_FCS_DL_REG1	R	16	0x2E
GEA_FCS_DL_REG2	R	16	0x30
GEA_SWITCH_REG	R/W	16	0x32
GEA_DATA16_REG	R/W	16	0x34
GEA_DATA8_REG	R/W	16	0x36
GEA_REVNU	R	16	0x38

15.4.1.2 GEA3 Module Register Descriptions

15.4.1.2.1 Control and Status Registers

Table 15-5 through Table 15-7 describe the individual bit fields of the control and status registers of the GEA3 module.

Table 15-5. Control Register (GEA_CNTL_REG)

Physical Address = MPU: 0xFFFF C000; Offset = 0x00				
Bit	**Name**	**Function**	**Type**	**Reset**
15:6	Reserved	–	R/W	Undefined
5	IT_EN	Controls interrupt generation after processing is completed: 0: No interrupt generated. 1: An interrupt is generated.	R/W	0
4	UL_DL	Selects module processing for uplink or downlink: 0: uplink (ciphering + FCS) 1: downlink (deciphering + FCS check).	R/W	0
3	CLOCK_EN	Controls the internal clock 0: Clock off 1: Clock on	R/W	0
2	START	Starts the module. Select the uplink/downlink state machine by using the UL_DL bit. This bit is cleared by internal logic.	R/W	0
1	NRESET_DL	Set this bit to 0 to reset the downlink module (deciphering + FCS). The current processing is cancelled and the internal variables are reset. This bit is cleared by internal logic.	R/W	1
0	NRESET_UL	Set this bit to 0 to reset the uplink module (ciphering + FCS). The current processing is cancelled and the internal variables are reset. This bit is cleared by internal logic.	R/W	1

Notes:

- NRESET_UL and NRESET_DL can be used to reset UL or DL configuration/frame/mode only after the current data frame processing is completed; to prevent memory pointer problems, process the current frame (discard it) even if it is not required. In addition, ensure that there are as many data-write accesses (to memory) as there are read accesses.
- NRESET_UL and NRESET_DL are resynchronized and, hence, fully taken in account only when the clock is enabled. As a result, do not disable the clock until the reset has completed (reset to the initialization value).
- Reset on a part of a frame is not available.
- The START bit is used to start the module. Fill all module control bits/words and buffers before starting the module. When this bit is set, the WORKING bit of the GEA_STATUS_REG[0] is automatically set and the START bit is automatically reset.
- The CLOCK_EN bit is used to enable the internal module clock. The clock must be stable when the module is started. To ensure clock stability, refer to Section 15.2.3*GEA3 Programming Guide*.
- The UL_DL bit is used to select the part of the module to be enabled. The GEA3 module can be considered split into two internal modules: one for ciphering/FCS generation and one for deciphering/FCS checking. The UL_DL bit is different from the D bit of the GEA_CONF_xL_REG1[6] registers.
- The IT_EN bit allows disabling of the ciphering interrupt. If the ciphering bit is disabled, the MPU has to make polling on the WORKING bit of the GEA_STATUS_REG[0] register and wait for a return to 0. The interrupt can also be masked by the interrupt handler.

Table 15-6. Status Register (GEA_STATUS_REG)

Physical Address = MPU: 0xFFFF C002; ; Offset = 0x02				
Bit	**Name**	**Function**	**Type**	**Reset**
15:2	Reserved	–	R	Undefined
1	FCS_STATUS	FCS status. 0: FCS is good. 1: FCS is bad. This bit is valid in downlink only and on the last frame of the processed LLC_PDU, only if FCS compute bit (GEA_CONF_DL_REG1[5]) is set to 1.	R	0
0	WORKING	This bit indicates that the module is processing. 0: Module is not working. 1: Module is working. It is automatically enabled when the START bit of GEA_CNTL_REG has been set.	R	0

Table 15-7. Status Register for Interrupts (GEA_STATUS_IR_REG)

Physical Address = MPU: 0xFFFF C004; Offset = 0x04				
Bit	**Name**	**Function**	**Type**	**Reset**
15:1	Reserved	–	R	Undefined
0	LLC_IT	Set to 1 when the module has finished its processing It remains 1 until read. When read, this bit is automatically reset to 0.	R	0

Note: The LLC_IT bit can be used only when the IT_EN bit of GEA_CNTL_REG[5] register is enabled. When an interrupt occurs, software must check this register to identify whether the GEA3 module is the interrupt source. When read, this bit is reset to 0.

15.4.1.2.2 Configuration Registers (GEA_CONF_xL_REGi, i=[1:5])

This section describes the configuration registers of the GEA3 module. These registers are divided into two groups: uplink (UL) and downlink (DL) registers.

Uplink Registers (GEA_CONF_UL_REGi, i=[1:5])

Table 15-8 through Table 15-13 describe the individual bit fields of the uplink registers.

Table 15-8. GEA_CONF_UL_REG1

Physical Address = MPU: 0xFFFF C006; Offset = 0x06				
Bit	**Name**	**Function**	**Type**	**Reset**
15:9	Reserved	–	R/W	Undefined
8:7	AS	GEA3 algorithm selection: 00: First GEA algorithm—GEA/1 01: Second GEA algorithm—GEA/2 1x: Third GEA algorithm—GEA/3	R/W	00
6	D	Direction bit This bit is used as an input of the ciphering algorithm. Its value depends on whether the user is located on the MS or the SGSN part of the network. It is different from the UL_DL bit of GEA_CNTL_REG[4]. 0: LLC frame transmission is from the mobile station (MS) to the Serving GPRS Support Node (SGSN) (uplink). 1: LLC frame transmission is from the SGSN to the MS (downlink).	R/W	0

Table 15-8. GEA_CONF_UL_REG1 (continued)

Physical Address = MPU: 0xFFFF C006; Offset = 0x06				
Bit	**Name**	**Function**	**Type**	**Reset**
5	F	FCS computation bit 0: No FCS computed. The FCS is included in the LLC-PDU data and the GEA3 module does not compute it. PM bit (GEA_CONF_UL_REG1[4]) and N202 field (GEA_CONF_UL_REG3[7:0]) are ignored when the F bit is set to 0. 1: FCS computed. The GEA3 module computes the FCS.	R/W	0
4	PM	Protection mode bit 0: FCS computed on frame header + first N202 bytes (GEA_CONF_UL_REG3{7:0]) of the information field 1: FCS computed on frame header + info field (LLC-PDU size)	R/W	0
3	E	Encryption mode bit 0: Encryption disabled. No encryption is applied to the frame. 1: Encryption enabled. The information and FCS fields of the frame are encrypted (ciphered).	R/W	0
2	Reserved	–	R/W	Undefined
1	OS	Output buffer shift 0: No action 1: The first byte in the output buffer is set to 0x00 and the output data are shifted by one byte.	R/W	0
0	IS	Input buffer shift 0: No action 1: Byte 0 of the LLC frame is ignored. First byte sent to ciphering/FCS computation is byte 1.	R/W	0

Table 15-9. GEA_CONF_UL_REG2

Physical Address = MPU: 0xFFFF C008; Offset = 0x08				
Bit	**Name**	**Function**	**Type**	**Reset**
15:0	PDU_SIZE	LLC-PDU size in bytes. Number of bytes of the LLC-PDU contained in the buffer (IS bit (GEA_CONF_UL_REG1[0]) has no influence on this size) If the FCS computation is disabled (F bit (GEA_CONF_UL_REG1[5]) set to 0), this size takes into account the LLC header, the LLC information field, and the FCS (full LLC-PDU size). If the FCS computation is enabled (F bit (GEA_CONF_UL_REG1[5]) set to 1), this size takes into account the LLC header and the LLC information field, but not the FCS (full LLC-PDU size-3).	R/W	0x0000

Table 15-10. GEA_CONF_UL_REG3

Physical Address = MPU: 0xFFFF C00A; Offset = 0x0A				
Bit	**Name**	**Function**	**Type**	**Reset**
15:8	HEADER_SIZE	LLC-PDU header size Number of bytes in the LLC frame header. It is used to identify the starting point of the information field (for ciphering and unprotected mode) and, together with N202, the size of the data to be protected. It is not mandatory to fill this field if no ciphering and protected mode are used.	R/W	0x00

Table 15-10. GEA_CONF_UL_REG3 (continued)

Physical Address = MPU: 0xFFFF C00A; Offset = 0x0A				
Bit	**Name**	**Function**	**Type**	**Reset**
7:0	N202	N202 variable size. Used only when the PM bit (GEA_CONF_UL_REG1[4]) is set to 0 (UI frames only). When PM is set to 1, this field is ignored. Number of information field bytes used to calculate the FCS. When the length of the information field is less than N202 bytes, the FCS covers the complete information field.	R/W	0x00

Note: It is assumed that HEADER_SIZE and N202 values given for each PDU are error free. It is up to the software driving the GEA3 hardware module to take care that these values are consistent and to not disturb this module. For example, (HEADER_SIZE + N202) must not be greater than PDU size.

Table 15-11. GEA_CONF_UL_REG4

Physical Address = MPU: 0xFFFF C00C; Offset = 0x0C				
Bit	**Name**	**Function**	**Type**	**Reset**
15:0	CIPH_IN_LSB	LSB part of the CIPH_IN register (message key)	R/W	0x0000

Table 15-12. GEA_CONF_UL_REG5

Physical Address = MPU: 0xFFFF C00E; Offset = 0x0E				
Bit	**Name**	**Function**	**Type**	**Reset**
15:0	CIPH_IN_MSB	MSB part of the CIPH_IN register (message key)	R/W	0x0000

Table 15-13. Bit Mapping for GEA_CONF_UL_REG4 and GEA_CONF_UL_REG5

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GEA_CONF_UL_REG5	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GEA_CONF_UL_REG4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note: GEA_CONF_UL_REG4 contains the LSB and GEA_CONF_UL_REG5 contains the MSB of the CIPH_IN control word.

Example: CIPH_IN = 0x12345678 → GEA_CONF_UL_REG4 = 0x5678 and GEA_CONF_UL_REG5 = 0x1234

Downlink Registers (GEA_CONF_DL_REGi, i=[1:5])

Table 15-14 through Table 15-19 describe the individual bit fields of the Downlink registers.

Table 15-14. GEA_CONF_DL_REG1

Physical Address = MPU: 0xFFFF C010; Offset = 0x10				
Bit	**Name**	**Function**	**Type**	**Reset**
15:9	Reserved	–	R/W	Undefined
8:7	AS	GEA3 algorithm selection: 00: First GEA algorithm—GEA/1 01: Second GEA algorithm—GEA/2 1x: Third GEA algorithm—GEA/3	R/W	00

Table 15-14. GEA_CONF_DL_REG1 (continued)

Physical Address = MPU: 0xFFFF C010; Offset = 0x10				
Bit	**Name**	**Function**	**Type**	**Reset**
6	D	Direction bit. This bit is used as an input of the ciphering algorithm. Its value depends on which part of the network the user is located (MS or SGSN). It is different from the UL_DL bit of GEA_CNTL_REG[4] 0: LLC frame transmission is from the mobile station (MS) to the Serving GPRS Support Node (SGSN) (uplink). 1: LLC frame transmission is from the SGSN to the MS (downlink).	R/W	0
5	F	FCS computation bit 0: No FCS computed. The FCS is included in the LLC-PDU data and the GEA3 module does not compute it. PM bit (GEA_CONF_DL_REG1[4]) and N202 field (GEA_CONF_DL_REG3[7:0]) are ignored when F bit is set to 0. 1: FCS computed. The GEA3 module computes the FCS.	R/W	0
4	PM	Protection mode bit 0: FCS computed on frame header + first N202 bytes (GEA_CONF_DL_REG3{7:0]) of the information field. 1: FCS computed on frame header + info field (LLC-PDU size).	R/W	0
3	E	Encryption mode bit 0: Encryption disabled. No encryption is applied to the frame. 1: Encryption enabled. The information and FCS fields of the frame are encrypted (ciphered).	R/W	0
2	Reserved	–	R/W	Undefined
1	OS	Output buffer shift 0: No action 1: The first byte in the output buffer is set to 0x00 and the output data are shifted by one byte.	R/W	0
0	IS	Input buffer shift 0: No action 1: Byte 0 of the LLC frame is ignored. First byte sent to ciphering/FCS computation is byte 1.	R/W	0

Table 15-15. GEA_CONF_DL_REG2

Physical Address = MPU: 0xFFFF C012; Offset = 0x12				
Bit	**Name**	**Function**	**Type**	**Reset**
15:0	PDU_SIZE	LLC-PDU size in bytes with FCS Number of bytes of the LLC-PDU contained into the buffer (IS bit (GEA_CONF_DL_REG1[0]) has no influence on this size). This size takes into account the LLC header, the LLC information field, and the LLC FCS. It is used to identify when the end of the frame is reached. Upon reaching the end of the frame, all internal variables must be reset in preparation to process a new frame. This value is not known in advance for downlinks because it is only known on the last RLC/MAC block received. This is why the MPU must set it only for processing the last part of the frame; otherwise, it must be set to 0. The module reads it every time the start bit is set; if its value is different from 0, then this is the last part of the LLC-PDU. If FCS is computed, the last RLC/MAC block must maintain at least 3 FCS bytes in one shot. Write PDU_SIZE before completing processing of all PDU bytes.	R/W	0x0000

Note: This register can not be modified by the GEA3 module. Instead, the MPU has to set/reset it using the correct values.

Table 15-16. GEA_CONF_DL_REG3

Physical Address = MPU: 0xFFFF C014; Offset = 0x14				
Bit	**Name**	**Function**	**Type**	**Reset**
15:8	HEADER_SIZE	LLC-PDU header size. Number of bytes in the LLC frame header. It is used to identify the starting point of the information field (for ciphering and unprotected mode) and, together with N202, the size of the data to be protected. It is not mandatory to fill it if no ciphering and protected mode are used.	R/W	0x00
7:0	N202	N202 variable size. It is used only when the PM bit is set to 1. Used only when the PM bit (GEA_CONF_UL_REG1[4]) is set to 0 (UI frames only). When PM is set to 1, this field is ignored. Number of information field bytes used to calculate the FCS. When the length of the information field is less than N202 bytes, the FCS covers the complete information field.	R/W	0x00

Note: It is assumed that HEADER_SIZE and N202 values given for each PDU are error free. It is up to the software driving the GEA3 hardware module to take care that these values are consistent and to not disturb this module. For example, (HEADER_SIZE + N202) must not be greater than PDU size.

Table 15-17. GEA_CONF_DL_REG4

Physical Address = MPU: 0xFFFF C016; Offset = 0x16				
Bit	**Name**	**Function**	**Type**	**Reset**
15:0	CIPH_IN_LSB	LSB part of the CIPH_IN register (message key)	R/W	0x0000

Table 15-18. GEA_CONF_DL_REG5

Physical Address = MPU: 0xFFFF C018; Offset = 0x18				
Bit	**Name**	**Function**	**Type**	**Reset**
15:0	CIPH_IN_MSB	MSB part of the CIPH_IN register (message key)	R/W	0x0000

Table 15-19. Bit Mapping for CONF_DL_REG4 and CONF_DL_REG5

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GEA_CONF_DL_REG4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GEA_CONF_DL_REG5	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Note: GEA_CONF_DL_REG4 contains the LSB and GEA_CONF_DL_REG5 contains the MSB of the CIPH_IN control word.

Example: CIPH_IN = 0x12345678 → GEA_CONF_DL_REG4 = 0x5678 and GEA_CONF_DL_REG5 = 0x1234

15.4.1.2.3 Kc Registers (GEA_KC_REGi, i=[1:8])

These registers contain ciphering key Kc (64 bits) for GEA/1 and GEA/2 or ciphering key CK (128 bits) for the GEA/3 algorithm (internal 128-bit input to the KGCORE function). It is split into eight 16-bit registers; the bit order is given in Table 15-20.

Table 15-20. Kc Registers Bit Order

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GEA_KC_REG1	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
GEA_KC_REG2	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
GEA_KC_REG3	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
GEA_KC_REG4	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49
GEA_KC_REG5	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65
GEA_KC_REG6	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81
GEA_KC_REG7	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97
GEA_KC_REG8	128	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113

This variable (key) is set by the MPU and is taken into account by the GEA3 module at the beginning of the processing of each LLC-PDU. This means that the KC/CK variable can be modified as soon as one byte of the LLC-PDU has been processed. This new value is used only on the next LLC-PDU.

When the GEA/3 GPRS encryption algorithm is selected, a 128-bit input key, which is a combination of cipher key Kc, must be programmed. To allow for possible future enhancements to the standards, the algorithm (GEA/3 only) allows key lengths between 64 and 128, inclusive.

If the KLEN parameter is the length of Kc in bits, between 64 and 128 inclusive, a pseudo key CK is defined for being used by the KGCORE/KASUMI GEA3 architecture inside these registers:

```
CK[0] . . . CK[KLEN-1] = Kc [0] . . . Kc[KLEN-1]

If KLEN < 128 then     CK[KLEN] . . . CK[127] = Kc[0] . . . Kc[127-KLEN]

(so in particular, if KLEN = 64 then CK = Kc || Kc)
```

In order to use the GEA/3 algorithm, be sure to program the 128-bit CK cipher key and not the Kc key value directly.

15.4.1.2.4 FCS Registers (GEA_FCS_xL_REGi, i=[1:2])

These are the results of the FCS computation and are generated if the F bit of GEA_CONF_xL_REG1[5] is set to 1 (FCS computation bit).

Uplink Registers (GEA_FCS_UL_REGi, i=[1:2])

The two FCS uplink registers corresponds to the 24-bit FCS computed by the LLC module on the uplink LLC-PDU. It is split on two 16-bit read-only registers. The highest-order terms of the FCS are transmitted first, so they are located in the LSB of the register. These registers contain the plain or the ciphered FCS, depending on the E bit of GEA_CONF_UL_REG1[3]. Table 15-21 gives the register bit order.

Table 15-21. FCS Register Bit Order

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GEA_FCS_UL_REG1	B8	B9	…											…	B22	B23
GEA_FCS_UL_REG2	0	0	0	0	0	0	0	0	B0	B1	B2	…			…	B7

Downlink Registers (GEA_FCS_DL_REGi, i=[1:2])

The two FCS downlink registers corresponds to the 24-bit FCS computed by the LLC module on the downlink LLC-PDU. It is split over two 16-bit read-only registers. The highest-order terms of the FCS are received first, so the computed FCS is displayed in the same order. These registers always contain the plain FCS. Table 15-22 gives the register bit order.

Table 15-22. Downlink Registers Bit Order(1)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GEA_FCS_DL_REG1	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20	B21	B22	B23
GEA_FCS_DL_REG2	0	0	0	0	0	0	0	0	B0	B1	B2	B3	B4	B5	B6	B7

(1) The FCS is computed on the LLC header and information field, but not on the received FCS.

15.4.1.2.5 Switch Clock Register (GEA_SWITCH_REG)

Table 15-23 describes the individual bit fields of the switch clock register.

Table 15-23. Switch Clock Register (GEA_SWITCH_REG)

Physical Address = MPU: 0xFFFF C032; Offset = 0x32				
Bit	**Name**	**Function**	**Type**	**Reset**
15:1	Reserved	–	R	Undefined
0	SWITCH_CLOCK	Set to 1 when a GEA3 memory access is required (Needed for read and write access to the GEA_DATA16_REG and GEA_DATA8_REG registers) Set to 0 when GEA3 memory accesses are completed.	R/W	0

Note: Before each access to GEA3 memory, software must set the SWITCH_CLOCK bit to 1 to switch the internal memory clock. When all accesses to memory (read or write) are complete, software can deactivate this bit (reset to 0).

By default, the SWITCH_CLOCK bit is set to 0 and GEA3_FCLK is the functional internal-memory clock that allows to process data (processing/working). For an external access, for example, to write data to memory, the memory clock must be switched to the TIPB clock (ClkBridge52) by setting the SWITCH_CLOCK bit field to 1.

The SWITCH_CLOCK bit must be set to 1 before accessing memory (external access), and then set back to 0 when finished.

15.4.1.2.6 DATA16 Register (GEA_DATA16_REG)

The DATA 16 register is used to send/receive data to/from the GEA3 module. Data is redirected to an internal buffer of the GEA3 module (there is only one internal buffer that can be used for uplink or downlink). This internal buffer can contain up to 1560 bytes (see Section 15.2.2.8, *Memory Management*, for details).

The size of the current block/frame (of LLC_PDU) is automatically determined by an internal counter that counts the number of write accesses to the register before the START bit is enabled. This data is in 16-bit word format and uses the bit order given in Table 15-24. Bit 0 is processed first.

Table 15-24. GEA_DATA16_REG Register Data Format

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GEA_DATA16_REG	DATA_MSB								DATA_LSB							

Table 15-25 describes the individual bit fields of the DATA16 register.

Table 15-25. GEA_DATA16_REG (1)

Physical Address = MPU: 0xFFFF C034; Offset = 0x34				
Bit	**Name**	**Function**	**Type**	**Reset**
15:8	DATA_MSB	MSB part of the 16-bit word	R/W	0x00
7:0	DATA_LSB	LSB part of the 16-bit word	R/W	0x00

(1) To manage this interface efficiently, the MPU is considered to be in little endian order only. For example, data 0x1234, located in RAM, is copied as 0x1234 into the DATA register (no swap) with DATA LSB equal to 0x34 and DATA MSB equal to 0x12.

Note: To manage this interface efficiently, the MPU is considered to be in little endian order only. For example, data 0x1234, located in RAM, is copied as 0x1234 into the GEA_DATA16_REG register (no swap) with DATA_LSB equal to 0x34 and DATA_MSB equal to 0x12.

Bit Order

An LLC frame contains a header, and information field, and a checksum (FCS).

In uplink, the FCS can be given or not to the module, depending on the F bit.

In downlink, this FCS must always be given to the module.

Table 15-26. GEA_DATA16_REG Register Processing Order

Bit	**b0**	**b1**	**b2** …	… **bn**
LLC frame	H	Information field		FCS

This frame is split into 16-bit words that are sent to the GEA3 module for processing.

The bits are processed in the order b0…b15 b16…b32… with b0 the first bit of the header of the LLC frame (see Table 15-27). However, FCS (in downlink) is given with the highest-order terms first.

Table 15-27. GEA_DATA16_REG Register Bit Order

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
b47	b46	b45	b44	b43	b42	b41	b40	b39	b38	b37	b36	b35	b34	b33	b32
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
bf8	bf9	bf10	bf11	bf12	bf13	bf14	bf15	bf16	bf17	bf18	bf19	bf20	bf21	bf22	bf23
0	0	0	0	0	0	0	0	bf0	bf1	bf2	bf3	bf4	bf5	bf6	bf7

Note: In Table 15-27, the bf bits are FCS bits

Frame Splitting

The LLC frames have a dynamic size (two LLC frames do not have necessarily the same size). These frames are split into RLC/MAC frames for transmission, which have a fixed size that depends on the protection chosen. The LLC frames are transmitted continuously, i.e., no padding exists between two frames; therefore, a RLC/MAC frame can contain parts of more than one LLC frame.

Most of the time, the GEA3 processing (especially in downlink) is done as soon as new data are received. Therefore, the processing is done on a RLC/MAC frame basis (every 20 ms).

Uplink and downlink LLC frames are multiplexed, i.e., the GPRS can start ciphering a part of an uplink LLC frame and then start to process a part of a received LLC frame, and so on. Therefore, the GEA3 module keeps in memory all the static variables used for uplink or downlink.

15.4.1.2.7 Data8 Register (GEA_DATA8_REG)

Table 15-28 describes the individual bit fields of the DATA8 register.

Table 15-28. GEA_DATA8_REG

Physical Address = MPU: 0xFFFF C036; Offset = 0x36				
Bit	**Name**	**Function**	**Type**	**Reset**
15:8	Reserved	–	R/W	Undefined
7:0	DATA8	8-bits of data	R/W	0x00

Note: Even if the default data access by DATA 16 register is in 16-bit format (using IS and OS bits to adapt even/odd frame), this DATA 8 register is still available. Two accesses to this register is equivalent to one 16-bit access. If an 8-bit write access is followed by a 16-bit write access, the data is continuous in the memory.

Example:

8 bits write 0x11, followed by 16 bits write 0x3322. Data is stored in memory as follows:

MSB Byte	LSB Byte
0x22	0x11
	0x33

15.4.1.2.8 Revision Number Register (REVNU_REG)

Table 15-29 describes the individual bit fields of the revision number register.

Table 15-29. Revision Number Register (GEA_REVNU)

Physical Address = MPU: 0xFFFF C038; Offset = 0x38				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:8	Reserved	–	R	Undefined
7:0	REV	IP revision The four LSBs indicate a minor revision. The four MSBs indicate a major revision. Example: 0x10 for 1.0, 0x21 for 2.1	R	(1)

(1) TI internal data

15.4.2 Cipher_A5 Register Manual

All registers are controlled directly by the DSP Private TIPB. Only the DSP can access to them. Table 15-30 shows the base address and address space for the GEA3 module.

Table 15-30. Cipher_A5 Module Instance Summary

Module Name	DSP Base Address	Size
Cipher_A5	0x2800	4K bytes

15.4.2.1 Cipher_A5 Module Register Mapping

Table 15-31 lists the Cipher_A5 module's registers.

Table 15-31. Cipher_A5 Module Register Offsets

Register Name	Type	Register Width (Bits)	Offset
CNTL_REG	R/W	16	0x00
STATUS_IRQ_REG	R	16	0x01
STATUS_WORK_REG	R	16	0x02
KC_REG1	R/W	16	0x03
KC_REG2	R/W	16	0x04
KC_REG3	R/W	16	0x05
KC_REG4	R/W	16	0x06
KC_REG5	R/W	16	0x07
KC_REG6	R/W	16	0x08
KC_REG7	R/W	16	0x09

Table 15-31. Cipher_A5 Module Register Offsets (continued)

Register Name	Type	Register Width (Bits)	Offset
KC_REG8	R/W	16	0x0A
COUNT_REG1	R/W	16	0x0B
COUNT_REG2	R/W	16	0x0C
DECI_REG_1	R	16	0x0D
DECI_REG_2	R	16	0x0E
DECI_REG_3	R	16	0x0F
DECI_REG_4	R	16	0x10
DECI_REG_5	R	16	0x11
DECI_REG_6	R	16	0x12
DECI_REG_7	R	16	0x13
DECI_REG_8	R	16	0x14
DECI_REG_9	R	16	0x15
DECI_REG_10	R	16	0x16
DECI_REG_11	R	16	0x17
DECI_REG_12	R	16	0x18
DECI_REG_13	R	16	0x19
DECI_REG_14	R	16	0x1A
DECI_REG_15	R	16	0x1B
DECI_REG_16	R	16	0x1C
DECI_REG_17	R	16	0x1D
DECI_REG_18	R	16	0x1E
DECI_REG_19	R	16	0x1F
DECI_REG_20	R	16	0x20
DECI_REG_21	R	16	0x21
DECI_REG_22	R	16	0x22
ENCI_REG_1	R	16	0x23
ENCI_REG_2	R	16	0x24
ENCI_REG_3	R	16	0x25
ENCI_REG_4	R	16	0x26
ENCI_REG_5	R	16	0x27
ENCI_REG_6	R	16	0x28
ENCI_REG_7	R	16	0x29
ENCI_REG_8	R	16	0x2A
ENCI_REG_9	R	16	0x2B
ENCI_REG_10	R	16	0x2C
ENCI_REG_11	R	16	0x2D
ENCI_REG_12	R	16	0x2E
ENCI_REG_13	R	16	0x2F
ENCI_REG_14	R	16	0x30
ENCI_REG_15	R	16	0x31
ENCI_REG_16	R	16	0x32
ENCI_REG_17	R	16	0x33
ENCI_REG_18	R	16	0x34
ENCI_REG_19	R	16	0x35
ENCI_REG_20	R	16	0x36
ENCI_REG_21	R	16	0x37
ENCI_REG_22	R	16	0x38

Table 15-31. Cipher_A5 Module Register Offsets (continued)

Register Name	Type	Register Width (Bits)	Offset
REVNU_REG	R	16	0x39

15.4.2.2 Cipher_A5 Module Register Descriptions

15.4.2.2.1 Control and Status Registers

Table 15-32 through Table 15-34 describe the individual bit fields of the control and status registers.

Table 15-32. Control Register (CNTL_REG)

Physical Address = DSP: 0x2800; Offset = 0x00				
Bit	**Name**	**Function**	**Type**	**Reset**
15:6	Reserved	—	R/W	Undefined
5	TYPE	Type of encryption 0: GSM (default)—BLOCK size: 114 bits 1: ECSD (EDGE)—BLOCK size: 348 bits	R/W	0
4	CLK_EN	Enables the internal clock 0: Clock disabled 1: Clock enabled All Cipher_A5 module's registers are only accessible if the internal clock is enabled.	R/W	0
3:2	MODE	Algorithm selection Defines which algorithm is used 00: no algorithm 01: algorithm A5/1 10: algorithm A5/2 11: algorithm A5/3	R/W	00
1	RESET_SW	This bit resets the internal blocks of the Cipher_A5 module. It does not reset the registers (control, status, and data registers). The reset must be performed before each processing. Refer to Section 15.4.2*Cipher_A5 Module Programming Guide* section. Active low	R/W	0
0	START	An event on start (rising edge only) starts the process. Active on rising edge Toggle bit (reset to 0 when read)	R/W	0

Note: In Kc key registers (KC_REG1-8, with key value), bits ordering depends on the TYPE parameter: for the A5/1 and A5/2 algorithms, a 54-bit key must be entered (refer to Section 15.3.3 *Cipher_A5 Programming Guide*) and for A5/3, a 128 bit key must be entered (fill using software if less than 128 bits).

Table 15-33. Status Register (STATUS_IRQ_REG)

Physical Address = DSP: 0x2801; Offset = 0x01				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1	Reserved	—	R	0x0000
0	IT_FIN	Set to 1 when ciphering is completed; remains 1 until read.	R	0

Table 15-34. Status Register (STATUS_WORK_REG)

Physical Address = DSP: 0x2802; Offset = 0x02				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1	Reserved	—	R	0x0000
0	WORKING	Set to 1 when the module is working	R	0

15.4.2.2.2 Kc Registers (KC_REGi, i=[1:8])

Table 15-35 describes the individual Kc registers bit fields.

Table 15-35. Kc Registers (KC_REGi, i=[1:8])

Physical Address = DSP: 0x2803 – 0x280A; Offset = 0x03 – 0x0A				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	KC_i	Contains 16 bits of Kc	R/W	0x0000

15.4.2.2.2.1 For A5/1 and A5/2 Algorithms (GSM Mode Only)

The 54-bit Kc key is directly written into the Kc registers, as described in Table 15-36.

Table 15-36. KC_REGi Bits Ordering (A5/1 and A5/2)

		b_{15}	…													…	b_0
First word	KC_REG1	6	5	4	3	2	1	0	0	0	0	0	0	0	0	0	0
Second word	KC_REG2	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
Third word	KC_REG3	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23
Fourth word	KC_REG4	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39
KC_REG5 to KC_REG8 not used		0	0	0	…									…	0	0	0

Note: Bit 1 is the LSB of the Kc key (b10 on KC_REG1), bit 54 is the MSB of the Kc key (b15 of KC_REG4).

15.4.2.2.2.2 For the A5/3 Algorithm (GSM or ECSD Mode)

When using algorithm A5/1 or A5/2, cipher key length (Kc) is fixed at 54 bits (see Table 15-36. For the A5/3 algorithm, a 128-bit input key is required. This key is a combination of the cipher Kc key and is named CK.

The 128-bit CK key is written into the Kc registers as described in Table 15-37.

Table 15-37. KC_REGi Bits Ordering (A5/3)

	b_{15}	…													…	b_0
KC_REG1	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
…	…															…
KC_REG5	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65
…	…															…
KC_REG7	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97
KC_REG8	128	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113

Note: Bit 1 is LSB of CK (b0 of KC_REG1), bit 127 is MSB of CK (b15 of KC_REG8).

To allow future enhancements to the standards, the algorithm (A5/3 only) allows key length between 64 and 128 bits, inclusive. If the KLEN parameter is the length of Kc in bits, between 64 and 128, inclusive, a pseudo key CK is defined which is used by KGCORE/KASUMI architecture (written in KC_REGi registers):

If KLEN < 128, then CK[127] … CK[128-KLEN] = Kc[KLEN-1] … Kc[0]
CK[127-KLEN] … CK[0] = Kc[KLEN-1] … Kc[2*KLEN-128]

(So, in particular, if KLEN = 64, then CK = Kc || Kc)

If KLEN = 128, then CK[127] … CK[0] = Kc[127] … Kc[0]

So, to use the A5/3 algorithm, the 128-bit cipher key CK must be programmed instead of Kc.

Example:

Kc key: 0x5ACB1D644C0D51204EA55ACB1D644C0D - KLEN = 80

Therefore, CK = 0x5ACB1D644C0D51204EA55ACB1D644C0D5ACB1D644C0D (128bits)

15.4.2.2.3 Count Registers (COUNT_REGi, i=[1:2])

These registers contain the frame number COUNT, as specified in Table 15-38.

Bit 1 is LSB of COUNT; bit 22 is MSB of COUNT.

They should be scaled correctly for COUNT before the process starts.

Table 15-38. COUNT_REGi Bits Ordering

		b_{15}	…													…	b_0
First word	COUNT_REG1	0	0	0	0	0	11	10	9	8	7	6	5	4	3	2	1
Second word	COUNT_REG2	0	0	0	0	0	22	21	20	19	18	17	16	15	14	13	12

Table 15-39 describes the individual bit fields of the count registers.

Table 15-39. COUNT_REGi Register Bits

Physical Address = DSP: 0x280B– 0x280C; Offset = 0x0B –0x0C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:11	Reserved	–	R	0x00
10:0	COUNT_i	Contains 11 bits of COUNT	R/W	0x000

15.4.2.2.4 Decipher Data Registers (DECI_REG_i, i=[1:22])

Table 15-40 describes the decipher data register bit fields. Table 15-41 describes the register bit order in GSM mode, and Table 15-42 describes the bit ordering in ECSD mode.

Table 15-40. DECI_REG_i Register Bits

Physical Address = DSP: 0x280D– 0x2822; Offset = 0x0D – 0x22				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DECI_i	Contains 16 decipher data bits	R	0x00

These registers contain the 114 bits of BLOCK1 (GSM mode) or 348 bits of BLOCK1 (ECSD mode).

Table 15-41. DECI_REG_i Bits Ordering (GSM Mode)

		b_{15}	…													…	b_0
First word	DECI_REG_1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Second word	DECI_REG_2	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
…		…															…
Seventh word	DECI_REG_7	…													…	111	112
Eighth word	DECI_REG_8	113	114	0	0	0	0	0	0	0	0	0	0	0	0	0	0

For GSM encryption, the least-significant bit of BLOCK 1 is in DECI_REG_1[15] and the most-significant bit is in DECI_REG_8[14].

Note: DECI_REG_9 to DECI_REG_22 registers are ignored (default value: 0x0000).

Table 15-42. DECI_REG_i Bits Ordering (ECSD Mode)

		b_{15}	…									…	b_4	…		…	b_0
First word	DECI_REG_1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Second word	DECI_REG_2	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
…		…															…
Twentieth word	DECI_REG_20	…													…	319	320
Twenty first word	DECI_REG_21	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336
Twenty second word	DECI_REG_22	337	338	339	340	341	342	343	344	345	346	347	348	0	0	0	0

For ECSD encryption, the least-significant bit of BLOCK 1 is in DECI_REG_1[15], and the most-significant bit is in DECI_REG_22[4].

15.4.2.2.5 Encipher Data Registers (ENCI_REG_i, i=[1:22])

Table 15-43 describes encipher data register bit fields. Table 15-44 describes the register bit ordering in GSM mode, and Table 15-45 describes the bit ordering in ECSD mode.

Table 15-43. Encipher Data Registers 1-22 (ENCI_REG_i, i=[1:22])

Physical Address = DSP: 0x2823 – 0x2838; Offset = 0x23 – 0x38				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	ENCI_i	Contains 16 encipher data bits	R	0x0000

These registers contain the 114 bits of BLOCK2 (GSM mode) or 348 bits of BLOCK2 (ECSD mode).

Table 15-44. ENCI_REG_i Bits Ordering (GSM Mode)

		b_{15}	…													…	b_0
First word	ENCI_REG_1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Second word	ENCI_REG_2	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
…		…															…
Seventh word	ENCI_REG_7	…													…	111	112
Eighth word	ENCI_REG_8	113	114	0	0	0	0	0	0	0	0	0	0	0	0	0	0

For GSM encryption, the least significant bit of BLOCK2 is in ENCI_REG_1[15] and the most significant bit is in ENCI_REG_8[14].

Note: ENCI_REG_9 to ENCI_REG_22 registers are ignored (default value: 0x0000).

Table 15-45. ENCI_REG_i Bits Ordering (ECSD Mode)

		b_{15}	…									…	b_4	…		…	b_0
First word	ENCI_REG_1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Second word	ENCI_REG_2	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
…		…															…
Twentieth word	ENCI_REG_20	…													…	319	320
Twenty first word	ENCI_REG_21	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336
Twenty second word	ENCI_REG_22	337	338	339	340	341	342	343	344	345	346	347	348	0	0	0	0

For ECSD Encryption, the least significant bit of BLOCK2 is in ENCI_REG_1[15] and the most significant bit is in ENCI_REG_22[4].

15.4.2.2.6 Revision Number Register

Table 15-46 describes the individual bit fields of the revision number register.

Table 15-46. Revision Number Register (REVNU_REG)

Physical Address = DSP: 0x2839; Offset = 0x39				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:8	Reserved	–	R	Undefined
7:0	REV	IP revision	R	(1)
		The four LSBs indicate a minor revision.		
		The four MSBs indicate a major revision.		
		Example: 0x10 for 1.0, 0x21 for 2.1		

(1) TI internal data

Time Processing Unit

This chapter describes the LOCOSTO time processing unit (TPU) module and the associated TPU2OCP module.

Topic **Page**

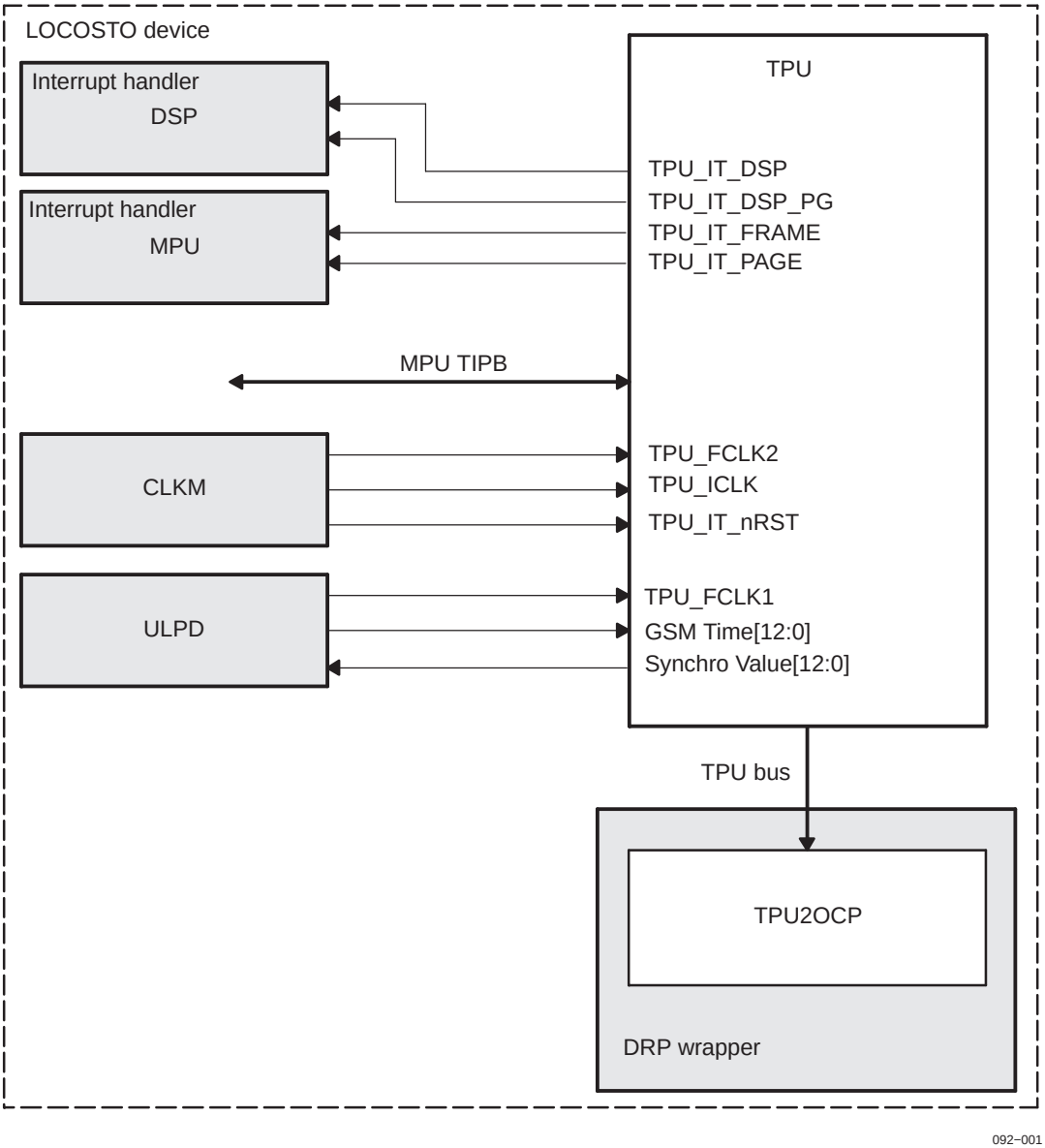
16.1 Time Processing Unit (TPU)

This section describes the TPU module of the LOCOSTO device.

16.1.1 TPU Overview

Figure 16-1 shows the TPU.

Figure 16-1. Time Processing Unit



The TPU real-time sequencer is dedicated to monitoring GSM baseband processing. Working from an event table referred to a GSM TDMA time base, the TPU activates tasks to control DSP peripherals with respect to the time constraints related to GSM sequencing.

To store the real-time microinstructions of the sequencer, the TPU includes a two-port RAM of 1024 16-bit words with dual-page addressing capability. The microprocessor unit (MPU) can access the full RAM in write mode only when the TPU is running.

The TPU schedules hardware tasks when time accuracy is incompatible with a software management.

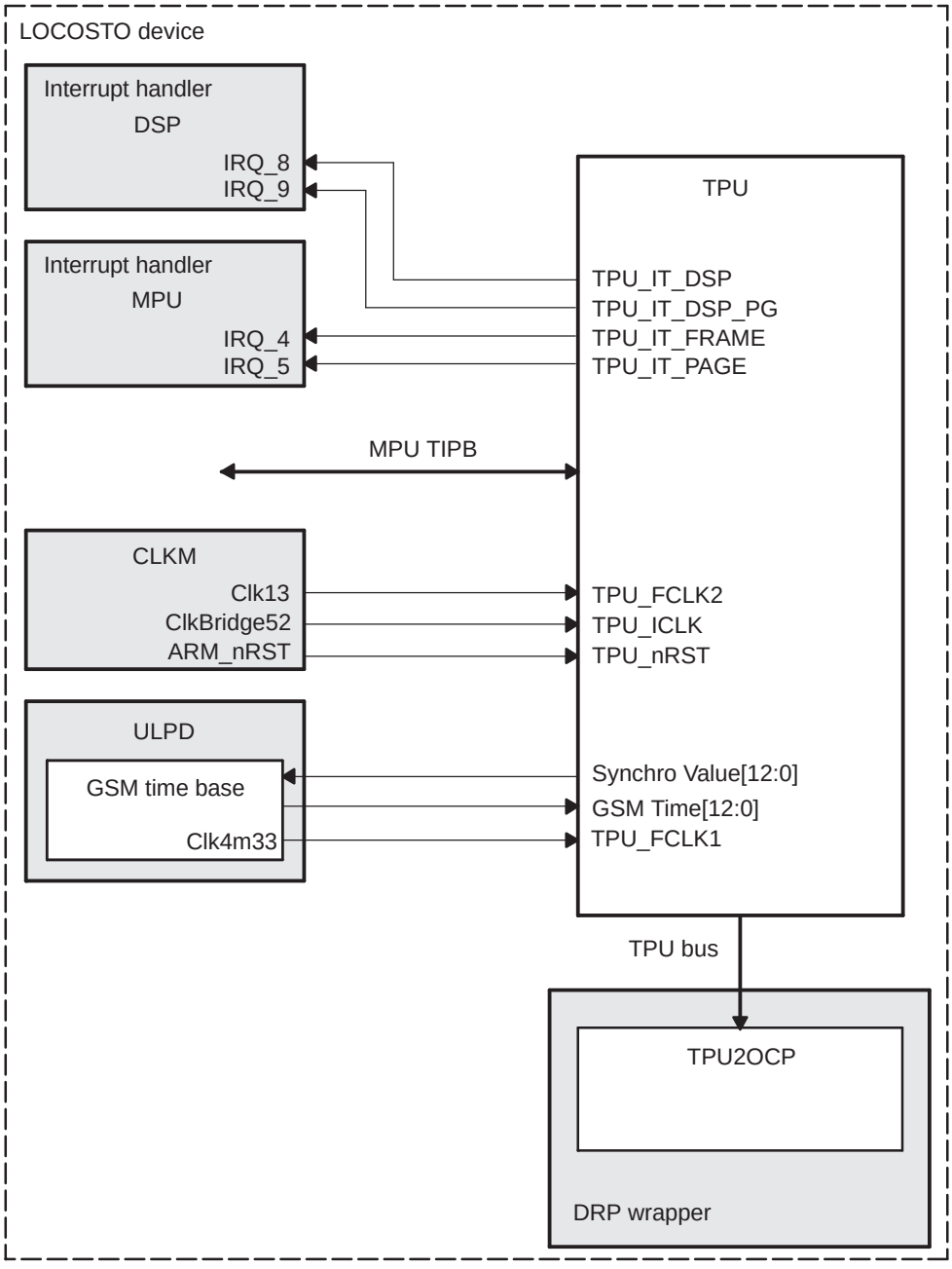
16.1.2 TPU Environment

Because the TPU module has no external connections with the environment of the LOCOSTO device, this document does not detail the TPU environment.

16.1.3 TPU Integration

This section describes TPU integration inside the LOCOSTO device. Figure 16-2 shows TPU integration with interrupt handlers, CLKM, and interconnections.

Figure 16-2. TPU Integration



16.1.3.1 Clocking, Reset, and Power-Management Scheme

The TPU module receives its reset and clock from the ultralow power-down (ULPD) module (see Table 16-1.

Table 16-1. TPU Clocking and Resets

Module	Functional Clock	Interface Clock	Resets
TPU	Clk4m33	ClkBridge52	**Hardware:** ARM_nRST
	Clk13		**Software:** TPU.REG_TPU_CTRL[0] TPU_RESET bit

16.1.3.1.1 Clocks

The TPU module is clocked with two functional clocks and one interface clock:

- TPU_FCLK1: 4.33-MHz clock (Clk4m33) generated by the ULPD module
- TPU_FCLK2: 13-MHz clock (Clk13) generated by the CLKM module
- TPU_ICLK: interface clock with the MPU private TIPB (ClkBridge52); runs at 52 MHz and triggers access to the TPU module registers

The TPU_FCLK1 clocks can be internally gated through the TPU.REG_TPU_CTRL[10] TPU_CK_ENABLE bit. This bit must be set to 1 to enable the clocks (reset value is 0).

The TPU module also provides one clock to the TPU2OCP module through the TPU bus that triggers access to some TPU2OCP module registers (see Section 16.2, *TPU2OCP Module*).

16.1.3.1.2 Resets

16.1.3.1.2.1 Hardware Reset

The ARM_nRST reset signal performs a TPU module hardware reset.

For more information on the ARM_nRST reset signal, see Chapter 6, *Power, Reset, and Clock Management*.

16.1.3.1.2.2 Software Reset

To perform the TPU software reset, set the TPU.REG_TPU_CTRL[0] TPU_RESET bit to 1. (The default value of this field is 1.)

CAUTION

The TPU_RESET bit must be cleared by software after 1/4 GSM bit time to deactivate the reset before using the TPU module.

16.1.3.2 Hardware Requests

16.1.3.2.1 DMA Requests

The TPU module does not handle DMA requests.

16.1.3.2.2 Interrupt Requests

The TPU module handles four interrupt lines (see Table 16-2).

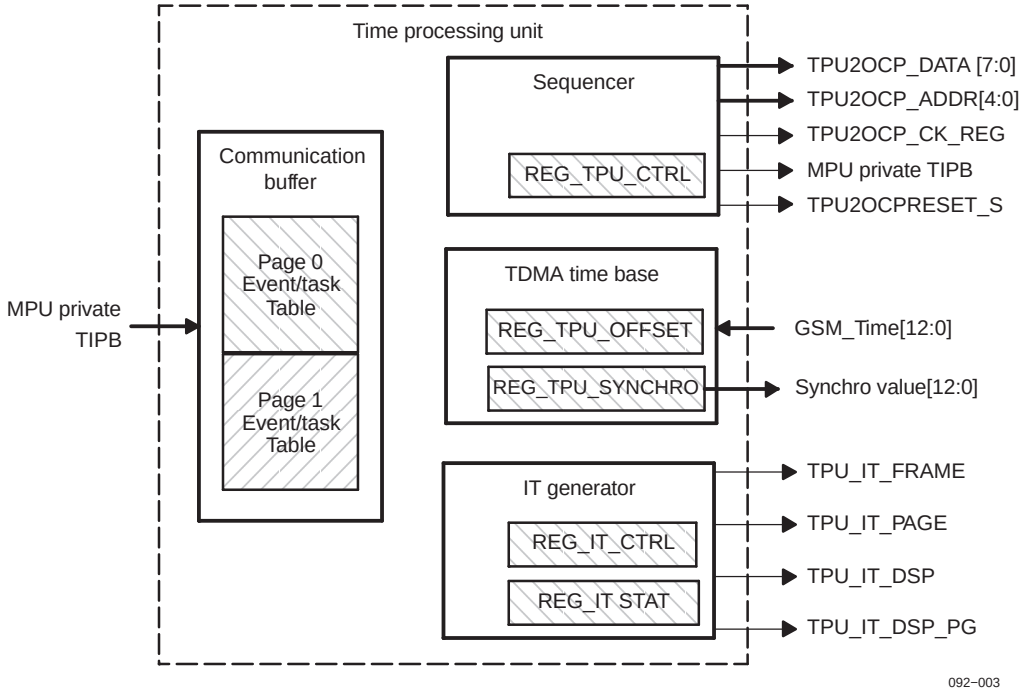
Table 16-2. TPU Interrupt Names and MPU/DSP IRQ Mapping

Interrupt Name	Mapping	Comments	Domain
TPU_IT_FRAME	IRQ_4	Frame interrupt	MPU
TPU_IT_PAGE	IRQ_5	Page interrupt	MPU
TPU_IT_DSP	IRQ_8	DSP interrupt	DSP
TPU_IT_DSP_PG	IRQ_9	Programmable interrupt	DSP

16.1.4 TPU Functional Description

Figure 16-3 shows the TPU.

Figure 16-3. Time Processing Unit



The TPU can be distributed in several functional blocks:

- The TDMA time base schedules the execution of scenarios in the TPU and gives the frame tempo to the MPU and the DSP.
- The microprogrammed sequencer is scheduled on the TDMA time base.
- The communication buffer is the memory space where the MPU transfers its scenarios to the TPU.
- The interrupt generator is responsible for the TPU interrupts generation.

16.1.4.1 TDMA Time Base

The TDMA time base gets the GSM time from the GSM time base module (for more information, see Chapter 6, *Power, Reset, and Clock Management*). The GSM time is in 1/4 GSM bit units modulo 1 TDMA frame. Thus, the GSM time range is 0 up to 5000 (5000 x 1/4 GSM bit = 1250 GSM bit = 8 x 156.25 GSM bit = 8 slots = 1 TDMA frame).

The TDMA time base gives an absolute time independently of any synchronization on a base-station and must be considered as a reference time source from which the current time of the network is computed.

Note: The whole GSM time (frame, multiframe, and super-frame) is computed in the MPU based on the frame tempo delivered by the interrupt generation.

16.1.4.1.1 Offset Principle

The principle of the TPU is to define the network time as the sum of this arbitrary reference time and an offset time calculated during a synchronization phase of the equipment.

The network time (sum of the reference GSM time and the offset) schedules the execution of the scenario in the TPU through the use of dedicated instruction (for details, see Section 16.1.4.2.4.1, *AT Instruction*).

The offset value can be dynamically modified at any time by the dedicated TPU.REG_TPU_OFFSET register writable with the OFFSET instruction of the TPU (see OFFSET instruction details). This can handle the timing advance of any transmit windows and the delta-time, neighbor cell versus serving cell, for any monitoring processing (SCH, FCH, and BCCH bursts).

16.1.4.1.2 Synchronization Principle

Using the same principle as computing network time, synchronization time is defined as the sum of the reference time and an offset value called delta synchro.

The synchronization time can be dynamically modified at any time by the dedicated TPU.REG_TPU_SYNCHRO register writable with the SYNCHRO instruction of the TPU (see the SYNCHRO instruction details). This is done for the hand-over phases or to track the serving cell.

The synchronization time (sum of the reference GSM time and the delta synchro) gives the TDMA frame tempo to both the MPU and the DSP (see Section 16.1.4.4, *Interrupt Generator*, for more information) using the TPU_IT_FRAME and the TPU_IT_DSP interrupts generation.

16.1.4.2 TPU Sequencer

16.1.4.2.1 Functional Description

The TPU sequencer is a microprogrammable machine that executes an auto scheduled program code. The corresponding time (923.1 ns), which corresponds to 1/4 GSM bit time, is the time accuracy for the execution of one microinstruction.

16.1.4.2.2 Microinstructions Set Definition

The set of microinstructions is limited to the essential instructions to minimize the complexity of the decoder.

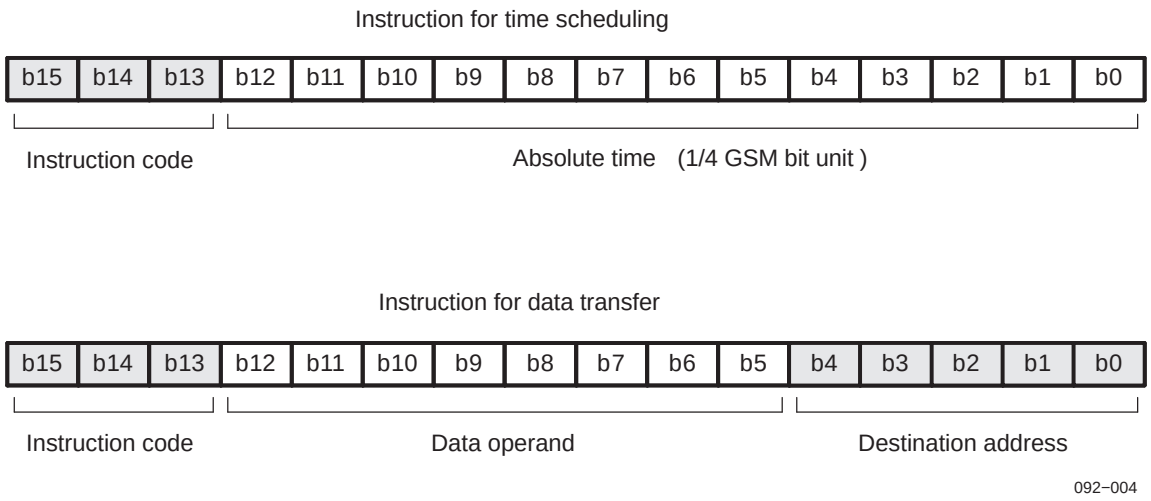
16.1.4.2.3 Structure of Microinstructions

The microinstructions are coded on 16 bits for consistency with the word format manipulated by the MPU. Split into several fields, the microinstructions have a format specific to each category of instructions.

As shown in Figure 16-4, there are two types of microinstruction:

- Instruction for time scheduling
- Instruction for data transfer

Figure 16-4. TPU Instruction Format



16.1.4.2.4 Microinstruction for Time Scheduling

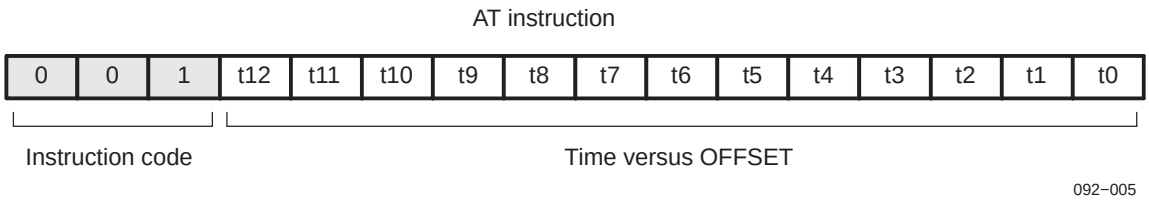
The time scheduling instructions include:

- AT: Starts a process at a relative time in the frame
- OFFSET: Loads the offset value for the network time
- SYNCHRO: Loads the offset value for the synchronization time
- WAIT: Loads the waiting time before executing the next instruction
- SLEEP: Stops the sequencer

16.1.4.2.4.1 AT Instruction

The AT instruction allows a process to start at a specific time in the GSM TDMA frame. The time is relative to the network time. The time value is stored on 13 bits and is expressed in 1/4 GSM bit units. The dynamic range is 1 TDMA frame (0 up to 4999) with a 1/4 GSM bit time accuracy (see Figure 16-5).

Figure 16-5. TPU AT Instruction

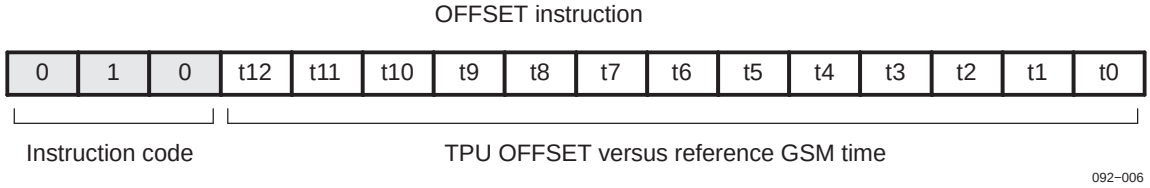


16.1.4.2.4.2 OFFSET Instruction

The OFFSET instruction loads the offset in the TPU offset register (TPU.REG_TPU_OFFSET) of the TDMA time base. The time value is stored on 13 bits and is expressed in1/4 GSM bit units. The dynamic range is 1 TDMA frame (0 up to 4999) with a 1/4 GSM bit time-accuracy.

The MPU can read the OFFSET value in the TPU.REG_TPU_OFFSET[12:0] register but cannot modify it (read-only access). See Figure 16-6.

Figure 16-6. TPU OFFSET Instruction

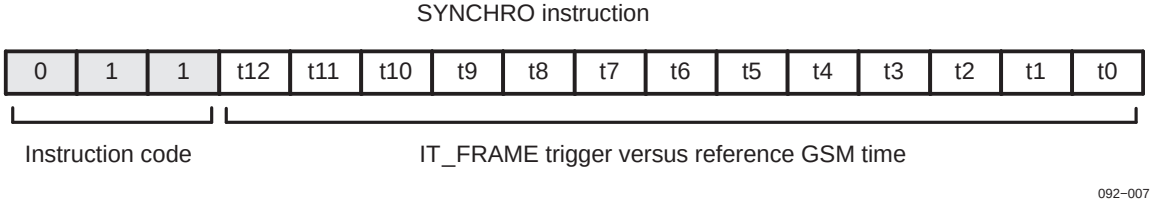


16.1.4.2.4.3 SYNCHRO Instruction

The SYNCHRO instruction simultaneously loads the synchronization value into the TPU synchronization register (TPU.REG_TPU_SYNCHRO) and the offset register of the TDMA time base. The time value is stored on 13 bits and is expressed in 1/4 GSM bit units. The dynamic range is 1 TDMA frame (0 up to 4999) with a 1/4 GSM bit time-accuracy.

The MPU can read the SYNCHRO value in the TPU.REG_TPU_SYNCHRO[12:0] register but cannot modify it (read-only access). (See Figure 16-7.)

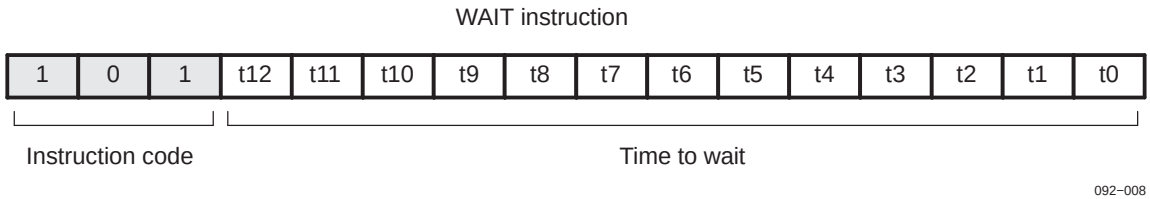
Figure 16-7. TPU SYNCHRO Instruction



16.1.4.2.4.4 WAIT Instruction

The WAIT instruction injects a waiting-period between the executions of two instructions and can be used as a time-relative AT. This time value is stored on 13 bits and is expressed in 1/4 GSM bit units. The total waiting time is equal to the programmed waiting period plus 1/4 GSM bit time interval, which corresponds to the execution time of the WAIT instruction (see Figure 16-8).

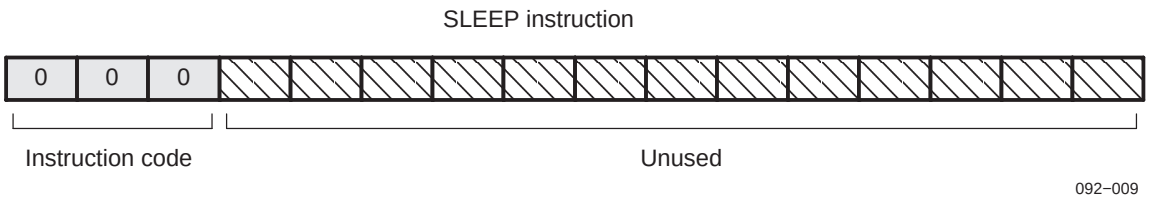
Figure 16-8. TPU WAIT Instruction



16.1.4.2.4.5 SLEEP Instruction

The SLEEP instruction disables the TPU_EN bit of the control register TPU.REG_TPU_CTRL, which stops the sequencer. To restart the TPU, the MPU must set the TPU.REG_TPU_CTRL[2] TPU_EN bit to 1 (see Figure 16-9).

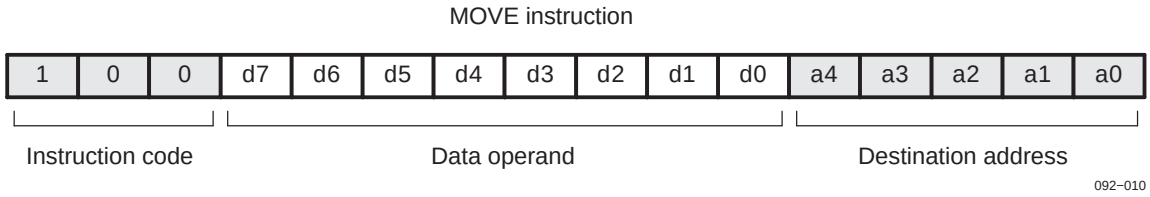
Figure 16-9. TPU SLEEP Instruction



16.1.4.2.5 Microinstruction for Data Transfer

The instruction for data processing (MOVE) writes an 8-bit max word into a register of the TPU2OCP peripheral or of the TPU. The register address is coded on 5 bits, thus defining 32 potentially-addressable registers (see Table 16-5 and Table 16-21 for TPU register offset details). Figure 16-10 shows the TPU MOVE instruction.

Figure 16-10. TPU MOVE Instruction



16.1.4.3 The Communication Buffer

The scenarios are transferred from an external host (MPU) to the TPU through a communication buffer based on a double-page memory space. This buffer lets the TPU execute one scenario during the loading of the next scenario, thus providing the possibility to chain the scenario without idle time.

From a hardware point of view, the TPU does not integrate the communication buffer, which is built from a part or the whole of a double-port or dual-access RAM. However, the memory space allocated for this communication function can be considered as intrinsic to the TPU from a functional point of view.

16.1.4.3.1 Functional Description

The scenarios are described in the form of microprograms that are elaborated by the MPU and transferred to the TPU through the communication buffer.

The transfer of data is unidirectional from the MPU to the TPU.

The TPU communication buffer is built on a double-page memory space of identical size: 1024 x 16 bit. In normal mode, only one page is accessible by the MPU at any time.

The management of the TPU communication buffer is based on a mixed status/control signal gathered in the TPU.REG_TPU_CTRL register:

- The TPU.REG_TPU_CTRL[1] TPU_PAGE bit represents the page number (page 0 or page 1) used by the TPU. This bit is toggled by the TPU.
- The TPU.REG_TPU_CTRL[2] TPU_EN bit is set by the MPU when a new scenario is available in the TPU communication buffer. This bit is cleared by the TPU.
- The TPU.REG_TPU_CTRL[8] TPU_IDLE bit represents the activity of the TPU module. It is set when the TPU is running, and reset when the TPU is in idle mode (SLEEP instruction).
- The TPU.REG_TPU_CTRL[9] TPU_WAIT bit is set when the TPU is in WAIT or AT instruction execution, and reset if it is executing another instruction.
- The TPU.REG_TPU_CTRL[0] TPU_RESET bit forces the TPU to enter in reset state (the TPU module is disable, and TPU_PAGE is 0). This bit is set and reset by the MPU.

The MPU always transfers a new scenario in the complementary page, no matter if the TPU is active or idle. Thus, the MPU can transfer a new scenario in the TPU communication buffer while the TPU executes the previous one. This allows the TPU to execute scenarios on-the-fly without any blank time.

Note: The page allocated to the MPU to write a scenario is hardware-selected by the TPU; thus, the MPU sees only a single page and does not manage the double-page allocation. The page selection is completely controlled by the TPU (the MSB address bit is managed by the TPU); the MPU has no need to take care of the page it is addressing.

However, for debug, read and write access to the TPU RAM can be given to the MPU through the TPU.REG_TPU_CTRL[6] MPU_RAM_ACCESS and the TPU.REG_TPU_CTRL[11] FULL_WRITE bits. See Table 16-6 for more information.

16.1.4.4 Interrupt Generator

Because the activities of the MPU and the DSP must be synchronized on the GSM time, the TPU generates to these processors the interrupts described in Section 16.1.4.4.1 through Section 16.1.4.4.4. All of these interrupts are edge-sensitive.

16.1.4.4.1 TPU_IT_FRAME (Frame Interrupt)

This interrupt is generated at the beginning of each new TDMA frame (GSM time = synchro value) and is provided to the MPU.

Note: To save power, the MPU must stay asleep during the GSM idle periods; therefore, the TPU_IT_FRAME interrupt is inhibited. This inhibition is automatic, but can be bypassed for debugging reasons through the TPU.REG_IT_CTRL[0] ITF_M bit.

16.1.4.4.2 TPU_IT_PAGE (Page Interrupt)

This interrupt is synchronous to any state changes of the TPU_PAGE bit field and is generated at the beginning of execution of the new instruction page of the TPU communication buffer. The page interrupt is provided to both the MPU and the DSP interrupt handlers, thus communicating to the MPU/DSP when a scenario is beginning and when a scenario can be loaded.

The validation and inhibition of this interrupt is under the control of the MPU through the TPU.REG_IT_CTRL[1] ITP_M bit.

16.1.4.4.3 TPU_IT_DSP (DSP Interrupt)

This interrupt is generated at the beginning of each new TDMA frame (GSM time = synchro value) and is provided to the DSP.

The inhibition or the forcing of the generation of this interrupt is controlled through two command bits:

- REG_IT_CTRL[2] ITD_M bit: interrupt mask
- REG_IT_CTRL[3] ITD_F bit: force interrupt

16.1.4.4.4 TPU_IT_DSG_PG (Programmable Interrupt)

This interrupt enables the scheduling of the DSP activation on a TPU instruction execution and is generated as soon as a MOVE of 1 in the TPU.REG_IT_DSP_PG register instruction is executed.

There is no mask bit for this interrupt.

16.1.4.5 Debug Feature

You can track the correct execution of a TPU scenario versus the corresponding TDMA frame by generating an interrupt to the MPU.

Each TDMA frame is associated with a parity bit that toggles when switching from the current frame to the next one (synchronously with the TPU_IT_FRAME interrupt occurrence). This bit is readable by the MPU through the TPU.REG_FRAME_PARITY[0] FRAME_PARITY bit.

Similarly, a parity bit is associated to the TPU scenario and is stored in the TPU.REG_PARITY_SCENARIO[0] PARITY_SCENARIO bit. This bit is updated on a TPU MOVE instruction only.

A comparison between the TDMA frame parity bit and the TPU scenario parity bit is done at the beginning of each new TDMA frame synchronously with the TPU_IT_FRAME interrupt. The result of this comparison is stored in the TPU.REG_IT_STAT[3] ITPAR_ERROR bit.

When a comparison fails, a parity error interrupt is generated to the MPU/DSP. The interrupt is:

- Edge-sensitive
- Maskable with the same mask bit as the one defined for the IT_FRAME interrupt
- Combined with the TPU page interrupt on the same physical channel

16.1.5 TPU Register Manual

16.1.5.1 Instance Summary

Table 16-3 lists the base address and address space for the TPU module.

Table 16-3. Instance Summary

Module Name	MPU Base Address	Size
TPU register	0xFFFF F000	2K bytes
TPU RAM	0xFFFF 9000	2K bytes

16.1.5.2 Module Register Mapping Summary

Table 16-4 and Table 16-5 list the TPU registers. Table 16-6 through Table 16-14 describe the register bits.

Table 16-4. TPU Register Accessible by the MPU

Register Name	Type	Register Width (Bits)	Offset
REG_TPU_CTRL	R/W	16	0x00
REG_IT_CTRL	R/W	16	0x02
REG_IT_STAT	R	16	0x04
REG_FRAME_PARITY	R	16	0x06
REG_TPU_OFFSET	R	16	0x0C
REG_TPU_SYNCHRO	R	16	0x0E

Table 16-5. TPU Register Accessible by the TPU Only

Register Name	Type	Register Width (Bits)	Offset
REG_IT_DSP_PG	W	16	0x10
REG_PARITY_SCENARIO	W	16	0x12

TPU.REG_IT_DSP_PG and TPU.REG_PARITY_SCENARIO are not accessible (read or write) by the MPU.

Table 16-6. TPU Control Register (REG_TPU_CTRL)

Address Offset	0x00		
Physical Address	MPU: 0xFFFF F000	**Instance**	TPU
Description	This register allows the MPU to reset the TPU and the TPU2OCP modules, check the TPU activity, and control the TPU communication buffer.		

Table 16-6. TPU Control Register (REG_TPU_CTRL) (continued)

Type	R/W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				FULL_WRITE	TPU_CK_ENABLE	TPU_WAIT	TPU_IDLE	TPU2OCP_RESET	MPU_RAM_ACCESS	Reserved	DSP_EN	Reserved	TPU_EN	TPU_PAGE	TPU_RESET

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Reserved	—	—
11	FULL_WRITE	Enables the MPU to write anywhere in the RAM even if the TPU is executing a scenario. Must be used in conjunction with MPU_RAM_ACCESS. See Table 16-7. 0: Full write access disabled 1: Full write access enabled **Warning:** Handle with care because there is no protection preventing the MPU from writing on the address read by the TPU.	R/W	0
10	TPU_CK_ENABLE	Enable the functional clock (TPU_FCLK1) of the TPU module 0: TPU clock disabled 1: TPU clock enabled	R/W	0
9	TPU_WAIT	WAIT or AT state of TPU 0: Other state 1: WAIT or AT state active	R	0
8	TPU_IDLE	Status of TPU scenario execution 0: TPU is in idle mode (SLEEP instruction) 1: TPU is running	R	0
7	TPU2OCP_RESET	Software reset for TPU2OCP module 0: No effect 1: Reset active	R/W	1
6	MPU_RAM_ACCESS	RAM read access. Must be used in conjunction with FULL_WRITE (see Table 16-7). 0: RAM read access not allowed to MPU 1: RAM read access allowed to MPU	R/W	0
5	Reserved	Reserved	—	—
4	DSP_EN	Enable TPU_IT_DSP generation on next TPU_IT_FRAME. Set by the MPU. Reset when TPU_IT_DSP occurs.	R/W	0
3	Reserved	Reserved	—	—
2	TPU_EN	Enables execution of the new scenario loaded in the TPU buffer at page TPU_PAGE. Set by the MPU. Reset by the TPU.	R	0
1	TPU_PAGE	Page of TPU buffer visible by TPU 0: Page 0 1: Page 1	R	0
0	TPU_RESET	Reset TPU module (except GSM time base) 0: No effect 1: Reset active	R/W	1

Table 16-7 compares the FULL_WRITE vs. MPU_RAM_ACCESS logic.

Table 16-7. FULL_WRITE vs. MPU_RAM_ACCESS Logic

MPU_RAM_ACCESS	FULL_WRITE	Mode
0	0	Write page access mode
0	1	Full write mode
1	X	Full read/write mode

Table 16-8. TPU Interrupt Control Register (REG_IT_CTRL)

Address Offset	0x02		
Physical Address	MPU: 0xFFFF F002	**Instance**	TPU
Description	This register allows the MPU to mask the frame interrupt (TPU_IT_FRAME), the page interrupt (TPU_IT_PAGE), and the DSP interrupt (TPU_IT_DSP) generations.		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												ITD_F	ITD_M	ITP_M	ITF_M

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	—	—
3	ITD_F	Force frame interrupt for DSP 0: Inactive 1: Active	R/W	0
2	ITD_M	Mask on frame interrupt for DSP 0x0: Unmask 0x1: Mask	R/W	1
1	ITP_M	Mask on page interrupt 0x0: Unmask 0x1: Mask	R/W	1
0	ITF_M	Mask on frame interrupt for MPU 0x0: Unmask 0x1: Mask	R/W	1

Table 16-9. TPU Interrupt Status Register (REG_IT_STAT)

Address Offset	0x04		
Physical Address	MPU: 0xFFFF F004	**Instance**	TPU
Description	This register informs the MPU of the origin of the interrupt.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												ITPAR_ERROR	ITD	ITP	ITF

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	—	—
3	ITPAR_ERROR	Parity error occurrence. Scenario executed in a bad frame. Active level is high.	R	0
2	ITD	DSP interrupt occurrence. Active level is low.	R	0
1	ITP	Page interrupt occurrence. Execution of a new scenario. Active level is low.	R	0
0	ITF	Frame interrupt occurrence. Active level is low.	R	0

Table 16-10. TPU Frame Parity Register (REG_FRAME_PARITY)

Address Offset	0x06		
Physical Address	MPU: 0xFFFF F006	**Instance**	TPU
Description	This register toggles when switching from the current frame to the next one.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															FRAME_PARITY

Bits	Field Name	Description	Type	Reset
15:1	Reserved	Reserved	—	—
0	FRAME_PARITY	Frame parity	R	0

Table 16-11. TPU Offset Register (REG_TPU_OFFSET)

Address Offset	0x0C		
Physical Address	MPU: 0xFFFF F00C	**Instance**	TPU
Description	This register allows setting the value of the OFFSET operand.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TPU_OfFSET							

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	—	—
12:0	TPU_OfFSET	Value of OFFSET operand	R	0x0000

Table 16-12. TPU Sychronization Register (REG_TPU_SYNCHRO)[(1)]

Address Offset	0x0E		
Physical Address	MPU: 0xFFFF F00E	**Instance**	TPU
Description	This register allows setting the value of the SYNCHRO operand.		
Type	R		

(1) The content of this register is not latched, so its value can be corrupted if MPU read access is simultaneous with a TPU write operation.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TPU_SYNCHRO							

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	—	—
12:0	TPU_SYNCHRO	Value of SYNCHRO operand	R	0x0000

Table 16-13. TPU DSP Interrupt Generation Register (REG_IT_DSP_PG)

Address Offset	0x10 (for a TPU MOVE instruction)		
Physical Address	MPU: N/A	**Instance**	TPU
Description	This register enables generation of a DSP interrupt (TPU_IT_DSP_PG) by executing a MOVE instruction to this register.		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															IT_DWSP_PG

Bits	Field Name	Description	Type	Reset
15:1	Reserved	Reads return 0s.	—	—
0	IT_DWSP_PG	DSP programmable interrupt occurrence. Active level is low.	W	1

Table 16-14. TPU Parity Scenario Register (REG_PARITY_SCENARIO)

Address Offset	0x12 (for a TPU MOVE instruction)		
Physical Address	MPU: N/A	**Instance**	TPU
Description	This register enables the setting of a parity bit for a TPU scenario by executing a MOVE instruction to this register.		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															PARITY_SCENARIO

Bits	Field Name	Description	Type	Reset
15:1	Reserved	Reserved	—	—
0	PARITY_SCENARIO	Parity of TPU scenario	W	0

16.2 TPU2OCP Module

16.2.1 TPU2OCP Module Overview

The TPU2OCP module is the interface between the TPU module and the DRP2, allowing the TPU to access the digital radio processor (DRP) internal memory space. The TPU2OCP module also includes a register to manage the gauging of the 32-kHz clock in the ULPD.

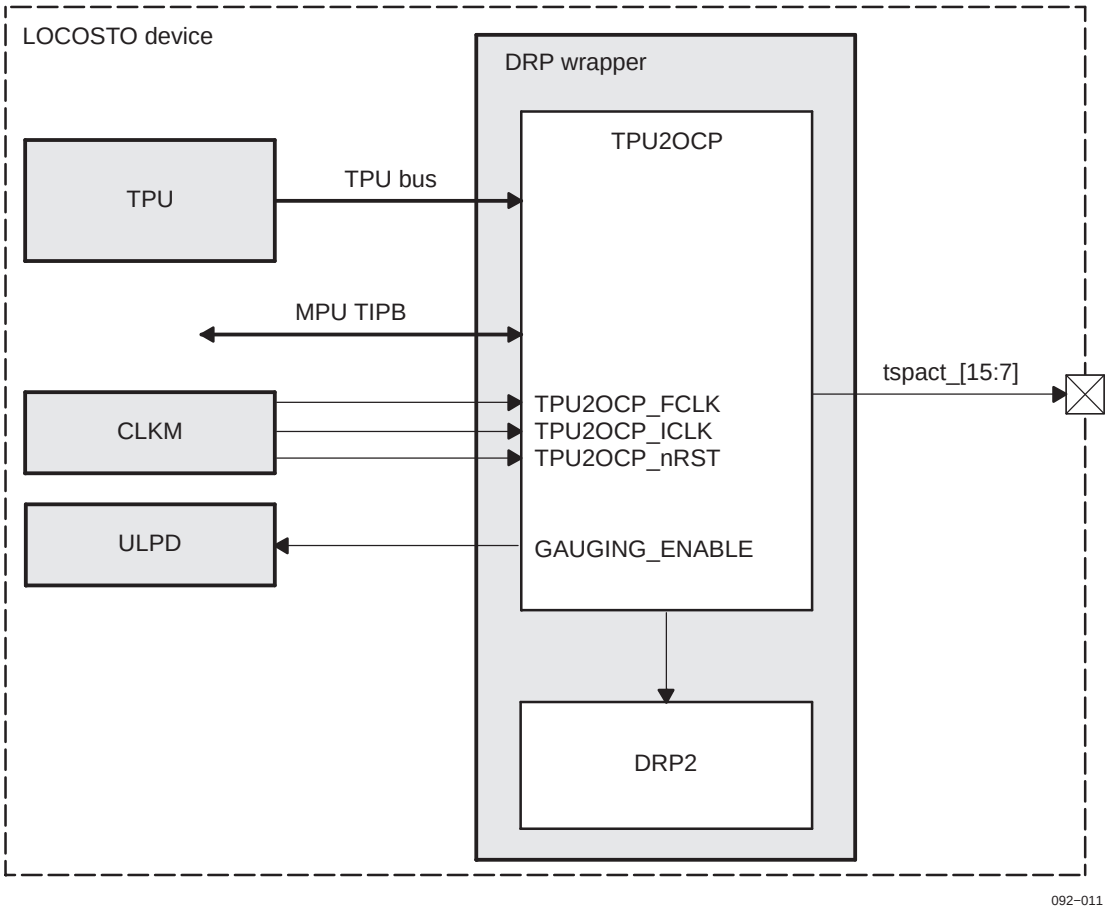
Note: For more information, see the *DRP Control Using the TPU Application Note (APN223)*.

The TPU2OCP module includes a parallel bit interface that monitors activation and deactivation of nine external signals (called tspact_i, with i=[15:7]) and seven internal signals (called TSPACT_i, with i=[6:0]) with a 1/4 GSM bit time accuracy.

The TPU controls all of these registers (write access only). The MPU accesses the registers for debugging (read access only).

Figure 16-11 shows an overview of the TPU2OCP module.

Figure 16-11. TPU2OCP Module Overview



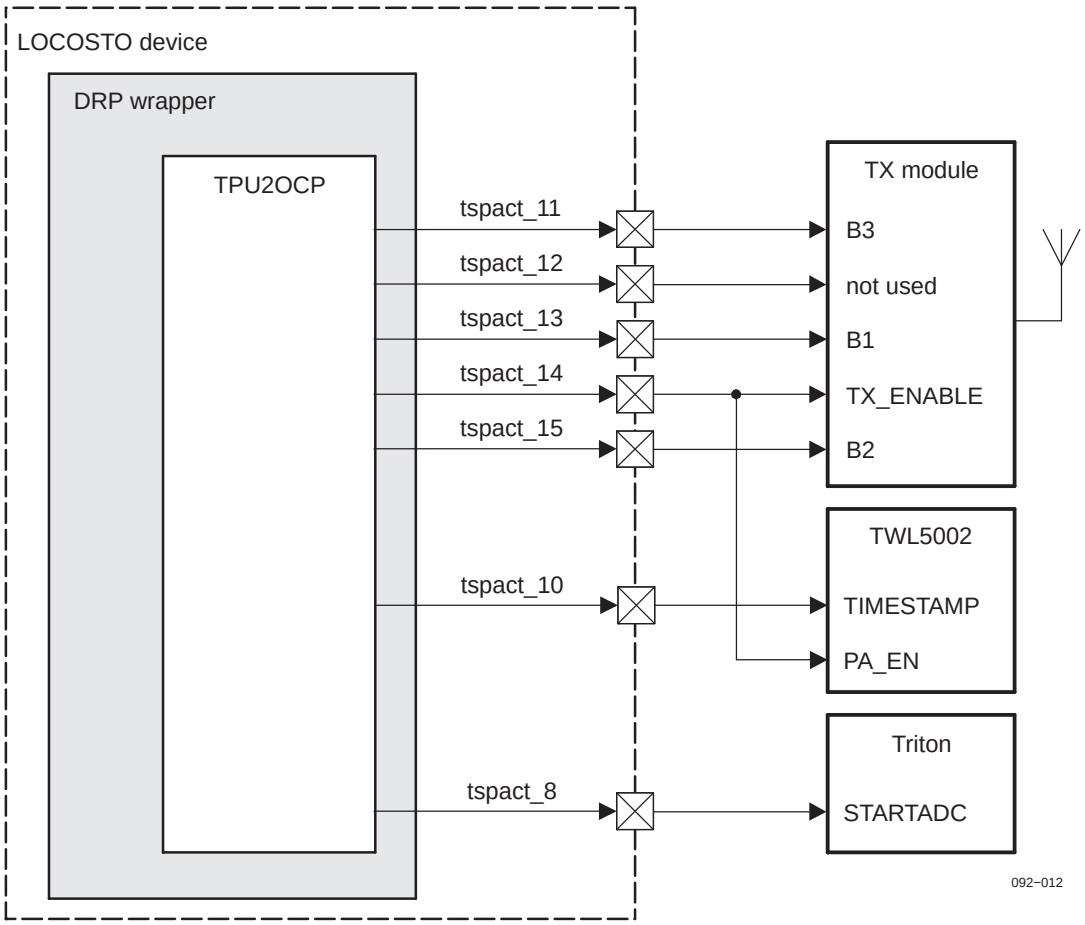
16.2.2 TPU2OCP Module Environment

The TPU2OCP external signals control external ICs that require high-accuracy signals.

16.2.2.1 Basic TPU2OCP Pins Connection

Figure 16-12 shows the basic connection of the TPU2OCP external tspact_i signals with the TX module for RF transmission, the TWL5002, and the TWL3029. Contact your TI representative for more information about these external ICs.

Figure 16-12. Module Interface Signals



Note: Figure 16-12 does not show tspact_7 and tspact_9.

16.2.2.2 Interface Description

Table 16-15 lists the nine TPU2OCP signals available at the LOCOSTO device boundary, depending on the pin multiplexing configuration.

Table 16-15. TPU2OCP External Signal Description

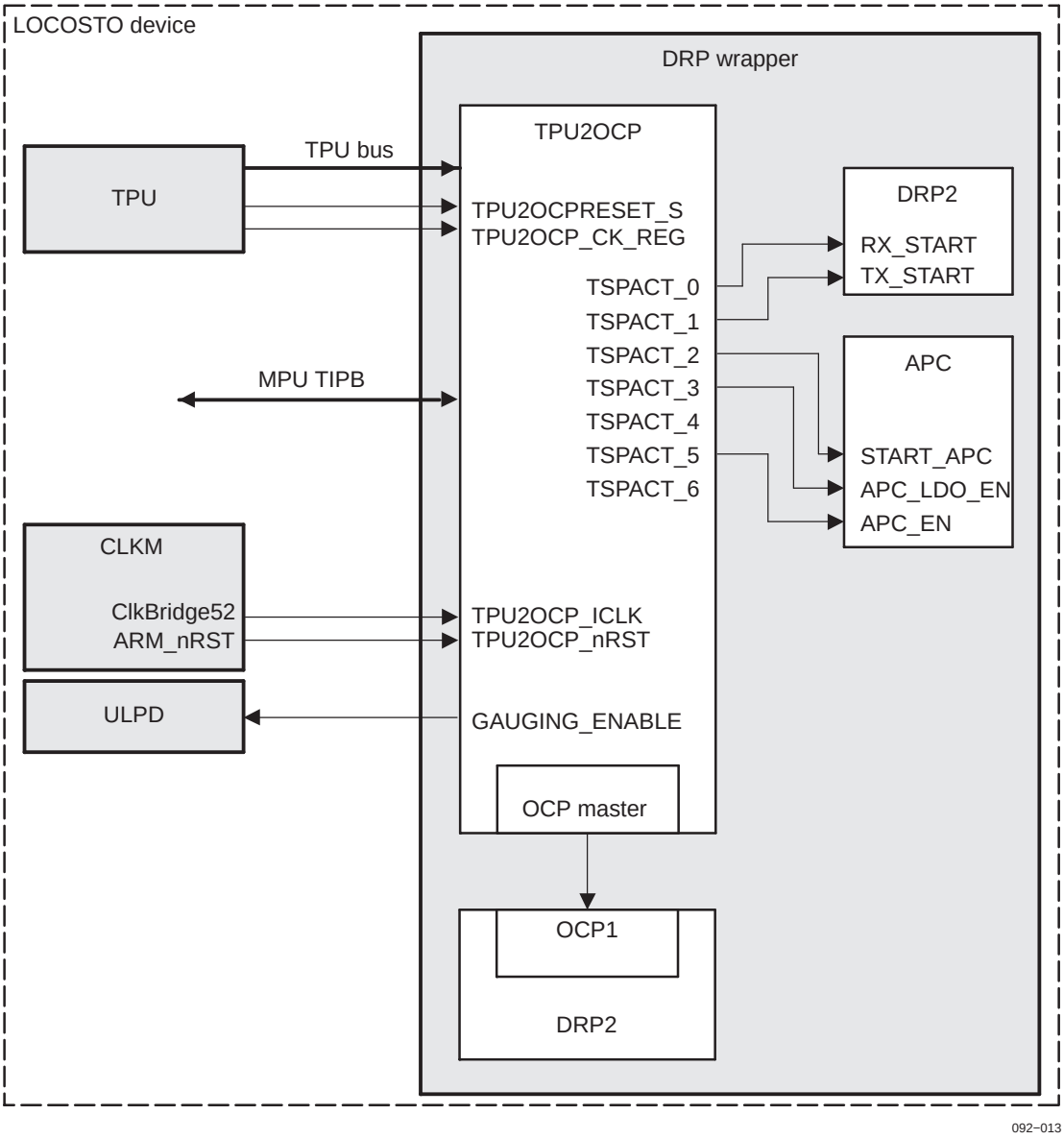
Signal Name	I/O(1)	Description	Reset Value
tspact_[15:7]	O	TPU2OCP external activation signals	0

(1) I=Input, O=Output

16.2.3 TPU2OCP Module Integration

This section describes the TPU2OCP integration inside the LOCOSTO device. Figure 16-13 shows the TPU2OCP internal connections with the TPU, the ULPD, the DRP2, and other submodules of the DRP wrapper.

Figure 16-13. TPU2OCP Integration



16.2.3.1 Clocking, Reset, and Power-Management Scheme

The TPU2OCP module receives its clocks and reset signals from the CLKM module (see Table 16-16).

Table 16-16. TPU2OCP Clocking and Resets

Module	Functional Clock	Interface Clock	Reset
TPU2OCP	Clk13	ClkBridge52	**Hardware:** ARM_nRST **Software:**TPU.REG_TPU_CTRL[7] TPU2OCP_RESET bit

16.2.3.1.1 Clocks

The TPU2OCP module is clocked with three interface clocks:

- TPU2OCP_CK_REG is the interface clock with the TPU module, runs at TPU bus clock speed (1/4 GSM bit, or 4.33/3 MHz), and is used to trigger access to some of the TPU2OCP module registers.

- TPU2OCP_ICLK is the interface clock with the MPU private TIPB, runs at 52 MHz, and is used to trigger access to some of the TPU2OCP module registers (access is limited to read-only for debug purposes).
- An interface clock with the DRP2 (through the OCP master interface) runs at 52 MHz and is used to access to the DRP2 memory space.

16.2.3.2 Hardware and Software Reset

16.2.3.2.1 Hardware Reset

A TPU2OCP module hardware reset is performed by the ARM_nRST reset signal. For more information on these signals, see Chapter 6, *Power, Reset, and Clock Management*.

16.2.3.2.2 Software Reset

The TPU module performs a TPU2OCP module software reset (through the TPU2OCPRESET_S signal). To reset the TPU2OCP module, set the TPU.REG_TPU_CTRL[7] TPU2OCP_RESET bit to 1 (reset value).

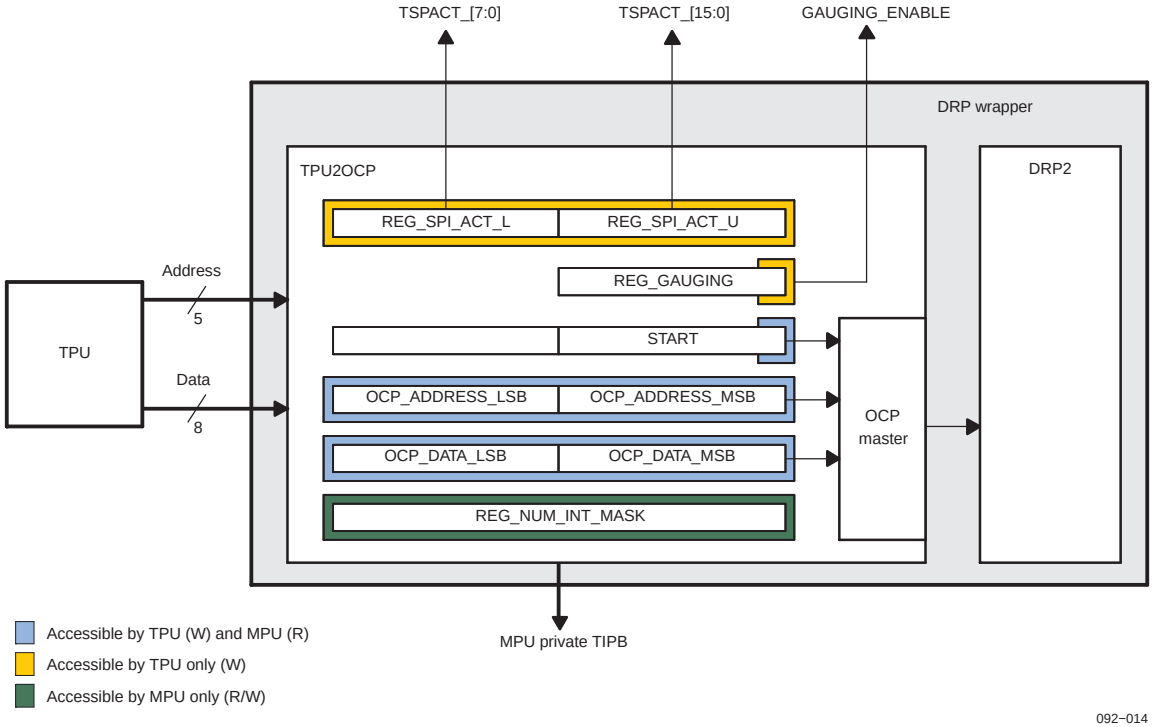
CAUTION

The TPU2OCP_RESET bit is not cleared automatically. Software must clear this bit after at least 1/4 GSM bit time before using the TPU2OCP module to deactivate the reset.

16.2.4 TPU2OCP Module Functional Description

Figure 16-14 shows the architecture of the TPU2OCP module.

Figure 16-14. TPU2OCP Block Diagram



The TPU2OCP contains the following:

- Two 8-bit registers to control the parallel interface: TPU2OCP.REG_SPI_ACT_U and TPU2OCP.REG_SPI_ACT_L
- One 1-bit register to control the GAUGING_ENABLE sig
- One 1-bit TPU2OCP.START register, two 8-bit address registers (TPU2OCP.OCP_ADDRESS_LSB and TPU2OCP.OCP_ADDRESS_ MSB) and two 8-bit data registers (TPU2OCP.OCP_DATA_LSB and TPU2OCP.OCP_DATA_MSB) to control the OCP master interface.
- One 16-bit register TPU2OCP.REG_NUM_INT_MASK to mask a certain number of interrupt coming from the DRP2. Only the six least-significant bits (LSBs) are available at the TPU2OCP module boundary.

All of these registers are mapped to the TPU address space (Table 18-98, *Module Register Mapping Summary*, details register mapping for TPU access). The TPU uses a 5-bit address bus and an 8-bit data bus to write to these registers. The TPU can write to these registers (with a MOVE instruction) but cannot read from these registers.

The MPU can read the TPU2OCP.START, TPU2OCP.OCP_ADDRESS_xSB, and TPU2OCP.OCP_DATA_xSB registers through the MPU private TIPB for debugging, and can write to or read from the TPU2OCP.REG_NUM_INT_MASK register.

16.2.4.1 Parallel Bit Interface

The parallel bit interface directly monitors 16 output signals, TSPACT_i with i = [15:0]. The interface can independently control the activation and deactivation of each output signal with a 1/4 GSM bit time accuracy (923 ns).

The interface is based on two 8-bit registers: TPU2OCP.REG_SPI_ACT_U and TPU2OCP.REG_SPI_ACT_L, which are loaded by the TPU module with a MOVE instruction. TPU2OCP.REG_SPI_ACT_U monitors the TSPACT_[15:8] signals, and TPU2OCP.REG_SPI_ACT_L monitors the TSPACT_[7:0] signals.

As soon as 1 bit of these registers is updated, the corresponding signal is activated or deactivated (activation is defined as high-level, and deactivation as low-level).

16.2.4.1.1 External TSPACT_i

The external TSPACT_i signals (band select, PA enable, etc.) are stored in the 8-bit register TPU2OCP.REG_SPI_ACT_U and in the upper bit 7 of the TPU2OCP.REG_SPI_ACT_L register, as defined in Table 16-17.

Table 16-17. TSPACT External Signals

REG_SPI_ACT_U Bit	Description	REG_SPI_ACT_L Bit	Description
7	TSPACT_15: B2 input of the TX module	7	TSPACT_7: Not used
6	TSPACT_14: TX_ENABLE input of the TX module and PA_EN input of the TWL5002		
5	TSPACT_13: B1 input of the TX module		
4	TSPACT_12: Not used		
3	TSPACT_11: B3 input of the TX module		
2	TSPACT_10: TIMESTAMP input of the TWL5002		
1	TSPACT_9: Not used		
0	TSPACT_8: START_ADC input of TWL3029		

16.2.4.1.2 Internal TSPACT_i

The internal TSPACT_i signals are stored in the lower 7 bits of the 8-bit TPU2OCP.REG_SPI_ACT_L register, as defined in Table 16-18.

Table 16-18. ISPACT Internal Signals

Bit	Description
6	TSPACT_6: Not used
5	TSPACT_5: APC_EN: Enable the automatic power control (APC) module.
4	TSPACT_4: Not used
3	TSPACT_3: APC_LDO_EN: Enable the APC internal low dropout (LDO).
2	TSPACT_2: START_APC: Start of the APC module
1	TSPACT_1: TX_START: TX start of the DRP2
0	TSPACT_0: RX_START: RX start of the DRP2

16.2.4.2 OCP Master Interface

This interface programs DRP2 registers and is connected to the DRP2 OCP slave1 interface (OCP1). Whenever the bit 0 of the START register is set to 1, the OCP master places the TPU2OCP.OCP_DATA_xSB and TPU2OCP.OCP_ADDRESS_xSB registers content on respective buses and issues a write command.

Table 16-19 details the memory space accessible through the OCP master interface.

Table 16-19. OCP Master Interface Memory Mapping

OCP_ADDRESS_MSB and OCP_ADDRESS_LSB		Memory Address Accessed		Size (byte)	Access
Register Start Value	Register End Value	Memory Start Address	Memory End Address		
0x0000	0x3FFF	0xFFFF 0000	0xFFFF 3FFF	16K	16-bit W
0x4000	0x47FF	0xFFFF 4000	0xFFFF 47FF	2K	16-bit W
0x4800	0x4FFF	0xFFFF 4800	0xFFFF 4FFF	2K	16-bit W
0x5000	0XFFFF		Not allowed		

16.2.4.3 Gauging Bit Interface

The GAUGING_ENABLE signal goes to the ULPD and is internally ORed with the ULPD.GAUGING_CTRL_REG[0] GAUGING_EN bit of the ULPD. Thus, the TPU can launch the gauging process to gauge the 32-kHz clock by setting the TPU2OCP.REG_GAUGING[0] GAUGING_ENABLE bit.

For more information about gauging the 32-kHz clock, see Chapter 6, *Power, Reset, and Clock Management*.

16.2.5 Programming Model

16.2.5.1 Reset

Before accessing or using the TPU2OCP module, the local host must reset it as follows:

1. The MPU sets the TPU.REG_TPU_CTRL[7] TPU2OCP_RESET. Software must wait at least 1/4 GSM bit time to ensure that the reset is effective.
2. The MPU must reset the TPU2OCP_RESET bit to deactivate the reset of the TPU2OCP module.

16.2.6 TPU2OCP Register Manual

16.2.6.1 Instance Summary

The TPU2OCP module registers are accessible by the TPU (write-only) through a dedicated bus and by the MPU through the MPU private TIPB. Table 16-20 lists the base address and address space of the TPU2OCP module for MPU access.

Table 16-20. Instance Summary

Module Name	MPU Base Address	Size
TPU2OCP	0xFFFE 0800	2K bytes

16.2.6.2 Module Register Mapping Summary

Table 16-21 lists the TPU2OCP registers offsets for TPU access. All of these registers are write-only. The addresses are coded on 5 bits (see Section 16.1.4.2.5, *Microinstruction for Data Transfer*).

Table 16-21. TPU2OCP Registers Offsets for TPU Access

Register Name	Type	Register Width (Bits)	TPU Offset
START	W	8	0x01
OCP_DATA_LSB	W	8	0x04
OCP_DATA_MSB	W	8	0x05
REG_SPI_ACT_L	W	8	0x06
REG_SPI_ACT_U	W	8	0x07
OCP_ADDRESS_LSB	W	8	0x0A
OCP_ADDRESS_MSB	W	8	0x0B
REG_GAUGING	W	8	0x11

Table 16-22 lists the TPU2OCP registers offsets for MPU access. TPU2OCP.START, TPU2OCP.OCP_ADDRESS, and TPU2OCP.OCP_DATA are read-only registers and reflect the values written in the TPU2OCP.START, TPU2OCP.OCP_ADDRESS_xSB, and TPU2OCP.OCP_DATA_xSB registers. The TPU2OCP.REG_NUM_INT_MASK register is read/write to the MPU.

Table 16-22. TPU2OCP Registers Offsets for MPU Access(1)

Register Name	Type	Register Width (Bits)	TPU Offset
START	R	16	0x00
OCP_ADDRESS	R	16	0x02
OCP_DATA	R	16	0x04
REG_NUM_INT_MASK	R/W	16	0x06

(1) OCP_ADDRESS = OCP_ADDRESS_LSB && OCP_ADDRESS_MSB
OCP_DATA = OCP_DATA_LSB && OCP_DATA_MSB

16.2.6.3 Register Description

16.2.6.3.1 TPU Access Registers

Table 16-23 through Table 16-30 describe the TPU2OCP registers, which are configurable by the TPU with a MOVE instruction.

Table 16-23. TPU2OCP Start Register Bit Description (START)

Address Offset	0x01 (for a TPU MOVE instruction)		
Physical Address	MPU: N/A	**Instance**	TPU2OCP
Description	This register is dedicated to the activation of the OCP master interface.		
Type	W		

7	6	5	4	3	2	1	0
Reserved							START

Bits	Field Name	Description	Type	Reset
7:1	Reserved	Reserved	—	0x00
0	START	OCP master Interface activation When set to 1, the OCP master interface executes a write access to the DRP2 internal memory. This bit is auto-cleared.	W	0

Table 16-24. TPU2OCP Data LSB Register Bit Description (OCP_DATA_LSB)

Address Offset	0x04 (for a TPU MOVE instruction)		
Physical Address	MPU: N/A	**Instance**	TPU2OCP
Description	This register contains the lower 8 bits of the data to write through the OCP master interface.		
Type	W		

7	6	5	4	3	2	1	0
DATA_LSB							

Bits	Field Name	Description	Type	Reset
7:0	DATA_LSB	Lower 8 bits of data to write	W	0x00

Table 16-25. TPU2OCP Data MSB Register Bit Description (OCP_DATA_MSB)

Address Offset	0x05 (for a TPU MOVE instruction)		
Physical Address	MPU: N/A	**Instance**	TPU2OCP
Description	This register contains the upper 8 bits of the data to write through the OCP master interface.		
Type	W		

7	6	5	4	3	2	1	0
DATA_MSB							

Bits	Field Name	Description	Type	Reset
7:0	DATA_MSB	Upper 8 bits of data to write	W	0x00

Table 16-26. TPU2OCP Parallel Signal Activation LSB Register Bit Description (REG_SPI_ACT_L)

Address Offset	0x06 (for a TPU MOVE instruction)		
Physical Address	MPU: N/A	**Instance**	TPU2OCP
Description	This register controls independently the activation and deactivation of the TSPACT_i signal, i = [7:0] with a 1/4 GSM time accuracy.		
Type	W		

7	6	5	4	3	2	1	0
REG_SPI_ACT_L							

Bits	Field Name	Description	Type	Reset
7:0	REG_SPI_ACT_L	Activation of the TSPACT_[7:0] signals	W	0x00

Note: TSPACT_[6:0] are internal signal only. They are not mapped at the LOCOSTO device boundary.

Table 16-27. TPU2OCP Parallel Signal Activation MSB Register Bit Description (REG_SPI_ACT_U)

Address Offset	0x07 (for a TPU MOVE instruction)		
Physical Address	MPU: N/A	**Instance**	TPU2OCP
Description	This register controls independently the activation and deactivation of the TSPACT_i signal, i = [15:8] with a 1/4 GSM time accuracy.		
Type	W		

7	6	5	4	3	2	1	0
REG_SPI_ACT_U							

Bits	Field Name	Description	Type	Reset
7:0	REG_SPI_ACT_U	Activation of the TSPACT_[15:8] signals	W	0x00

Table 16-28. TPU2OCP OCP Address LSB Register Bit Description (OCP_ADDRESS_LSB)

Address Offset	0x0A (for a TPU MOVE instruction)		
Physical Address	MPU: N/A	**Instance**	TPU2OCP
Description	This register contains the lower 8 bits of the address to access through the OCP master interface.		
Type	W		

7	6	5	4	3	2	1	0
OCP_ADDRESS_LSB							

Bits	Field Name	Description	Type	Reset
7:0	OCP_ADDRESS_LSB	Lower 8 bits of the address to access	W	0x00

Table 16-29. TPU2OCP OCP Address MSB Register Bit Description (OCP_ADDRESS_MSB)

Address Offset	0x0B (for a TPU MOVE instruction)		
Physical Address	MPU: N/A	**Instance**	TPU2OCP
Description	This register contains the upper 8 bits of the address to access through the OCP master interface.		
Type	W		

7	6	5	4	3	2	1	0
OCP_ADDRESS_MSB							

Bits	Field Name	Description	Type	Reset
7:0	OCP_ADDRESS_MSB	Upper 8 bits of the address to access	W	0x00

Table 16-30. TPU2OCP Gauging Control Register Bit Description (REG_GAUGING)

Address Offset	0x11 (for a TPU MOVE instruction)		
Physical Address	MPU: N/A	**Instance**	TPU2OCP
Description	This register controls the activation of the GAUGING_ENABLE signal sent to the ULPD.		
Type	W		

7	6	5	4	3	2	1	0
Reserved							GAUGING_ENABLE

Bits	Field Name	Description	Type	Reset
7:1	Reserved	Reserved	—	0x00
0	GAUGING_ENABLE	Activation of the GAUGING_ENABLE signal sent to the ULPD The TPU must clear this bit to deactivate the GAUGING_ENABLE signal; this bit is not cleared automatically.	W	0

16.2.6.3.2 MPU Access Registers

Table 16-31 through Table 16-34 describe the TPU2OCP registers that are accessible by the MPU (read-only).

Table 16-31. TPU2OCP Start Register Bit Description (START)

Address Offset	0x00		
Physical Address	MPU: 0xFFFE 0800	**Instance**	TPU2OCP
Description	This register reflects the START bit status configured by the TPU.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															START

Bits	Field Name	Description	Type	Reset
15:1	Reserved	Reserved	—	0x0000
0	START	OCP master interface executes a write access when this bit is set to 1. This bit is auto-cleared.	R	0

Table 16-32. TPU2OCP OCP Address Register Bit Description (OCP_ADDRESS)

Address Offset	0x02		
Physical Address	MPU: 0xFFFE 0802	**Instance**	TPU2OCP
Description	This register contains the 16 bits of the address to access through the OCP master interface.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCP_ADDRESS															

Bits	Field Name	Description	Type	Reset
15:0	OCP_ADDRESS	Address to access	R	0x0000

Table 16-33. TPU2OCP Data Register Bit Description (OCP_DATA)

Address Offset	0x04		
Physical Address	MPU: 0xFFFE 0804	**Instance**	TPU2OCP
Description	This register contains the 16 bits of the data to write through the OCP master interface.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCP_ADDRESS															

Bits	Field Name	Description	Type	Reset
15:0	OCP_ADDRESS	Address to access	R	0x0000

Table 16-34. TPU2OCP Interrupt Mask Register Bit Description (REG_NUM_INT_MASK)

Address Offset	0x06		
Physical Address	MPU: 0xFFFE 0806	**Instance**	TPU2OCP
Description	This register indicates the number of masked RX interrupt coming from the DRP2.		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								INT_MASK							

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Reserved	—	0x000
5:0	INT_MASK	Number of RX interrupts masked	R/W	0x10

Camera Interface

This chapter describes the LOCOSTO camera interface.

Topic **Page**

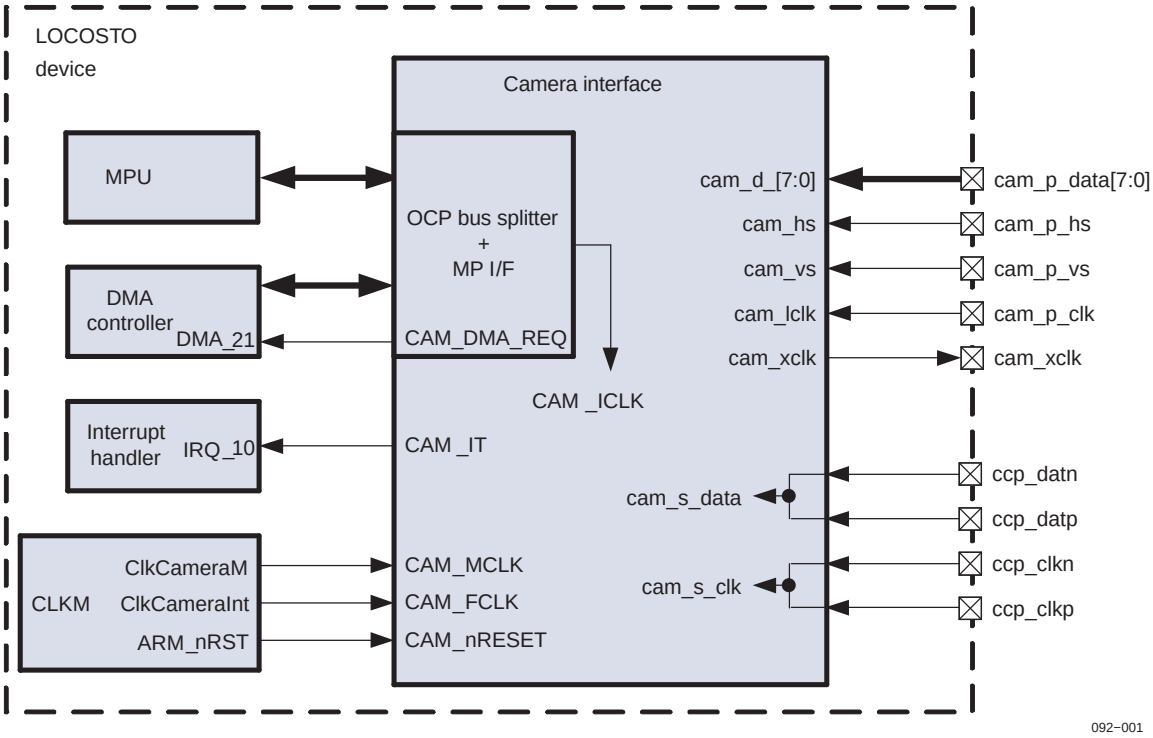
17.1 Camera Interface Overview

17.1.1 Introduction

The camera interface of the LOCOSTO device allows it to interface with a variety of external image sensors. It stores image data in a FIFO and can generate direct memory access (DMA) requests. Figure 17-1 shows the camera interface overview.

Note: The camera interface is not available on the LOCOSTO Lite device.

Figure 17-1. Camera Interface Overview



17.1.2 Features

The main part of the camera interface is the camera core, which ensures image sensor data format detection and transmission to the system. The other part of the camera interface is the wrapper, which allows the camera core to communicate with other modules of the LOCOSTO device (DMA, microprocessor unit [MPU] subsystem, and so forth) through open-core protocol (OCP) buses. No details are needed for this wrapper, so only the camera core is considered in this chapter.

The camera core module supports two image-sensor interfaces, serial and parallel. The interface selection is exclusive; only one at a time can be set.

Both serial and parallel interfaces have the following:

- 128 × 32-bit FIFO
  - DMA access with enable/disable capability
  - Configurable FIFO trigger level
  - One interrupt line supporting programmable interrupts
  - Start/end of line

  - Start/end of image
  - FIFO overflow and data threshold status
- Byte swapping capability/32-bit word packet supply

17.1.3 Description

17.1.3.1 Serial Interface

The compact camera port (CCP) interface is a unidirectional differential interface that connects to a camera sensor in compliance with MIPI Rev1.0. The physical link is the sub-LVDS differential (clock and data) type for low EMI supporting external clocking signals up to 208 MHz. The CCP bit stream has the following embedded synchronization signals:

- Frame start
- Line start
- Frame end
- Line end

Table 17-1 lists the image data formats supported by the CCP module.

Table 17-1. CCP Image Data Format

Data Type	Format
YUV4:2:2 image data	YUV422
YUV4:2:0 image data	YUV420
RGB888 image data	RGB888
RGB565 image data	RGB565
RGB444 image data	RGB444
Raw Bayer 8-bit image data	RAW8
Raw Bayer 10-bit image data	RAW10
Raw Bayer 12-bit image data	RAW12
JPEG 8-bit data FSP	JPEG8FSP
JPEG 8-bit data	JPEG8

17.1.3.2 Parallel Interface

This 8-bit interface includes camera horizontal/vertical synchronization signals, a camera reference clock, and a pixel-capture clock operating up to 52 MHz. This interface provides a camera reference clock (CAM_XCLK) to the camera sensor based on on-chip APLL (48 MHz) or MPU (52 MHz) clock sources.

The interface supports two operating modes:

- BT656: ITU-R BT656-compatible parallel interface. This BT656 block extracts the start-of-active video (SAV), the end-of-active video (EAV), and the image data from the 8-bit data flow.
- NoBT: General parallel interface with camera horizontal/vertical synchronization. This NoBT detects valid data, and includes the reference clock and the pixel-capture clock.

The camera interface complies with the open OCP v2.2 (not backward-compatible with v2.1). It has two slave interfaces:

- OCP slave port I is used to configure the camera core module.
- OCP slave port II is used by the DMA controller to read data from the FIFO.

Data words are stored in a FIFO buffer and sent over the 32-bit OCP slave II bus under DMA or MPU control. These data can be read by the host (DMA or MPU) using burst accesses to get the best performance.

The general-purpose I/O pins can also be used to interface with the camera sensor:

- The gpio_4 pin (PU at reset) can be used for nEN for the level shifter or for the camera nReset; the gpio_2 pin (PD at reset and main peripheral reset) can be used for EN for the level shifter or for the camera reset (for more information on the reset signal, see Chapter 6, *Power, Reset, and Clock Management*).
- The gpio_17 pin can be connected to the CMOS sensor power-down (PWRDN) signal to save power. Any other GPIO pin can be used for the PWRDN signal.

For more information about the configuration, see Chapter 18, *Configuration*.

Note: The LCD and the camera are multiplexed. The APN212 application note includes an example of how to configure camera and LCD at the same time.

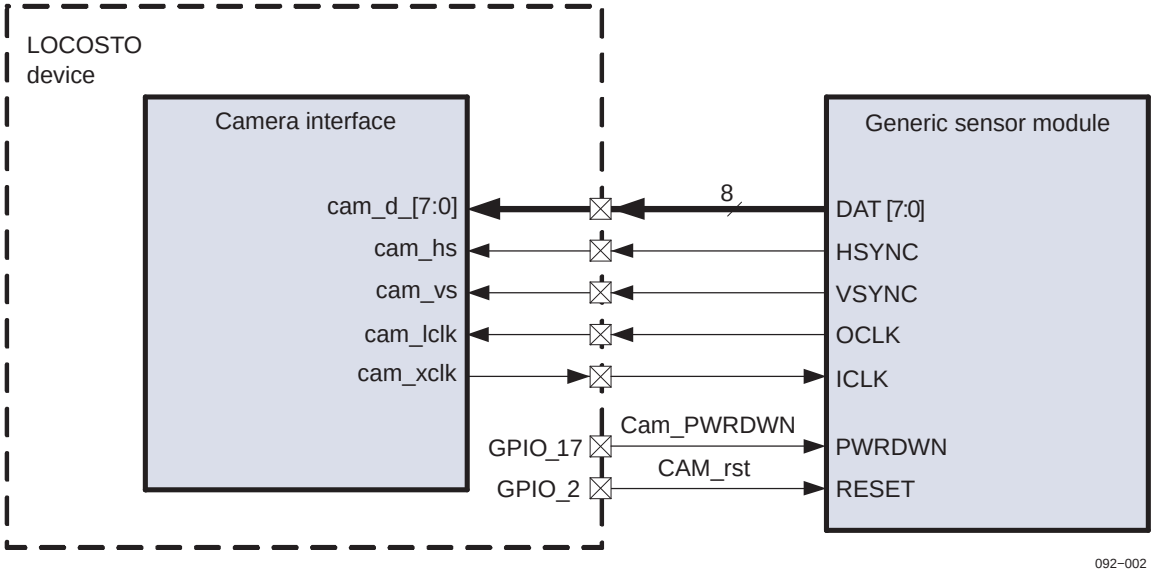
17.2 Camera Interface Environment

The camera interface can be used in three different configurations (generic parallel interface, ITU-R BT.656 parallel interface, and CCP serial interface), as shown in Figure 17-2 through Figure 17-4.

Note: GPIO_2 and GPIO_17 for sensor reset and power-down, respectively, are used as examples in these configurations. There are other possibilities.

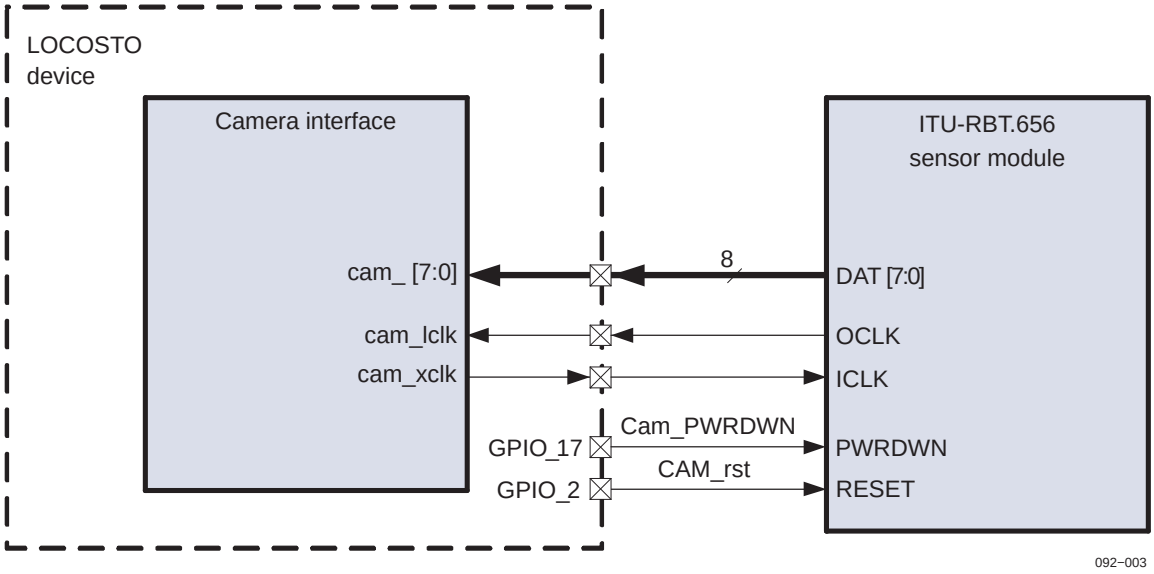
17.2.1 Generic Parallel Interface

Figure 17-2. Parallel Camera Interface in Generic Configuration



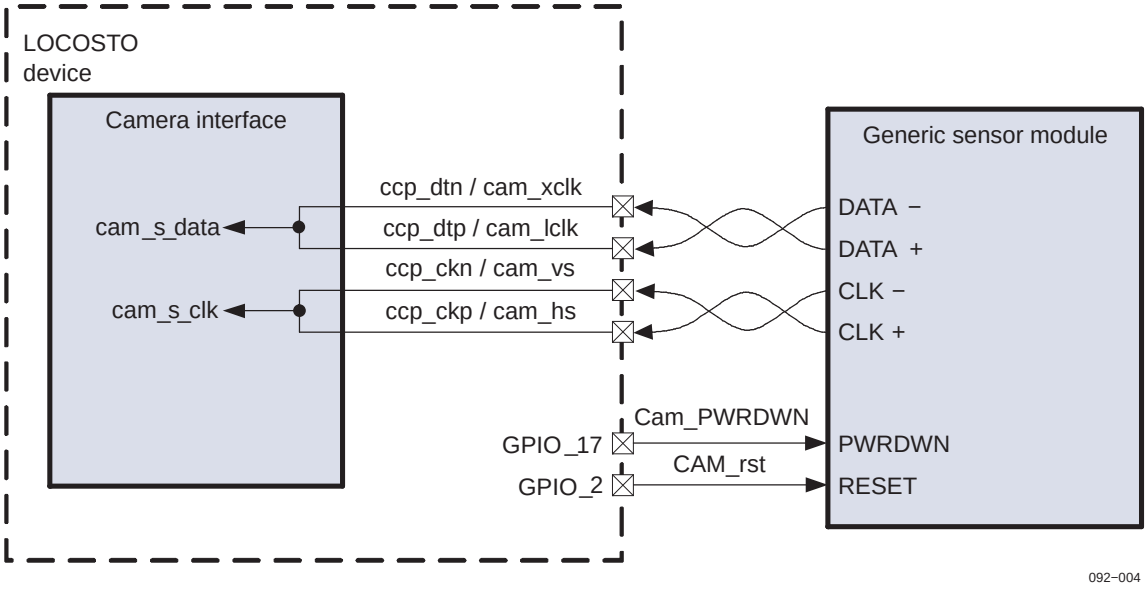
17.2.2 ITU-R BT.656 Parallel Interface

Figure 17-3. Parallel Camera Interface in ITU-R BT.656 Configuration



17.2.3 CCP Serial Interface

Figure 17-4. CCP Serial Interface Configuration

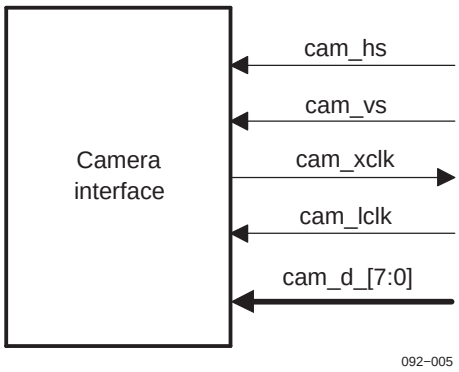


17.2.4 Generic Parallel Camera Interface Configuration

17.2.4.1 Generic Parallel Camera Interface Signals

shows the camera signals of the generic parallel configuration.

Figure 17-5. Signals for the Generic Parallel Camera Interface



17.2.4.2 Generic Parallel Camera Interface Signal Description

Table 17-2 describes the I/O signals of the generic parallel camera interface.

Table 17-2. Generic Parallel Camera Interface

Signal Name	I/O[1]	Description	Value at Reset
cam_hs	I	Line trigger input signal	N/A
cam_vs	I	Frame trigger input signal	N/A
cam_xclk	O	External clock for the image sensor module	0
cam_lclk	I	Latch clock for the parallel input data	N/A
cam_d_[7:0]	I	Input data bits 0 to 7	N/A

(1) I = Input,O = Output

17.2.4.3 Generic Parallel Camera Interface Protocol and Data Format

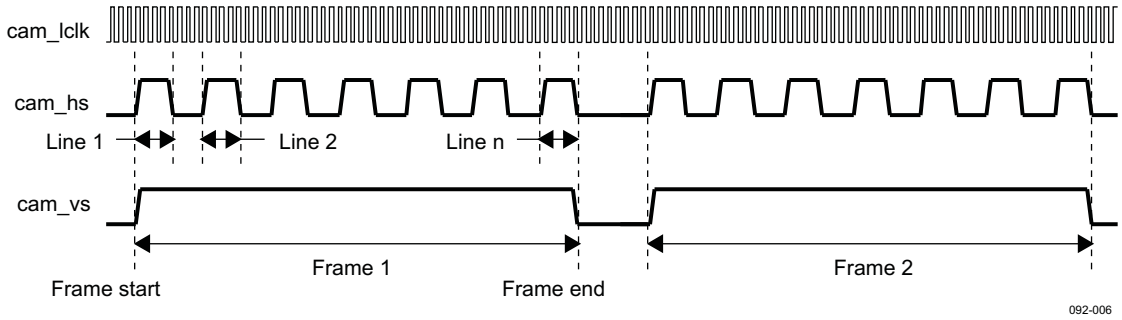
This configuration is also called parallel NoBT.656 because it uses all BT.656 signals, plus the cam_hs and cam_vs signals, to recognize valid data. In this configuration, no assumptions are made about the data format of pixels, but the dynamic is limited to 8 bits (data can be pure luminance for black and white sensor, RGB444, Bayer RGB, etc.).

The pixel data is presented on cam_d, and the data bus is sampled for every cam_lclk rising edge (or falling edge, depending on the configuration of cam_lclk polarity). For more information, see Section 17.4, *Camera Interface Functional Description*).

Active pixels are identified by a combination of two additional timing signals: horizontal synchronization (cam_hs) and vertical synchronization (cam_vs). During the image sensor readout, these signals define when a row of valid data begins and ends and when a frame starts and ends.

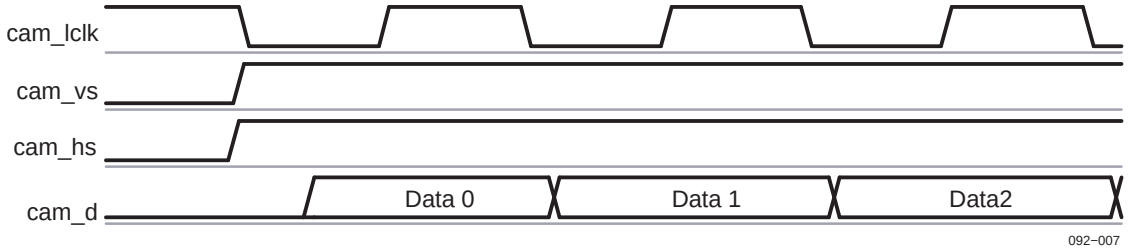
Figure 17-6 and Figure 17-7 show the frame and data timing based on the synchronization signals in the parallel NoBT configuration.

Figure 17-6. Synchronization Signals and Frame Timing in NoBT Protocol



Note: The clock cam_lclk provided by the camera sensor must continue to run during blanking periods (cam_hs and cam_vs inactive). A minimum of 10 clock cycles is required between consecutive cam_vs active frames (frame n and frame n+1) for proper operation when the line is not a multiple of 12 bytes; otherwise, 1 clock cycle is enough to detect cam_vs and work properly.

Figure 17-7. Synchronization Signals and Data Timing in NoBT Protocol

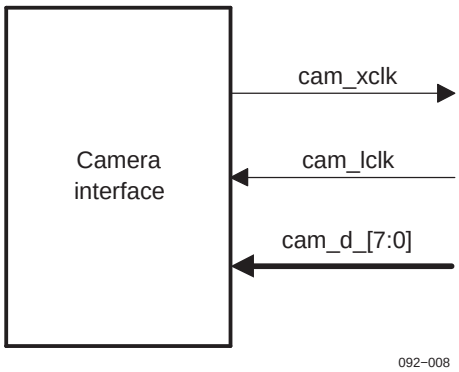


17.2.5 ITU-R BT.656 Parallel Camera Interface Configuration

17.2.5.1 ITU-R BT.656 Parallel Camera Interface Signals

Figure 17-8 shows the possible interfaces of the camera core in the ITU-R BT.656 parallel configuration.

Figure 17-8. Signals of the ITU-R BT.656 Parallel Camera Interface



17.2.5.2 ITU-R BT.656 Parallel Camera Interface Signal Description

Table 17-3 describes the BT.656 parallel camera interface I/O.

Table 17-3. ITU-R BT.656 Parallel Camera Interface I/O Description

Signal Name	I/O(1)	Description	Value at Reset
cam_xclk	O	External clock for the image-sensor module	0
cam_d_[7:0]	I	Input data bits 0 to 7	Unknown
cam_lclk	I	Latch clock for parallel input data	Unknown

(1) I = Input, O = Output

17.2.5.3 ITU-R BT.656 Camera Interface Protocol and Data Format

The camera interface supports data in ITU-R BT.656 format.

The ITU-R BT.656 standard specifies a method for transferring YUV422 data over an 8-bit interface.

In BT.656, data words in which the 8 most-significant bits (MSBs) are all set to 1 or all set to 0 are reserved. Only 254 of the possible 256 8-bit word values are used to represent signal values.

The data is multiplexed in the following order: Cb_0, Y_0, Cr_0, Y_1, Cb_2, Y_2, Cr_2, Y_3, etc., where the byte sequence Cb2n, Y2n, Cr2n refers to cosited (coincident) luminance and chroma samples and the following byte (Y2n+1) corresponds to the next luminance sample. Chroma samples are cosited with alternate luminance samples.

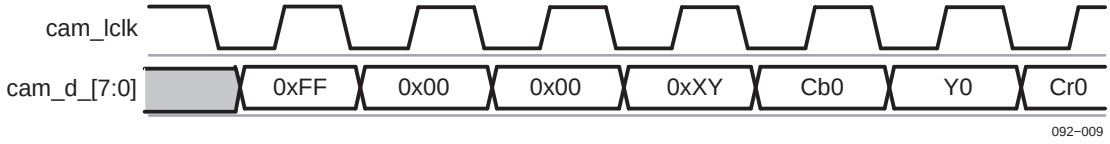
The BT.656 protocol uses a unique timing reference signal embedded in the video stream. Synchronization signals cam_hs and cam_vs are not required. This reduces the number of wires required for a BT.656 video interface.

There are two timing reference codes. The SAV reference code precedes each video data block, and the EAV code follows each video block. Each timing reference signal consists of a 4-byte sequence in the following hexadecimal format: 0xFF 00 00 XY. The first 3 bytes are a fixed preamble. For more information, see the ITU-R BT.656 specification.

The fourth byte contains information about field identification (F), blanking (V), and Start of Active Video (SAV)/End of Active Video (EAV) information (H), plus 4 parity bits that are calculated as a function of F, V, and H.

Figure 17-9 shows the data timing with SAV and EAV synchronization signals.

Figure 17-9. Data Timing in ITU-R BT, 8-Bit Case



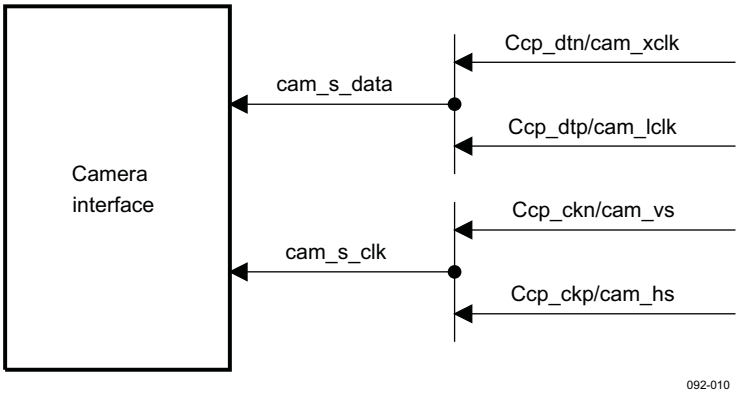
Note: The APN212 application note includes a sample environment composed of the Agilent ADCM-2700 CMOS sensor and the Phillips LPH8754-1 LCD.

17.2.6 CCP Serial Camera Interface Configuration

17.2.6.1 CCP Serial Camera Interface Signals

Figure 17-10 shows all of the camera interface signals in the CCP serial configuration.

Figure 17-10. Signals of the CCP Serial Camera Interface



17.2.6.2 CCP Serial Camera Interface Signal Description

Table 17-4 describes the CCP serial configuration interface.

Table 17-4. CCP Serial Camera Interface I/O Description

Signal Name	I/O(1)	Description	Value at Reset
cam_s_data	I	Serial data input	N/A
cam_s_clk	I	Serial clock input	N/A

(1) I = Input, O = Output

Sub-LVDS cells provide the CCP differential signal conversion. On the LOCOSTO device, only four dedicated pads are sub-LVDS pins. The two pairs of sub-LVDS pins are as follows:

- ccp_dtn./cam_xclk and ccp_dtp/cam_lclk pins (balls A5 and C6)
- ccp_ckn/cam_vs and ccp_ckp/cam_hs pins (balls E7 and F8)

17.2.6.3 CCP Serial Camera Interface Protocol and Data Format

The CCP is a serial interface to an image sensor. Inputs from the camera are serial input data (cam_s_data) and a clock signal (cam_s_clk).

The CCP protocol uses four synchronization codes embedded in the serial bitstream:

- The frame-start code (FSC) identifies the start of a new frame.

- The line-start code (LSC) identifies the start of a new line.
- The line-end code (LEC) identifies the end of a line. It is received for every line, except for the last line, which ends with a frame-end code.
- The frame-end code (FEC) identifies the end of the last line and the end of the current frame.

The CCP interface has several image data operating modes, listed in Table 17-5.

Table 17-5. CCP Image Data Operating Modes

Mode	Description
YUV422	YUV4:2:2 image data
YUV420	YUV4:2:0 image data
RGB888	RGB888 image data
RGB565	RGB565 image data
RGB444	RGB444 image data
RAW8	Raw Bayer 8-bit image data
RAW10	Raw Bayer 10-bit image data
RAW12	Raw Bayer 12-bit image data
JPEG8	JPEG 8-bit data
JPEG8 FSP	JPEG 8-bit data + FSP

Figure 17-11 through Figure 17-19 show the CCP image data operating modes.

Figure 17-11. YUV422

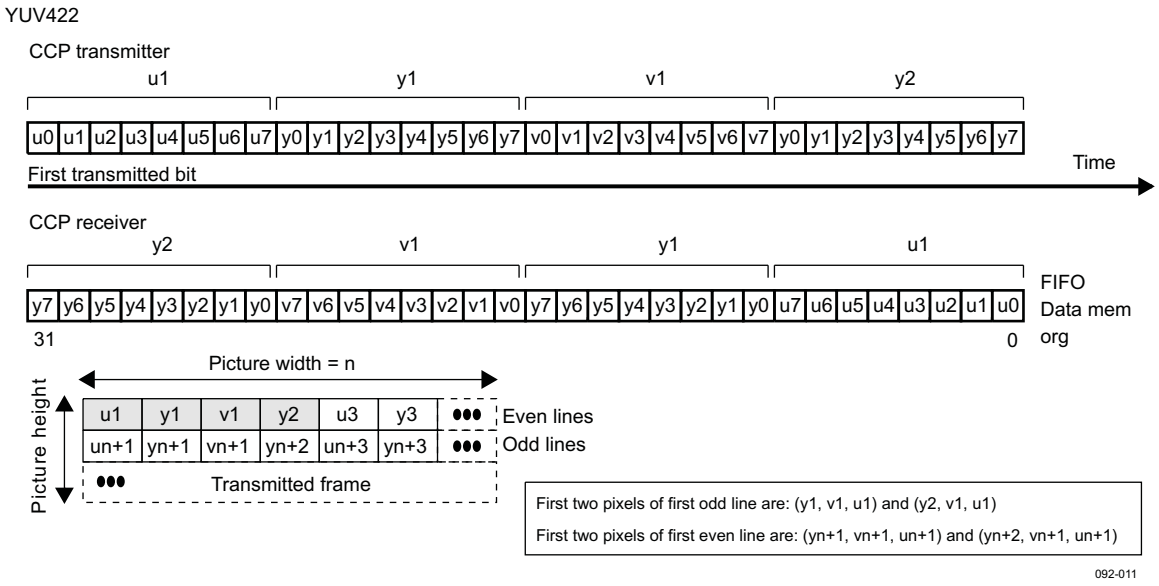


Figure 17-12. YUV420

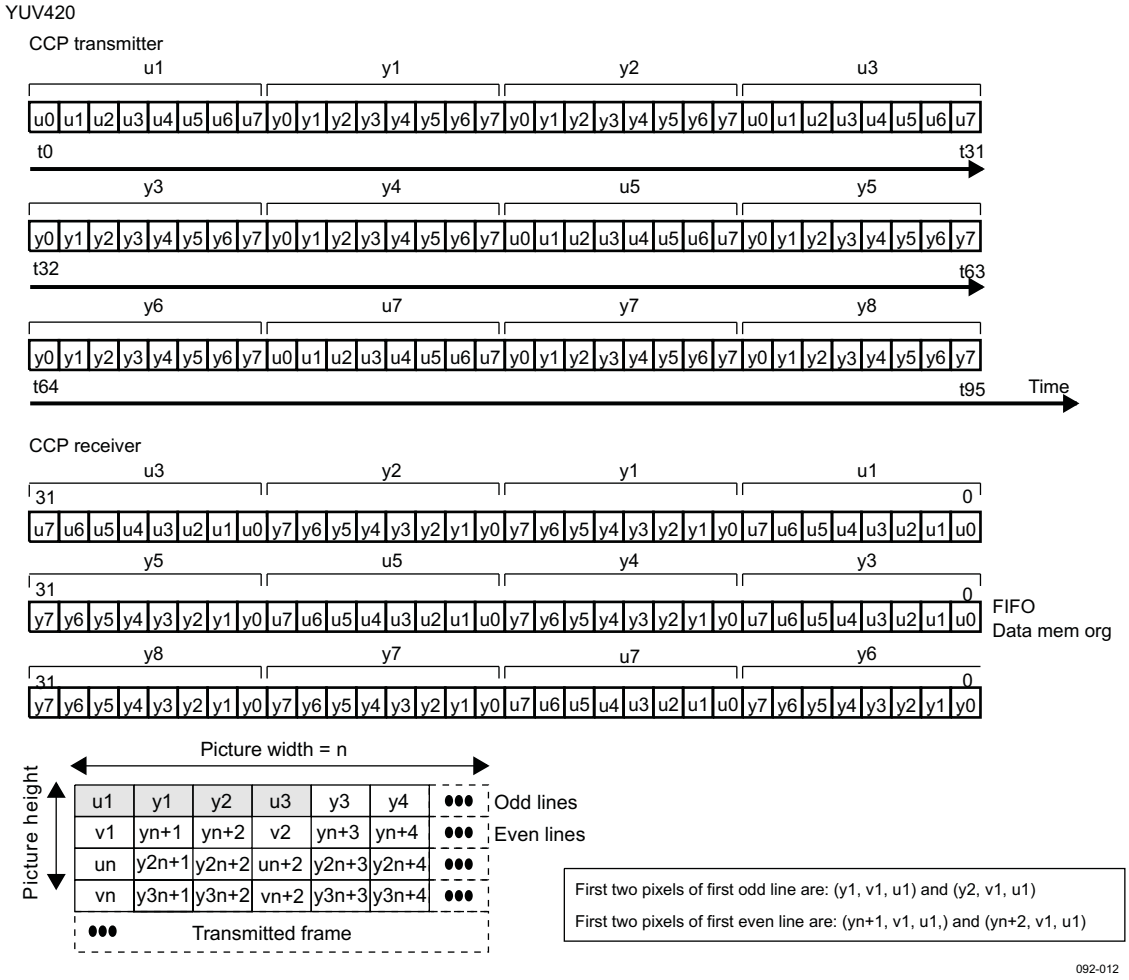


Figure 17-13. RGB888

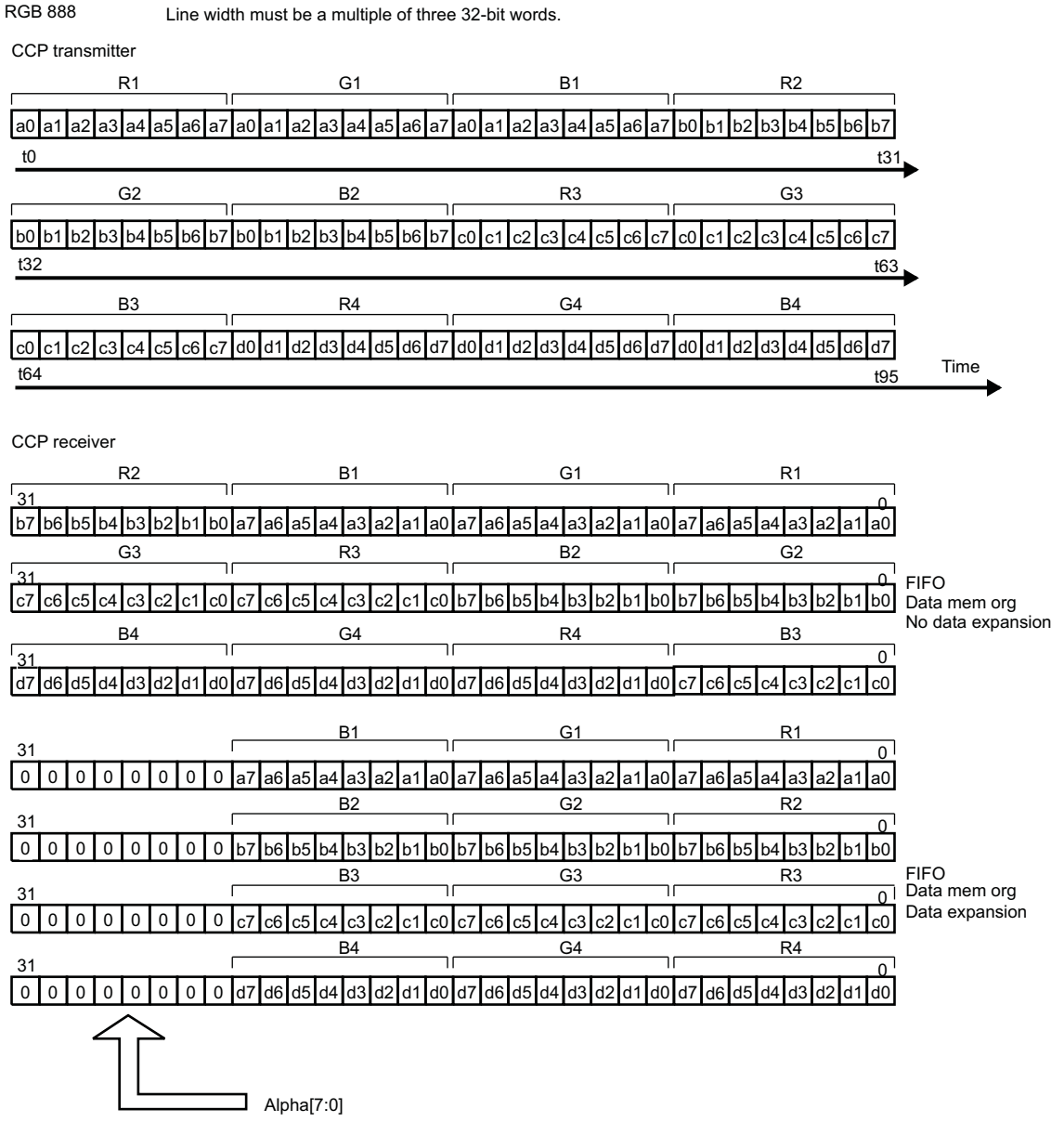


Figure 17-14. RGB565

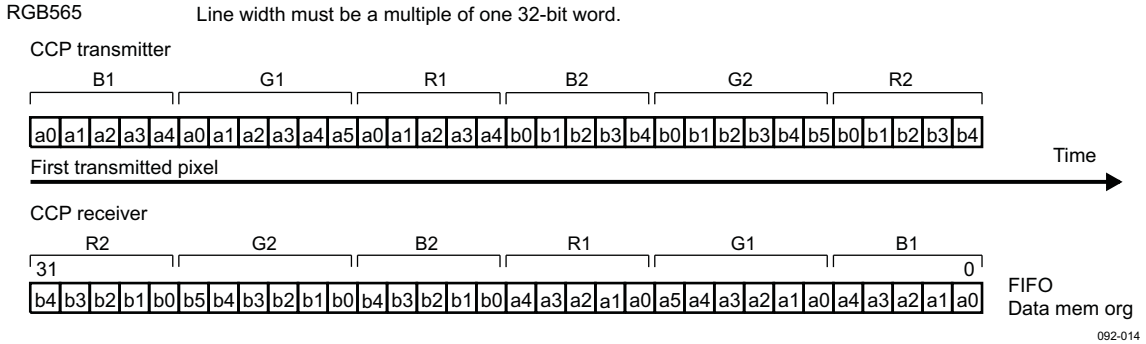


Figure 17-15. RGB444

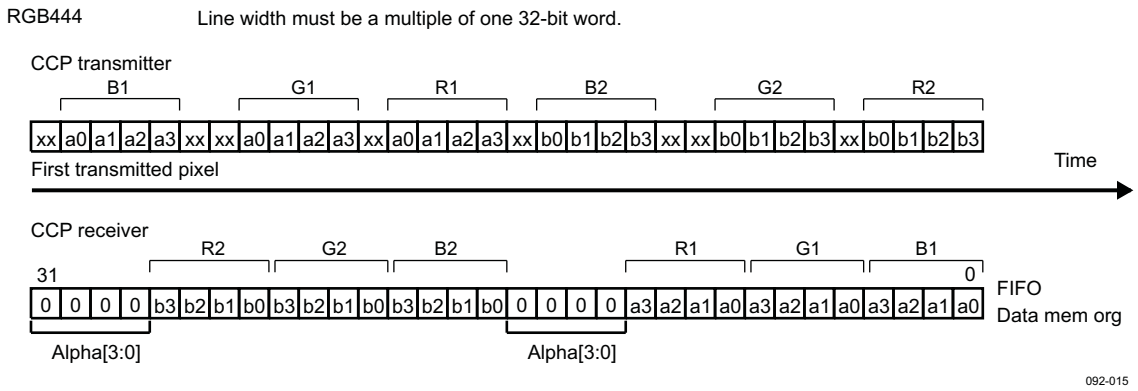


Figure 17-16. RAW8

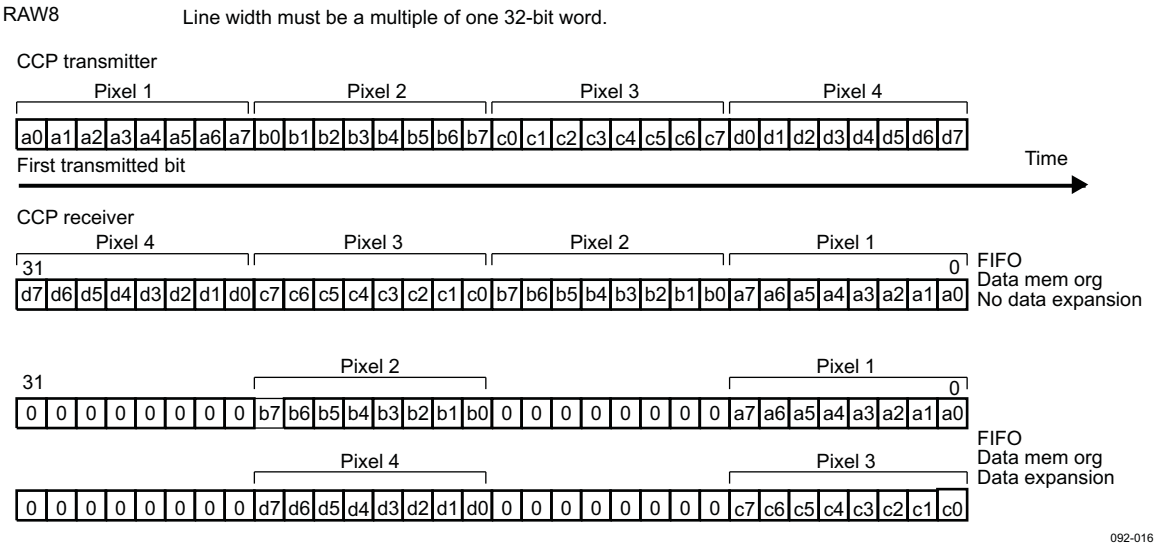


Figure 17-17. RAW10

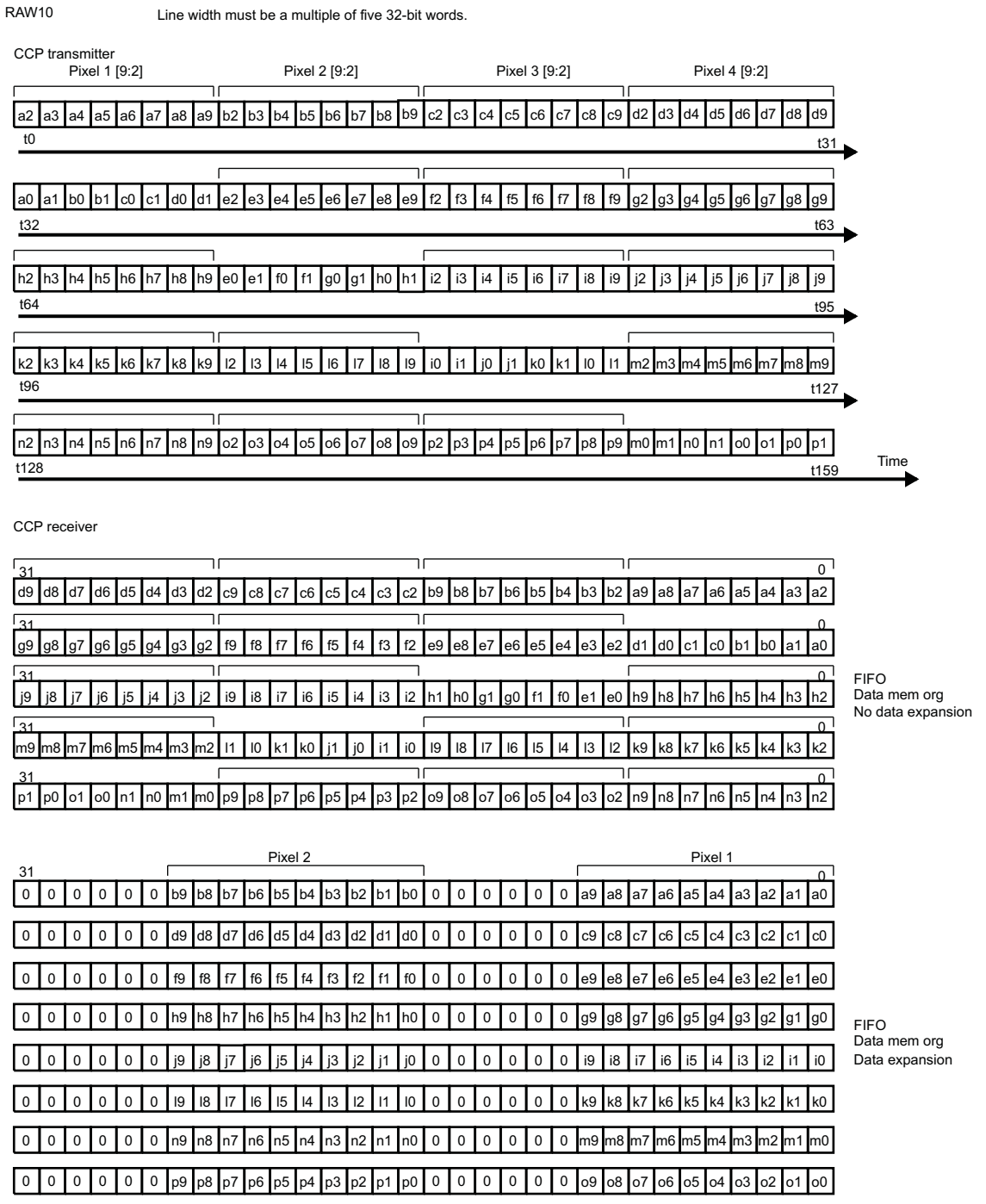


Figure 17-18. RAW12

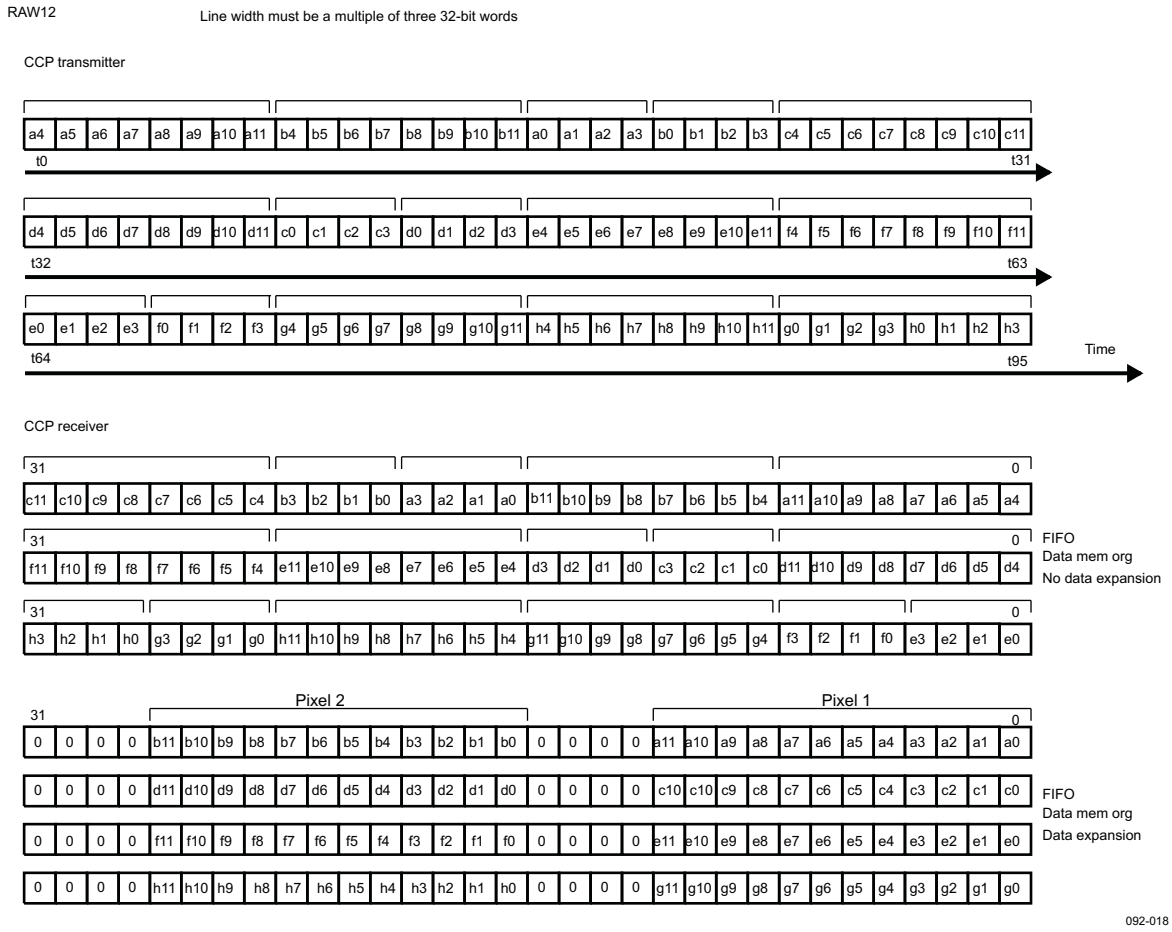
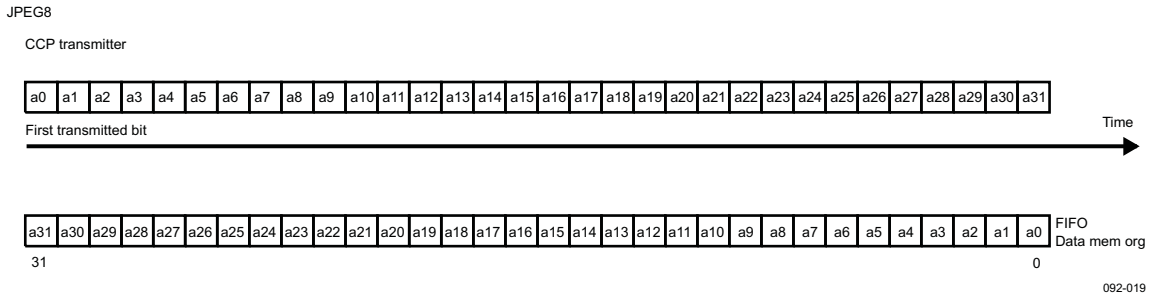


Figure 17-19. JPEG8 and JPEG8 FSP

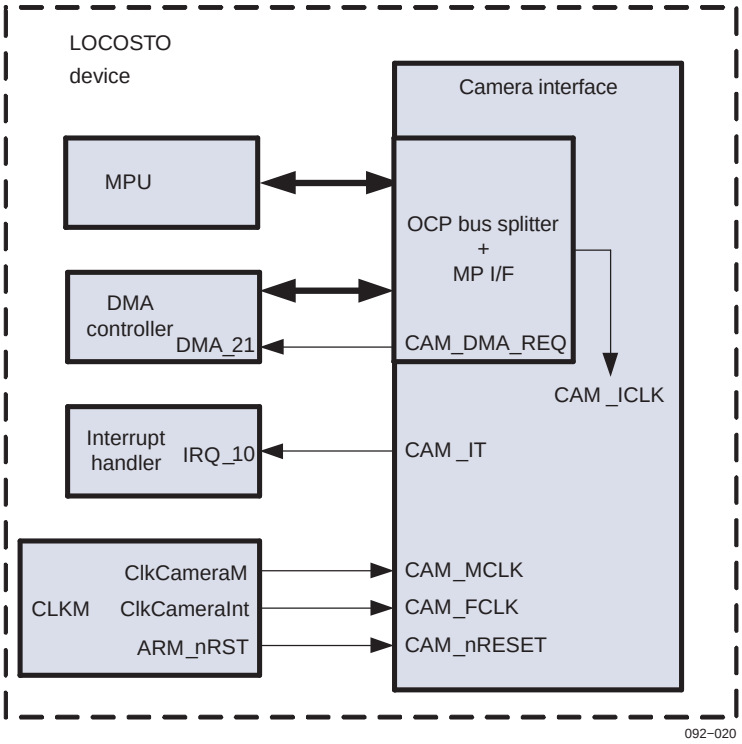


Note: JPEG8 FSP is a JPEG format in which FSP is a decoding performed by the receiving end on the JPEG8 format. FSP decoding removes the 0xA5 data inserted by the FSP encoder to suppress any violating combinations on the data stream. FSP decoding occurs after the frame-start synchronization code is received.

17.3 Camera Interface Integration

Figure 17-20 shows the camera interface integration in the LOCOSTO.

Figure 17-20. Camera Core Integration



17.3.1 Clock and Reset Scheme

Table 17-6 lists the clocks and reset.

Table 17-6. Clocks and Reset

Module	Functional Clock	Master Clock	Resets
Camera core	CAM_FCLK	CAM_MCLK	**Hardware:** CAM_nRESET
			Software: CAM.CAM_CC_SYSCONFIG[1] SOFTWRESET bit

17.3.1.1 Camera Core Clocks

The clock domains in the camera core are as follows:

- Serial camera sensor
- Parallel camera sensor
- Functional

17.3.1.1.1 Serial Camera Sensor Clock Domain

cam_s_clk is the serial sensor clock provided through the cam_vs and cam_hs differential clock inputs from an external serial image sensor. The cam_s_clk frequency can be up to 208 MHz (for more information, see Section 17.2.6, *CCP Serial Camera Interface Configuration*).

17.3.1.1.2 Parallel Camera Sensor Clock Domain

cam_lclk is the parallel sensor clock provided through the cam_lclk clock input from an external parallel image sensor. The cam_lclk frequency can be up to 48 MHz.

17.3.1.1.3 Functional Clock Domain

The parallel camera module is provided by the ClkCameraM signal from the clock and reset management (CLKM) module and can provide a camera reference clock (CAM_XCLK) to the camera sensor based on on-chip APLL (48 MHz) or MPU (52 MHz) clock sources. The generated clock is not used by the camera core module. The camera core clock generator can be disabled if the chip can provide the required clock to the sensor. The configuration of the clock divider is programmable by setting the configuration register (CAM.CAM_CC_CTRL_XCLK.XCLK_DIV[4:0]).

For more information about the cam_xclk generation, see Section 17.4, *Functional Description*.

17.3.1.2 Camera Core Reset Scheme

17.3.1.2.1 Hardware Reset

Global reset of the module is performed by the ARM_nRST hardware reset signal.

17.3.1.2.2 Software Reset

The camera core also provides three types of resets:

- Reset of the camera core:

  To reset the camera core, set the CAM.CAM_CC_SYSCONFIG[1] SOFTRESET bit to 1. This bit is automatically reset by hardware. During reads, it always returns 0.
- Reset of the image sensor interface:

  The internal state-machines of the image sensor interface are reset by setting the CAM.CAM_CC_CTRL[16] CC_EN bit to 1. This ensures proper operation of the CCP and the BT when the clock is off between frames.
- Reset of the FIFO and DMA control:

  The internal state-machines of the FIFO and DMA control circuitry are software-reset by setting the CAM.CAM_CC_CTRL[18] CC_RST bit to 1. This software reset must be applied when the image sensor interface is disabled (that is, when the CAM.CAM_CC_CTRL[16] CC_EN bit is set to 0).

17.3.2 Hardware Requests

17.3.2.1 DMA Requests

The camera core DMA request mapping is listed in Table 17-7. For more information about system DMA (sDMA), see Chapter 13, *Direct Memory Access*.

Table 17-7. Camera Core DMA Request

DMA Request Name	Mapping	Destination
CAM_DMA_REQ	DMA_21	DMA controller

17.3.2.2 Interrupt Requests

The interrupt mapping from the camera core is shown in Table 17-8. CAM_IT is an interrupt to the MPU subsystem interrupt handler. For more information about the MPU interrupt handler, see Chapter 12, *Interrupt Handlers*.

Table 17-8. Camera Core Interrupt Names and MPU IRQ Mapping

Interrupt Name	Mapping	Comments	Domain
CAM_IT	IRQ_10	Interrupt from camera core	MPU

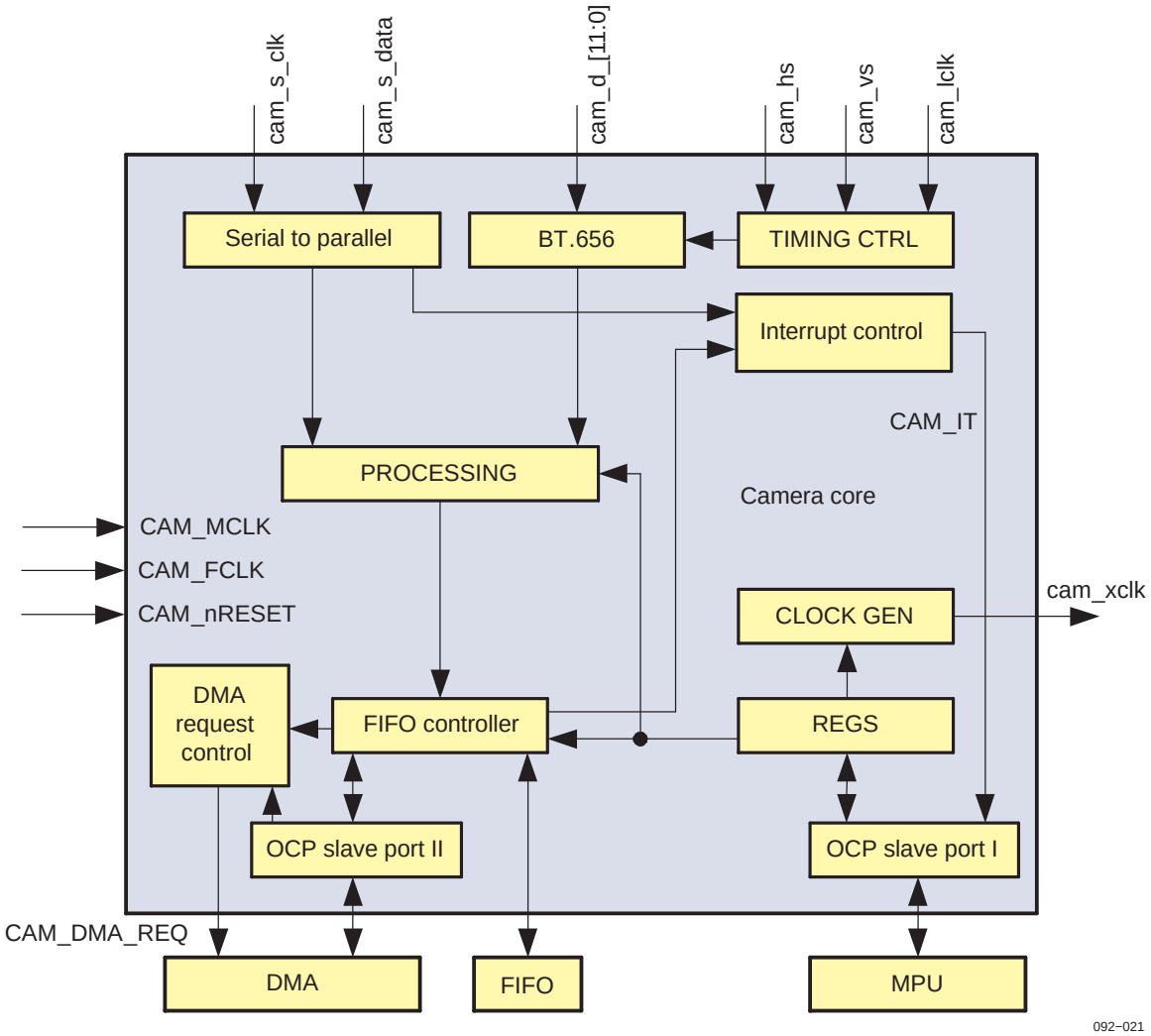
Note: CAM_IT is an interrupt request generated by the camera core. The interrupt source can be determined using the CAM.CAM_CC_IRQSTATUS register fields. This interrupt can be masked by using the CAM.CAM_CC_ IRQENABLE register fields. See Section 17.6.2, *Camera Core Register Descriptions*.

17.4 Camera Interface Functional Description

17.4.1 Block Diagram

Figure 17-21 is a block diagram of the camera core.

Figure 17-21. Camera Core Block Diagram



The camera core transfers data from the image sensor to the buffer (FIFO) and generates DMA requests. Two DMA requests are connected to the same CAM_DMA_REQ line: The first is triggered when the FIFO threshold level is reached, and the second is triggered at the end of a frame, if data remain in the FIFO.

This block can be used in several configurations to accommodate different image sensors, but the serial and parallel interfaces cannot be active at the same time.

The camera core includes a 128 × 32-bit FIFO with a programmable trigger level. When the write FIFO counter reaches this trigger level, a DMA request is generated. This request can be enabled or disabled (DMA or MPU mode).

A programmable interrupt generator is embedded in the camera core. An interrupt can be generated to indicate the start and end of a line, the start and end of an image, FIFO overflow, and data threshold status.

Each time 4 bytes are received from the camera sensor, they are packed and stored in the FIFO. Optional byte swapping can be enabled. The MPU or the sDMA reads the data out of the FIFO for additional processing.

The camera core can be used in several configurations to accommodate different image sensors:

- CCP configuration (serial): The CCP bit stream has embedded synchronization signals: frame start, line start, line end, and frame end.
- BT.656 configuration (parallel): The BT.656 block extracts the SAV and EAV signals and the data signals.
- NoBT.656 configuration (parallel): The BT.656 block uses the cam_hs and cam_vs signals to determine the validity of the data.

The serial and parallel interfaces cannot be active at the same time.

17.4.2 Compact Camera Port

17.4.2.1 CCP Features

The CCP interface has several image-data operating modes.

Each data format has a constraint in terms of number of 32-bit words per line. Table 17-9 lists the constraints for the CCP data formats.

Table 17-9. Constraints on Line Length for CCP Data Formats

Data Format	Constraint on Line Length
YUV422, YUV420, RGB565, RGB444, RAW8, JPEG8, JPEG8 FSP	Line must be a multiple of one 32-bit word.
RGB888	Line must be a multiple of three 32-bit words.
RAW10	Line must be a multiple of five 32-bit words.
RAW12	Line must be a multiple of three 32-bit words.

If these constraints are not met, an error is flagged in the interrupt status register. The CAM.CAM.CC_IRQSTATUS[10] FW_ERR_IRQ bit is set to 1.

YUV422, YUV420, RGB888 (no data expansion), RGB565, RAW8 (no data expansion), RAW10 (no data expansion), RAW12 (no data expansion), and JPEG8 formats are identical for the CCP receiver.

RGB888 (data expansion), RGB444, RAW8 (data expansion), RAW10 (data expansion), and RAW12 (data expansion) formats require dedicated treatment by the CCP receiver.

The data mode is selected by the CCP data format select register. Table 17-10 summarizes the programming of the CAM.CAM.CC_CCPDFR[3:0] DATAFORMATSELECT[3:0] field.

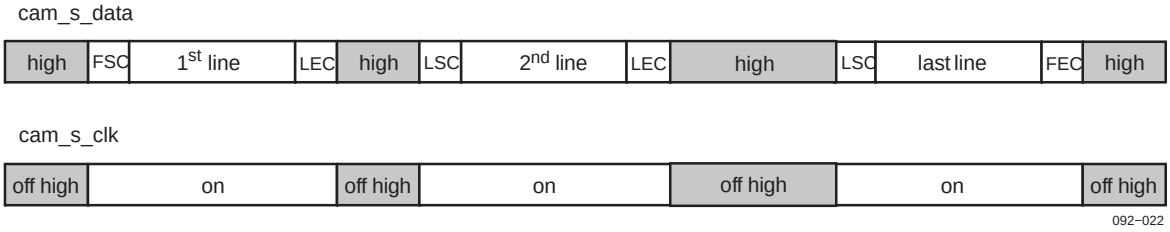
Table 17-10. DATAFORMATSELECT[3:0] Field Programming in CAM.CAM_CC_CCPDFR

Data Format	DATAFORMATSELECT[3:0]
YUV422 big endian	0000
YUV422	0001
YUV420	0010
RGB444	0100
RGB565	0101
RGB888 (no data expansion)	0110
RGB888 (data expansion)	0111
RAW8 (no data expansion)	1000
RAW8 (data expansion)	1001
RAW10 (no data expansion)	1010
RAW10 (data expansion)	1011
RAW12 (no data expansion)	1100
RAW12 (data expansion)	1101
JPEG8 FSP	1110
JPEG8	1111

17.4.2.2 CCP Timing

Figure 17-22 shows the timing diagrams of CCP serial interface signals cam_s_data and cam_s_clk. The CCP bit stream has embedded synchronization signals: frame start (FSC), line start (LSC), line end (LEC), and frame end (FEC). cam_s_data is registered at the rising edge of the cam_s_clk clock.

Figure 17-22. Timing Diagrams of cam_s_data and cam_s_clk



The clock signal cam_s_clk can be either free-running or off between the lines and frames. The camera core receiver reacts to the frame-end synchronization code immediately after its reception to support the shutdown of the clock. When the transmitter has sent the last bit of the synchronization code, the clock signal remains high until the next frame starts. The receiver does not need extra transmission clock cycles after the FEC.

17.4.2.3 CCP Synchronization Code

The reception of data from the image sensor module uses four synchronization codes, which are embedded in the serial bit stream:

- The FSC identifies the start of a new frame.
- The LSC identifies the start of a new line.
- The LEC identifies the end of a line. It is received for every line, except for the last line, which ends with a frame-end code.
- The FEC identifies the end of the last line and the end of the current frame.

The CCP synchronization codes after hardware reset are listed in Table 17-11. Every synchronization code is transmitted byte-wise, least-significant bit (LSB) first. For example, the code 0xFF00 0002 transmitted from the image sensor corresponds to the following bit stream:

0b11111111 – 0b00000000 – 0b00000000 – 0b01000000

Every default code starts with a set of eight 1s and sixteen 0s that are never received in pixel data (because having eight 1s and sixteen 0s is not allowed in pixel data).

The CCP synchronization codes are programmable through camera core configuration registers CAM.CAM_CC_CCPFSCR, CAM.CAM_CC_CCPLECR, CAM.CAM_CC_CCPLSCR, and CAM.CAM_CC_CCPFECR.

Table 17-11. CCP Synchronization Codes

Data Types	Code	Configuration Register	Reset Value
All data types	Line start	CAM.CAM_CC_CCPLSCR	0xFF00 0000
	Line end	CAM.CAM_CC_CCPLECR	0xFF00 0001
	Frame start	CAM.CAM_CC_CCPFSCR	0xFF00 0002
	Frame end	CAM.CAM_CC_CCPFECR	0xFF00 0003

Synchronization is operated bit-wise. Every line consists of a finite number of 32-bit CCP words.

Two synchronization errors can occur:

- FALSESYNCCODE (FSC) indicates that the detection of the synchronization code was out of order. When this occurs, the CAM.CAM_CC_IRQSTATUS[9] FSC_ERR_IRQ bit is flagged.
- SHIFTEDSYNCCODE (SSC) indicates that a detection of a synchronization code on a non-32-bit boundary occurred. In this case, the CAM.CAM_CC_IRQSTATUS[8] SSC_ERR_IRQ bit is flagged. This synchronization error occurs only with LECs and FECs.

17.4.3 Parallel NoBT656 (NoBT)

17.4.3.1 NoBT Features

No assumptions are made on the data format in this configuration, but the parallel interface size is limited to 8 bits.

17.4.3.2 Description

Pixel data is presented on cam_d_[7:0]. One pixel is latched on every cam_lclk rising edge (or falling edge, depending on the configuration of the cam_lclk polarity).

The cam_lclk polarity is defined through the CAM.CAM_CC_CTRL[10] (camera core control register) PAR_CLK_POL bit.

There are additional pixel times between rows that represent a blanking period. The active pixels are identified by a combination of two additional timing signals: cam_hs (horizontal synchronization) and cam_vs (vertical synchronization).

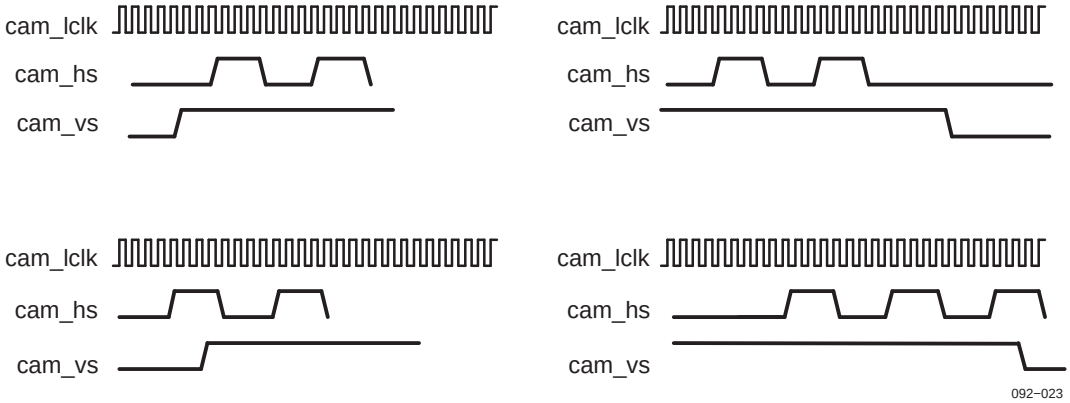
During the image sensor readout, these signals define when a row of valid data begins and ends and when a frame starts and ends.

The CAM.CAM_CC_CTRL[9] (camera core control register) NoBT_HS_POL bit sets cam_hs polarity, and the CAM.CAM_CC_CTRL[8] (camera core control register) NoBT_HS_POL bit sets bitcam_vs polarity.

The acquisition can start either on the beginning of a new frame (cam_vs inactive and then active) or immediately in function of the CAM.CAM_CC_CTRL[13] NoBT_SYNCHRO bit. Set this last one to 1 to ensure a clean acquisition of the frame. The scenarios shown in Figure 17-23 through Figure 17-26 are also allowed in NoBT configuration; in other words, data is accepted as long as both cam_hs and cam_vs are active.

In the scenario shown in Figure 17-23, data is valid when both cam_hs and cam_vs are active (high in this example).

Figure 17-23. Different Scenarios of cam_hs and cam_vs



The camera core supports decimation from the image sensor where cam_hs toggles between pixels. Figure 17-24 shows this scenario.

Figure 17-24. cam_hs Toggles Between Pixels in Decimation

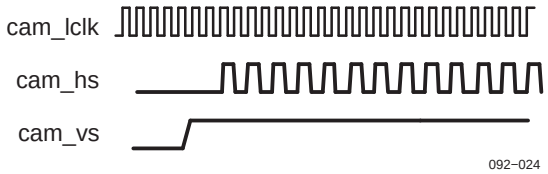
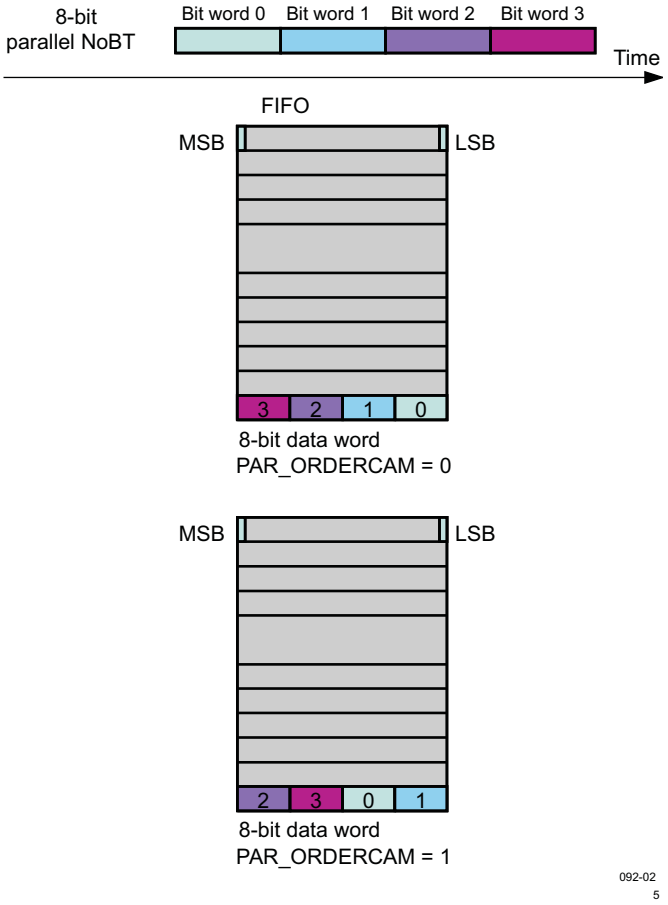


Image data is stored differently in the FIFO, depending on the setting of the CAM.CAM_CC_CTRL[11] PAR_ORDERCAM bit, as shown in Figure 17-25.

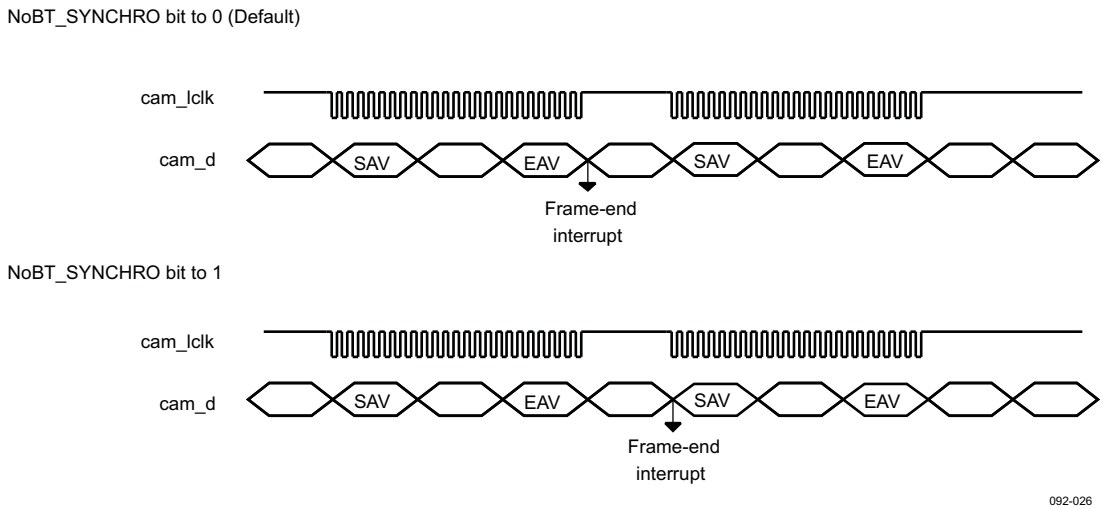
Figure 17-25. FIFO 8-Bit Data in NoBT Format



Note: Blanking periods are automatically removed by the camera core module.

When the CAM.CAMC1_CC_CTRL[13] NoBT_SYNCHRO bit is set to 1, a pipeline register is selected for all sensor inputs, to improve performance. If the clock is off between lines or frames, the last data is delayed until the start of the next frame (next clock cycle). Line-end/frame-end interrupts are also delayed, as shown in Figure 17-26, and DMA requests can be impacted.

Figure 17-26. Frame-End Interrupt Generation in BT Mode



Note: Frame-end interrupts can never be generated for a 1-shot acquisition.

Note: A free-running clock with the CAM.CAMC1.CC_CTRL[1] NoBT_SYNCHRO bit set to 1 must be used to avoid a delay.

17.4.4 ITU-R BT.656

17.4.4.1 ITU-R BT.656 Features

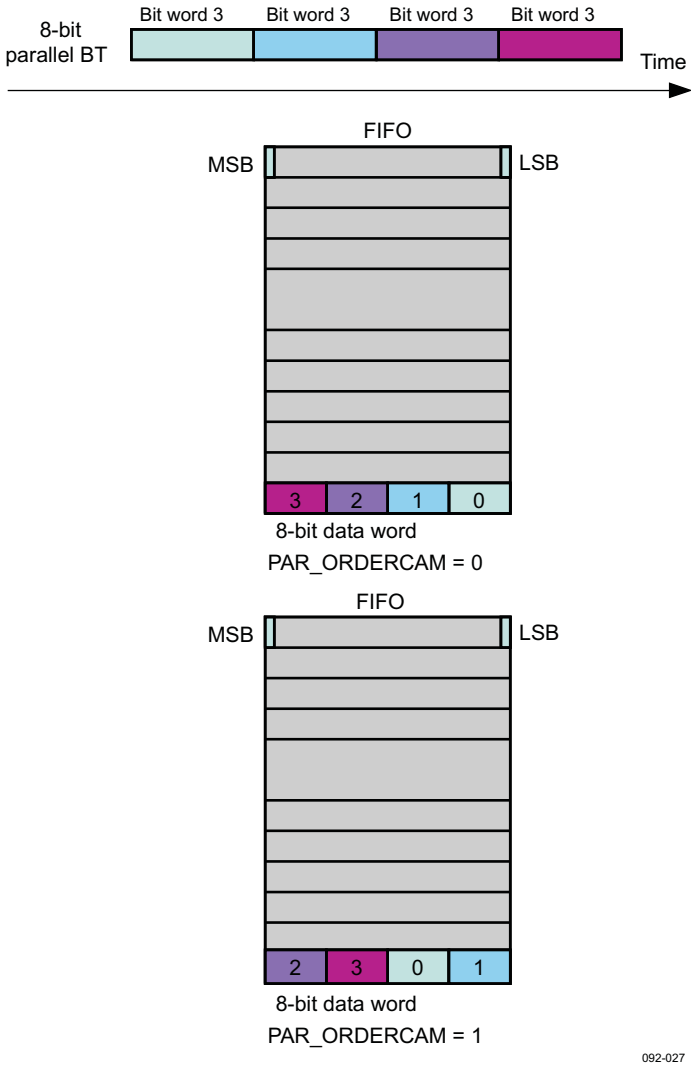
The camera interface supports data in ITU-R BT.656 format. The ITU-R BT.656 standard specifies a method to transfer YUV422 data over an 8-bit video interface and store it in the FIFO. The parallel interface size is limited to 8 bits.

17.4.4.2 Description

The error-correcting capability can be enabled or disabled by configuring the CAM.CAM_CC_CTRL[12] BT_CORRECT bit.

Image data is stored differently in the FIFO, depending on the setting of the CAM.CC_CTRL[11] PAR_ORDERCAM bit, as shown in Figure 17-27.

Figure 17-27. FIFO Image 8-Bit Data in BT Format



Note: Blanking periods are automatically removed by the camera core module.

17.4.5 FIFO Transfer

The FIFO is a buffer that operates based on a first-in, first-out principle and is synchronous with the functional clock. The FIFO receives data from either the CCP serial sensor interface or the parallel interface, as defined by the CAM.CAM_CC_CTRL[0] CCP_MODE bit.

The FIFO depth is set by the generic parameter (the CAM.CAM_CC_GENPAR[2:0] CAM_CC_FIFO_DEPTH field).

The FIFO goes into overflow when a write is attempted to a full FIFO. This means that either the software is too slow or the throughput is too high. The content of the full FIFO is not corrupted by any further writes. During FIFO overflow, it is still possible to read data from the FIFO. Once the FIFO is no longer full, more writes are possible.

When a read is attempted from an empty FIFO, the FIFO goes into underflow because the software has too many read accesses. After an underflow, it is still possible to write. When the FIFO is no longer empty, it is possible to read from the FIFO.

The FIFO is reset by writing 1 to the CAM.CAM_CC_CTRL[18] CC_RST bit. If the CAM.CAM_CC_IRQENABLE[1] FIFO_OF_EN bit or the CAM.CAM_CC_IRQENABLE[0] FIFO_UF_IRQ_EN bit is enabled, an overflow or an underflow generates an interrupt. The interrupt is cleared by writing 1 to the FIFO_OF_IRQ_EN bit or the FIFO_UF_IRQ_EN bit. There is no need to apply CC_RST beforehand.

It is possible to access the FIFO from OCP port 1 (read/write) or OCP port 11 (read).

Table 17-12 lists the FIFO write and read accesses, and it shows how it is possible to access the FIFO by setting the CCP_MODE, PAR_MODE, and DMA_EN fields in the CAM.CAM_CC_CTRL register.

Table 17-12. FIFO Write and Read Access

FIFO Write	FIFO Read	CCP_MODE	PAR_MODE[2:0]	DMA_EN
Serial sensor	OCP 2	1	Don't care	1
Parallel sensor	OCP 2	0	000, 001, 010, 100, 101	1
OCP 1 (CAM_CC_ FIFODATA)	OCP 2	0	111	1
Serial sensor	OCP 1 (CAM_CC_ FIFODATA)	1	Don't care	0
Parallel sensor	OCP 1 (CAM_CC_ FIFODATA)	0	000, 001, 010, 100, 101	0
OCP 1 (CAM_CC_ FIFODATA)	OCP 1 (CAM_CC_ FIFODATA)	0	111	0

17.4.6 Clock Generation

The module can provide the CAM_XCLK clock to the camera sensor based on one input clock (CAM_MCLK). The generated clock is not used by the camera core module. The camera core clock generator can be disabled if the chip can provide the required clock to the sensor. The configuration of the clock divider is programmable by setting the configuration register (CAM_CC_CTRL_XCLK).

The clock divider configuration is programmable through the CAM.CAM_CC_CTRL_XCLK[4:0] XCLK_DIV field, as listed in Table 17-13.

Table 17-13. Ratio of the CAM_XCLK Frequency Generator

XCLK_DIV[4:0]	CAM_XCLK
0	Stable low-level divider not enabled (default)
1	Stable high-level divider not enabled
2	cam_xclk/2
3	cam_xclk/3
…	
30	cam_xclk/30
31	Bypass (cam_xclk frequency = CAM_FCLK frequency)

To generate a error-free CAM_XCLK signal clock, the input CAM_FCLK clock must be error-free before enabling the clock divider generator. It is possible to switch modes without a problem, but if this is done, the camera sensor or the camera module must be disabled to avoid data capture during the frequency transition phase. The duty cycle generated depends on the selected mode, as follows:

- Mode 0 and 1 (divider not enabled): No duty cycle can be defined.
- Mode 31 (bypass mode): CAM_XCLK and CAM_MCLK duty cycles are the same.
- Other modes: 50 percent duty cycle

17.4.7 Interrupt Generation

The camera core provides one active-low interrupt line (CAM_IT) mapped on MPU interrupt handler (IRQ_1).

The CAM_IT line is asserted (active low) when one of the following events occurs:

- FIFO underflow occurs—notified by the CAM.CAM_CC_IRQSTATUS[0] FIFO_UF_IRQ bit.
- FIFO overflow occurs—notified by the CAM.CAM_CC_IRQSTATUS[1] FIFO_OF_IRQ bit.
- FIFO threshold is reached—notified by the CAM.CAM_CC_IRQSTATUS[2] FIFO_THR_IRQ bit.
- FIFO is full—notified by the CAM.CAM_CC_IRQSTATUS[3] FIFO_FULL_IRQ bit.
- FIFO is not empty—notified by the CAM.CAM_CC_IRQSTATUS[4] FIFO_NOEMPTY_IRQ bit.
- A shifted synchronization code is detected—notified by the CAM.CAM_CC_IRQSTATUS[8] SSC_ERR_IRQ bit.
- A false synchronization code is detected—notified by the CAM.CAM_CC_IRQSTATUS[9] FSC_ERR_IRQ bit.
- A frame width error occurs—notified by the CAM.CAM_CC_IRQSTATUS[10] FW_ERR_IRQ bit.
- An FSP decoding code error occurs—notified by the CAM.CAM_CC_IRQSTATUS[11] FSP_ERR_IRQ bit.
- A frame end occurs—notified by the CAM.CAM_CC_IRQSTATUS[16] FE_IRQ bit.
- A line start occurs—notified by the CAM.CAM_CC_IRQSTATUS[17] LS_IRQ bit.
- A line end occurs—notified by the CAM.CAM_CC_IRQSTATUS[18]LE_IRQ bit.
- A frame start occurs—notified by the CAM.CAM_CC_IRQSTATUS[19] FS_IRQ bit.

The CAM.CAM_CC_IRQSTATUS register must be read to determine the cause of the camera core interrupt.

Table 17-14 lists and clarifies which camera core interrupts are operational in CCP, parallel NoBT, and parallel BT modes.

Table 17-14. Operational Interrupts in CCP, Parallel NoBT, and Parallel BT Modes

Interrupt Name	CCP	Parallel NoBT	Parallel BT
FIFO_UF_IRQ	Yes	Yes	Yes
FIFO_OF_IRQ	Yes	Yes	Yes
FIFO_THR_IRQ	Yes	Yes	Yes
FIFO_FULL	Yes	Yes	Yes
FIFO_NOEMPTY	Yes	Yes	Yes
SSC_ERR_IRQ	Yes	No	No
FSC_ERR_IRQ	Yes	No	Yes
FW_ERR	Yes	No	No
FSP_ERR	Yes	No	No
FE_IRQ	Yes	Yes	Yes
LS_IRQ	Yes	No	No
LE_IRQ	Yes	No	No
FS_IRQ	Yes	No	No

When the CAM.CAM_CC_IRQSTATUS register is read, it is not automatically reset. Write 1 to the corresponding bit to reset the interrupt.

Each event that generates an interrupt can be individually enabled through the interrupt enable register (CAM.CAM_CC_IRQENABLE).

Unlike the corresponding register in CCP mode, when a particular event is not enabled (for example, CAM.CAM_CC_IRQENABLE[4] = 0), the corresponding status bit (CAM.CAM_CC_IRQSTATUS[4] = 1) is flagged if the corresponding event occurs. This has no effect on the interrupt line, but can be used by any software to poll the status.

17.4.8 DMA Interface

The camera core module interfaces with a DMA controller. At system level, the advantage is to discharge the CPU or sDMA of the data transfers.

The module can generate two DMA requests:

- One request is generated when the FIFO reaches the threshold programmed into the CAM.CAM_CC_CTRL_DMA[6:0] FIFO_THRESHOLD field.
- The other request is generated when the frame ends and data still remains in the FIFO. This can occur when the frame is not a multiple of the threshold.

When the frame is not a multiple of the threshold, the second DMA request is generated when the DMA request based on threshold cannot be generated. This constrains the software to handle the frame before the next incoming frame. This constraint is always met by considering the blanking period between frames (enough time for the software to handle the previous frame acquisition).

The CAM.CAM_CC_CTRL_DMA[9] DMA1_DISABLE bit generates only the first DMA request based on the threshold. In this case, pictures can be received continuously but must be handled by software.

The deassertion of the DMA request occurs when a number of 32-bit words equal to the FIFO threshold has been read by the DMA controller.

Note: It is easy to confuse a frame, in DMA terminology, with a camera frame. A DMA frame corresponds to THRESHOLD_VALUE size. When this size is reached, a DMA request is asserted.

This applies to source-packed synchronized transfers. The DMA does not transfer one camera frame per DMA request, but a part of camera frame.

The maximum transfer size, regardless of the packet size, is always as follows:

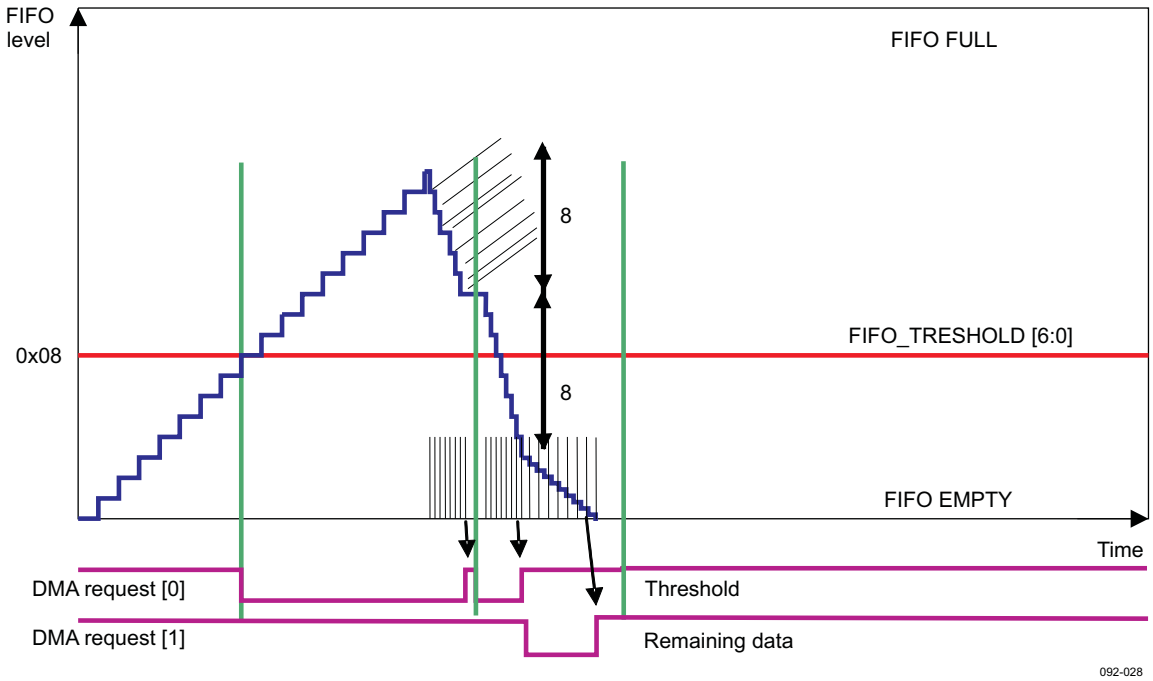
Block_Size = Number_of_Frame_in_Block x Number_of_ Element_in_Frame x Element_Size

Example:

If the video frame is 176 × 144 pixels of 1 byte (25,344 bytes or 6336 32-bit words), and the threshold is 0x0A (32-bit words), the module generates 633 of DMA request 0 and 1 DMA request 1 for the remaining six 32-bit words.

Figure 17-28 shows the DMA assertion and deassertion, and a FIFO threshold of 8. Some additional data to end the frame acquisition is assumed.

Figure 17-28. Camera Core DMA Requests During Frame Acquisition



The DMA interface can be configured using the control DMA register (CAM.CAM_CC_CTRL_DMA). The CAM.CAM_CC_CTRL_DMA[8] DMA_EN bit enables or disables the CAM_DMA_REQ request line.

The CAM.CAM_CC_CTRL_DMA[9] DMA1_DISABLE bit generates only the first request, based only on the threshold.

The CAM.CAM_CC_CTRL_DMA[6:0] FIFO_THRESHOLD field sets the threshold the FIFO can reach before a DMA request is generated.

For more information about the sDMA, see Chapter 13, *Direct Memory Access*.

17.4.9 System Power Management

17.4.9.1 Autoidle Mode

The camera core provides an autoidle capability in its interconnect clock domain.

The interconnect clock autoidle power-saving mode is enabled or disabled by the camera core CAM.CAM_CC_SYSCONFIG[0] AUTOIDLE bit. When this mode is enabled and there is no activity on the interconnect interface, the interconnect clock (CAM_ICLK) is disabled inside the module, thus reducing power consumption. When there is new activity on the interconnect interface, the interconnect clock is restarted with no latency penalty. This mode is disabled by default after a reset.

Enabling this mode is recommended to reduce power consumption. For software configuration, see Section 17.5, *Camera Interface Programming Model*.

17.4.9.2 Slave-Idle Mode

In addition to autoidle mode, the camera core provides a slave-idle mode on the CAM_ICLK interface clock by the CLKM.

The camera core can be configured through the CAM.CAM_CC_ SYSCONFIG SIDLEMODE[1:0] field to be in one of these idle modes:

- No-idle mode: The module never goes into idle state.

- Force-idle mode: The module goes into idle state immediately after receiving the idle request from the CLKM.

For more information about power management, see Chapter 6, *Power, Reset, and Clock Management*.

For software configuration, see Section 17.5, *Camera Interface Programming Model*.

17.5 Camera Interface Programming Model

A programming sequence is listed in this section. However, a full example with the Agilent ADCM-2700 CMOS sensor is detailed in the *APN212* application note on the video acquisition chain.

17.5.1 Camera Core Reset

The camera core module can accept a general software reset, propagated throughout the hierarchy. This reset can be performed as follows to initialize the module and has the same effect as the hardware reset:

1. Set the CAM.CAM_CC_SYSCONFIG[1] SOFTRESET bit to 1.
2. Read the CAM.CAM_CC_SYSSTATUS[0] RESETDONE bit to check if it is 1, indicating that the reset took place.

If the RESETDONE bit is still 0 after five consecutive reads, an error occurred during the reset stage.

17.5.2 Enable Picture/Video Acquisition

Always check that the camera core module is not under reset before starting any picture or video acquisition.

After the reset is complete (the RESETDONE bit is set to 1), a configuration step is required before actual acquisition.

Note: In this section, THRESHOLD_VALUE refers to the amount of 32-bit data read by the system DMA from the camera core for each sDMA request.

Use the following programming sequence to set the camera core module:

1. Set the CAM.CAM_CC_SYSCONFIG[0] AUTOIDLE bit to 1 to enable power-saving circuitry.
2. Configure the interrupt generation as required using the CAM.CAM_CC_IRQSTATUS and CAM.CAM_CC_IRQENABLE registers (the most common interrupts are overflow and underflow).
3. Set the CAM.CAM_CC_CTRL_DMA[6:0] FIFO_THRESHOLD field to THRESHOLD_VALUE – 1 and the CAM.CAM_CC_CTRL_DMA[8] DMA_EN bit to 1 for normal usage of the module. (You can also configure a transfer with OCP 1.)
4. To request external clock generation, configure the CAM.CAM_CC_CTRL_XCLK[4:0] XCLK_DIV field.
5. If using the CCP, configure the CAM.CAM_CC_CCPDFR register to select the data format (CAM.CAM_CC_CCPDFR[3:0] DATA_FORMAT_ SELECT field and alpha (CAM.CAM_CC_CCPDFR[15:8] ALPHA) field.
6. Enable picture acquisition using the CAM.CAM_CC_CTRL[16] CC_EN bit. This register lets you select the functional mode, select CCP or BT format, and specify when the module starts and ends the acquisition.

Setting the CAM.CAM_CC_CTRL[17] CC_FRAME_TRIG bit and the CAM.CAM_CC_CTRL[13] NoBT_SYNCHRO bit to 1 is recommended when the CC_EN bit is set to 1 to start the acquisition. The same behavior occurs for the end of the frame when the CC_EN bit is disabled, and the CC_FRAME_TRIG bit is required to cleanly stop the acquisition. This ensures that the DMA is not lost and that the picture written into the SDRAM is full and not partially received.

For the software to acquire only one frame acquisition, use the CAM.CAM_CC_CTRL[20] CC_ONE_SHOT bit. In this case, the module is automatically disabled by itself at the end of the frame.

17.5.3 Disable Picture/Video Acquisition

Picture acquisition is disabled through the camera core.

Set the CAM.CAM_CC_CTRL[16] CC_EN bit to 0 and set the CAM.CAM_CC_CTRL[17] CC_FRAME_TRIG bit to 1 to correctly disable the frame acquisition. In this case, the camera core either ends the current frame at the end or stops immediately, if there is no ongoing frame.

17.5.4 Interrupt Handling

When an interrupt request occurs (with CAM_IT asserted low), the source is identified in the CAM.CAM_CC_IRQSTATUS register.

1. Read the camera core CAM.CAM_CC_IRQSTATUS register to identify the source.
2. Set the related bit to 1 in the camera core CAM.CAM_CC_IRQSTATUS register to clear the source.

17.5.4.1 FIFO Overflow (CAM.CAM_CC_IRQSTATUS[1] Set to 1)

1. Set the CAM.CAM_CC_CTRL[17] CC_FRAME_TRIG bit to 0 so the module can be disabled immediately.
2. Set the CAM.CAM_CC_CTRL[16] CC_EN bit to 0 to disable the sensor interface.
3. Clear the interrupt source by setting the CAM.CAM_CC_IRQSTATUS[1] FIFO_OF_IRQ bit to 1.

CAUTION

If the CAM.CAM_CC_CTRL_DMA[8] DMA_EN bit is set to 1, the MPU can terminate the DMA transfer immediately or let it run until no DMA requests remain. Ensure that the desired transfers are terminated before performing Step 4.

If the CAM.CAM_CC_CTRL_DMA[8] DMA_EN bit is set to 0, the MPU can either finish the FIFO read (CAM.CAM_CC_FIFODATA register) or stop reading it. Ensure that the desired transfers are terminated before performing Step 4.

4. Reset the FIFO pointers and camera core internal state-machines by setting the CAM.CAM_CC_CTRL[18] CC_RST bit to 1.
5. Set the CAM.CAM_CC_CTRL[16] CC_EN bit to 1 to re-enable the data flow from the image sensor.

Note: If an overflow occurs, reset the entire data flow path (for example, the DMA controller) to restart cleanly.

17.5.4.2 FIFO Underflow (CAM.CAM_CC_IRQSTATUS[0] Set to 1)

1. Set the CAM.CAM_CC_CTRL[17] CC_FRAME_TRIG bit to 0 so the module can be disabled immediately.
2. Set the CAM.CAM_CC_CTRL[16] CC_EN bit to 0 to disable the sensor interface.
3. Clear the interrupt source by setting the CAM.CAM_CC_IRQSTATUS[0] FIFO_UF_IRQ bit to 1.
4. Reset the FIFO pointers and camera core internal state-machines by setting the CAM.CAM_CC_CTRL[18] CC_RST bit to 1.
5. Set the CAM.CAM_CC_CTRL[16] CC_EN bit to 1 to re-enable the data flow from the image sensor.

Note: If an underflow occurs, reset the entire data flow path (for example, the DMA controller) to restart cleanly.

17.5.4.3 SSC_ERR_IRQ (Shift Sync Code), FSC_ERR_IRQ (Frame Sync Code), FW_ERR (Frame Width), FSP_ERR (FSP Code)

A noisy line can cause a problem during data reception. One solution is to be aware of noisy transmission lines, which can result in overflow/underflow interrupt generation.

Note: For more information about the video acquisition chain, see the *APN212* application note.

17.5.5 Power Management

17.5.5.1 Autoidle Mode

To reduce the power consumption of the module, select the autoidle capability using the CAM.CAM_CC_SYSCONFIG[0] AUTOIDLE bit. Table 17-15 lists the power-management behavior based on the values of autoidle (the CAM.CAM_CC_SYSCONFIG[0] CC_EN bit, the CAM.CAM_CC_CTRL[16] CCP_MODE bit, and the CAM.CAM_CC_CTRL[0] XCLK_DIV bit).

Table 17-15. Power-Management Behavior

Autoidle	CC_EN	CCP_MODE	XCLK_DIV[4:0]	Functionality
0	X	X	X	All internal clocks are free-running.
1	X	X	Value equal to 0, 1, or 31	The XCLK clock is not divided; the divider is shut off.
1	0	X	X	The clocks provided by the camera interface are shut off because the sensor is no longer required for data capture. Then, if the FIFO is empty when the module is disabled, the functional clock for the FIFO and the DMA is shut off. Any access over the OCP 1 and OCP 2 interfaces wakes up the system.
1	1	0	X	Parallel-clock input is on for data acquisition in parallel mode.
1	1	1	X	Serial-clock input is on for data acquisition in CCP mode.

17.5.5.2 Slave-Idle Mode

Power saving on the camera core can also be configured using the CAM.CAM_CC_SYSCONFIG[4:3] SIDLEMODE field to set one of the following idle modes:

- No-idle mode: The module never goes into idle state if the CAM.CAM_CC_SYSCONFIG[4:3] SIDLEMODE field is set to 0x01.
- Force-idle mode: The module goes into the idle state immediately after receiving the request from the ULPDR module if the CAM.CAM_CC_ SYSCONFIG[4:3] SIDLEMODE field is set to 0x00.

17.6 Camera Interface Registers

Table 17-16 contains an instance summary for the camera core registers.

Table 17-16. Instance Summary

Module Name	Base Address	Size	Description
Camera interface	0x0970 0000	1M byte	Camera core

17.6.1 Camera Interface Register Mapping Summary

Table 17-17 lists the camera core registers and their respective type, widths, and address offset values.

Table 17-17. Camera Core Register Offset Addresses

Register Name	Type	Register Width (Bits)	Offset
CAM_CC_REVISION	R	32	0x00
CAM_CC_SYSCONFIG	RW	32	0x10
CAM_CC_SYSSTATUS	R	32	0x14
CAM_CC_IRQSTATUS	RW	32	0x18
CAM_CC_IRQENABLE	RW	32	0x1C
CAM_CC_CTRL	RW	32	0x40
CAM_CC_CTRL_DMA	RW	32	0x44
CAM_CC_CTRL_XCLK	RW	32	0x48
CAM_CC_FIFODATA	RW	32	0x4C
CAM_CC_TEST	R	32	0x50
CAM_CC_GENPAR	R	32	0x54
CAM_CC_CCPFSCR	RW	32	0x58
CAM_CC_CCPFECR	RW	32	0x5C
CAM_CC_CCPLSCR	RW	32	0x60
CAM_CC_CCPLECR	RW	32	0x64
CAM_CC_CCPDFR	RW	32	0x68

Note: The camera core registers are limited to 32-bit access. Data accesses of 16 bits and 8 bits generate an error. For more information, see Chapter 5, *Interconnect*.

17.6.2 Camera Core Register Descriptions

Table 17-18 through Table 17-33 describe the camera interface registers.

Table 17-18. CAM_CC_REVISION

Address Offset	0x00		
Physical Address	0x0970 0000	**Instance**	Camera core
Description	Contains the IP revision code		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REV							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	REV	IP revision code	R	0x22

Table 17-19. CAM_CC_SYSCONFIG

Address Offset	0x10		
Physical Address	0x0970 0010	**Instance**	Camera core
Description	Controls the parameters of the interface (CCP and parallel mode)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											SIDLEMODE		Reserved	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
4:3	SIDLEMODE	Slave interface power management, req/ack control	RW	0x0
		0x00: Force-idle. An idle request is acknowledged unconditionally.		
		0x01: No-idle. An idle request is never acknowledged.		
		0x10: Reserved. Do not use.		
		0x11: Reserved. Do not use.		
2	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
1	SOFTRESET	Software reset. Set this bit to trigger a module reset. The bit is automatically reset by the hardware. During reads, it always returns 0.	RW	0
		0x0: Normal mode		
		0x1: The module is reset.		
0	AUTOIDLE	Internal OCP gating strategy	RW	0
		0x0: The OCP clock is free-running.		
		0x1: Automatic OCP clock-gating strategy is applied, based on OCP interface activity.		

Table 17-20. CAM_CC_SYSSTATUS

Address Offset	0x14		
Physical Address	0x0970 0014	**Instance**	Camera core
Description	Provides status information about the module, excluding interrupt status information		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															RESETDONE

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Reserved for module-specific status information. Read returns 0.	R	0x000000
7:1	Reserved	Reserved for OCP socket status information. Read returns 0.	R	0x00
0	RESETDONE	Internal reset monitoring	R	0
		0x00: Internal module reset is ongoing.		
		0x01: Reset completed		

Table 17-21. CAM_CC_IRQSTATUS

Address Offset	0x18		
Physical Address	0x0970 0018	**Instance**	Camera core
Description	Regroups statuses of module internal events that can generate interrupts (CCP and parallel mode)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												FS_IRQ	LE_IRQ	LS_IRQ	FE_IRQ	Reserved				FSP_ERR_IRQ	FW_ERR_IRQ	FSC_ERR_IRQ	SSC_ERR_IRQ	Reserved			FIFO_NO_EMPTY_ IRQ	FIFO_FULL_IRQ	FIFO_THR_IRQ	FIFO_OF_IRQ	FIFO_UF_IRQ

Bits	Field Name	Description	Type	Reset
31:20	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x000
19	FS_IRQ	Frame start has occurred. Read 0x0: Event false Write 0x0: Event status bit unchanged Read 0x1: Event is true (pending). Write 0x1: Event status bit is reset.	RW	1
18	LE_IRQ	Line end has occurred. Read 0x0: Event false Write 0x0: Event status bit unchanged Write 0x1: Event status bit is reset. Read 0x1: Event is true (pending).	RW	0
17	LS_IRQ	Frame end has occurred. Read 0x0: Event false Write 0x0: Event status bit unchanged Read 0x1: Event is true (pending). Write 0x1: Event status bit is reset.	RW	0
16	FE_IRQ	Frame end has occurred. Read 0x0: Event false Write 0x0: Event status bit unchanged Read 0x1: Event is true (pending). Write 0x1: Event status bit is reset.	RW	0
15:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
11	FSP_ERR_IRQ	FSP code error Read 0x0: Event false Write 0x0: Event status bit unchanged Read 0x1: Event is true (pending). Write 0x1: Event status bit is reset.	RW	0
10	FW_ERR_IRQ	Frame height error Read 0x0: Event false Write 0x0: Event status bit unchanged Read 0x1: Event is true (pending). Write 0x1: Event status bit is reset.	RW	0
9	FSC_ERR_IRQ	False synchronization code	RW	0

Bits	Field Name	Description	Type	Reset
		Read 0x0: Event false		
		Write 0x0: Event status bit unchanged		
		Read 0x1: Event is true (pending).		
		Write 0x1: Event status bit is reset.		
8	SSC_ERR_IRQ	Shifted synchronization code	RW	0
		Read 0x0: Event false		
		Write 0x0: Event status bit unchanged		
		Read 0x1: Event is true (pending).		
		Write 0x1: Event status bit is reset.		
7:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
4	FIFO_NO_EMPTY_ IRQ	FIFO is not empty.	RW	0
		Read 0x0: Event false		
		Write 0x0: Event status bit unchanged		
		Read 0x1: Event is true (pending).		
		Write 0x1: Event status bit is reset.		
3	FIFO_FULL_IRQ	FIFO is full.	RW	0
		Read 0x0: Event false		
		Write 0x0: Event status bit unchanged		
		Read 0x1: Event is true (pending).		
		Write 0x1: Event status bit is reset.		
2	FIFO_THR_IRQ	FIFO threshold has been reached.	RW	0
		Read 0x0: Event false		
		Write 0x0: Event status bit unchanged		
		Read 0x1: Event is true (pending).		
		Write 0x1: Event status bit is reset.		
1	FIFO_OF_IRQ	FIFO overflow has occurred.	RW	0
		Read 0x0: Event false		
		Write 0x0: Event status bit unchanged		
		Read 0x1: Event is true (pending).		
		Write 0x1: Event status bit is reset.		
0	FIFO_UF_IRQ	FIFO underflow has occurred.	RW	0
		Read 0x0: Event false		
		Write 0x0: Event status bit unchanged		
		Read 0x1: Event is true (pending).		
		Write 0x1: Event status bit is reset.		

Table 17-22. CAM_CC_IRQENABLE

Address Offset	0x1C		
Physical Address	0x0970 001C	**Instance**	Camera core
Description	Allows enabling/disabling of module internal sources of interrupt on an event-by-event basis (CCP and parallel mode)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												FS_IRQ_EN	LE_IRQ_EN	LS_IRQ_EN	FE_IRQ_EN	Reserved				FSP_ERR_IRQ_EN	FW_ERR_IRQ_EN	FSC_ERR_IRQ_EN	SSC_ERR_IRQ_EN	Reserved			FIFO_NO_EMPTY_ IRQ_EN	FIFO_FULL_IRQ_EN	FIFO_THR_IRQ_EN	FIFO_OF_IRQ_EN	FIFO_UF_IRQ_EN

Bits	Field Name	Description	Type	Reset
31:20	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000
19	FS_IRQ_EN	Frame-start interrupt enable	RW	0
		0x0: Event is masked.		
		0x1: Event generates an interrupt when it occurs.		
18	LE_IRQ_EN	Line-end interrupt enable	RW	0
		0x0: Event is masked.		
		0x1: Event generates an interrupt when it occurs.		
17	LS_IRQ_EN	Line-start interrupt enable	RW	0
		0x0: Event is masked.		
		0x1: Event generates an interrupt when it occurs.		
16	FE_IRQ_EN	Frame-end interrupt enable	RW	0
		0x0: Event is masked.		
		0x1: Event generates an interrupt when it occurs.		
15:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
11	FSP_ERR_IRQ_EN	FSP code error	RW	0
		0x0: Event is masked.		
		0x1: Event generates an interrupt when it occurs.		
10	FW_ERR_IRQ_EN	Frame-height error interrupt enable	RW	0
		0x0: Event is masked.		
		0x1: Event generates an interrupt when it occurs.		
9	FSC_ERR_IRQ_EN	False synchronization code interrupt enable	RW	0
		0x0: Event is masked.		
		0x1: Event generates an interrupt when it occurs.		
8	SSC_ERR_IRQ_EN	False synchronization code interrupt enable	RW	0
		0x0: Event is masked.		
		0x1: Event generates an interrupt when it occurs.		
7:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
4	FIFO_NO_EMPTY_ IRQ_EN	FIFO not empty interrupt enable	RW	0
		0x0: Event is masked.		
		0x1: Event generates an interrupt when it occurs.		
3	FIFO_FULL_IRQ_EN	FIFO full interrupt enable	RW	0
		0x0: Event is masked.		
		0x1: Event generates an interrupt when it occurs.		
2	FIFO_THR_IRQ_EN	FIFO threshold interrupt enable	RW	0
		0x0: Event is masked.		
		0x1: Event generates an interrupt when it occurs.		
1	FIFO_OF_IRQ_EN	FIFO overflow interrupt enable	RW	0
		0x0: Event is masked.		

Bits	Field Name	Description	Type	Reset
		0x1: Event generates an interrupt when it occurs.		
0	FIFO_UF_IRQ_EN	FIFO underflow interrupt enable	RW	0
		0x0: Event is masked.		
		0x1: Event generates an interrupt when it occurs.		

Table 17-23. CAM_CC_CTRL

Address Offset	0x40		
Physical Address	0x0970 0040	**Instance**	Camera core
Description	Controls the mode of operation of the camera core block (CCP and parallel mode)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											CC_ONE_SHOT	CC_IF_SYNCHRO	CC_RST	CC_FRAME_TRIG	CC_EN	Reserved		NoBT_SYNCHRO	BT_CORRECT	PAR_ORDERCAM	PAR_CLK_POL	NoBT_HS_POL	NoBT_VS_POL	Reserved				PAR_MODE			CCP_MODE

Bits	Field Name	Description	Type	Reset
31:21	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
20	CC_ONE_SHOT	One-shot capability (one frame captured) This must be set the same time as CC_EN is set to 1. One frame acquisition starts/stops automatically. Read returns 0. 0: No synchro (most applications) 1: Synchro enabled (should never be required)	RW/ Dual	0x0
19	CC_IF_SYNCHRO	Synchronize all camera sensor inputs This must be set during the configuration phase before CC_EN is set to 1. This can be used in very high frequency to avoid dependency on the I/O tiimings. 0: No synchro (most applications) 1: Synchro enabled (should never be required)	RW	0x0
18	CC_RST	Resets the internal finite state-machines of the camera core module by writing 1 to this bit Must be applied when CC_EN = 0 Reads returns 0.	RW	0
17	CC_FRAME_TRIG	Set the modality in which CC_EN works when sensor camera core must be disabled: If CC_FRAME_TRIG = 1, writing 0 to CC_EN disables the module at the end of the frame. If CC_FRAME_TRIG = 0, writing 0 to CC_EN disables the module immediately.	RW	0x0
16	CC_EN	Enables the sensor interface of the camera core module: Writing 1 to this field enables the module. Writing 0 to this field disables the module at the end of the frame if CC_FRAM_TRIG =1, and disables the module immediately if CC_FRAM_TRIG = 0.	RW	0x0
15:14	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
13	NoBT_SYNCHRO	Enables a start at the beginning of the frame or not in NoBT: 0: Acquisition starts when vertical synchro is high. 1: Acquisition starts when vertical synchro goes from low to high (beginning of the frame) _UnicodeEncodeError_ recommended.	RW	0x1

Bits	Field Name	Description	Type	Reset
12	BT_CORRECT	Enables correction within the sync codes in BT mode: 0: Correction is not enabled. 1: Correction is enabled.	RW	0x1
11	PAR_ORDERCAM	Enables swap between image data in parallel mode: 0: Swap is not enabled. 1: Swap is enabled.	RW	0x0
10	PAR_CLK_POL	Inverts the clock coming from the sensor in parallel mode: 0: Clock not inverted—data sampled on rising edge 1: Clock inverted—data sampled on falling edge	RW	0x0
9	NoBT_HS_POL	Sets the polarity of the synchronization signals in NoBT parallel mode: 0: CAM_P_HS is active high. 1: CAM_P_HS is active low.	RW	0x0
8	NoBT_VS_POL	Sets the polarity of the synchronization signals in NoBT parallel mode: 0: CAM_P_VS is active high. 1: CAM_P_VS is active low.	RW	0x0
7:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	
3:1	PAR_MODE	Sets the protocol mode of the camera core module in parallel mode (when CCP_MODE = 0): 000: Parallel NoBT 8-bit 001: Reserved 010: Reserved 011: Reserved 100: Parallel BT 8-bit 101: Reserved 110: Reserved 111: FIFO test mode (see Table 17-12)	RW	0x0
0	CCP_MODE	Sets the camera core in CCP mode: 0: CCP mode disabled 1: CCP mode enabled	RW	0x1

CAUTION

In CCP mode (CCP_MODE set to 1), software developers must configure PAR_MODE to 0x0 (reset value).

Table 17-24. CAM_CC_CTRL_DMA

Address Offset	0x44		
Physical Address	0x0970 0044	**Instance**	Camera core
Description	Controls the DMA mode of the camera core (CCP and parallel mode)		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						DMA1_DISABLE	DMA_EN	Reserved							
																								FIFO_ THRESHOLD							

Bits	Field Name	Description	Type	Reset
31:10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000000
9	DMA1_DISABLE	DMA1 disable feature. Uses only DMA0 (threshold-based)	RW	0x0
		0x0: DMA1 line can be activated.		
		0x1: DMA1 line cannot be activated.		
8	DMA_EN	Sets the number of DMA request lines:	RW	1
		0x0: DMA interface disabled (The DMA request line stays inactive.)		
		0x1: DMA interface enabled (The DMA request line is operational.)		
7	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
6:0	FIFO_ THRESHOLD	Sets the threshold of the FIFO	RW	0x07
		The assertion of the DMA request line occurs when the threshold is reached:		
		0b0000000 (0x00) threshold set to 1		
		0b0000001 (0x01) threshold set to 2		
		…		
		0b1111111 (0x7F) threshold set to 128		

Table 17-25. CAM_CC_CTRL_XCLK

Address Offset	0x48		
Physical Address	0x0970 0048	**Instance**	Camera core
Description	This register controls the value of the clock divisor used to generate the external clock (parallel mode).		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								XCLK_DIV							

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
4:0	XCLK_DIV	Sets the clock divisor value for CAM_XCLK generation based on CAM_MCLK Other values: reserved	RW	0x00
		0x0: CAM_XCLK stable low-level divider not enabled		
		0x1: CAM_XCLK stable high-level divider not enabled		
		From 0x2 to 0x1E: CAM_XCLK = CAM_MCLK/XCLK_DIV		
		0x1F: CAM_XCLK = CAM_MCLK (bypass mode)		

Note: See Table 17-13 for details about the ratio of the CAM_XCLK frequency.

Table 17-26. CAM_CC_FIFODATA

Address Offset	0x4C		
Physical Address	0x0970 004C	**Instance**	Camera core
Description	This register enables writing to and reading from the FIFO (CCP and parallel mode).		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFO_DATA																															

Bits	Field Name	Description	Type	Reset
31:0	FIFO_DATA	Writes the 32-bit word into the FIFO	RW	0x00000000

Note: See Table 17-12 for details about the FIFO write and read accesses to this register bank.

Table 17-27. CAM_CC_TEST

Address Offset	0x50		
Physical Address	0x0970 0050	**Instance**	Camera core
Description	This register shows the status of some variables of the camera core module (CCP and parallel mode).		
Type	R		

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
FIFO_RD_POINTER	FIFO_WR_POINTER	FIFO_LEVEL	FIFO_LEVEL_PEAK

Bits	Field Name	Description	Type	Reset
31:24	FIFO_RD_POINTER	Value of the FIFO read pointer. Expected values range from 0 to 127.	R	0x00
23:16	FIFO_WR_POINTER	Value of the FIFO write pointer Expected values range from 0 to 127.	R	0x00
15:8	FIFO_LEVEL	FIFO level (number of 32-bit words the FIFO contains). Can assume values between 0 and 128.	R	0x00
7:0	FIFO_LEVEL_PEAK	FIFO level peak (maximum value of the FIFO level). Can assume values between 0 and 64.	R	0x00

Table 17-28. CAM_CC_GENPAR

Address Offset	0x54		
Physical Address	0x0970 0054	**Instance**	Camera core
Description	This register shows the values of the generic parameters (CCP and parallel mode).		
Type	R		

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Reserved			CAM_CC_FIFO_ DEPTH

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000000
2:0	CAM_CC_FIFO_ DEPTH	FIFO depth generic parameter	R	0x7

Table 17-29. CAM_CC_CCPFSCR

Address Offset	0x58		
Physical Address	0x0970 0058	**Instance**	Camera core
Description	This register contains the frame-start code (CCP mode).		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCPFSCR																															

Bits	Field Name	Description	Type	Reset
31:0	CCPFSCR	Frame-start code	RW	0xFF000002

Table 17-30. CAM_CC_CCPFECR

Address Offset	0x5C		
Physical Address	0x0970 005C	**Instance**	Camera core
Description	This register contains the frame-end code (CCP mode).		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCPFECR																															

Bits	Field Name	Description	Type	Reset
31:0	CCPFECR	Frame-end code	RW	0xFF000003

Table 17-31. CAM_CC_CCPLSCR

Address Offset	0x60		
Physical Address	0x0970 0060	**Instance**	Camera core
Description	This register contains the line-start code (CCP mode).		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCPLSCR																															

Bits	Field Name	Description	Type	Reset
31:0	CCPLSCR	Line-start code register	RW	0xFF000000

Table 17-32. CAM_CC_CCPLECR

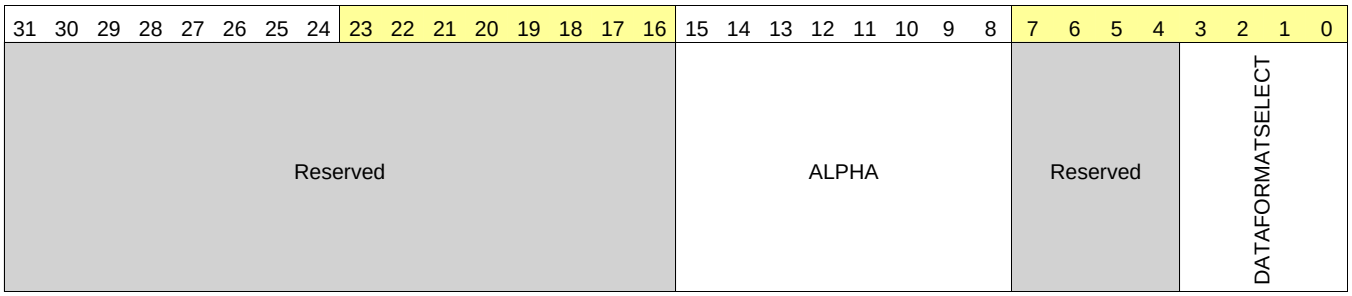
Address Offset	0x64		
Physical Address	0x0970 0064	**Instance**	Camera core
Description	This register contains the line-end code (CCP mode).		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCPLECR																															

Bits	Field Name	Description	Type	Reset
31:0	CCPLECR	Line-end code	RW	0xFF000001

Table 17-33. CAM_CC_CCPDFR

Address Offset	0x68		
Physical Address	0x0970 0068	**Instance**	Camera core
Description	This register sets the data format and controls the receive state-machine (CCP mode).		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:8	ALPHA	Alpha parameter for RGB888 and RGB444 (transparency)	RW	0x00
7:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
3:0	DATAFORMATSELECT	Sets the data format in CCP mode	RW	0x0
		0x0: YUV422 big endian		
		0x1: YUV422		
		0x2: YUV420		
		0x3: Reserved		
		0x4: RGB444		
		0x5: RGB565		
		0x6: RGB888 (no data expansion)		
		0x7: RGB888 (data expansion)		
		0x8: RAW8 (no data expansion)		
		0x9: RAW8 (data expansion)		
		0xA: RAW10 (no data expansion)		
		0xB: RAW10 (data expansion)		
		0xC: RAW12 (no data expansion)		
		0xD: RAW12 (data expansion)		
		0xE: JPEG8FSP		
		0xF: JPEG8		

Note: In JPEG8FSP format, the synchronization codes (CCPFSCR, CCPFECR, CCPLSCR, and CCPLECR field values) must be the defaults, or at least FF0000xx, where xx is different from 0xA5.

Configuration

This chapter discusses the configuration of the LOCOSTO device.

Topic **Page**

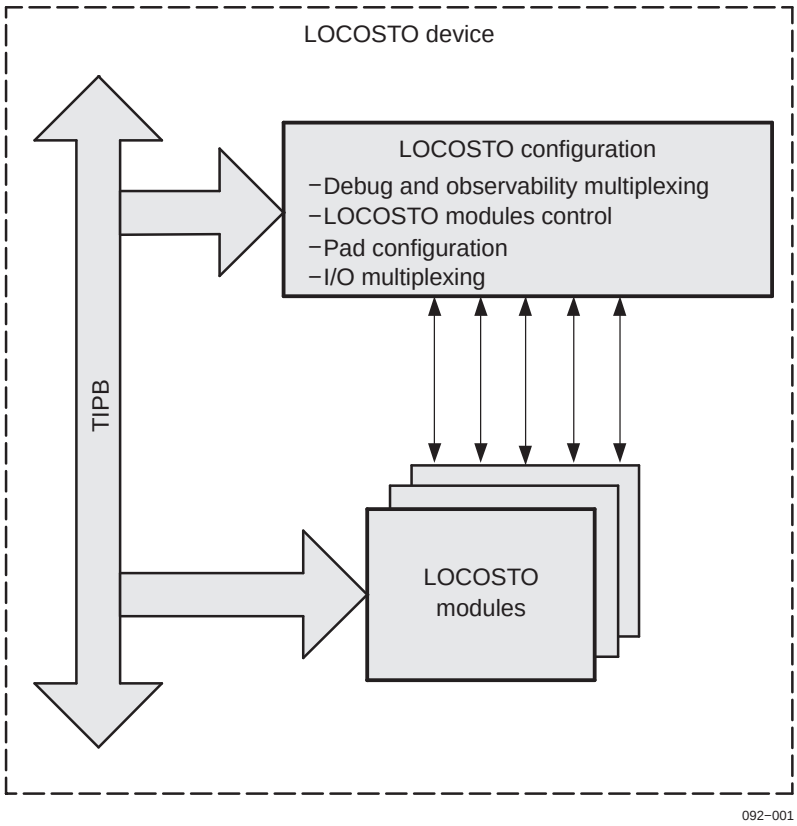
18.1 Configuration Overview

The LOCOSTO configuration is the primary point of control for the following LOCOSTO device modules:

- Static device configuration
- Debug and observability input/output (I/O) multiplexing
- Functional I/O multiplexing
- Pad configuration (pullup or pulldown enable)

Figure 18-1 shows the configuration of the LOCOSTO device.

Figure 18-1. Configuration Highlight



The LOCOSTO configuration implements a bank of registers accessible (read/write) by software. These registers are divided into the following classes:

- Static device configuration registers (16-bit read/write registers for module-specific configuration)
- Debug and observability I/O multiplexing registers (16- and 8-bit read/ write registers)
- Pad functional multiplexing and configuration registers (16-bit registers)

The LOCOSTO device can be configured for the following functions:

- Universal serial bus (USB) mode control
- Universal subscriber identity module (USIM) voltage control
- Camera, liquid crystal display (LCD), and NAND flash shared interface control
- Software debug override configuration

The configuration module includes device controls used for the following:

- I/O multiplexing
- Debug signal multiplexing

18.2 Functional Configuration

This section describes registers used by software to configure functions of the LOCOSTO device.

18.2.1 LOCOSTO Functional Configuration Registers Summary

Table 18-1 lists the functional configuration registers and their physical addresses.

Table 18-1. Functional Configuration Registers and Physical Addresses

Register Name	Type	Register Width (Bits)	Physical Addresses
CONF_CORE_REG	R/W	16	0xFFFE F01C
CONF_LCD_CAM_ND	R/W	16	0xFFFE F01E
CONF_LOCOSTO_DEBUG	R/W	16	0xFFFE F020

18.2.2 LOCOSTO Functional Configuration Registers Description

Table 18-2 through Table 18-4 describe the individual register bits.

Table 18-2. CONF_CORE_REG

Address Offset	0x1C		
Physical Address	0xFFFE F01C	**Instance**	Configuration
Description	This register contains the core configuration.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAM_TEST_ FAST_CTRL	CAM_CCP_MODE	USIM_FMODE		USIM_VMODE	REG_USBX_ SYNCHRO	USB_TRX_MODE		USB_PRT		USB_FILTER_CD				USB_VBUS_MODE	USB_VBUS_CTRL

Bits	Field Name	Description	Type	Reset
15	CAM_TEST_ FAST_CTRL	0: Normal mode when in CCP mode 1: Fast mode when in CCP mode	R/W	0b0
14	CAM_CCP_MODE	0: CCP mode is not enabled. CONF_GPIO [22:19] is transparently controlling the multiplexing on the GPIO [22:19] pads. 1: CCP mode is enabled. CKP is selected on gpio_19. CKN is selected on gpio_20. DTP is selected on gpio_21. DTN is selected on gpio_22.	R/W	0b0
13:12	USIM_FMODE	00: Off 3 V 01: Fast start 3 V 10: Fast start 3 V 11: High impedance 3 V	R/W	0b00
11	USIM_VMODE	0: 1.8-V mode 1: Different 3-V modes depending on USIM_FMODE	R/W	0b1
10	REG_USBX_ SYNCHRO	This bit is used to avoid glitches on the host and client path, if delay is not balanced, adding a delay of one clock cycle. The USB module clock is 48 MHz. 0: No delay is present on the output path. 1: One clock cycle delay is added to the output signals.	R/W	0b1

Bits	Field Name	Description	Type	Reset
9:7	USB_TRX_MODE	This bit selects the 3-pin or 4-pin configuration of USB. Based on the choice of number of pins, the configuration of USB changes. 000: 3-pin interface 001: 4-pin interface Others: Reserved	R/W	0b000
6:5	USB_PRT	USB port is available. 00: USB port is not available. 01: USB port is available. Others: Reserved	R/W	0b01
4:2	USB_FILTER_CD	This field controls the USB filter time. Each time 1 is added to the field, 5 ms is added to the filter time. 000: 0 ms or no filter 001: 05 ms 010: 10 ms 011: 15 ms 100: 20 ms 101: 25 ms 110: 30 ms 111: 0 ms or no filter	R/W	0b000
1	USB_VBUS_MODE	This bit selects the USB vbus input source, used for USB insertion/disconnection detection. 0: USB vbus pin uses USB_VBUS_CTRL bit value. 1: USB vbus pin uses UIS480 detection cell output.	R/W	0b0
0	USB_VBUS_CTRL	This bit indicates an external USB insertion/ disconnection. 0: Indicates an external USB disconnection 1: Indicates an external USB insertion	R/W	0b0

Table 18-3. CONF_LCD_CAM_ND[(1)]

Address Offset	0x1E		
Physical Address	0xFFFE F01E	**Instance**	Configuration
Description	This register indicates to override the individual pad programming capability for the shared interface LCD_DATA [7:0] bus. The complete LCD, camera, or NAND flash can be switched completely from one interface to another by programming this register.		
Type	R/W		
Not relevant	Not relevant		

(1) This register also overrides the configuration register value for the pads corresponding to the secondary multiplexing of NAND flash and camera. This register must be programmed to 0 if NAND flash or camera is required on the secondary interface. One-register programming does not exist for the secondary interface of the camera and NAND flash.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													SIM_PWRDN	CONF_LCD_ CAM_ND	

Bits	Field Name	Description	Type	Reset
15:3	RESERVED		R/W	0x0000
2	SIM_PWRDN	Power down control to the SIM IO interface = Default value 1 0: SIM interface is powered and normal functionality is ensured. This bit should not be zero before switching on the power to the SIM IF. 1: SIM interface is powered down.	R/W	0x1

Bits	Field Name	Description	Type	Reset
1:0	CONF_LCD_ CAM_ND	00: The CONF_LCD [7:0] controls the choice of functionality mapped. 01: CAM_D [7:0] is present on LCD_DATA [7:0]. 10: ND[7:0] is present on LCD_DATA [7:0]. 11: LCD_DATA [7:0] is present on the LCD_DATA [7:0].	R/W	0x0

For more information on how Table 18-3 is used in USB, see Chapter 23, *USB Client*.

Table 18-4. CONF_LOCOSTO_DEBUG

Address Offset	0x20		
Physical Address	0xFFFE F020	**Instance**	Configuration
Description	This register contains the software debug override configuration.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											CLK13M_OFF_ VALUE	CUT_CLK13M	SECURE	ALL_ZERO_ MODE	ALL_ONE_MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED			
4	CLK13M_OFF_ VALUE	When CUT_CLK13M is at 1 after synchronization, the CLK13M_OUT follows the value stored in this bit. (It is then usable as general-purpose output.)	R/W	0b0
3	CUT_CLK13M	0: Enable Clock 13 MHz 1: Disable to CLK13M_OFF_VALUE	R/W	0b0
2	SECURE	This bit can be programmed to enable the security feature, when no software debug signals inside the core can be monitored. 0: Normal debug values are available from the debug module. 1: All debug observation nodes are forced to 0.	R/W	0b0
1	ALL_ZERO_ MODE	This bit can be programmed to force to 0 all outputs from the LOCOSTO software debug module. 0: Normal debug values are available from the debug module. 1: All debug observation nodes are forced to 1.	R/W	0b0
0	ALL_ONE_MODE	This bit can be programmed to force to 1 all outputs from the LOCOSTO software debug module. 0: Normal debug values are available from the debug module. 1: All debug observation nodes are forced to 0.	R/W	0b0

Table 18-5. FORCE_KBC

Address Offset	0x1EA		
Physical Address	0xFFFE F000	**Instance**	Configuration
Description			
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															FORCE_KBC

Bits	Field Name	Description	Type	Reset
15:1	RESERVED		RW	undefined
0	FORCE_KBC	The functionality is built in to Locosto to disable the cross-coupling of the Key press noise to the RF when active. The TPU (TSPACT4) can be used to exercise such a control based on the SW scenario. The key board scan output during the suspend state can be configured as under: 0x0: KBC output pads forced to 1 0x1: KBC output pads are tri-stated.	R/W	0

18.3 Pinout Configuration

18.3.1 Pinout Overview

18.3.1.1 Terminology

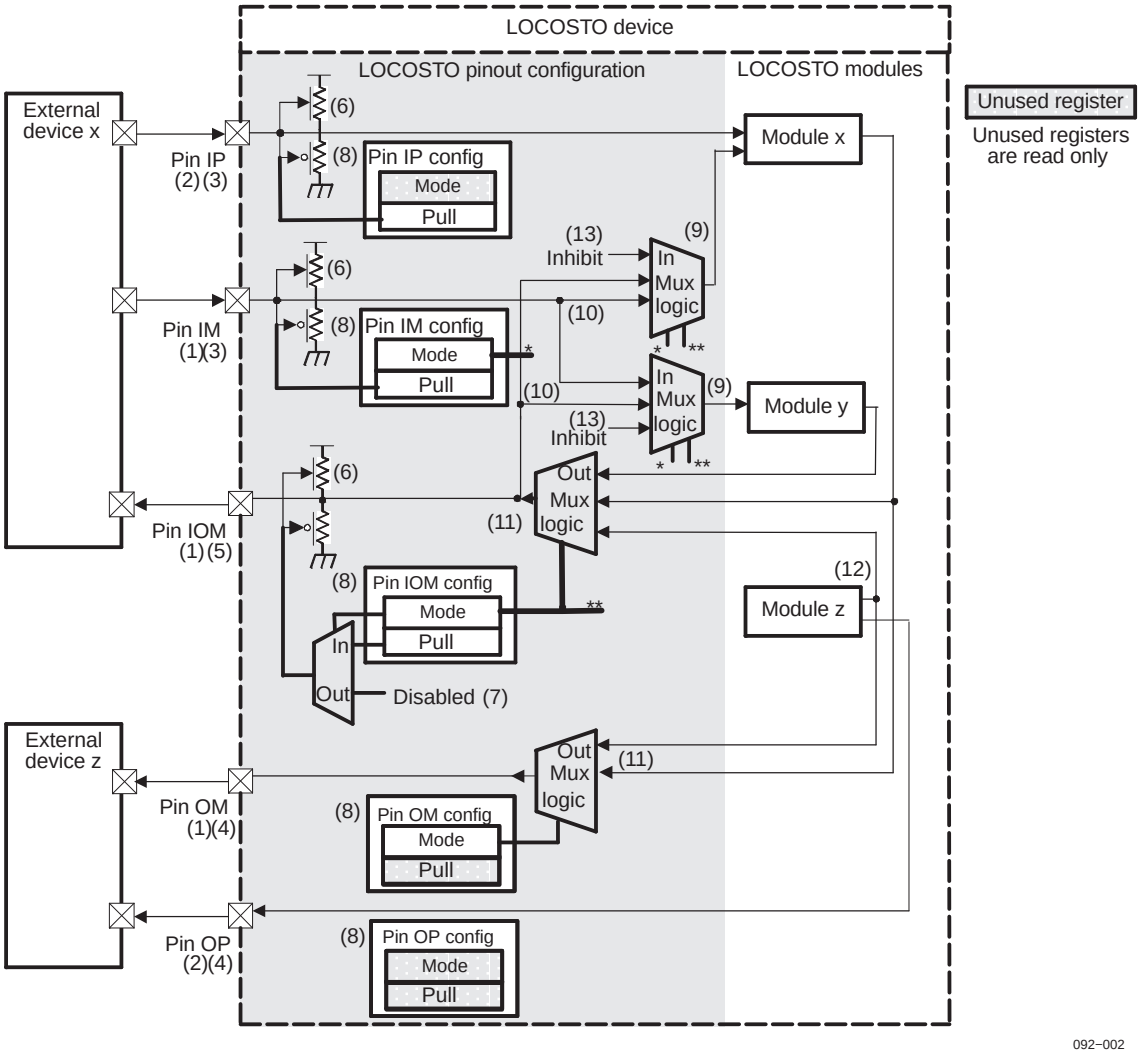
- Pure input: A pure input pin can be multiplexed only on module inputs.
- Pure output: A pure output pin can be multiplexed only between module outputs.
- Muxed pin: A pin is multiplexed when its pin configuration register field can be reconfigured by software to change the function associated with the pin.
- In mux logic: Combinational logic applied to input functional signal multiplexing
- Out mux logic: Combinational logic used in output signal functional multiplexing
- Dedicated pin: A dedicated pin is hardwired to only one function.
- Pinout configuration register: 16-bit register that contains up to two different pin configuration register fields
- Default mode: The mode selected at the release of the power-on reset. It corresponds to mode 0.
- Mode: The 3-bit field of the pin configuration register used to change the mode. Mode programming is assumed by software selecting a function on the external interface of the LOCOSTO device.
- Pull: The 2-bit field of the pin configuration register that enables the pull on input and selects pullup or pulldown

18.3.1.2 LOCOSTO Device Pinout Overview

The LOCOSTO device can be interfaced easily with any external system because of its programmable pinout. Pinout control enables flexible configuration of each pin to match system requirements. The default configuration, which has been chosen to limit electrical contention risks, can be changed by simple software access to the pinout configuration registers. Numerous combinations of internal modules and pins are available to cover a maximum number of applicative solutions.

Although Figure 18-2 is not all inclusive, it shows the different pin categories with their respective control logic.

Figure 18-2. Pinout Overview



NOTE: PI = Pure Input, MPI = Muxed Pure Input, MIO = Muxed Input/Output, MPO = Muxed Pure Output, PO = Pure Output

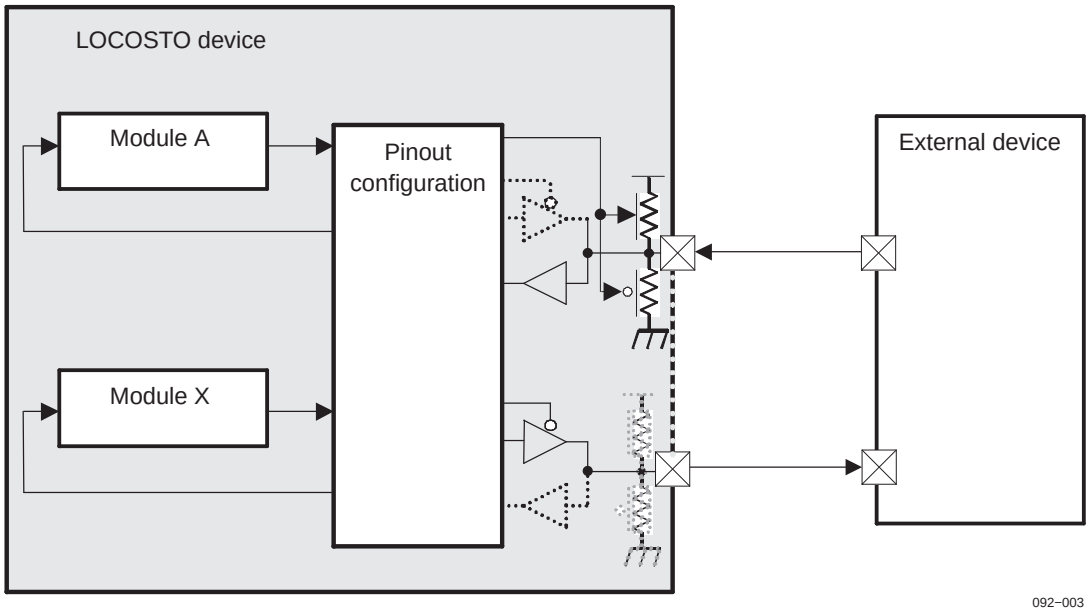
The LOCOSTO device shown in Figure 18-2 has the following features (the numbers in parentheses correspond to those in the figure):

- 163 pins controlled for applicative purposes on the LOCOSTO device
- Support for both muxed (1) and dedicated pins (2)
- Support for pure inputs (3), pure outputs (4), and I/O pins (5)
- Software-controlled combinational pullup/pulldown (6)
- Pull automatically disabled when configured pad is in output (7)
- One control register for each pin (8)
- Support access from different pins to one module input (9)
- Ability to map one pin on inputs from different modules (10)
- Ability to configure one pin to support outputs from different modules (11)
- Ability to configure one module output on several different pads (12)
- Hardware protection to avoid internal electrical contention in case of a bad configuration (13)

18.3.2 Pinout Environment

Pinout configuration enables the routing of the internal module I/O to the device boundary for interconnection with an external device (see Figure 18-3).

Figure 18-3. Pinout Environment Overview



The software must ensure proper programming of the pinout configuration module for each external device interfacing with the LOCOSTO device. Several pins can be independently programmed to achieve maximum flexibility.

18.3.3 Pinout Integration

The pinout configuration module contains a bank of registers that directly control the I/O pins. The pinout configuration is included in the core power domain. The pinout configuration can be reset only by the internal power-on reset (core_ON_nOFF).

18.3.3.1 Clocking, Reset, and Power-Management Scheme

- Clocking: The main sequential logic within the pinout configuration is accessible in a register file through the TI peripheral bus (TIPB). The only clock provided to the pinout configuration is the interface clock.
- Hardware reset: The pinout configuration is sensitive only to the core_ON_nOFF reset, which is the power-on reset.
- Software reset: No software reset is available for the pinout configuration module.
- Power domain: The pinout configuration module is part of the core power domain.

18.3.3.2 Application Pin Characteristics

Table 18-6 lists the characteristics of each application pad configuration. 163 pins are dedicated to applicative pins on the LOCOSTO device package. The following describes the application pin characteristics:

- Ball Address: The ball index of the pin on the package (see the LOCOSTO device package)
- Pin Name: Name of the pin. This name refers to the primary function of the pin (equivalent to Mode 0).
- Dir: I for pure input, O for pure output, and IO for IO pins
- Mux Type: Muxed when the pin can be multiplexed on several different functions. Dedicated when the pin is hardwired to only one function.
- Configuration Register Name: Name of the pinout configuration register. This register contains the

pinout configuration register field for pin configuration.

- Register Address: Physical address for the 16 bit pinout configuration register. See Section 18.3.7, *LOCOSTO I/O Multiplexing Registers*, for a detailed description of the LOCOSTO configuration registers.
- Register Field: Gives the position of the pinout configuration register fields used inside the 16-bit pinout configuration register. Unused bits are read-only. See Section 18.3.7, *LOCOSTO I/O Multiplexing Registers*, for more details on LOCOSTO configuration register fields.
- Register Reset Value: Gives the default value of a pinout configuration register
- Pull: Defines the type of pull cell available internally on the pin:
  - PUPD: Combination of a pullup and a pulldown (software programmable)
  - PU: Pullup (enable or disable by software programming)
  - PD: Pulldown (enable or disable by software programming)

  Gray shading in Table 18-6 indicates that no internal pull is available for this pin.
- Pull Reset Status: Gives the state of the pull on the pin:
  - U: Pullup active by default (at release of the power on reset)
  - D: Pulldown active by default (at release of the power on reset)
  - _UnicodeEncodeError_: No pull active by default

Gray shading in Table 18-6 indicates that no internal pull is available for this pin.

Note: Yellow shading in Table 18-6 indicates a static configuration that cannot be modified by software.

Table 18-6. LOCOSTO Application Pin Characteristics

Pin			Pinout Configuration Register				Alternate Functions						Pull	
Pin Name	**Dir**	**Mux Type**	**Configuration Register Name**	**Register Address**	**Register Field**	**Reset Register Value**	**Mode0**	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**	**PU PD Typ e**	**Pull Reset Status**
tck	I	Muxed	CONF_ TCK	0xFFFE F188	[4:3]	01000b	tck						PU PD	D
sense	O	Dedicated					sense							
tms	I	Muxed	CONF_ TMS	0xFFFE F182	[4:3]	11000b	tms						PU PD	U
gpio_ 21	IO	Muxed	CONF_ GPIO_21	0xFFFE F16E	[1:0]	00000b	gpio_21	cam_ lclk	test_ port_ 30					
gpio_ 18	IO	Muxed	CONF_ GPIO_18	0xFFFE F158	[4:3] and [0:0]	11000b	gpio_18	nd_we					PU PD	U
tdi	I	Muxed	CONF_ TDI	0xFFFE F18C	[4:3]	11000b	tdi						PU PD	U
lcd_ stb	IO	Muxed	CONF_ LCD_STB	0xFFFE F150	[4:3] and [1:0]	00000b	lcd_stb	gpio_14	test_ port_ 11				PU PD	_Unicode EncodeEr ror_
gpio_ 12	IO	Muxed	CONF_ GPIO_12	0xFFFE F14A	[4:3] and [1:0]	01000b	gpio_12	lpg	tspact_10				PU PD	D
kbc_4	IO	Muxed	CONF_ GPIO_9	0xFFFE F144	[4:3] and [1:0]	10000b	kbc_4	gpio_9	test_ port_ 13				PU PD	_Unicode EncodeEr ror_
i_force	I	Dedicated					i_force							
tdo	O	Dedicated	CONF_ TDO	0xFFFE F1BC	Reserved	00000b	tdo							
gpio_ 30	IO	Muxed	CONF_ GPIO_30	0xFFFE F180	[4:3] and [1:0]	01000b	gpio_30	lt3	ndf5	cam_d_5			PD	D
lcd_ data_7	IO	Muxed	CONF_ LCD_ DATA_7	0xFFFE F168	[4:3] and [1:0]	00000b	lcd_ data_7	cam0_ d_7	test_ port_0	ndf_ dyn_7				
lcd_ data_4	IO	Muxed	CONF_ LCD_ DATA_4	0xFFFE F162	[4:3] and [1:0]	00000b	lcd_ data_4	cam0_ d_4	test_ port_3	ndf_ dyn_4				
lcd_ data_1	IO	Muxed	CONF_ LCD_ DATA_1	0xFFFE F15C	[1:0]	00000b	lcd_ data_1	cam0_ d_1	test_ port_6	ndf_ dyn_1				
gpio_ 17	IO	Muxed	CONF_ GPIO_17	0xFFFE F156	[4:3] and [1:0]	11000b	gpio_17	lcd_ncs1	test_ port_8				PU PD	U
lcd_ nrst	IO	Muxed	CONF_ LCD_NRST	0xFFFE F14E	[4:3] and [0:0]	00000b	lcd_nrst	test_ port_12					PD	_Unicode EncodeEr ror_
nemu0	IO	Muxed	CONF_ GPIO_10	0xFFFE F146	[4:3] and [0:0]	11000b	nemu0	gpio_10					PU PD	U

Table 18-6. LOCOSTO Application Pin Characteristics (continued)

Pin			Pinout Configuration Register				Alternate Functions						Pull	
Pin Name	Dir	Mux Type	Configuration Register Name	Register Address	Register Field	Reset Register Value	Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	PU PD Type	Pull Reset Status
kbr_2	IO	Muxed	CONF_KBR_2	0xFFFEF13E	[4:3] and [0:0]	11000b	kbr_2	est_port_20					PU PD	U
tspact_13	IO	Dedicated	CONF_TSPACT_13	0xFFFEF134	Reserved	00000b	tspact_13							
kbc_3	O	Muxed	CONF_KBC_3	0xFFFEF12E	[0:0]	00000b	kbc_3	test_port_14						
kbc_1	O	Muxed	CONF_KBC_1	0xFFFEF12A	[0:0]	00000b	kbc_1	test_port_16						
gpio_34	IO	Muxed	CONF_GPIO_34	0xFFFEF190	[4:3] and [1:0]	11000b	gpio_34	nd_rdy	test_port_24				PU	U
gpio_31	IO	Muxed	CONF_GPIO_31	0xFFFEF186	[4:3] and [1:0]	11000b	gpio_31	nd_re	test_port_27				PU	U
gpio_28	IO	Muxed	CONF_GPIO_28	0xFFFEF17C	[4:3] and [1:0]	11000b	gpio_28	spi_ncs2	ndf7	cam_d_7			PU	U
gpio_25	IO	Muxed	CONF_GPIO_25	0xFFFEF176	[4:3] and [0:0]	01000b	gpio_25	spi_data_mosi					PD	D
gpio_22	IO	Muxed	CONF_GPIO_22	0xFFFEF170	[2:0]	00000b	gpio_22	cam_xclk	cam_d_3	test_port_29	ndf3			
lcd_data_6	IO	Muxed	CONF_LCD_DATA_6	0xFFFEF166	[4:3] and [1:0]	00000b	lcd_data_6	cam0_d_6	test_port_1	ndf_ dyn_6				
lcd_data_2	IO	Muxed	CONF_LCD_DATA_2	0xFFFEF15E	[1:0]	00000b	lcd_data_2	cam0_d_2	test_port_5	ndf_ dyn_2				
nemu1	IO	Muxed	CONF_GPIO_11	0xFFFEF148	[4:3] and [0:0]	11000b	nemu1	gpio_11					PU PD	U
xtal	I	Dedicated					xtal							
gpio_36	IO	Muxed	CONF_GPIO_36	0xFFFEF196	[4:3] and [1:0]	11000b	gpio_36	ncs1	ndf2				PU	U
nd_ce1	O	Muxed	CONF_ND_CE1	0xFFFEF192	[0:0]	00000b	nd_ce1	test_port_23						
lcd_data_5	IO	Muxed	CONF_LCD_DATA_5	0xFFFEF164	[4:3] and [1:0]	00000b	lcd_data_5	cam0_d_5	test_port_2	ndf_ dyn_5				
lcd_rs	IO	Muxed	CONF_LCD_RS	0xFFFEF154	[4:3] and [1:0]	00000b	lcd_rs	gpio_16	test_port_9				PU PD	_UnicodeEncodeError_

Table 18-6. LOCOSTO Application Pin Characteristics (continued)

Pin			Pinout Configuration Register				Alternate Functions						Pull	
Pin Name	**Dir**	**Mux Type**	**Configuration Register Name**	**Register Address**	**Register Field**	**Reset Register Value**	**Mode0**	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**	**PU PD Typ e**	**Pull Reset Status**
gpio_13	IO	Muxed	CONF_GPIO_13	0xFFFEF14C	[4:3] and [0:0]	11000b	gpio_13	lcd_ncs0					PU PD	U
kbr_1	IO	Muxed	CONF_KBR_1	0xFFFEF13C	[4:3] and [0:0]	11000b	kbr_1	test_port_21					PU PD	U
tspact_14	IO	Muxed	CONF_TSPACT_14	0xFFFEF136	Reserved	00000b	tspact_14							
kbc_2	O	Muxed	CONF_KBC_2	0xFFFEF12C	[0:0]	00000b	kbc_2	test_port_15						
vref	O	Dedicated					vref							
ncs0	IO	Muxed	CONF_GPIO_38	0xFFFEF19A	[4:3] and [2:0]	00000b	ncs0	gpio_38	lt1	tspact_7	ndf0		PU	_UnicodeEncodeError_
nd_nwp	IO	Muxed	CONF_ND_NWP	0xFFFEF184	[1:0]	00000b	nd_nwp	ndf4	cam_d_4	test_port_28				
gpio_26	IO	Muxed	CONF_GPIO_26	0xFFFEF178	[4:3] and [0:0]	01000b	gpio_26	spi_ncs0					PU PD	D
gpio_20	IO	Muxed	CONF_GPIO_20	0xFFFEF16C	[2:0]	00000b	gpio_20	cam_vs	cam_d_3	test_port_ 31	ndf3			
lcd_data_3	IO	Muxed	CONF_LCD_DATA_3	0xFFFEF160	[1:0]	00000b	lcd_data_3	cam0_d_3	test_port_4	ndf_dyn_3				
lcd_data_0	IO	Muxed	CONF_LCD_DATA_ 0	0xFFFEF15A	[1:0]	00000b	lcd_data_0	cam0_d_0	test_port_7	ndf_dyn_0				
add_17	O	Dedicated					add_17							
kbr_4	IO	Muxed	CONF_GPIO_8	0xFFFEF142	[4:3] and [1:0]	11000b	kbr_4	gpio_8	test_port_18				PU PD	U
kbr_0	IO	Muxed	CONF_KBR_0	0xFFFEF13A	[4:3] and [0:0]	11000b	kbr_0	test_port_22					PU PD	U
tspact_11	IO	Muxed	CONF_TSPACT_11	0xFFFEF130	[0:0]	00000b	tspact_11	clkm_clk						
txlb	O	Dedicated					txlb							
add_18	O	Dedicated					add_18							

Table 18-6. LOCOSTO Application Pin Characteristics (continued)

Pin			Pinout Configuration Register				Alternate Functions						Pull	
Pin Name	Dir	Mux Type	Configuration Register Name	Register Address	Register Field	Reset Register Value	Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	PU PD Typ e	Pull Reset Status
add_ 21	IO	Muxed	CONF_ ADD_21	0xFFFE F1B4	[4:3] and [0:0]	00000b	add_21	gpio_6					PD	_Unicode EncodeEr ror_
gpio_ 35	IO	Muxed	CONF_ GPIO_35	0xFFFE F194	[4:3] and [1:0]	11000b	gpio_35	ncs2	ndf3				PU	U
gpio_ 32	IO	Muxed	CONF_ GPIO_32	0xFFFE F18A	[4:3] and [1:0]	01000b	gpio_32	nd_cle	test_ port_26				PD	D
gpio_ 24	IO	Muxed	CONF_ GPIO_24	0xFFFE F174	[4:3] and [0:0]	01000b	gpio_24	spi_ data_ miso					PD	D
gpio_ 19	IO	Muxed	CONF_ GPIO_19	0xFFFE F16A	[2:0]	00000b	gpio_19	cam_hs	cam_ d_3	test_ port_32	ndf3			
lcd_ rnw	IO	Muxed	CONF_ LCD_RNW	0xFFFE F152	[4:3] and [1:0]	00000b	lcd_rnw	gpio_15	test_ port_10				PU PD	_Unicode EncodeEr ror_
add_ data_ 14	IO	Dedicated					add_ data_14							
tspact_ 15	IO	Dedicated	CONF_ TSPACT_ 15	0xFFFE F138	Reserved	00000b	tspact_15							
kbc_0	O	Muxed	CONF_ KBC_0	0xFFFE F128	[0:0]	00000b	kbc_0	test_ port_17						
txhb	O	Dedicated					txhb							
add_ data_13	IO	Dedicated					add_ data_13							
add_16	O	Dedicated					add_16							
gpio_39	IO	Muxed	CONF_ GPIO_39	0xFFFE F19C	[4:3] and [1:0]	01000b	gpio_39	add_22	ndf0				PD	D
gpio_7	IO	Muxed	CONF_ GPIO_7	0xFFFE F1BA	[4:3] and [2:0]	01000b	gpio_7	nfwp	tspact_7	lt1	ndf2	cam_ d_2	PD	D
gpio_29	IO	Muxed	CONF_ GPIO_29	0xFFFE F17E	[4:3] and [1:0]	01000b	gpio_29	bu	ndf6	cam_d_6			PD	D
gpio_27	IO	Muxed	CONF_ GPIO_27	0xFFFE F17A	[4:3] and [0:0]	11000b	gpio_27	spi_ncs1					PU PD	U
gpio_23	IO	Muxed	CONF_ GPIO_23	0xFFFE F172	[4:3] and [0:0]	01000b	gpio_23	spi_clk					PD	D

Table 18-6. LOCOSTO Application Pin Characteristics (continued)

Pin			Pinout Configuration Register				Alternate Functions						Pull	
Pin Name	Dir	Mux Type	Configuration Register Name	Register Address	Register Field	Reset Register Value	Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	PU PD Typ e	Pull Reset Status
tspact_ 12	IO	Dedicated	CONF_ TSPACT_12	0xFFFE F132	Reserved	00000b	tspact_12							
add_ data_10	IO	Dedicated					add_ data_10							
add_ data_11	IO	Dedicated					add_ data_11							
add_ data_15	IO	Dedicated					add_ data_15							
add_ data_12	IO	Dedicated					add_ data_12							
add_19	O	Dedicated					add_19							
gpio_37	IO	Muxed	CONF_ GPIO_37	0xFFFE F198	[4:3] and [1:0]	01000b	gpio_37	add_23	ndf1				PD	D
gpio_33	IO	Muxed	CONF_ GPIO_33	0xFFFE F18E	[4:3] and [1:0]	01000b	gpio_33	nd_ale	test_ port_25				PD	D
kbr_3	IO	Muxed	CONF_ KBR_3	0xFFFE F140	[4:3] and [0:0]	11000b	kbr_3	test_ port_19					PU PD	U
rxpcsm	I	Dedicated					rxpcsm							
add_ data_7	IO	Dedicated					add_ data_7							
add_ data_6	IO	Dedicated					add_ data_6							
add_ data_8	IO	Dedicated					add_ data_8							
add_ data_5	IO	Dedicated					add_ data_5							
add_ data_9	IO	Dedicated					add_ data_9							
adv	IO	Muxed	CONF_ADV	0xFFFE F1A0	[4:3] and [0:0]	00000b	adv	gpio_41					PU	_Unicode EncodeEr ror_
add_20	O	Dedicated					add_20							
rxpcsp	I	Dedicated					rxpcsp							
rxdcsm	I	Dedicated					rxdcsm							
add_ data_4	IO	Dedicated					add_ data_4							

Table 18-6. LOCOSTO Application Pin Characteristics (continued)

Pin			Pinout Configuration Register				Alternate Functions						Pull	
Pin Name	Dir	Mux Type	Configuration Register Name	Register Address	Register Field	Reset Register Value	Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	PU PD Typ e	Pull Reset Status
add_ data_3	IO	Dedicated					add_ data_3							
add_ data_0	IO	Dedicated					add_ data_0							
add_ data_2	IO	Dedicated					add_ data_2							
fdp	O	Dedicated					fdp							
gpio_45	IO	Muxed	CONF_ GPIO_45	0xFFFE F1A8	[4:3] and [1:0]	11000b	gpio_45	mcsi_tx	cdo	test_ port_35			PU	U
gpio_1	IO	Muxed	CONF_ GPIO_1	0xFFFE F102	[4:3] and [1:0]	01000b	gpio_1	pwt	pmc_ reset				PU PD	D
vfsrx	I	Muxed	CONF_ VFSRX	0xFFFE F1B8	[4:3]	00000b	vfsrx						PD	_Unicode EncodeEr ror_
rnw	O	Muxed	CONF_RNW	0xFFFE F1DE	[0:0]	00000b	rnw	force_0						
rxegsm p	I	Dedicated					rxegsmp							
rxdcsp	I	Dedicated					rxdcsp							
add_ data_1	IO	Dedicated					add_ data_1							
nbhe	O	Dedicated					nbhe							
ncs3	O	Muxed	CONF_NCS3	0xFFFE F1E2	[0:0]	00000b	ncs3	force_0						
gpio_42	IO	Muxed	CONF_ GPIO_42	0xFFFE F1A2	[4:3] and [0:0]	11000b	gpio_42	ckm					PU	U
uart_rx	IO	Muxed	CONF_ UART_RX	0xFFFE F1AE	[4:3]	00000b	uart_rx						PD	_Unicode EncodeEr ror_
usb_ se0	IO	Muxed	CONF_ USB_SE0	0xFFFE F10A	[4:3] and [0:0]	00000b	usb_se0	uart_tx					PU	_Unicode EncodeEr ror_
apc spare1	IO	Dedicated					apcspare1							
nmoe	O	Muxed	CONF_NMOE	CONF_ NMOE	[0:0]	00000b	nmoe	force_0						

Table 18-6. LOCOSTO Application Pin Characteristics (continued)

Pin			Pinout Configuration Register				Alternate Functions						Pull	
Pin Name	**Dir**	**Mux Type**	**Configuration Register Name**	**Register Address**	**Register Field**	**Reset Register Value**	**Mode0**	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**	**PU PD Typ e**	**Pull Reset Status**
sim_ pwrctrl	O	Muxed	CONF_SIM_ PWCTRL	0xFFFE F126	[4:3]	00000b	sim_ pwrctrl						PU PD	_Unicode EncodeEr ror_
apcout	IO	Dedicated					apcout							
rxegsm m	I	Dedicated					rxegsmm							
rxgsmp	I	Dedicated					rxgsmp							
nble	O	Dedicated					nble							
nrdy	IO	Muxed	CONF_NRDY	0xFFFE F19E	[4:3] and [0:0]	00000b	nrdy	gpio_40					PU	_Unicode EncodeEr ror_
gpio_44	IO	Muxed	CONF_ GPIO_44	0xFFFE F1A6	[4:3] and [1:0]	11000b	gpio_44	mcsi_fs	csync_o	test_ port_34			PU	U
gpio_0	IO	Muxed	CONF_ GPIO_0	0xFFFE F100	[4:3] and [1:0]	11000b	gpio_0	dcd_txir	cam_d_1				PU	U
usb_ txen	O	Muxed	CONF_ USB_TXEN	0xFFFE F10E	[4:3] and [0:0]	00000b	usb_txen	uart_rts						
vdr	I	Muxed	CONF_VDR	0xFFFE F1E6	[4:3]	00000b	vdr						PD	_Unicode EncodeEr ror_
csync	IO	Muxed	CONF_ CSYNC	0xFFFE F114	[4:3]	01000b	csync						PD	D
on_noff	I	Dedicated					on_noff							
sim_rst	O	Muxed	CONF_ SIM_RST	0xFFFE F120	[4:3]	00000b	sim_rst						PU PD	_Unicode EncodeEr ror_
rxgsmm	I	Dedicated					rxgsmm							
gpio_43	IO	Muxed	CONF_ GPIO_43	0xFFFE F1A4	[4:3] and [1:0]	11000b	gpio_43	mcsi_ck	csclk_o	test_ port_33			PU	U
gpio_47	IO	Muxed	CONF_ GPIO_47	0xFFFE F1B6	[4:3] and [1:0]	11000b	gpio_47	dsr_rxir	cam_d_0				PU	U
i2c_sda	IO	Dedicated					i2c_sda							
scl_trit	IO	Dedicated					scl_trit							
ckout_ 1 3mhz	O	Dedicated					ckout_ 13mhz							
gpio_5	IO	Muxed	CONF_ GPIO_5	0xFFFE F11C	[4:3] and [0:0]	01000b	gpio_5	tspact_8					PU PD	D

Table 18-6. LOCOSTO Application Pin Characteristics (continued)

Pin			Pinout Configuration Register				Alternate Functions						Pull	
Pin Name	Dir	Mux Type	Configuration Register Name	Register Address	Register Field	Reset Register Value	Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	PU PD Typ e	Pull Reset Status
sim_clk	O	Muxed	CONF_ SIM_CLK	0xFFFE F124	[4:3]	00000b	sim_clk						PU PD	_Unicode EncodeEr ror_
gpio_46	IO	Muxed	CONF_ GPIO_46	0xFFFE F1AA	[4:3] and [1:0]	11000b	gpio_46	mcsi_rx	test_ port_36				PU	U
uart_tx	O	Dedicated	CONF_ UART_TX	0xFFFE F1AC	Reserved	00000b	uart_tx							
abb_irq	IO	Muxed	CONF_ABB_ IRQ	0xFFFE F112	[4:3]	11000b	abb_irq						PU	U
csclk	IO	Muxed	CONF_ CSCLK	0xFFFE F116	[4:3]	01000b	csclk						PD	D
sim_io	IO	Muxed	CONF_SIM_ IO	0xFFFE F122	[4:3] and [0:0]	00000b	sim_io	force_0					PU PD	_Unicode EncodeEr ror_
apcldo filter	IO	Dedicated					apcldofilter							
apcvref	IO	Dedicated					apcvref							
iref	O	Dedicated					iref							
vref1	I	Dedicated					vref1							
xanat st6	IO	Dedicated					xanatst6							
uart_cts	IO	Muxed	CONF_ UART_CTS	0xFFFE F1B0	[4:3]	00000b	uart_cts						PD	_Unicode EncodeEr ror_
rts_ sdirda	O	Dedicated					rts_sdirda							
i2c_scl	IO	Dedicated					i2c_scl							
usb_dat	IO	Muxed	CONF_ USB_DAT	0xFFFE F10C	[4:3] and [0:0]	00000b	usb_dat	uart_rx					PU PD	_Unicode EncodeEr ror_
sda_trit	IO	Dedicated					sda_trit							
vclkrx	I	Muxed	CONF_ VCLKRX	0xFFFE F1E8	[4:3]	00000b	vclkrx						PD	_Unicode EncodeEr ror_
wakeup_ req	O	Dedicated					wakeup_ req							
gpio_4	IO	Muxed	CONF_ GPIO_4	0xFFFE F11A	[4:3] and [1:0]	11000b	gpio_4	cdi	tspact_9	lt2			PU PD	U

Table 18-6. LOCOSTO Application Pin Characteristics (continued)

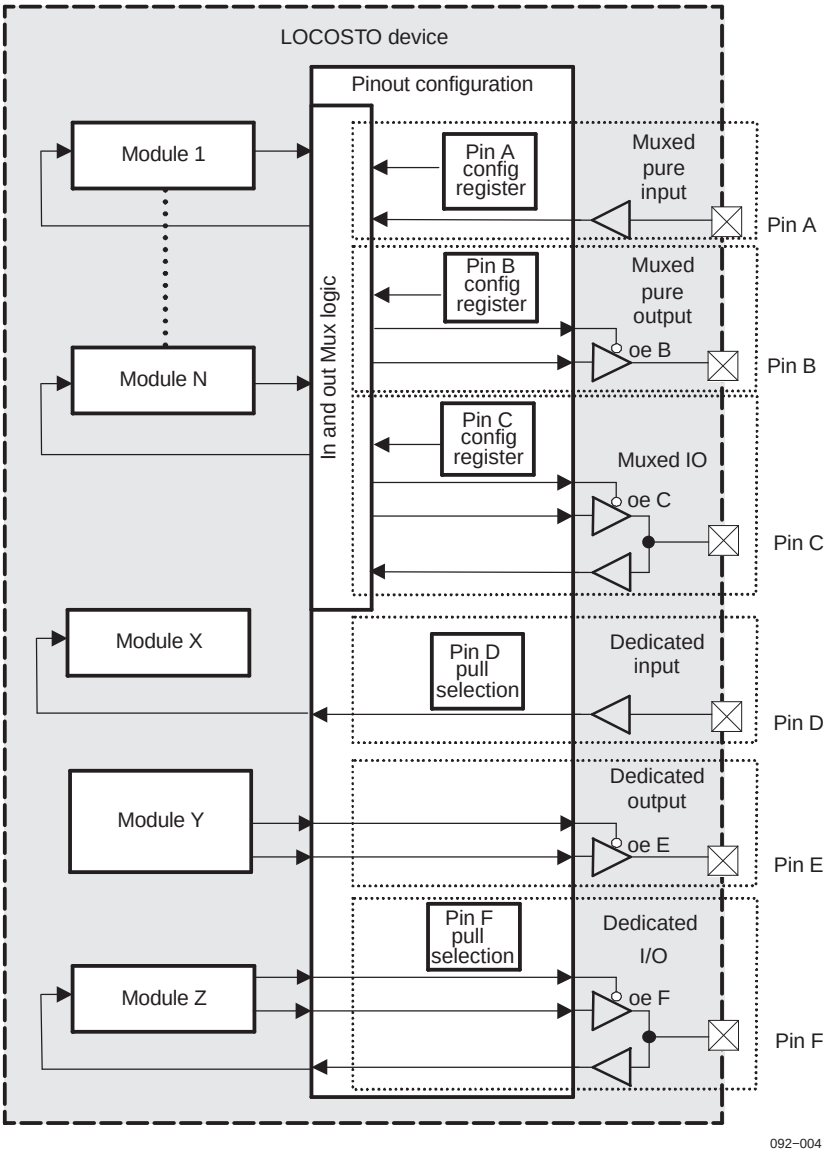
Pin			Pinout Configuration Register				Alternate Functions						Pull	
Pin Name	**Dir**	**Mux Type**	**Configuration Register Name**	**Register Address**	**Register Field**	**Reset Register Value**	**Mode0**	**Mode1**	**Mode2**	**Mode3**	**Mode4**	**Mode5**	**PU PD Typ e**	**Pull Reset Status**
nbscan	I	Muxed	CONF_ NBSCAN	0xFFFE F1B2	[4:3]	11000b	nbscan						PU	U
usb_ boot	IO	Muxed	CONF_ USB_BOOT	0xFFFE F11E	[1:0]	00000b	usb_boot	gpio_3	lpg					
sim_ pbias	IO	Dedicated					sim_pbias							
xanatst5	IO	Dedicated					xanatst5							
anatst1	IO	Dedicated					anatst1							
trstn	I	Muxed	CONF_TRST	0xFFFE F110	[4:3]	01000b	trstn						PU PD	D
gpio_2	IO	Muxed	CONF_ GPIO_2	0xFFFE F104	[4:3] and [1:0]	01000b	gpio_2	pwl	pmc_clk				PU PD	D
usb_rcv	I	Muxed	CONF_ USB_RCV	0xFFFE F108	[4:3] and [0:0]	00000b	usb_rcv	uart_cts					PD	_Unicode EncodeEr ror_
vdx	O	Dedicated					vdx							
ck 13mh z_ en	I	Muxed	CONF_ CLK13MHZ_ EN	0xFFFE F1E4	[4:3]	00000b	ck13mhz_ en						PU	_Unicode EncodeEr ror_
ckin_ 32kHz	I	Dedicated					ckin_ 32khz							
cdo	O	Dedicated	CONF_CDO	0xFFFE F118	Reserved	00000b	cdo							
spare_3	IO	Muxed	CONF_ SPARE3	0xFFFE F106	[4:3]	00000b	spare_3						PU	_Unicode EncodeEr ror_
xanatst3	IO	Dedicated					xanatst3							
xanat st4	IO	Dedicated					xanatst4							
anatst2	IO	Dedicated					anatst2							
spare_2	IO	Dedicated					spare_2							

18.3.4 Pinout Functional Description

18.3.4.1 Block Diagram

Figure 18-4 shows the pinout block diagram.

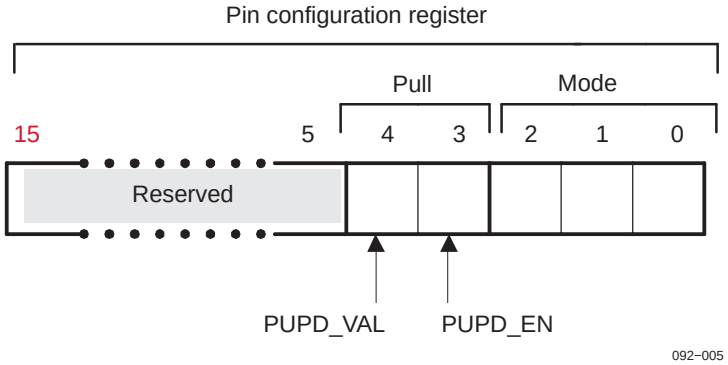
Figure 18-4. Pinout Block Diagram



18.3.4.2 Pin Configuration Register

All muxed pins and some dedicated pins can be configured by software using their corresponding pin configuration register. Although pin configuration registers are 16 bits wide, only 5 bits are used. Figure 18-5 shows the pin configuration register.

Figure 18-5. Pin Configuration Register



The lower 5 bits of the pin configuration registers are divided into the following two fields:

- Mode: 3 bits are used for mode selection; a mode corresponds to the selection of the functionality, which is mapped on the pin. There are up to 6 (0 to 5) possible functional modes for each pin.

  In some registers, only 1 or 2 bits are available. In some dedicated pins, the entire field is reserved and unused.
- Pull: 2 bits are used for combinational pullup/down configuration:
  - PUPD_VAL: Selection between the pullup and the pulldown for the pin
  - PUPD_EN: Activation of the pull

    In some registers, the pull field is reserved and unused.

For more details on the pin configuration registers, see Section 18.3.7, *LOCOSTO I/O Multiplexing Registers*.

18.3.4.2.1 Mode Selection

Table 18-7. Pin Selected Modes

Mode	Mode Selected
000	Default mode = Mode 0
001	Mode 1
010	Mode 2
011	Mode 3
100	Mode 4
101	Mode 5

Mode 0 is the default mode; when mode 0 is set, the function mapped on the pin corresponds to the name of the pin. There is always a function mapped on the default mode. Mode 0 is automatically configured at the release of the power-on reset (core_ON_nOFF).

Mode 1 to mode 5 are possible modes for alternate functions. On each pin, some modes are used effectively for alternate functions; other modes are unused and correspond to nonfunctional configurations.

18.3.4.2.2 Pull Selection

Regardless of the pull value configuration, when a pin is configured as an output, the pull is automatically disabled. Not all input pins can be pullup- or pulldown- selected. Table 18-8 lists the pull selection.

Table 18-8. Pull Selection

Pull		Pad Behavior
PUPD_VAL	PUPD_EN	
0	0	Pulldown selected but not activated
0	1	Pulldown selected and activated if pin configured in input
1	0	Pullup selected but not activated
1	1	Pullup selected and activated if pin configured in input

18.3.4.3 Dedicated Pins

Dedicated pins are used exclusively by one interface. In this case, the pin is hardwired to the module interface; no alternate modes are available for this pin.

18.3.4.3.1 Dedicated Pure Input Pins

If a pull exists on these pins, pullup/pulldown configuration is available using the pull bits of the pin configuration register field.

18.3.4.3.2 Dedicated Pure Output Pin

This pin cannot be configured. The output and the output-enable (oe) signal are hardwired to the module. No pullup/pulldown is available on this pin.

18.3.4.3.3 Dedicated I/O Pin

This pin combines a pure input and a pure output. When the oe signal is driven to 0 by the module (the pin is in output), the pull is automatically disabled. When the oe signal is driven to 1 by the module (the pin is configured in input), the pull can be configured by the pull bits of the pin configuration register field.

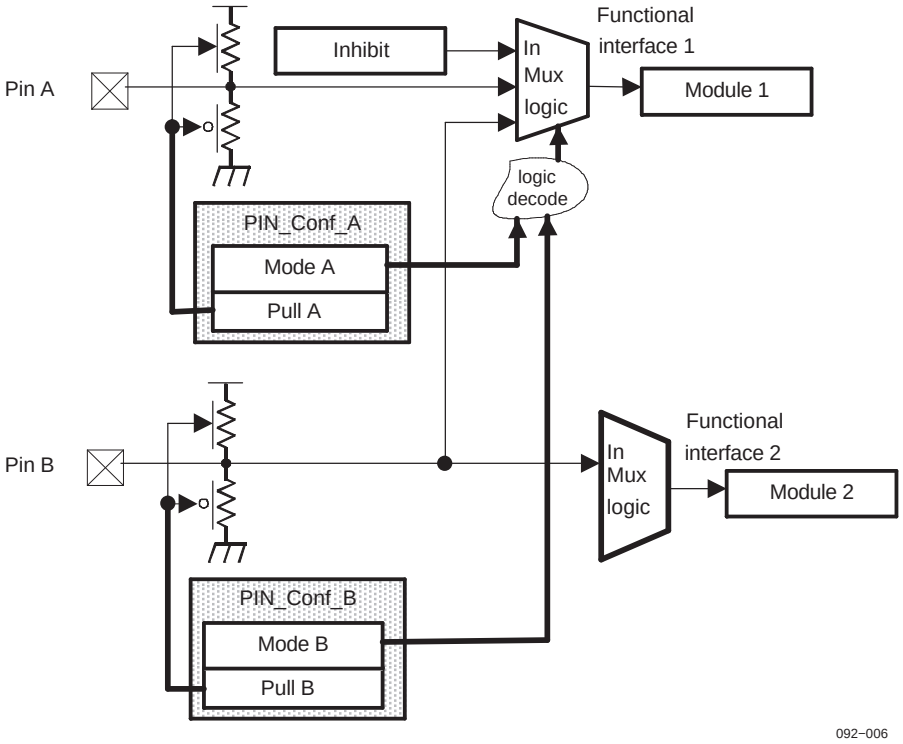
18.3.4.4 Muxed Pins

Muxed pins correspond to pins on which several different functions can be mapped, depending on the MODE field of the pin configuration register.

18.3.4.4.1 Muxed Pure Input Pin

Only input interfaces can be mapped on this type of pin. If there is a pull, it can be configured by writing to the pull bits of the pin-configuration register field (see Figure 18-6).

Figure 18-6. Muxed Pure Input



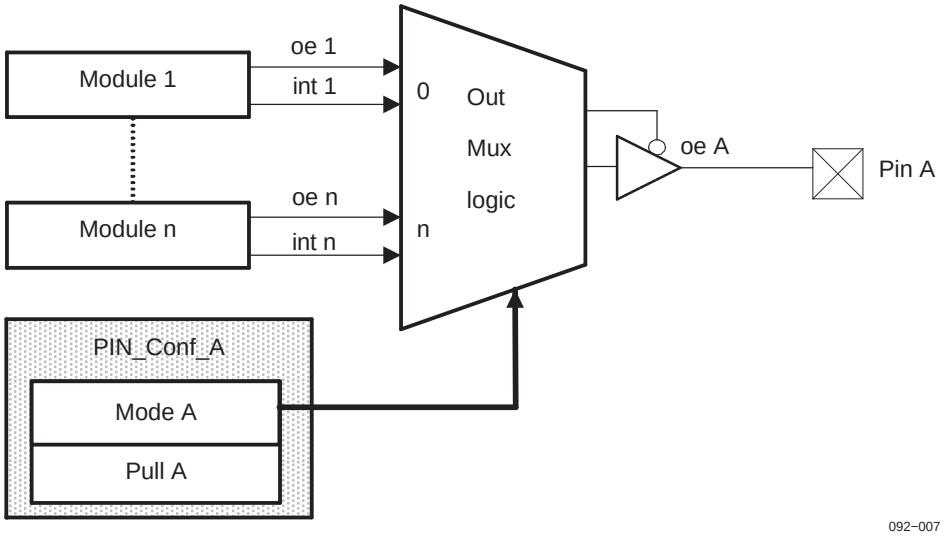
Hardware protection is implemented to prevent any damage in case of faulty programming:

- If no functional interface is selected for a module, the value driven is the inhibit value, which is the hardwired value chosen to keep the module inactive.
- If several input pins are asserted to the same functional interface, conflict is automatically solved by giving priority to the pin with the lowest MODE field value.

18.3.4.4.2 Muxed Pure Output Pins

No pull is available for these pins. Multiple alternate functions can be mapped on a pin chosen by configuring the MODE bits of the pin configuration register (see Figure 18-7).

Figure 18-7. Muxed Pure Output



18.3.4.4.3 Muxed I/O

Muxed I/O is the combination of a muxed input and a muxed output. The MODE field of the pin configuration register defines the I/O type depending on the interface mapped. The muxed I/O pin behavior is then strictly equivalent to a pure input or a pure output.

18.3.5 Pinout Functional Interface

This section describes the pins that can be configured for each module, with signals classified by their functions.

All possible interfaces for each function are summarized in the appropriate table.

18.3.5.1 Interfaces Pin Multiplexing Example

Table 18-9 is an example of the general organization of Table 18-10 through Table 18-35.

Note: The signals and functions in this example are fictitious and do not correspond to actual LOCOSTO device signals or functions.

Table 18-9. Pin Multiplexing Table Example

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
funcn_ sig1	Signal 1 of function n Description	I	Muxed	CONF_FUNCN_ SIG1	PD	funcn_ sig1	funcx_ sig2	funcy_ sig1	funck_ sig5	funck_ sig7	funcx_ sig3
funcn_ sig2	Signal 2 of function n Description	I	Dedicated			funcn_ sig2					
funcn_ sig3	Signal 3 of function n Description	IO	Muxed	CONF_FUNCZ_ SIG1	PUPD	funcz_ sig1	funcn_ sig3	funcy_ sig2	funck_ sig8	funcx_ sig6	
funcn_ sig4	Signal 4 of function n Description	O	Muxed	CONF_FUNCZ_ SIG2	PU	funcz_ sig2	funcn_ sig4				
			Muxed	CONF_FUNCZ_ SIG3	PU	funcy_ sig3	funcz_ sig5	funcy_ sig8	funcn_ sig4		

Table 18-9 relates to function n. The following explain the table:

- Signal Name: All interface signals related to function n. A maximum of four signals (funcn_sig1, funcn_sig2, funcn_sig3, and funcn_sig4) are available for function n.
- Description: A short description of each signal
- Dir: The direction of the signal:
  - The direction is related to the signal, not to the pin. The signal funcn_sig1 is input only, but the pin can be an I/O pin (for information about the pin, see Table 18-6).
  - Information related to the direction of alternate functions is not in this table, but it is in the dedicated table for the alternate function. Funcx.sig2 direction, for example, is available in the function x table.
- Mux Type: The multiplexed type of the pin:
  - Dedicated pins are used exclusively by one interface. In this case, the pin is hardwired to the module interface; no alternate modes are available.
  - Muxed pins correspond to pins on which several different functions can be mapped, depending on the value of the MODE field in the pin configuration register.
- Configuration Register: The name of the register to use to configure the pin multiplexing and the pull
- PUPD Type: The pull available on the pin:
  - PU: Pullup
  - PD: Pulldown
- Alternate Functions: The signals that can be multiplexed on the same pin. In this example, if mode 1 is programmed in the CONF_FUNCN_SIG1 register, then funcx_sig2 is available in spite of funcn_sig1. Alternate function visibility lets the programmer identify potential resource conflicts on the pin. If several signals that must be used simultaneously in a user application are on the same line of the table, it must be determined whether the alternate signals can be multiplexed on other pins.

- Yellow cells in Table 18-9 indicate that the pull or the configuration register do not exist or that the particular mode is not used. In the example, modes 2, 3, 4, and 5 are not used in the CONF_FUNCZ_SIG2 register.
- Gray cells: A signal is effectively associated to a pin only when the corresponding mode is selected by using the pin configuration register programming. The mode associated with the described function is highlighted in gray in the corresponding cell in Table 18-9.

In this example:

- The MODE field CONFIG.CONF_FUNCN_SIG1[2:0] must be written to 000 to select funcn_sig1.
- The signal funcn_sig2 is dedicated; therefore, it is always selected.
- The signal funcn_sig3 is selected if mode 1 is programmed in the CONF_FUNCZ_SIG1 register.
- The signal funcn_sig4 is selected if mode 1 is programmed in the CONF_FUNCZ_SIG2 register or if mode 3 is programmed in the CONF_FUNCZ_SIG3 register.

Table 18-10. APC Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
apcldofilter	APC	IO	Dedicated			apcldofilter					
apcvref	APC	IO	Dedicated			apcvref					
apcspare1	APC	IO	Dedicated			apcspare1					
apcout	APC	IO	Dedicated			apcout					

Table 18-11. USIM Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
sim_io	Smart Card interface data I/O line	IO	Muxed	CONF_SIM_IO	PUPD	sim_io	force_0				
sim_clk	Smart Card clock _UnicodeEncodeError_ 1.8 V	O	Muxed	CONF_SIM_ CLK	PUPD	sim_clk					
sim_ pwrctrl	Smart Card power supply ctrl_Unicode EncodeError _ 1.8 V	O	Muxed	CONF_SIM_ PWCTRL	PUPD	sim_ pwrctrl					

Table 18-11. USIM Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
sim_rst	Smart Card reset _UnicodeEncodeError_ 1.8 V	O	Muxed	CONF_SIM_RST	PUPD	sim_rst					
force_0	Dummy signal for Spinner	O	Muxed	CONF_SIM_IO	PUPD	sim_io	force_0				
				CONF_RNW		rnw	force_0				
				CONF_NMOE		nmoe	force_0				
				CONF_NCS3		ncs3	force_0				
sim_ pbias	Backup solution in case Pbias generator cell does not work	IO	Dedicated			sim_ pbias					

Table 18-12. I²C Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
i2c_sda	I²C serial bidirectional data	IO	Dedicated			i2c_sda					
i2c_scl	I²C master serial clock	IO	Dedicated			i2c_scl					

Table 18-13. I²C TWL4030 Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
sda_trit	I²C serial bidirectional data	IO	Dedicated			sda_trit					
scl_trit	I²C master serial clock	IO	Dedicated			sda_trit					

Table 18-14. USB Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
usb_rcv	USB differential receiver input	I	Muxed	CONF_USB_ RCV	PD	usb_rcv	uart_cts				
usb_se0	USB single-ended zero	IO	Muxed	CONF_USB_ SE0	PU	usb_se0	uart_tx				
usb_dat	USB transmit data	IO	Muxed	CONF_USB_ DAT	PUPD	usb_dat	uart_rx				
usb_txen	USB transmit enable	O	Muxed	CONF_USB_ TXEN		usb_txen	uart_rts				

Table 18-15. Codec Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
csync_o	Codec frame synchronization interface	O	Muxed	CONF_ GPIO_44	PU	gpio_ 44	mcsi_fs	csync_ o	test_ port_34		
csclk_o	Codec serial clock interface	O	Muxed	CONF_ GPIO_43	PU	gpio_ 43	mcsi_ ck	csclk_o	test_ port_33		
csync	Codec frame synchronization interface	IO	Muxed	CONF_ CSYNC	PD	csync					
csclk	Codec serial clock interface	IO	Muxed	CONF_CSCLK	PD	csclk					
cdi	Codec serial data input	I	Muxed	CONF_ GPIO_4	PUPD	gpio_4	cdi	tspact_ 9	lt2		
cdo	Codec serial data output interface	O	Dedicated	CONF_CDO		cdo					
			Muxed	CONF_ GPIO_45	PU	gpio_ 45	mcsi_tx	cdo	test_ port_35		
pmc_clk	Codec master clock input interface	I	Muxed	CONF_ GPIO_2	PUPD	gpio_2	pwl	pmc_ clk			

Table 18-15. Codec Module Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
pmc_ reset	Codec reset: Provided only to avoid C-port test case porting for LOCOSTO	O	Muxed	CONF_ GPIO_1	PUPD	gpio_1	pwt	pmc_ reset			

Table 18-16. IRQ Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
abb_ irq	External interrupt from ABB	I	Muxed	CONF_ABB_ IRQ	PU	abb_irq					

Table 18-17. ULPD Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
ck13mhz_ en	Enable 13-MHz system clock generation	I	Muxed	CONF_ CK13MHZ_ EN	PU	ck13mhz_ en					
ckout_ 13mhz	13-kHz system clock	O	Dedicated			ckout_13 mhz					
ckin_32khz	32-kHz input clock	I	Dedicated			ckin_ 32khz					
wakeup_ req	Wake-up request to TWL4030	O	Dedicated			wakeup_ req					
on_noff	Power ON reset	I	Dedicated			on_noff					
clkm_clk	ARM7 CLKM clock	I	Muxed	CONF_ TSPACT_11		tspact_11	clkm_ clk				

Table 18-18. LCD Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
lcd_ data_0	LCD pixel data bit 0	IO	Muxed	CONF_LCD_ DATA_0		lcd_ data_0	cam0_ d_0	test_ port_7	ndf_ dyn_0		
lcd_ data_1	LCD pixel data bit 1	IO	Muxed	CONF_LCD_ DATA_1		lcd_ data_1	cam0_ d_1	test_ port_6	ndf_ dyn_1		
lcd_ data_2	LCD pixel data bit 2	IO	Muxed	CONF_LCD_ DATA_2		lcd_ data_2	cam0_ d_2	test_ port_5	ndf_ dyn_2		
lcd_ data_3	LCD pixel data bit 3	IO	Muxed	CONF_LCD_ DATA_3		lcd_ data_3	cam0_ d_3	test_ port_4	ndf_ dyn_3		
lcd_ data_4	LCD pixel data bit 4	IO	Muxed	CONF_LCD_ DATA_4		lcd_ data_4	cam0_ d_4	test_ port_3	ndf_ dyn_4		
lcd_ data_5	LCD pixel data bit 5	IO	Muxed	CONF_LCD_ DATA_5		lcd_ data_5	cam0_ d_5	test_ port_2	ndf_ dyn_5		
lcd_ data_6	LCD pixel data bit 6	IO	Muxed	CONF_LCD_ DATA_6		lcd_ data_6	cam0_ d_6	test_ port_1	ndf_ dyn_6		
lcd_ data_7	LCD pixel data bit 7	IO	Muxed	CONF_LCD_ DATA_7		lcd_ data_7	cam0_ d_7	test_ port_0	ndf_ dyn_7		
lcd_rs	LCD data/control selection	O	Muxed	CONF_LCD_RS	PUPD	lcd_rs	gpio_ 16	test_ port_9			
lcd_rnw	LCD read-not write control/ write enable clock	O	Muxed	CONF_LCD_ RNW	PUPD	lcd_rnw	gpio_ 15	test_ port_10			
lcd_ncs0	LCD chip-select	O	Muxed	CONF_GPIO_13	PUPD	gpio_ 13	lcd_ ncs0				
lcd_ncs1	LCD chip-select	O	Muxed	CONF_GPIO_17	PUPD	gpio_ 17	lcd_ ncs1	test_ port_8			
lcd_stb	LCD strobe enable	O	Muxed	CONF_LCD_ STB	PUPD	lcd_stb	gpio_ 14	test_ port_11			
lcd_nrst	LCD controller dedicated reset	O	Muxed	CONF_LCD_ NRST	PD	lcd_ nrst	test_ port_12				

Table 18-19. GPIO Modules Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
gpio_0	GPIO 0	IO	Muxed	CONF_GPIO_0	PU	gpio_0	dcd_txir	cam_d_1			
gpio_1	GPIO 1	IO	Muxed	CONF_GPIO_1	PUPD	gpio_1	pwt	pmc_reset			
gpio_2	GPIO 2	IO	Muxed	CONF_GPIO_2	PUPD	gpio_2	pwl	pmc_clk			
gpio_3	GPIO 3	IO	Muxed	CONF_USB_ BOOT		usb_boot	gpio_3	lpg			
gpio_4	GPIO 4	IO	Muxed	CONF_GPIO_4	PUPD	gpio_4	cdi	tspact_9	lt2		
gpio_5	GPIO 5	IO	Muxed	CONF_GPIO_5	PUPD	gpio_5	tspact_8				
gpio_6	GPIO 6	IO	Muxed	CONF_ADD_21	PD	add_21	gpio_6				
gpio_7	GPIO 7	IO	Muxed	CONF_GPIO_7	PD	gpio_7	nfwp	tspact_7	lt1	ndf2	cam_ d_2
gpio_8	GPIO 8	IO	Muxed	CONF_GPIO_8	PUPD	kbr_4	gpio_8	test_port_ 18			
gpio_9	GPIO 9	IO	Muxed	CONF_GPIO_9	PUPD	kbc_4	gpio_9	test_port_ 13			
gpio_ 10	GPIO 10	IO	Muxed	CONF_GPIO_10	PUPD	nemu0	gpio_10				
gpio_ 11	GPIO 11	IO	Muxed	CONF_GPIO_11	PUPD	nemu1	gpio_11				
gpio_ 12	GPIO 12	IO	Muxed	CONF_GPIO_12	PUPD	gpio_12	lpg	tspact_10			
gpio_ 13	GPIO 13	IO	Muxed	CONF_GPIO_13	PUPD	gpio_13	lcd_ncs0				
gpio_ 14	GPIO 14	IO	Muxed	CONF_LCD_STB	PUPD	lcd_stb	gpio_14	test_port_11			
gpio_ 15	GPIO 15	IO	Muxed	CONF_LCD_RNW	PUPD	lcd_rnw	gpio_15	test_port_ 10			
gpio_ 16	GPIO 16	IO	Muxed	CONF_LCD_RS	PUPD	lcd_rs	gpio_16	test_port_9			
gpio_ 17	GPIO 17	IO	Muxed	CONF_GPIO_17	PUPD	gpio_17	lcd_ncs1	test_port_8			
gpio_ 18	GPIO 18	IO	Muxed	CONF_GPIO_18	PUPD	gpio_18	nd_we				
gpio_ 19	GPIO 19	IO	Muxed	CONF_GPIO_19		gpio_19	cam_hs	cam_d_3	test_port_ 32	ndf3	
gpio_ 20	GPIO 20	IO	Muxed	CONF_GPIO_20		gpio_20	cam_vs	cam_d_3	test_port_ 31	ndf3	
gpio_ 21	GPIO 21	IO	Muxed	CONF_GPIO_21		gpio_21	cam_lclk	test_port_ 30			
gpio_ 22	GPIO 22	IO	Muxed	CONF_GPIO_22		gpio_22	cam_xclk	cam_d_3	test_port_ 29	ndf3	
gpio_ 23	GPIO 23	IO	Muxed	CONF_GPIO_23	PD	gpio_23	spi_clk				
gpio_ 24	GPIO 24	IO	Muxed	CONF_GPIO_24	PD	gpio_24	spi_data_miso				
gpio_ 25	GPIO 25	IO	Muxed	CONF_GPIO_25	PD	gpio_25	spi_data_mosi				
gpio_ 26	GPIO 26	IO	Muxed	CONF_GPIO_26	PUPD	gpio_26	spi_ncs0				
gpio_ 27	GPIO 27	IO	Muxed	CONF_GPIO_27	PUPD	gpio_27	spi_ncs1				
gpio_ 28	GPIO 28	IO	Muxed	CONF_GPIO_28	PU	gpio_28	spi_ncs2	ndf7	cam_d_7		
gpio_ 29	GPIO 29	IO	Muxed	CONF_GPIO_29	PD	gpio_29	bu	ndf6	cam_d_6		
gpio_ 30	GPIO 30	IO	Muxed	CONF_GPIO_30	PD	gpio_30	lt3	ndf5	cam_d_5		
gpio_ 31	GPIO 31	IO	Muxed	CONF_GPIO_31	PU	gpio_31	nd_re	test_port_ 27			

Table 18-19. GPIO Modules Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
gpio_ 32	GPIO 32	IO	Muxed	CONF_GPIO_32	PD	gpio_32	nd_cle	test_port_ 26			
gpio_ 33	GPIO 33	IO	Muxed	CONF_GPIO_33	PD	gpio_33	nd_ale	test_port_ 25			
gpio_ 34	GPIO 34	IO	Muxed	CONF_GPIO_34	PU	gpio_34	nd_rdy	test_port_ 24			
gpio_ 35	GPIO 35	IO	Muxed	CONF_GPIO_35	PU	gpio_35	ncs2	ndf3			
gpio_ 36	GPIO 36	IO	Muxed	CONF_GPIO_36	PU	gpio_36	ncs1	ndf2			
gpio_ 37	GPIO 37	IO	Muxed	CONF_GPIO_37	PD	gpio_37	add_23	ndf1			
gpio_ 38	GPIO 38	IO	Muxed	CONF_GPIO_38	PU	ncs0	gpio_38	lt1	tspact_7	ndf0	
gpio_ 39	GPIO 39	IO	Muxed	CONF_GPIO_39	PD	gpio_39	add_22	ndf0			
gpio_ 40	GPIO 40	IO	Muxed	CONF_NRDY	PU	nrdy	gpio_40				
gpio_ 41	GPIO 41	IO	Muxed	CONF_ADV	PU	adv	gpio_41				
gpio_ 42	GPIO 42	IO	Muxed	CONF_GPIO_42	PU	gpio_42	ckm				
gpio_ 43	GPIO 43	IO	Muxed	CONF_GPIO_43	PU	gpio_43	mcsi_ck	csclk_o	test_port_ 33		
gpio_ 44	GPIO 44	IO	Muxed	CONF_GPIO_44	PU	gpio_44	mcsi_fs	csync_o	test_port_ 34		
gpio_ 45	GPIO 45	IO	Muxed	CONF_GPIO_45	PU	gpio_45	mcsi_tx	cdo	test_port_ 35		
gpio_ 46	GPIO 46	IO	Muxed	CONF_GPIO_46	PU	gpio_46	mcsi_rx	test_port_ 36			
gpio_ 47	GPIO 47	IO	Muxed	CONF_GPIO_47	PU	gpio_47	dsr_rxir	cam_d_0			

Table 18-20. TPU2OCP Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
tspact_ 7	TPU2OCP synchronous activation signal	O	Muxed	CONF_GPIO_38	PU	ncs0	gpio_ 38	lt1	tspact_ 7	ndf0	
				CONF_GPIO_7	PD	gpio_7	nfwp	tspact_ 7	lt1	ndf2	cam_ d_2
tspact_ 8	TPU2OCP synchronous activation signal	O	Muxed	CONF_GPIO_5	PUPD	gpio_5	tspact_ 8				
tspact_ 9	TPU2OCP synchronous activation signal	O	Muxed	CONF_GPIO_4	PUPD	gpio_4	cdi	tspact_ 9	lt2		

Table 18-20. TPU2OCP Module Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
tspact_ 10	TPU2OCP synchronous activation signal	O	Muxed	CONF_GPIO_12	PUPD	gpio_ 12	lpg	tspact_ 10			
tspact_ 11	TPU2OCP synchronous activation signal	O	Muxed	CONF_ TSPACT_11		tspact_ 11	clkm_ clk				
tspact_ 12	TPU2OCP synchronous activation signal	O	Dedicated	CONF_ TSPACT_12		tspact_ 12					
tspact_ 13	TPU2OCP synchronous activation signal	O	Dedicated	CONF_ TSPACT_13		tspact_ 13					
tspact_ 14	TPU2OCP synchronous activation signal	O	Dedicated	CONF_ TSPACT_14		tspact_ 14					
tspact_ 15	TPU2OCP synchronous activation signal	O	Dedicated	CONF_ TSPACT_15		tspact_ 15					

Table 18-21. JTAG Emulation Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
nemu0	Test emulation 0	IO	Muxed	CONF_GPIO_10	PUPD	nemu0	gpio_10				
nemu1	Test emulation 1	IO	Muxed	CONF_GPIO_11	PUPD	nemu1	gpio_11				
tdi	Test data input	I	Muxed	CONF_TDI	PUPD	tdi					
tdo	Test data output	O	Dedicated	CONF_TDO		tdo					
tms	Test mode select	I	Muxed	CONF_TMS	PUPD	tms					
tck	Test clock	I	Muxed	CONF_TCK	PUPD	tck					
nbscan	Boundary scan	I	Muxed	CONF_NBSCAN	PU	nbscan					
trstn	Test reset	I	Muxed	CONF_TRST	PUPD	trstn					

Table 18-22. Camera Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
cam_ d_0	Camera digital image data bit 0	I	Muxed	CONF_GPIO_47	PU	gpio_ 47	dsr_rxir	cam_ d_0			
cam_ d_1	Camera digital image data bit 1	I	Muxed	CONF_GPIO_0	PU	gpio_0	dcd_ txir	cam_ d_1			
cam_ d_2	Camera digital image data bit 2	I	Muxed	CONF_GPIO_7	PD	gpio_7	nfwp	tspact_ 7	lt1	ndf2	cam_ d_2
cam_ d_3	Camera digital image data bit 3	I	Muxed	CONF_GPIO_19		gpio_ 19	cam_ hs	cam_ d_3	test_ port_32	ndf3	
				CONF_GPIO_20		gpio_ 20	cam_vs	cam_ d_3	test_ port_31	ndf3	
				CONF_GPIO_22		gpio_ 22	cam_ xclk	cam_ d_3	test_ port_29	ndf3	
cam_ d_4	Camera digital image data bit 4	I	Muxed	CONF_ND_NWP		nd_ nwp	ndf4	cam_ d_4	test_ port_28		
cam_ d_5	Camera digital image data bit 5	I	Muxed	CONF_GPIO_30	PD	gpio_ 30	lt3	ndf5	cam_ d_5		
cam_ d_6	Camera digital image data bit 6	I	Muxed	CONF_GPIO_29	PD	gpio_ 29	bu	ndf6	cam_ d_6		
cam_ d_7	Camera digital image data bit 7	I	Muxed	CONF_GPIO_28	PU	gpio_2 8	spi_ ncs2	ndf7	cam_d _7		
cam0_ d_0	Camera digital image data bit 0	I	Muxed	CONF_LCD_ DATA_0		lcd_ data_0	cam0_ d_0	test_ port_7	ndf_ dyn_0		
cam0_ d_1	Camera digital image data bit 1	I	Muxed	CONF_LCD_ DATA_1		lcd_ data_1	cam0_ d_1	test_ port_6	ndf_ dyn_1		
cam0_ d_2	Camera digital image data bit 2	I	Muxed	CONF_LCD_ DATA_2		lcd_ data_2	cam0_ d_2	test_ port_5	ndf_ dyn_2		
cam0_ d_3	Camera digital image data bit 3	I	Muxed	CONF_LCD_ DATA_3		lcd_ data_3	cam0_ d_3	test_ port_4	ndf_ dyn_3		
cam0_ d_4	Camera digital image data bit 4	I	Muxed	CONF_LCD_ DATA_4		lcd_ data_4	cam0_ d_4	test_ port_3	ndf_ dyn_4		
cam0_ d_5	Camera digital image data bit 5	I	Muxed	CONF_LCD_ DATA_5		lcd_ data_5	cam0_ d_5	test_ port_2	ndf_ dyn_5		
cam0_ d_6	Camera digital image data bit 6	I	Muxed	CONF_LCD_ DATA_6		lcd_ data_6	cam0_ d_6	test_ port_1	ndf_ dyn_6		
cam0_ d_7	Camera digital image data bit 7	I	Muxed	CONF_LCD_ DATA_7		lcd_ data_7	cam0_ d_7	test_ port_0	ndf_ dyn_7		
cam_hs	Camera horizontal synchronization	I	Muxed	CONF_GPIO_19		gpio_ 19	cam_ hs	cam_ d_3	test_ port_32	ndf3	

Table 18-22. Camera Module Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
cam_vs	Camera vertical synchronization	I	Muxed	CONF_GPIO_20		gpio_ 20	cam_vs	cam_ d_3	test_ port_31	ndf3	
cam_lclk	Camera image data latch clock	I	Muxed	CONF_GPIO_21		gpio_ 21	cam_ lclk	test_ port_30			
cam_ xclk	Camera output clock	O	Muxed	CONF_GPIO_22		gpio_ 22	cam_ xclk	cam_ d_3	test_ port_29	ndf3	

Table 18-23. SPI Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
spi_data_ mosi	SPI data: Master out / slave in	IO	Muxed	CONF_ GPIO_25	PD	gpio_ 25	spi_data_ mosi				
spi_data_ miso	SPI data: Master in / slave out	IO	Muxed	CONF_ GPIO_24	PD	gpio_ 24	spi_data_ miso				
spi_clk	SPI clk	IO	Muxed	CONF_ GPIO_23	PD	gpio_ 23	spi_clk				
spi_ncs0	SPI enable 0	IO	Muxed	CONF_ GPIO_26	PUPD	gpio_ 26	spi_ncs0				
spi_ncs1	SPI enable 1	O	Muxed	CONF_ GPIO_27	PUPD	gpio_ 27	spi_ncs1				
spi_ncs2	SPI enable 2	O	Muxed	CONF_ GPIO_28	PU	gpio_ 28	spi_ncs2	ndf7	cam_ d_7		

Table 18-24. NAND Flash Controller Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
ndf0	NAND flash data/address 0 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_GPIO_38	PU	ncs0	gpio_ 38	lt1	tspact_ 7	ndf0	
				CONF_GPIO_39	PD	gpio_ 39	add_22	ndf0			
ndf1	NAND flash data/address 1 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_GPIO_37	PD	gpio_ 37	add_23	ndf1			

Table 18-24. NAND Flash Controller Module Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
ndf2	NAND flash data/address 2 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_GPIO_36	PU	gpio_ 36	ncs1	ndf2			
				CONF_GPIO_7	PD	gpio_7	nfwp	tspact_ 7	lt1	ndf2	cam_ d_2
ndf3	NAND flash data/address 3 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_GPIO_19		gpio_ 19	cam_ hs	cam_ d_3	test_ port_32	ndf3	
				CONF_GPIO_20		gpio_ 20	cam_vs	cam_ d_3	test_ port_31	ndf3	
				CONF_GPIO_22		gpio_ 22	cam_ xclk	cam_ d_3	test_ port_29	ndf3	
				CONF_GPIO_35	PU	gpio_ 35	ncs2	ndf3			
ndf4	NAND flash data/address 4 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_ND_NWP		nd_ nwp	ndf4	cam_d _4	test_ port_28		
ndf5	NAND flash data/address 5 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_GPIO_30	PD	gpio_ 30	lt3	ndf5	cam_ d_5		
ndf6	NAND flash data/address 6 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_GPIO_29	PD	gpio_ 29	bu	ndf6	cam_ d_6		
ndf7	NAND flash data/address 7 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_GPIO_28	PU	gpio_ 28	spi_ncs 2	ndf7	cam_ d_7		
ndf_ dyn_0	NAND flash data/address 0 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_LCD_ DATA_0		lcd_ data_0	cam0_ d_0	test_ port_7	ndf_ dyn_0		

Table 18-24. NAND Flash Controller Module Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
ndf_ dyn_1	NAND flash data/address 1 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_LCD_ DATA_1		lcd_ data_1	cam0_ d_1	test_ port_6	ndf_ dyn_1		
ndf_ dyn_2	NAND flash data/address 2 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_LCD_ DATA_2		lcd_ data_2	cam0_ d_2	test_ port_5	ndf_ dyn_2		
ndf_ dyn_3	NAND flash data/address 3 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_LCD_ DATA_3		lcd_ data_3	cam0_ d_3	test_ port_4	ndf_ dyn_3		
ndf_ dyn_4	NAND flash data/address 4 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_LCD_ DATA_4		lcd_ data_4	cam0_ d_4	test_ port_3	ndf_ dyn_4		
ndf_ dyn_5	NAND flash data/address 5 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_LCD_ DATA_5		lcd_ data_5	cam0_ d_5	test_ port_2	ndf_ dyn_5		
ndf_ dyn_6	NAND flash data/address 6 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_LCD_ DATA_6		lcd_ data_6	cam0_ d_6	test_ port_1	ndf_ dyn_6		
ndf_ dyn_7	NAND flash data/address 7 _UnicodeEncodeError_ 1.8 V	IO	Muxed	CONF_LCD_ DATA_7		lcd_ data_7	cam0_ d_7	test_ port_0	ndf_ dyn_7		
nd_cle	NAND flash common latch enable _UnicodeEncodeError_ 1.8 V	O	Muxed	CONF_GPIO_32	PD	gpio_ 32	nd_cle	test_ port_26			
nd_ale	NAND flash address latch enable _UnicodeEncodeError_ 1.8 V	O	Muxed	CONF_GPIO_33	PD	gpio_ 33	nd_ale	test_ port_25			

Table 18-24. NAND Flash Controller Module Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
nd_we	NAND flash write enable _UnicodeEncodeError_ 1.8 V	O	Muxed	CONF_GPIO_18	PUPD	gpio_ 18	nd_we				
nd_nwp	NAND flash write protect _UnicodeEncodeError_ 1.8 V	O	Muxed	CONF_ND_NWP		nd_ nwp	ndf4	cam_ d_4	test_ port_28		
nd_re	NAND flash read enable _UnicodeEncodeError_ 1.8 V	O	Muxed	CONF_GPIO_31	PU	gpio_ 31	nd_re	test_ port_27			
nd_rdy	NAND flash ready signal/busy signal _UnicodeEncodeError_ 1.8 V	I	Muxed	CONF_GPIO_34	PU	gpio_ 34	nd_rdy	test_ port_24			
nd_ce1	NAND flash chip enable 1 _UnicodeEncodeError_ 1.8 V	O	Muxed	CONF_ND_CE1		nd_ce1	test_ port_23				

Table 18-25. PWM Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
pwt	Pulse width tone	O	Muxed	CONF_GPIO_0	PUPD	gpio_1	pwt	pmc_ reset			
lpg	LED pulse generator (Drive = 4 mA)	O	Muxed	CONF_USB_ BOOT		usb_ boot	gpio_3	lpg			
				CONF_GPIO_12	PUPD	gpio_ 12	lpg	tspact_ 10			
lt1	Light output with PWM	O	Muxed	CONF_GPIO_38	PU	ncs0	gpio_ 38	lt1	tspact_ 7	ndf0	
				CONF_GPIO_7	PD	gpio_7	nfwp	tspact_ 7	lt1	ndf2	cam_ d_2
lt2	Light output with PWM	O	Muxed	CONF_GPIO_4	PUPD	gpio_4	cdi	tspact_ 9	lt2		
lt3	Light output with PWM	O	Muxed	CONF_GPIO_30	PD	gpio_ 30	lt3	ndf5	cam_ d_5		
bu	Buzzer output with PWM	O	Muxed	CONF_GPIO_29	PD	gpio_ 29	bu	ndf6	cam_ d_6		

Table 18-25. PWM Module Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
pwl	Pseudo-noise variable width light controller	O	Muxed	CONF_GPIO_2	PUPD	gpio_2	pwl	pmc_ clk			

Table 18-26. EMIF Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
rnw	Memory read (high) / write (low) control	O	Muxed	CONF_RNW		rnw	force_0				
nbhe	High byte enable, active low	O	Dedicated			nbhe					
nble	Low byte enable, active low	O	Dedicated			nble					
nmoe	Memory output enable, active low	O	Muxed	CONF_NMOE		nmoe	force_0				
nfwp	Flash write protect	O	Muxed	CONF_ GPIO_7	PD	gpio_7	nfwp	tspact_ 7	lt1	ndf2	cam_ d_2
fdp	Flash deep low power, active low	O	Dedicated			fdp					
ncs0	Chip-select 0	O	Muxed	CONF_ GPIO_38	PU	ncs0	gpio_ 38	lt1	tspact_ 7	ndf0	
ncs1	Chip-select 1	O	Muxed	CONF_ GPIO_36	PU	gpio_ 36	ncs1	ndf2			
ncs2	Chip-select 2	O	Muxed	CONF_ GPIO_35	PU	gpio_35	ncs2	ndf3			
ncs3	Chip-select 3	O	Muxed	CONF_NCS3		ncs3	force_0				
nrdy	External devices wait assertion	I	Muxed	CONF_NRDY	PU	nrdy	gpio_ 40				
adv	Address valid signal for synchronous burst	O	Muxed	CONF_ADV	PU	adv	gpio_ 41				
ckm	Clock signal for synchronous burst memory control	IO	Muxed	CONF_ GPIO_42	PU	gpio_ 42	ckm				
add_ data_0	EMIF address data muxed bit 0	IO	Dedicated			add_ data_0					
add_ data_1	EMIF address data muxed bit 1	IO	Dedicated			add_ data_1					
add_ data_2	EMIF address data muxed bit 2	IO	Dedicated			add_ data_2					

Table 18-26. EMIF Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
add_ data_3	EMIF address data muxed bit 3	IO	Dedicated			add_ data_3					
add_ data_4	EMIF address data muxed bit 4	IO	Dedicated			add_ data_4					
add_ data_5	EMIF address data muxed bit 5	IO	Dedicated			add_ data_5					
add_ data_6	EMIF address data muxed bit 6	IO	Dedicated			add_ data_6					
add_ data_7	EMIF address data muxed bit 7	IO	Dedicated			add_ data_7					
add_ data_8	EMIF address data muxed bit 8	IO	Dedicated			add_ data_8					
add_ data_9	EMIF address data muxed bit 9	IO	Dedicated			add_ data_9					
add_ data_10	EMIF address data muxed bit 10	IO	Dedicated			add_ data_10					
add_ data_11	EMIF address data muxed bit 11	IO	Dedicated			add_ data_11					
add_ data_12	EMIF address data muxed bit 12	IO	Dedicated			add_ data_12					
add_ data_13	EMIF address data muxed bit 13	IO	Dedicated			add_ data_13					
add_ data_14	EMIF address data muxed bit 14	IO	Dedicated			add_ data_14					
add_ data_15	EMIF address data muxed bit 15	IO	Dedicated			add_ data_15					
add_16	EMIF address 16	O	Dedicated			add_16					
add_17	EMIF address 17	O	Dedicated			add_17					
add_18	EMIF address 18	O	Dedicated			add_18					
add_19	EMIF address 19	O	Dedicated			add_19					
add_20	EMIF address 20	O	Dedicated			add_20					
add_21	EMIF address 21	O	Muxed	CONF_ ADD_21	PD	add_21	gpio_6				
add_22	EMIF address 22	O	Muxed	CONF_ GPIO_39	PD	gpio_39	add_22	ndf0			

Table 18-26. EMIF Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
add_23	EMIF address 23	O	Muxed	CONF_ GPIO_37	PD	gpio_37	add_23	ndf1			

Table 18-27. DSP Voice Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
vfsrx	DSP voice	I	Muxed	CONF_VFSRX	PD	vfsrx					
vdr	DSP voice	I	Muxed	CONF_VDR	PD	vdr					
vdx	DSP voice	O	Dedicated			vdx					
vclkrx	DSP voice	I	Muxed	CONF_VCLKRX	PD	vclkrx					

Table 18-28. DSP MCSI Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
i_force	PMC	I	Dedicated			i_force					
sense	PMC	O	Dedicated			sense					

Table 18-29. PMC Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
mcsi_ck	MCSI clock	IO	Muxed	CONF_ GPIO_43	PU	gpio_43	mcsi_ ck	csclk_o	test_ port_33		
mcsi_fs	MCSI frame synchronization	IO	Muxed	CONF_ GPIO_44	PU	gpio_44	mcsi_fs	csync_ o	test_ port_34		
mcsi_tx	MCSI transmit data	O	Muxed	CONF_ GPIO_45	PU	gpio_45	mcsi_tx	cdo	test_ port_35		
mcsi_rx	MCSI receive data	I	Muxed	CONF_ GPIO_46	PU	PU	mcsi_rx	test_ port_36			

Table 18-30. Boot Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
usb_boot	BOOT	I	Muxed	CONF_USB_BOOT		usb_boot	gpio_3	lpg			

Table 18-31. UART Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
rts_ sdirda	UART IrDA request to send	O	Dedicated			rts_ sdirda					
dsr_rxir	UART IrDA receive data	I	Muxed	CONF_GPIO_47	PU	gpio_ 47	dsr_rxir	cam_ d_0			
dcd_txir	UART IrDA transmit data	IO	Muxed	CONF_GPIO_0	PU	gpio_0	dcd_ txir	cam_ d_1			
uart_cts	UART clear to send	I	Muxed	CONF_USB_RCV	PD	usb_ rcv	uart_ cts				
			Muxed	CONF_UART_ CTS	PD	uart_ cts					
uart_tx	UART transmit data	O	Muxed	CONF_USB_SE0	PU	usb_ se0	uart_tx				
			Dedicated	CONF_UART_TX		uart_tx					
uart_rx	UART receive data	I	Muxed	CONF_USB_DAT	PUPD	usb_ dat	uart_rx				
			Muxed	CONF_UART_RX	PD	uart_rx					
uart_rts	UART request to send	O	Muxed	CONF_USB_ TXEN		usb_ txen	uart_rts				

Table 18-32. Keyboard Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
kbr_0	Keyboard matrix row 0	I	Muxed	CONF_KBR_0	PUPD	kbr_0	test_ port_22				
kbr_1	Keyboard matrix row 1	I	Muxed	CONF_KBR_1	PUPD	kbr_1	test_ port_21				
kbr_2	Keyboard matrix row 2	I	Muxed	CONF_KBR_2	PUPD	kbr_2	test_ port_20				
kbr_3	Keyboard matrix row 3	I	Muxed	CONF_KBR_3	PUPD	kbr_3	test_ port_19				

Table 18-32. Keyboard Module Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
kbr_4	Keyboard matrix row 4	I	Muxed	CONF_ GPIO_8	PUPD	kbr_4	gpio_8	test_ port_18			
kbc_0	Keyboard matrix column 0	O	Muxed	CONF_KBC_0		kbc_0	test_ port_17				
kbc_1	Keyboard matrix column 1	O	Muxed	CONF_KBC_1		kbc_1	test_ port_16				
kbc_2	Keyboard matrix column 2	O	Muxed	CONF_KBC_2		kbc_2	test_ port_15				
kbc_3	Keyboard matrix column 3	O	Muxed	CONF_KBC_3		kbc_3	test_ port_14				
kbc_4	Keyboard matrix column 4	O	Muxed	CONF_ GPIO_9	PUPD	kbc_4	gpio_9	test_ port_13			

Table 18-33. Spare Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
spare_2	Corner spare	IO	Dedicated			spare_2					
spare_3	Spare	IO	Muxed	CONF_ SPARE_3	PU	spare_3					

Table 18-34. DRP Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
iref	DRP	O	Dedicated			iref					
vref	DRP	O	Dedicated			vref					
vref1	DRP	I	Dedicated			vref1					
rxdcsm	DRP	I	Dedicated			rxdcsm					
rxdcsp	DRP	I	Dedicated			rxdcsp					
rxpcsm	DRP	I	Dedicated			rxpcsm					
rxpcsp	DRP	I	Dedicated			rxpcsp					
rxgsmm	DRP	I	Dedicated			rxgsmm					
rxgsmp	DRP	I	Dedicated			rxgsmp					
rxegsmm	DRP	I	Dedicated			rxegsmm					

Table 18-34. DRP Module Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
rxegsmp	DRP	I	Dedicated			rxegsmp					
txhb	DRP	O	Dedicated			txhb					
txlb	DRP	O	Dedicated			txlb					
xtal	DRP	I	Dedicated			xtal					
anatst1	DRP	IO	Dedicated			anatst1					
anatst2	DRP	IO	Dedicated			anatst2					
xanatst3	DRP	IO	Dedicated			xanatst3					
xanatst4	DRP	IO	Dedicated			xanatst4					
xanatst5	DRP	IO	Dedicated			xanatst5					
xanatst6	DRP	IO	Dedicated			xanatst6					

Table 18-35. Debug Module Pin Multiplexing

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
test_port_0	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_LCD_ DATA_7		lcd_ data_7	cam0_ d_7	test_ port_0	ndf_ dyn_7		
test_port_1	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_LCD_ DATA_6		lcd_ data_6	cam0_ d_6	test_ port_1	ndf_ dyn_6		
test_port_2	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_LCD_ DATA_5		lcd_ data_5	cam0_ d_5	test_ port_2	ndf_ dyn_5		
test_port_3	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_LCD_ DATA_4		lcd_ data_4	cam0_ d_4	test_po rt_3	ndf_ dyn_4		
test_port_4	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_LCD_ DATA_3		lcd_ data_3	cam0_ d_3	test_po rt_4	ndf_ dyn_3		

Table 18-35. Debug Module Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
test_port_5	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_LCD_DATA_2		lcd_ data_2	cam0_ d_2	test_po rt_5	ndf_ dyn_2		
test_port_6	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_LCD_DATA_1		lcd_ data_1	cam0_ d_1	test_po rt_6	ndf_ dyn_1		
test_port_7	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_LCD_DATA_0		lcd_ data_0	cam0_ d_0	test_po rt_7	ndf_ dyn_0		
test_port_8	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_GPIO_17	PUPD	gpio_ 17	lcd_ ncs1	test_po rt_8			
test_port_9	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_LCD_RS	PUPD	lcd_rs	gpio_ 16	test_po rt_9			
test_port_10	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_LCD_RNW	PUPD	lcd_rnw	gpio_ 15	test_po rt_10			
test_port_11	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_LCD_STB	PUPD	lcd_stb	gpio_ 14	test_po rt_11			
test_port_12	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_LCD_NRST	PD	lcd_ nrst	test_ port_12				
test_port_13	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_KBC_4	PUPD	kbc_4	gpio_9	test_po rt_13			
test_port_14	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_KBC_3		kbc_3	test_ port_14				

Table 18-35. Debug Module Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
test_port_15	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_KBC_2		kbc_2	test_ port_15				
test_port_16	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_KBC_1		kbc_1	test_ port_16				
test_port_17	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_KBC_0		kbc_0	test_ port_17				
test_port_18	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_KBR_4	PUPD	kbr_4	gpio_8	test_ port_18			
test_port_19	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_KBR_3	PUPD	kbr_3	test_ port_19				
test_port_20	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_KBR_2	PUPD	kbr_2 test_	test_ port_20				
test_port_21	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_KBR_1	PUPD	kbr_1	test_ port_21				
test_port_22	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_KBR_0	PUPD	kbr_0	test_ port_22				
test_port_23	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_ND_CE1		nd_ce1	test_ port_23				
test_port_24	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_GPIO_34	PU	gpio_ 34	nd_rdy	test_ port_24			

Table 18-35. Debug Module Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
test_port_25	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_GPIO_33	PD	gpio_ 33	nd_ale	test_ port_25			
test_port_26	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_GPIO_32	PD	gpio_ 32	nd_cle	test_ port_26			
test_port_27	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_GPIO_31	PU	gpio_ 31	nd_re	test_ port_27			
test_port_28	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_ND_NWP		nd_ nwp	ndf4	cam_ d_4	test_ port_28		
test_port_29	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_GPIO_22		gpio_ 22	cam_ xclk	cam_ d_3	test_ port_29	ndf3	
test_port_30	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_GPIO_21		gpio_ 21	cam_ lclk	test_ port_30			
test_port_31	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_GPIO_20		gpio_ 20	cam_vs	cam_ d_3	test_ port_31	ndf3	
test_port_32	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_GPIO_19		gpio_ 19	cam_ hs	cam_ d_3	test_ port_32	ndf3	
test_port_33	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_GPIO_43	PU	gpio_ 43	mcsi_ ck	csclk_o	test_ port_33		
test_port_34	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_GPIO_44	PU	gpio_ 44	mcsi_fs	csync_ o	test_ port_34		

Table 18-35. Debug Module Pin Multiplexing (continued)

Signal Name	Description	Dir	Mux Type	Configuration Register	PUPD Type	ALTERNATE FUNCTIONS					
						Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
test_port_35	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_GPIO_45	PU	gpio_ 45	mcsi_tx	cdo	test_ port_35		
test_port_36	Debug bus (for information on muxing, see Section 18.4.)	O	Muxed	CONF_GPIO_46	PU	gpio_ 46	mcsi_rx	test_ port_36			

18.3.6 Programming Model

18.3.6.1 Pinout Initialization Sequence

At release of the core _ON_nOFF global reset, pins take the default configuration (mode 0):

1. The default mode (mode 0) is set on each pad.
2. The pinout is ready to be configured for application purposes:
   a. Select required interfaces from the pinout functional interfaces table.
   b. Select the pin mapping using the pinout multiplexing table (see Table 18-6) to solve conflicts.
      For information on solving conflicts, see Section 18.3.7, *LOCOSTO I/O Multiplexing Registers*.
   c. Program each pinout configuration register using the pin characteristics table (see Table 18-6).

Reset sources other than the power-up reset cannot reset the I/O configuration.

18.3.6.2 Solving Conflicts on Muxed Pads

Some functional interfaces can be mapped on several pins. This following is a method for choosing the optimized configuration:

1. Identify all required interfaces that are mapped on only one pin. These interfaces are easily identified. Only one ball is proposed in the pinout functional interface table for these interfaces (see Table 18-10 through Table 18-35).
2. For each pin, confirm in the alternate function modes whether there is a required function:
   a. If there is no required function in the alternate functions list, the pin can be selected.
   b. If there is a required alternate function on this pin, determine which of the following applies:
      i. The alternate function can be found on another pin: Select the pin and the alternate pin for the alternate function.
      ii. There is no alternative pin available for the alternate function: There is no solution; this configuration is not valid. Choose between the function and the alternate function. It is not possible to use both functions at the same time on the LOCOSTO device.
3. Apply the same method on interfaces that are multiplexed in several places. The following describes three situations that can occur:
   a. Only one solution considers all multiplexed pins, but it is already used by other selected interfaces. There is no conflict; use this solution.
   b. There are several solutions (the function can be mapped on several pins). Choose the solution for the lowest value of the MODE field in the pin configuration register.
   c. No solution remains for this function (all pins for which the function is available are already used). The LOCOSTO device does not support the chosen configuration.

18.3.6.3 Pin Programming Example

The following sequence describes how to program pins for the universal asynchronous receiver/transmitter (UART) interface:

1. Select the UART signals required on the interface considering the target application:
   a. Required signals: uart_rx, uart_rts, uart_tx, uart_cts
   b. Unused signals: rts_sdirda, dsr_rxir, dcd_txir.
      Hypothesis: The UART does not perform the infrared data association (IrDA) function.
2. Select the pins (see Table 18-31). All UART signals (except for uart_rts, which is found on only one pin) can be found on two pins. Select the pin based on the alternate function used at the system level.
   Hypothesis: The USB is required in the system, and it uses pins multiplexed with the UART.
3. Make a decision on unused signals. Because rts_sdirda is a dedicated pin, it cannot be used by another module. The other IrDA signals can be used by the camera module or as GPIOs.
4. Identify the pinout configuration registers to program (see Table 18-6 and Table 18-31):
   a. CONF_UART_CTS: Address 0xFFFE F1B0
   b. CONF_UART_TX: Address 0xFFFE F1AC
   c. CONF_UART_RX: Address 0xFFFE F1AE

d. CONF_USB_TXEN: Address 0xFFFE F10E

5. Select pull activation on input only (it is automatically disabled for output). A programmable combination of pullup and pulldown is available on the following pins:
   a. uart_cts (pullup)
   b. uart_rx (pulldown)
6. Program the register mode 1 selected for the usb_txen signal (MODE = 1). Concatenate with the PULL field value:
   a. Write at address 0xFFFE F1B0 with value 0b11XXX.
   b. Write at address 0xFFFE F1AC with value 0bXXXXX.
   c. Write at address 0xFFFE F1AE with value 0b01XXX.
   d. Write at address 0xFFFE F10E with value 0bXXXX1.

   X represents a reserved bit; a write has no effect on this bit.

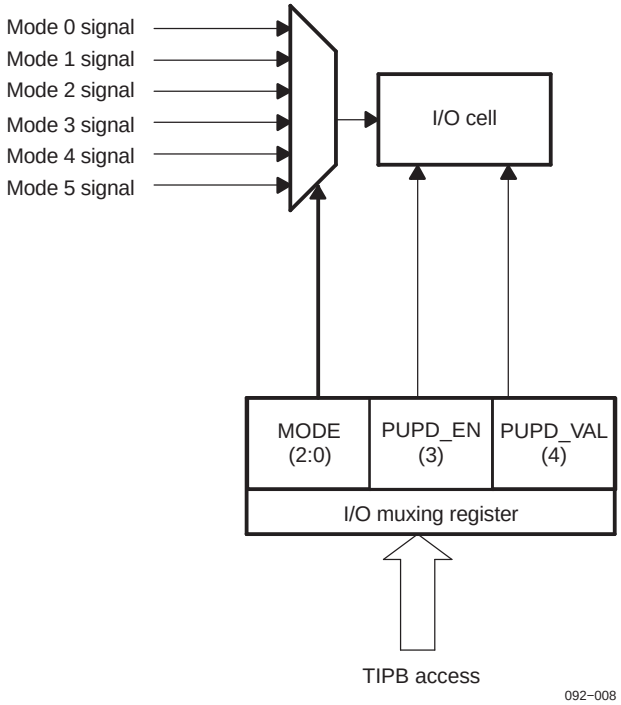
18.3.7 LOCOSTO I/O Multiplexing Registers

The I/O multiplexing configuration is set through registers that are available for each device pin. The I/O register MODE field (bits [2:0]) defines the signal routed to or from a device pin. The I/O can also be associated to a pullup or pulldown (PUPD_VAL), which is controlled by the pull enable (PUPD_EN).

The I/O reset configuration is mode 0.

Figure 18-8 shows the I/O multiplexing.

Figure 18-8. I/O Multiplexing Diagram



18.3.7.1 I/O Multiplexing Registers Mapping Summary

Table 18-36 lists the I/O multiplexing configuration registers.

Table 18-36. I/O Multiplexing Registers and Physical Addresses

Register Name	Type	Register Width (Bits)	Offset
CONF_GPIO_0	R/W	16	0xFFFE F100

Table 18-36. I/O Multiplexing Registers and Physical Addresses (continued)

Register Name	Type	Register Width (Bits)	Offset
CONF_GPIO_1	R/W	16	0xFFFE F102
CONF_GPIO_2	R/W	16	0xFFFE F104
CONF_USB_RCV	R/W	16	0xFFFE F108
CONF_USB_SE0	R/W	16	0xFFFE F10A
CONF_USB_DAT	R/W	16	0xFFFE F10C
CONF_USB_TXEN	R/W	16	0xFFFE F10E
CONF_TRST	R/W	16	0xFFFE F110
CONF_ABB_IRQ	R/W	16	0xFFFE F112
CONF_CSYNC	R/W	16	0xFFFE F114
CONF_CSCLK	R/W	16	0xFFFE F116
CONF_CDO	R/W	16	0xFFFE F118
CONF_GPIO_4	R/W	16	0xFFFE F11A
CONF_GPIO_5	R/W	16	0xFFFE F11C
CONF_USB_BOOT	R/W	16	0xFFFE F11E
CONF_SIM_RST	R/W	16	0xFFFE F120
CONF_SIM_IO	R/W	16	0xFFFE F122
CONF_SIM_CLK	R/W	16	0xFFFE F124
CONF_SIM_PWCTRL	R/W	16	0xFFFE F126
CONF_KBC_0	R/W	16	0xFFFE F128
CONF_KBC_1	R/W	16	0xFFFE F12A
CONF_KBC_2	R/W	16	0xFFFE F12C
CONF_KBC_3	R/W	16	0xFFFE F12E
CONF_TSPACT_11	R/W	16	0xFFFE F130
CONF_TSPACT_12	R/W	16	0xFFFE F132
CONF_TSPACT_13	R/W	16	0xFFFE F134
CONF_TSPACT_14	R/W	16	0xFFFE F136
CONF_TSPACT_15	R/W	16	0xFFFE F138
CONF_KBR_0	R/W	16	0xFFFE F13A
CONF_KBR_1	R/W	16	0xFFFE F13C
CONF_KBR_2	R/W	16	0xFFFE F13E
CONF_KBR_3	R/W	16	0xFFFE F140
CONF_GPIO_8	R/W	16	0xFFFE F142
CONF_GPIO_9	R/W	16	0xFFFE F144
CONF_GPIO_10	R/W	16	0xFFFE F146
CONF_GPIO_11	R/W	16	0xFFFE F148
CONF_GPIO_12	R/W	16	0xFFFE F14A
CONF_GPIO_13	R/W	16	0xFFFE F14C
CONF_LCD_NRST	R/W	16	0xFFFE F14E
CONF_LCD_STB	R/W	16	0xFFFE F150
CONF_LCD_RNW	R/W	16	0xFFFE F152
CONF_LCD_RS	R/W	16	0xFFFE F154
CONF_GPIO_17	R/W	16	0xFFFE F156
CONF_GPIO_18	R/W	16	0xFFFE F158
CONF_LCD_DATA_0	R/W	16	0xFFFE F15A
CONF_LCD_DATA_1	R/W	16	0xFFFE F15C
CONF_LCD_DATA_2	R/W	16	0xFFFE F15E

Table 18-36. I/O Multiplexing Registers and Physical Addresses (continued)

Register Name	Type	Register Width (Bits)	Offset
CONF_LCD_DATA_3	R/W	16	0xFFFE F160
CONF_LCD_DATA_4	R/W	16	0xFFFE F162
CONF_LCD_DATA_5	R/W	16	0xFFFE F164
CONF_LCD_DATA_6	R/W	16	0xFFFE F166
CONF_LCD_DATA_7	R/W	16	0xFFFE F168
CONF_GPIO_19	R/W	16	0xFFFE F16A
CONF_GPIO_20	R/W	16	0xFFFE F16C
CONF_GPIO_21	R/W	16	0xFFFE F16E
CONF_GPIO_22	R/W	16	0xFFFE F170
CONF_GPIO_23	R/W	16	0xFFFE F172
CONF_GPIO_24	R/W	16	0xFFFE F174
CONF_GPIO_25	R/W	16	0xFFFE F176
CONF_GPIO_26	R/W	16	0xFFFE F178
CONF_GPIO_27	R/W	16	0xFFFE F17A
CONF_GPIO_28	R/W	16	0xFFFE F17C
CONF_GPIO_29	R/W	16	0xFFFE F17E
CONF_GPIO_30	R/W	16	0xFFFE F180
CONF_TMS	R/W	16	0xFFFE F182
CONF_ND_NWP	R/W	16	0xFFFE F184
CONF_GPIO_31	R/W	16	0xFFFE F186
CONF_TCK	R/W	16	0xFFFE F188
CONF_GPIO_32	R/W	16	0xFFFE F18A
CONF_TDI	R/W	16	0xFFFE F18C
CONF_GPIO_33	R/W	16	0xFFFE F18E
CONF_GPIO_34	R/W	16	0xFFFE F190
CONF_ND_CE1	R/W	16	0xFFFE F192
CONF_GPIO_35	R/W	16	0xFFFE F194
CONF_GPIO_36	R/W	16	0xFFFE F196
CONF_GPIO_37	R/W	16	0xFFFE F198
CONF_GPIO_38	R/W	16	0xFFFE F19A
CONF_GPIO_39	R/W	16	0xFFFE F19C
CONF_NRDY	R/W	16	0xFFFE F19E
CONF_ADV	R/W	16	0xFFFE F1A0
CONF_GPIO_42	R/W	16	0xFFFE F1A2
CONF_GPIO_43	R/W	16	0xFFFE F1A4
CONF_GPIO_44	R/W	16	0xFFFE F1A6
CONF_GPIO_45	R/W	16	0xFFFE F1A8
CONF_GPIO_46	R/W	16	0xFFFE F1AA
CONF_UART_TX	R/W	16	0xFFFE F1AC
CONF_UART_RX	R/W	16	0xFFFE F1AE
CONF_UART_CTS	R/W	16	0xFFFE F1B0
CONF_NBSCAN	R/W	16	0xFFFE F1B2
CONF_ADD_21	R/W	16	0xFFFE F1B4
CONF_GPIO_47	R/W	16	0xFFFE F1B6
CONF_VFSRX	R/W	16	0xFFFE F1B8
CONF_GPIO_7	R/W	16	0xFFFE F1BA

Table 18-36. I/O Multiplexing Registers and Physical Addresses (continued)

Register Name	Type	Register Width (Bits)	Offset
CONF_TDO	R/W	16	0xFFFE F1BC
CONF_RNW	R/W	16	0xFFFE F1DE
CONF_NMOE	R/W	16	0xFFFE F1E0
CONF_NCS3	R/W	16	0xFFFE F1E2
CONF_CLK13MHZ_EN	R/W	16	0xFFFE F1E4
CONF_VDR	R/W	16	0xFFFE F1E6
CONF_VCLKRX	R/W	16	0xFFFE F1E8

18.3.7.2 I/O Multiplexing Registers Description

Table 18-37 through Table 18-137 describe the individual registers.

Table 18-37. CONF_GPIO_0

Address Offset	0x100		
Physical Address	0xFFFE F100	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_0: GPIO [IO] 01: dcd_txir: unused [IO] 10: cam_d_1: Camera digital image data bit 1 [I]	R/W	0b00

Table 18-38. CONF_GPIO_1

Address Offset	0x102		
Physical Address	0xFFFE F102	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_1: GPIO [IO] 01: pwt: Pulse Width Tone [O] 10: pmc_reset: Codec reset (Provided only to avoid C-port test case porting for LOCOSTO) [O]	R/W	0b00

Table 18-39. CONF_GPIO_2

Address Offset	0x104		
Physical Address	0xFFFE F104	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_2: GPIO [IO] 01: pwl: Pseudo-noise variable width light controller [O] 10: pmc_clk: Codec master clock input interface [I]	R/W	0b00

Table 18-40. CONF_SPARE_3

Address Offset	0x106		
Physical Address	0xFFFE F106	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:0	RESERVED		R	0b000

Table 18-41. CONF_USB_RCV

Address Offset	0x108		
Physical Address	0xFFFE F108	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: usb_rcv: Unused [I] 1: uart_cts: Unused [I]	R/W	0b0

Table 18-42. CONF_USB_SE0

Address Offset	0x10A		
Physical Address	0xFFFE F10A	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: usb_se0: Unused [IO] 1: uart_tx: Unused [O]	R/W	0b0

Table 18-43. CONF_USB_DAT

Address Offset	0x10C		
Physical Address	0xFFFE F10C	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: usb_dat: Unused [IO] 1: uart_rx: Unused [I]	R/W	0b0

Table 18-44. CONF_USB_TXEN

Address Offset	0x10E		
Physical Address	0xFFFE F10E	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															MODE

Bits	Field Name	Description	Type	Reset
15:1	RESERVED		R	0x000-
0	MODE	Select the pin mode: 0: usb_txen: Unused [O] 1: uart_rts: Unused [O]	R/W	0b0

Table 18-45. CONF_TRST

Address Offset	0x110		
Physical Address	0xFFFE F110	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:0	RESERVED		R	0b000

Table 18-46. CONF_ABB_IRQ

Address Offset	0x112		
Physical Address	0xFFFE F112	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:0	RESERVED		R	0b000

Table 18-47. CONF_CSYNC

Address Offset	0x114		
Physical Address	0xFFFE F114	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:0	RESERVED		R	0b000

Table 18-48. CONF_CSCLK

Address Offset	0x116		
Physical Address	0xFFFE F116	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:0	RESERVED		R	0b000

Table 18-49. CONF_CDO

Address Offset	0x118		
Physical Address	0xFFFE F118	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Bits	Field Name	Description	Type	Reset
15:0	RESERVED		R	0x0000

Table 18-50. CONF_GPIO_4

Address Offset	0x11A		
Physical Address	0xFFFE F11A	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_4: GPIO [IO] 01: cdi: Codec serial data input interface [I] 10: tspact_9: TPU2OCP synchronous activation signal [O] 11: lt2: Light output with PWM [O]	R/W	0b00

Table 18-51. CONF_GPIO_5

Address Offset	0x11C		
Physical Address	0xFFFE F11C	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: gpio_5: GPIO [IO] 1: tspact_8: TPU2OCP synchronous activation signal [O]	R/W	0b0

Table 18-52. CONF_USB_BOOT

Address Offset	0x11E		
Physical Address	0xFFFE F11E	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MODE	

Bits	Field Name	Description	Type	Reset
15:2	RESERVED		R	0x0000
1:0	MODE	Select the pin mode: 00: usb_boot: unused [I] 01: gpio_3: GPIO [IO] 10: lpg: LED pulse generator (Drive = 4 mA) [O]	R/W	0b00

Table 18-53. CONF_SIM_RST

Address Offset	0x120		
Physical Address	0xFFFE F120	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:0	RESERVED		R	0b000

Table 18-54. CONF_SIM_IO

Address Offset	0x122		
Physical Address	0xFFFE F122	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: sim_io: Smart Card interface data I/O line [IO] 1: force_0: Dummy signal for Spinner [O]	R/W	0b0

Table 18-55. CONF_SIM_CLK

Address Offset	0x124		
Physical Address	0xFFFE F124	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:0	RESERVED		R	0b000

Table 18-56. CONF_SIM_PWCTRL

Address Offset	0x126		
Physical Address	0xFFFE F126	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:0	RESERVED		R	0b000

Table 18-57. CONF_KBC_0

Address Offset	0x128		
Physical Address	0xFFFE F128	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															MODE

Bits	Field Name	Description	Type	Reset
15:1	RESERVED		R	0x0000
0	MODE	Select the pin mode: 0: kbc_0: Keyboard Matrix Column 0 [O] 1: test_port_17: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b0

Table 18-58. CONF_KBC_1

Address Offset	0x12A		
Physical Address	0xFFFE F12A	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															MODE

Bits	Field Name	Description	Type	Reset
15:1	RESERVED		R	0x0000
0	MODE	Select the pin mode: 0: kbc_1: Keyboard Matrix Column 0 [O] 1: test_port_16: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b0

Table 18-59. CONF_KBC_2

Address Offset	0x12C		
Physical Address	0xFFFE F12C	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															MODE

Bits	Field Name	Description	Type	Reset
15:1	RESERVED		R	0x0000
0	MODE	Select the pin mode: 0: kbc_2: Keyboard Matrix Column 2 [O] 1: test_port_15: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b0

Table 18-60. CONF_KBC_3

Address Offset	0x12E		
Physical Address	0xFFFE F12E	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															MODE

Bits	Field Name	Description	Type	Reset
15:1	RESERVED		R	0x0000
0	MODE	Select the pin mode: 0: kbc_3: Keyboard Matrix Column 3 [O] 1: test_port_14: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b0

Table 18-61. CONF_TSPACT_11

Address Offset	0x130		
Physical Address	0xFFFE F130	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															MODE

Bits	Field Name	Description	Type	Reset
15:1	RESERVED		R	0x0000
0	MODE	Select the pin mode: 0: tspact_11: TPU2OCP synchronous activation signal [O] 1: clkm_clk: ARM7 CLKM clock [I]	R/W	0b0

Table 18-62. CONF_TSPACT_12

Address Offset	0x132		
Physical Address	0xFFFE F132	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Bits	Field Name	Description	Type	Reset
15:0	RESERVED		R	0x0000

Table 18-63. CONF_TSPACT_13

Address Offset	0x134		
Physical Address	0xFFFE F134	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Bits	Field Name	Description	Type	Reset
15:0	RESERVED		R	0x0000

Table 18-64. CONF_TSPACT_14

Address Offset	0x136		
Physical Address	0xFFFE F136	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Bits	Field Name	Description	Type	Reset
15:0	RESERVED		R	0x0000

Table 18-65. CONF_TSPACT_15

Address Offset	0x138		
Physical Address	0xFFFE F138	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Bits	Field Name	Description	Type	Reset
15:0	RESERVED		R	0x0000

Table 18-66. CONF_KBR_0

Address Offset	0x13A		
Physical Address	0xFFFE F13A	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: kbr_0: Keyboard Matrix Row 0 [I] 1: test_port_22: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b0

Table 18-67. CONF_KBR_1

Address Offset	0x13C		
Physical Address	0xFFFE F13C	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: kbr_1: Keyboard Matrix Row 1 [I] 1: test_port_21: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b0

Table 18-68. CONF_KBR_2

Address Offset	0x13E		
Physical Address	0xFFFE F13E	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: kbr_2: Keyboard Matrix Row 2 [I] 1: test_port_20: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b0

Table 18-69. CONF_KBR_3

Address Offset	0x140		
Physical Address	0xFFFE F140	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: kbr_3: Keyboard Matrix Row 3 [I] 1: test_port_19: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b0

Table 18-70. CONF_GPIO_8

Address Offset	0x142		
Physical Address	0xFFFE F142	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: kbr_4: Keyboard Matrix Row 4 [I] 01: gpio_8: GPIO [IO] 10: test_port_18: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-71. CONF_GPIO_9

Address Offset	0x144		
Physical Address	0xFFFE F144	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON 1. After reset deassertion, PUPD_EN becomes 0. 2. During keyboard mode, when output is disabled, pull is enabled, provided PUPD_EN = 1.	R/W	0b0
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: kbc_4: Keyboard Matrix Row 4 [O] 01: gpio_9: GPIO [IO] 10: test_port_13: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-72. CONF_GPIO_10

Address Offset	0x146		
Physical Address	0xFFFE F146	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: nemu0: Test emulation 0 [I] 1: gpio_10: GPIO [IO]	R/W	0b0

Table 18-73. CONF_GPIO_11

Address Offset	0x148		
Physical Address	0xFFFE F148	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: nemu1: Test emulation 1 [I] 1: gpio_11: GPIO [IO]	R/W	0b0

Table 18-74. CONF_GPIO_12

Address Offset	0x14A		
Physical Address	0xFFFE F14A	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_12: GPIO [IO] 01: lpg: LED pulse generator (Drive = 4 mA) [O] 10: tspact_10: TPU2OCP synchronous activation signal [O]	R/W	0b00

Table 18-75. CONF_GPIO_13

Address Offset	0x14C		
Physical Address	0xFFFE F14C	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: gpio_13: GPIO [IO] 1: lcd_ncs0: LCD chip-select	R/W	0b0

Table 18-76. CONF_LCD_NRST

Address Offset	0x14E		
Physical Address	0xFFFE F14E	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: lcd_nrst: LCD controller dedicated reset [O] 1: test_port_12: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b0

Table 18-77. CONF_LCD_STB

Address Offset	0x150		
Physical Address	0xFFFE F150	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: lcd_stb: LCD strobe enable [O] 01: gpio_14: GPIO [IO] 10: test_port_11: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-78. CONF_LCD_RNW

Address Offset	0x152		
Physical Address	0xFFFE F152	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: lcd_rnw: unused [O] 01: gpio_15: GPIO [IO] 10: test_port_10: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-79. CONF_LCD_RS

Address Offset	0x154		
Physical Address	0xFFFE F154	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: lcd_rs: LCD data/control selection [O] 01: gpio_16: GPIO [IO] 10: test_port_9: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-80. CONF_GPIO_17

Address Offset	0x156		
Physical Address	0xFFFE F156	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_17: GPIO [IO] 01: lcd_ncs1: LCD chip-select [O] 10: test_port_8: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-81. CONF_GPIO_18

Address Offset	0x158		
Physical Address	0xFFFE F158	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: gpio_18: GPIO [IO] 1: nd_we: Flash write enable _UnicodeEncodeError_ 1.8 V [O]	R/W	0b0

Table 18-82. CONF_LCD_DATA_0

Address Offset	0x15A		
Physical Address	0xFFFE F15A	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MODE	

Bits	Field Name	Description	Type	Reset
15:2	RESERVED		R	0x0000
1:0	MODE	Select the pin mode: 00: lcd_data_0: LCD pixel data bit 0 [IO] 01: cam0_d_0: Camera digital image data bit 0 [I] 10: test_port_7: Debug bus [O] (For multiplexing, see Section 18.4.) 11: ndf_dyn_0: NAND flash data/address 0 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b00

Table 18-83. CONF_LCD_DATA_1

Address Offset	0x15C		
Physical Address	0xFFFE F15C	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MODE	

Bits	Field Name	Description	Type	Reset
15:2	RESERVED		R	0x0000
1:0	MODE	Select the pin mode: 00: lcd_data_1: LCD pixel data bit 1 [IO] 01: cam0_d_1: Camera digital image data bit 1 [I] 10: test_port_6: Debug bus [O] (For multiplexing, see Section 18.4.) 11: ndf_dyn_1: NAND flash data/address 1 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b00

Table 18-84. CONF_LCD_DATA_2

Address Offset	0x15E		
Physical Address	0xFFFE F15E	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MODE	

Bits	Field Name	Description	Type	Reset
15:2	RESERVED		R	0x0000
1:0	MODE	Select the pin mode: 00: lcd_data_2: LCD pixel data bit 2 [IO] 01: cam0_d_2: Camera digital image data bit 2 [I] 10: test_port_5: Debug bus [O] (For multiplexing, see Section 18.4.) 11: ndf_dyn_2: NAND flash data/address 2 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b00

Table 18-85. CONF_LCD_DATA_3

Address Offset	0x160		
Physical Address	0xFFFE F160	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MODE	

Bits	Field Name	Description	Type	Reset
15:2	RESERVED		R	0x0000
1:0	MODE	Select the pin mode: 00: lcd_data_3: LCD pixel data bit 3 [IO] 01: cam0_d_3: Camera digital image data bit 3 [I] 10: test_port_4: Debug bus [O] (For multiplexing, see Section 18.4.) 11: ndf_dyn_3: NAND flash data/address 3 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b00

Table 18-86. CONF_LCD_DATA_4

Address Offset	0x162		
Physical Address	0xFFFE F162	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MODE	

Bits	Field Name	Description	Type	Reset
15:2	RESERVED		R	0x0000
1:0	MODE	Select the pin mode: 00: lcd_data_4: LCD pixel data bit 4 [IO] 01: cam0_d_4: Camera digital image data bit 4 [I] 10: test_port_3: Debug bus [O] (For multiplexing,see Section 18.4.) 11: ndf_dyn_4: NAND flash data/address 4 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b00

Table 18-87. CONF_LCD_DATA_5

Address Offset	0x164		
Physical Address	0xFFFE F164	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MODE	

Bits	Field Name	Description	Type	Reset
15:2	RESERVED		R	0x0000
1:0	MODE	Select the pin mode: 00: lcd_data_5: LCD pixel data bit 5 [IO] 01: cam0_d_5: Camera digital image data bit 5 [I] 10: test_port_2: Debug bus [O] (For multiplexing, see Section 18.4.) 11: ndf_dyn_5: NAND flash data/address 5 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b00

Table 18-88. CONF_LCD_DATA_6

Address Offset	0x166		
Physical Address	0xFFFE F166	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MODE	

Bits	Field Name	Description	Type	Reset
15:2	RESERVED		R	0x0000
1:0	MODE	Select the pin mode: 00: lcd_data_6: LCD pixel data bit 6 [IO] 01: cam0_d_6: Camera digital image data bit 6 [I] 10: test_port_1: Debug bus [O] (For multiplexing, see Section 18.4.) 11: ndf_dyn_6: NAND flash data/address 6 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b00

Table 18-89. CONF_LCD_DATA_7

Address Offset	0x168		
Physical Address	0xFFFE F168	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MODE	

Bits	Field Name	Description	Type	Reset
15:2	RESERVED		R	0x0000
1:0	MODE	Select the pin mode: 00: lcd_data_7: LCD pixel data bit 7 [IO] 01: cam0_d_7: Camera digital image data bit 7 [I] 10: test_port_0: Debug bus [O] (For multiplexing, see Section 18.4.) 11: ndf_dyn_7: NAND flash data/address 7 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b00

Table 18-90. CONF_GPIO_19

Address Offset	0x16A		
Physical Address	0xFFFE F16A	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MODE	

Bits	Field Name	Description	Type	Reset
15:3	RESERVED		R	0x0000
2:0	MODE	Select the pin mode: 000: gpio_19: GPIO [IO] 001: cam_hs: Camera horizontal synchronization [I] 010: cam_d_3: Camera digital image data bit 3 [I] 011: test_port_32: Debug bus [O] (For multiplexing, see Section 18.4.) 100: ndf3: NAND flash data/address 3 _UnicodeEncodeError_ 1.8V [IO]	R/W	0b000

Table 18-91. CONF_GPIO_20

Address Offset	0x16C		
Physical Address	0xFFFE F16C	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								MODE							

Bits	Field Name	Description	Type	Reset
15:3	RESERVED		R	0x0000
2:0	MODE	Select the pin mode: 000: gpio_20: GPIO [IO] 001: cam_vs: Camera vertical synchronization [I] 010: cam_d_3: Camera digital image data bit 3 [I] 011: test_port_31: Debug bus [O] (For multiplexing, see Section 18.4.) 100: ndf3: NAND flash data/address 3 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b000

Table 18-92. CONF_GPIO_21

Address Offset	0x16E		
Physical Address	0xFFFE F16E	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MODE	

Bits	Field Name	Description	Type	Reset
15:2	RESERVED		R	0x0000
1:0	MODE	Select the pin mode: 00: gpio_21: GPIO [IO] 01: cam_lclk: Camera image data latch clock [I] 10: test_port_30: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-93. CONF_GPIO_22

Address Offset	0x170		
Physical Address	0xFFFE F170	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													MODE		

Bits	Field Name	Description	Type	Reset
15:3	RESERVED		R	0x0000
2:0	MODE	Select the pin mode: 000: gpio_22: GPIO [IO] 001: cam_xclk: Camera output clock [O] 010: cam_d_3: Camera digital image data bit 3 [I] 011: test_port_29: Debug bus [O] (For multiplexing, see Section 18.4.) 100: ndf3: NAND flash data/address 3 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b000

Table 18-94. CONF_GPIO_23

Address Offset	0x172		
Physical Address	0xFFFE F172	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: gpio_23: GPIO [IO] 1: spi_clk: SPI clk [IO]	R/W	0b0

Table 18-95. CONF_GPIO_24

	0x174		
Physical Address Address Offset	0xFFFE F174	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: gpio_24: GPIO [IO] 1: spi_data_miso: SPI data: Master in/Slave out [IO]	R/W	0b0

Table 18-96. CONF_GPIO_25

Address Offset	0x176		
Physical Address	0xFFFE F176	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: gpio_25: GPIO [IO] 1: spi_data_mosi: SPI data: Master out/Slave in [IO]	R/W	0b0

Table 18-97. CONF_GPIO_26

Address Offset	0x178		
Physical Address	0xFFFE F178	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: gpio_26: GPIO [IO] 1: spi_ncs0: SPI enable 0 [IO]	R/W	0b0

Table 18-98. CONF_GPIO_27

Address Offset	0x17A		
Physical Address	0xFFFE F17A	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: gpio_27: GPIO [IO] 1: spi_ncs1: SPI enable 1 [O]	R/W	0b0

Table 18-99. CONF_GPIO_28

Address Offset	0x17C		
Physical Address	0xFFFE F17C	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_28: GPIO [IO] 01: spi_ncs2: SPI Enable 2 [O] 10: ndf7: NAND flash data/address 7 _UnicodeEncodeError_ 1.8 V [IO] 11: cam_d_7: Camera digital image data bit 7 [I]	R/W	0b00

Table 18-100. CONF_GPIO_29

Address Offset	0x17E		
Physical Address	0xFFFE F17E	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0

Bits	Field Name	Description	Type	Reset
1:0	MODE	Select the pin mode: 00: gpio_29: GPIO [IO] 01: bu: Buzzer output with PWM [O] 10: ndf6: NAND flash data/address 6 _UnicodeEncodeError_ 1.8 V [IO] 11: cam_d_6: Camera digital image data bit 6 [I]	R/W	0b00

Table 18-101. CONF_GPIO_30

Address Offset	0x180		
Physical Address	0xFFFE F180	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_30: GPIO [IO] 01: lt3: Light output with PWM [O] 10: ndf5: NAND flash data/address 5 _UnicodeEncodeError_ 1.8 V [IO] 11: cam_d_5: Camera digital image data bit 5 [I]	R/W	0b00

Table 18-102. CONF_TMS

Address Offset	0x182		
Physical Address	0xFFFE F182	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1

Bits	Field Name	Description	Type	Reset
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:0	RESERVED		R	0b000

Table 18-103. CONF_ND_NWP

Address Offset	0x184		
Physical Address	0xFFFE F184	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MODE	

Bits	Field Name	Description	Type	Reset
15:2	RESERVED		R	0x0000
1:0	MODE	Select the pin mode: 00: nd_nwp: Flash write protect _UnicodeEncodeError_ 1.8 V [O] 01: ndf4: NAND flash data/address 4 _UnicodeEncodeError_ 1.8 V [IO] 10: cam_d_4: Camera digital image data bit 4 [I] 11: test_port_28: Debug bus [O] (For multiplexing, see Section 18.4.)	R	0b00

Table 18-104. CONF_GPIO_31

Address Offset	0x186		
Physical Address	0xFFFE F186	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_31: GPIO [IO] 01: nd_re: Flash read enable _UnicodeEncodeError_ 1.8 V [O] 10: test_port_27: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-105. CONF_TCK

Address Offset	0x188		
Physical Address	0xFFFE F188	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:0	RESERVED		R	0b000

Table 18-106. CONF_GPIO_32

Address Offset	0x18A		
Physical Address	0xFFFE F18A	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_32: GPIO [IO] 01: nd_cle: Flash common latch enable _UnicodeEncodeError_ 1.8 V [O] 10: test_port_26: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-107. CONF_TDI

Address Offset	0x18C		
Physical Address	0xFFFE F18C	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:0	RESERVED		R	0b000

Table 18-108. CONF_GPIO_33

Address Offset	0x18E		
Physical Address	0xFFFE F18E	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_33: GPIO [IO] 01: nd_ale: Flash address latch enable _UnicodeEncodeError_ 1.8 V [O] 10: test_port_25: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-109. CONF_GPIO_34

Address Offset	0x190		
Physical Address	0xFFFE F190	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_34: GPIO [IO] 01: nd_rdy: Flash ready signal/busy signal _UnicodeEncodeError_ 1.8 V [I] 10: test_port_24: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-110. CONF_ND_CE1

Address Offset	0x192		
Physical Address	0xFFFE F192	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															MODE

Bits	Field Name	Description	Type	Reset
15:1	RESERVED		R	0x0000
0	MODE	Select the pin mode: 0: nd_ce1: Flash chip enable 1 _UnicodeEncodeError_ 1.8 V [O] 1: test_port_23: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b0

Table 18-111. CONF_GPIO_35

Address Offset	0x194		
Physical Address	0xFFFE F194	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_35: GPIO [IO] 01: ncs2: Chip-select 2 [O] 10: ndf3: NAND flash data/address 3 _UnicodeEncodeError_ 1.8V [IO]	R/W	0b00

Table 18-112. CONF_GPIO_36

Address Offset	0x196		
Physical Address	0xFFFE F196	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_36: GPIO [IO] 01: ncs1: Chip-select 1 [O] 10: ndf2: NAND flash data/address 2 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b00

Table 18-113. CONF_GPIO_37

Address Offset	0x198		
Physical Address	0xFFFE F198	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_37: GPIO [IO] 01: add_23: EMIF address 23 [O] 10: ndf1: NAND flash data/address 1 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b00

Table 18-114. CONF_GPIO_38

Address Offset	0x19A		
Physical Address	0xFFFE F19A	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	MODE		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:0	MODE	Select the pin mode: 000: ncs0: Chip-select 0 [O] 001: gpio_38: GPIO [IO] 010: lt1: Light output with PWM [O] 011: tspact_7: TPU2OCP synchronous activation signal [O] 100: ndf0: NAND flash data/address 0 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b000

Table 18-115. CONF_GPIO_39

Address Offset	0x19C		
Physical Address	0xFFFE F19C	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_39: GPIO [IO] 01: add_22: EMIF address 22 [O] 10: ndf0: NAND flash data/address 0 _UnicodeEncodeError_ 1.8 V [IO]	R/W	0b00

Table 18-116. CONF_NRDY

Address Offset	0x19E		
Physical Address	0xFFFE F19E	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: nrdy: External devices wait assertion [I] 1: gpio_40: GPIO [IO]	R/W	0b0

Table 18-117. CONF_ADV

Address Offset	0x1A0		
Physical Address	0xFFFE F1A0	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: adv: Address valid signal for synchronous burst memory control [O] 1: gpio_41: GPIO [IO]	R/W	0b0

Table 18-118. CONF_GPIO_42

Address Offset	0x1A2		
Physical Address	0xFFFE F1A2	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: gpio_42: GPIO [IO] 1: ckm: Clock signal for synchronous burst memory control [IO]	R/W	0b0

Table 18-119. CONF_GPIO_43

Address Offset	0x1A4		
Physical Address	0xFFFE F1A4	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_43: GPIO [IO] 01: mcsi_ck: Unused [IO] 10: csclk_o: Codec serial clock interface [O] 11: test_port_33: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-120. CONF_GPIO_44

Address Offset	0x1A6		
Physical Address	0xFFFE F1A6	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0

Bits	Field Name	Description	Type	Reset
1:0	MODE	Select the pin mode: 00: gpio_44: GPIO [IO] 01: mcsi_fs: unused [IO] 10: csync_o: Codec frame synchronization interface [O] 11: test_port_34: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-121. CONF_GPIO_45

Address Offset	0x1A8		
Physical Address	0xFFFE F1A8	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_45: GPIO [IO] 01: mcsi_tx: Unused [O] 10: cdo: Codec serial data output interface [O] 11: test_port_35: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-122. CONF_GPIO_46

Address Offset	0x1AA		
Physical Address	0xFFFE F1AA	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1

Bits	Field Name	Description	Type	Reset
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_46: GPIO [IO] 01: mcsi_rx: Unused [I] 10: test_port_36: Debug bus [O] (For multiplexing, see Section 18.4.)	R/W	0b00

Table 18-123. CONF_UART_TX

Address Offset	0x1AC		
Physical Address	0xFFFE F1AC	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Bits	Field Name	Description	Type	Reset
15:0	RESERVED		R	0x0000

Table 18-124. CONF_UART_RX

Address Offset	0x1AE		
Physical Address	0xFFFE F1AE	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:0	RESERVED		R	0b00

Table 18-125. CONF_UART_CTS

Address Offset	0x1B0		
Physical Address	0xFFFE F1B0	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:0	RESERVED		R	0b00

Table 18-126. CONF_NBSCAN

Address Offset	0x1B2		
Physical Address	0xFFFE F1B2	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:0	RESERVED		R	0b000

Table 18-127. CONF_ADD_21

Address Offset	0x1B4		
Physical Address	0xFFFE F1B4	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		MODE

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:1	RESERVED		R	0b00
0	MODE	Select the pin mode: 0: add_21: EMIF address 21 [O] 1: gpio_6: GPIO [IO]	R/W	0b0

Table 18-128. CONF_GPIO_47

Address Offset	0x1B6		
Physical Address	0xFFFE F1B6	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED	MODE	

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b1
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2	RESERVED		R	0b0
1:0	MODE	Select the pin mode: 00: gpio_47: GPIO [IO] 01: dsr_rxir: Unused [I] 10: cam_d_0: Camera digital image data bit 0 [I]	R/W	0b00

Table 18-129. CONF_VFSRX

Address Offset	0x1B8		
Physical Address	0xFFFE F1B8	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:0	RESERVED		R	0b000

Table 18-130. CONF_GPIO_7

Address Offset	0x1BA		
Physical Address	0xFFFE F1BA	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	MODE		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b1
2:0	MODE	Select the pin mode: 000: gpio_7: GPIO [IO] 001: nfwp: Flash write protect [O] 010: tspact_7: TPU2OCP synchronous activation signal [O] 011: lt1: Light output with PWM [O] 100: ndf2: NAND flash data/address 2 _UnicodeEncodeError_ 1.8 V [IO] 101: cam_d_2: Camera digital image data bit 2 [I]	R	0b000

Table 18-131. CONF_TDO

Address Offset	0x1BC		
Physical Address	0xFFFE F1BC	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Bits	Field Name	Description	Type	Reset
15:0	RESERVED		R	0x0000

Table 18-132. CONF_RNW

Address Offset	0x1DE		
Physical Address	0xFFFE F1DE	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															MODE

Bits	Field Name	Description	Type	Reset
15:1	RESERVED		R	0x0000
0	MODE	Select the pin mode: 0: rnw: Memory read(high)/write (low) control [O] 1: force_0: Dummy signal for Spinner [O]	R/W	0b0

Table 18-133. CONF_NMOE

Address Offset	0x1E0		
Physical Address	0xFFFE F1E0	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															MODE

Bits	Field Name	Description	Type	Reset
15:1	RESERVED		R	0x0000
0	MODE	Select the pin mode: 0: nmoe: Memory output enable (active low) 1: force_0: Dummy signal for Spinner [O]	R/W	0b0

Table 18-134. CONF_NCS3

Address Offset	0x1E2		
Physical Address	0xFFFE F1E2	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															MODE

Bits	Field Name	Description	Type	Reset
15:1	RESERVED		R	0x0000
0	MODE	Select the pin mode: 0: ncs3: Chip-select 3 [O] 1: force_0: Dummy signal for Spinner [O]	R/W	0b0

Table 18-135. CONF_CLK13MHZ_EN

Address Offset	0x1E4		
Physical Address	0xFFFE F1E4	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:0	RESERVED		R	0b000

Table 18-136. CONF_VDR

Address Offset	0x1E6		
Physical Address	0xFFFE F1E6	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:0	RESERVED		R	0b000

Table 18-137. CONF_VCLKRX

Address Offset	0x1E8		
Physical Address	0xFFFE F1E8	**Instance**	Configuration
Description	I/O multiplexing register		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PUPD_VAL	PUPD_EN	RESERVED		

Bits	Field Name	Description	Type	Reset
15:5	RESERVED		R	0x0000
4	PUPD_VAL	Pull value: 0: Pulldown 1: Pullup	R/W	0b0
3	PUPD_EN	Pull activation: 0: Pull OFF 1: Pull ON	R/W	0b0
2:0	RESERVED		R	0b000

18.4 Debug Module

18.4.1 Introduction

The debug module is available for general-purpose (GP) and high-security (HS) LOCOSTO devices.

Note: The debug module allows for the observation of individual signals and processed signal combinations. Signals cannot be controlled from this module.

18.4.2 General Description

Figure 18-9 shows the debug module block diagram, and Table 18-138 describes the blocks in the diagram.

Figure 18-9. Debug Module Block Diagram

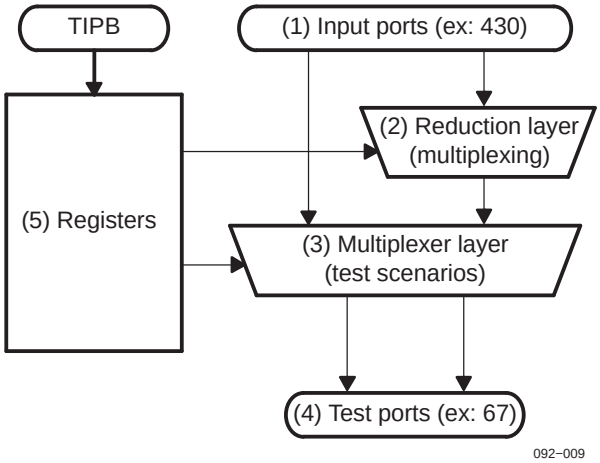


Table 18-138. Debug Module Blocks Description

Debug Module Blocks	Description
Input ports	Some, but not necessarily all, can be observed individually. Observed input signals include MPU IRQs, DSP INTs, DMA requests, and PRCM signals.
Reduction layer	The reduction layer uses multiplexers for certain signals, to reduce the number of input signals. In addition, it provides straight-through access for some input signals.
Multiplexer layer	This layer maps input signals from the reduction layer and direct input signals to the test ports. A maximum of eight signals (or eight modes) is available to each test port.
TIPB	TI peripheral bus (TIPB) ports. The address size is generated using the register offset range.
Test ports	Test port outputs
Registers	The registers are used to select signals to observe. There is one set of registers for the reduction layer and one set for the multiplexer layer.

For further information on specific debugging scenarios, see the *Debug Features* application note (APN211).

18.4.3 Observability Table

Table 18-139 describes the observable signals: MPU IRQs, DSP INTs, DMA requests, and PRCM signals.

Table 18-139. Observable Signals

Name Register	Muxed Signals	
	Signal Name	Description
CONF_DSP_VIEW_0	dsp_int[0:11]	DSP interrupt source

Table 18-139. Observable Signals (continued)

Name Register	Muxed Signals	
	Signal Name	**Description**
CONF_DSP_VIEW_1	dsp_int[0:11]	DSP interrupt source
CONF_DSP_VIEW_2	dsp_int[0:11]	DSP interrupt source
CONF_DMA_VIEW_0	dma_req[0:31]	DMA interrupt request
CONF_DMA_VIEW_1	dma_req[0:31]	DMA interrupt request
CONF_DMA_VIEW_2	dma_req[0:31]	DMA interrupt request
CONF_DMA_VIEW_3	dma_req[0:31]	DMA interrupt request
CONF_ARM_VIEW_0	arm_irq[0:31]	Internal interrupt to ARM
CONF_ARM_VIEW_1	arm_irq[0:31]	Internal interrupt to ARM
CONF_ARM_VIEW_2	arm_irq[0:31]	Internal interrupt to ARM
CONF_DMA_REQ_0	DMA_REQUEST_P0_0	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P1_0	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P2_0	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P3_0	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P4_0	Decimal encoding of DMA channel read/write
CONF_DMA_REQ_1	DMA_REQUEST_P0_1	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P1_1	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P2_1	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P3_1	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P4_1	Decimal encoding of DMA channel read/write
CONF_DMA_REQ_2	DMA_REQUEST_P0_2	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P1_2	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P2_2	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P3_2	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P4_2	Decimal encoding of DMA channel read/write
CONF_DMA_REQ_3	DMA_REQUEST_P0_3	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P1_3	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P2_3	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P3_3	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P4_3	Decimal encoding of DMA channel read/write
CONF_DMA_REQ_S	DMA_REQ_S[0:4]	DMA request stall
CONF_DMA_REQ_V	DMA_REQ_V[0:4]	DMA request valid
CONF_MUX_VIEW0	Drp_dbb_tx_req	DRP Tx Interrupt
	Drp_dbb_sysclk	DRP retimed clock
CONF_MUX_VIEW1	Drp_dbb_sinterrupt	DRP general-purpose interrupt
CONF_MUX_VIEW2	Drp_dbb_rx_req	DRP Rx interrupt
	Drp_dbb_rx_req	DRP Rx interrupt
CONF_MUX_VIEW3	Clk13_en	Clock enable
CONF_MUX_VIEW4	Dcxo_sysclken	Clock enable for dcxo
CONF_MUX_VIEW5	Reserved	
CONF_MUX_VIEW6	Dcxo_clrz	DRP reset signal
CONF_MUX_VIEW7	Reserved	
CONF_MUX_VIEW8	Reserved	
CONF_MUX_VIEW9	Reserved	
CONF_MUX_VIEW10	Reserved	
CONF_MUX_VIEW11	PP_CLK_EN	Parallel port clock enable
CONF_MUX_VIEW12	INT10n	DMA interrupt

Table 18-139. Observable Signals (continued)

Name Register	Muxed Signals	
	Signal Name	**Description**
	PP_CLK_IN	Parallel port clock input
CONF_MUX_VIEW13	ND_Flash_CLK_REQ	Clock request for data transfer
CONF_MUX_VIEW14	Reserved	
CONF_MUX_VIEW15	Reserved	
CONF_MUX_VIEW16	Reserved	
CONF_MUX_VIEW17	Reserved	
CONF_MUX_VIEW18	Reserved	
CONF_MUX_VIEW19	Reserved	
CONF_MUX_VIEW20	Reserved	
CONF_MUX_VIEW21	Reserved	
CONF_MUX_VIEW22	TPU_FRAME_IT	TPU frame interrupt
CONF_MUX_VIEW23	Reserved	
CONF_MUX_VIEW24	Reserved	
CONF_MUX_VIEW25	Reserved	
CONF_MUX_VIEW26	Reserved	
CONF_MUX_VIEW27	Reserved	
CONF_MUX_VIEW28	Reserved	
CONF_MUX_VIEW29	Reserved	
CONF_MUX_VIEW30	Drp_dbb_txdma_req	DRP data Tx DMA request
CONF_MUX_VIEW31	TPU_FRAME_IT	TPU frame interrupt
CONF_MUX_VIEW32	TPU_FRAME_IT	TPU frame interrupt
CONF_MUX_VIEW33	Reserved	
CONF_MUX_VIEW34	Drp_dbb_rx_req	DRP Rx interrupt
CONF_MUX_VIEW35	DRPTOPTESTCLKOUT(1)	DRP multiuse debug clock output
	DRPTOPTESTCLKOUT(0)	DRP multiuse debug clock output
CONF_MUX_VIEW36	Reserved	
CONF_DEBUG_SEL_ TST 0	DMA_REQUEST_P0_0	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P1_0	Decimal encoding of DMA channel read/write
	DMA_REQUEST_Px_0	Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_0)
	MUX_VIEW_0	Moved from reduction layer (see CONF_MUX_VIEW0)
CONF_DEBUG_SEL_ TST 1	DMA_REQUEST_P0_1	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P1_1	Decimal encoding of DMA channel read/write
	DMA_REQUEST_Px_1	Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_1)
	MUX_VIEW_1	Moved from reduction layer (see CONF_MUX_VIEW1)
CONF_DEBUG_SEL_ TST 2	DMA_REQUEST_P0_2	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P1_2	Decimal encoding of DMA channel read/write
	DMA_REQUEST_Px_2	Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_2)
	MUX_VIEW_2	Moved from reduction layer (see CONF_MUX_VIEW2)
CONF_DEBUG_SEL_ TST 3	DMA_REQUEST_P0_3	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P1_3	Decimal encoding of DMA channel read/write
	DMA_REQUEST_Px_3	Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_3)

Table 18-139. Observable Signals (continued)

Name Register	Muxed Signals	
	Signal Name	**Description**
	MUX_VIEW_3	Moved from reduction layer (see CONF_MUX_VIEW3)
CONF_DEBUG_SEL_ TST 4	DMA_REQ_S0	DMA request stall
	DMA_REQ_S1	DMA request stall
	DMA_REQ_Sx	DMA request stall multiplexed (see CONF_DMA_REQ_S)
	MUX_VIEW_4	Moved from reduction layer (see CONF_MUX_VIEW4)
CONF_DEBUG_SEL_ TST 5	DMA_REQ_V0	DMA request valid
	DMA_REQ_V1	DMA request valid
	DMA_REQ_Vx	DMA request valid multiplexed (see CONF_DMA_REQ_V)
	MUX_VIEW_5	Moved from reduction layer (see CONF_MUX_VIEW5)
CONF_DEBUG_SEL_ TST 6	DMA_REQUEST_P3_0	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P4_0	Decimal encoding of DMA channel read/write
	DMA_REQUEST_Px_0	Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_0)
	MUX_VIEW_6	Moved from reduction layer (see CONF_MUX_VIEW6)
CONF_DEBUG_SEL_ TST 7	DMA_REQUEST_P3_1	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P4_1	Decimal encoding of DMA channel read/write
	DMA_REQUEST_Px_1	Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_1)
	MUX_VIEW_7	Moved from reduction layer (see CONF_MUX_VIEW7)
CONF_DEBUG_SEL_ TST_8	DSP_nINT_VIEW_0	DSP interrupt source multiplexed (see CONF_DSP_VIEW_0)
	DMA_REQUEST_P3_2	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P4_2	Decimal encoding of DMA channel read/write
	DMA_REQUEST_Px_2	Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_2)
	ARM_nIRQ_VIEW_0	Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_0)
	MUX_VIEW_8	Moved from reduction layer (see CONF_MUX_VIEW8)
CONF_DEBUG_SEL_ TST_9	DSP_nINT_VIEW_1	DSP interrupt source multiplexed (see CONF_DSP_VIEW_1)
	DMA_REQUEST_P3_3	Decimal encoding of DMA channel read/write
	DMA_REQUEST_P4_3	Decimal encoding of DMA channel read/write
	DMA_REQUEST_Px_3	Decimal encoding of DMA channel read/write
	ARM_nIRQ_VIEW_1	Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_1)
	MUX_VIEW_9	Moved from reduction layer (see CONF_MUX_VIEW9)
CONF_DEBUG_SEL_ TST 10	Drp_dbb_rxdma_req	DRP data Rx DMA request
	DMA_REQ_S3	DMA request stall
	DMA_REQ_S4	DMA request stall
	DMA_REQ_Sx	DMA request stall multiplexed (see CONF_DMA_REQ_S)
	nDMA_REQ_VIEW_0	DMA interrupt request multiplexed (see CONF_DMA_VIEW_0)

Table 18-139. Observable Signals (continued)

Name Register	Muxed Signals	
	Signal Name	**Description**
	MUX_VIEW_10	Moved from reduction layer (see CONF_MUX_VIEW10)
CONF_DEBUG_SEL_ TST 11	INT10n	DMA interrupt
	DMA_REQ_V3	DMA request valid
	DMA_REQ_V4	DMA request valid
	DMA_REQ_Vx	DMA request valid multiplexed (see CONF_DMA_REQ_V)
	nDMA_REQ_VIEW_1	DMA interrupt request multiplexed (see CONF_DMA_VIEW_1)
	MUX_VIEW_11	Moved from reduction layer (see CONF_MUX_VIEW11)
CONF_DEBUG_SEL_ TST 12	IRQ4	TPU frame interrupt
	Drp_dbb_rxdma_req	DRP data Rx DMA request
	IACKn	Interrupt acknowledge
	nMREQ	ARM not memory request
	ARM_nIRQ_VIEW_1	Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_1)
	MUX_VIEW_12	Moved from reduction layer (see CONF_MUX_VIEW12)
CONF_DEBUG_SEL_ TST 13	DMA_REQUEST_P0_0	Decimal encoding of DMA channel read/write
	DSP_nINT_VIEW_0	DSP interrupt source multiplexed (see CONF_DSP_VIEW_0)
	nDMA_REQ_VIEW_0	DMA interrupt request multiplexed (see CONF_DMA_VIEW_0)
	DMA_REQUEST_Px_0	Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_0)
	MUX_VIEW_13	Moved from reduction layer (see CONF_MUX_VIEW13)
CONF_DEBUG_SEL_ TST 14	DMA_REQUEST_P0_1	Decimal encoding of DMA channel read/write
	DSP_nINT_VIEW_1	DSP interrupt source multiplexed (see CONF_DSP_VIEW_1)
	nDMA_REQ_VIEW_1	DMA interrupt request multiplexed (see CONF_DMA_VIEW_1)
	DMA_REQUEST_Px_1	Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_1)
	MUX_VIEW_14	Moved from reduction layer (see CONF_MUX_VIEW14)
CONF_DEBUG_SEL_ TST 15	DMA_REQUEST_P0_2	Decimal encoding of DMA channel read/write
	Drp_dbb_rxdma_req	DRP data Rx DMA request
	INT10n	DMA interrupt
	DMA_REQUEST_Px_2	Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_2)
	MUX_VIEW_15	Moved from reduction layer (see CONF_MUX_VIEW15)
CONF_DEBUG_SEL_ TST 16	DMA_REQUEST_P0_3	Decimal encoding of DMA channel read/write
	INT10n	DMA interrupt
	IRQ14	DMA interrupt
	DMA_REQUEST_Px_3	Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_3)
	MUX_VIEW_16	Moved from reduction layer (see CONF_MUX_VIEW16)

Table 18-139. Observable Signals (continued)

Name Register	Muxed Signals	
	Signal Name	**Description**
CONF_DEBUG_SEL_ TST_17	nDMA_REQ_VIEW_0	DMA interrupt request multiplexed (see CONF_DMA_VIEW_0)
	IRQ4	TPU frame interrupt
	ARM_nIRQ_VIEW_0	Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_0)
	DMA_REQ_Sx	DMA request stall multiplexed (see CONF_DMA_REQ_S)
	MUX_VIEW_17	Moved from reduction layer (see CONF_MUX_VIEW17)
CONF_DEBUG_SEL_ TST_18	nDMA_REQ_VIEW_1	DMA interrupt request multiplexed (see CONF_DMA_VIEW_1)
	IRQ14	DMA interrupt
	ARM_nIRQ_VIEW_1	Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_1)
	DMA_REQ_Vx	DMA request valid multiplexed (see CONF_DMA_REQ_V)
	MUX_VIEW_18	Moved from reduction layer (see CONF_MUX_VIEW18)
CONF_DEBUG_SEL_ TST 19	DMA_REQUEST_P3_0	Decimal encoding of DMA channel read/write
	INT4n	DRP DBB Tx interrupt
	DSP_nINT_VIEW_0	DSP interrupt source multiplexed (see CONF_DSP_VIEW_0)
	Drp_dbb_rx_req	DRP Rx interrupt
	nDMA_REQ_VIEW_0	DMA interrupt request multiplexed (see CONF_DMA_VIEW_0)
	MUX_VIEW_19	Moved from reduction layer (see CONF_MUX_VIEW19)
CONF_DEBUG_SEL_ TST 20	DMA_REQUEST_P3_1	Decimal encoding of DMA channel read/write
	nFIQ	ARM fast interrupts
	DSP_nINT_VIEW_1	DSP interrupt source multiplexed (see CONF_DSP_VIEW_1)
	NMIIT	DSP Nonmaskable interrupt
	nDMA_REQ_VIEW_1	DMA interrupt request multiplexed (see CONF_DMA_VIEW_1)
	MUX_VIEW_20	Moved from reduction layer (see CONF_MUX_VIEW20)
CONF_DEBUG_SEL_ TST 21	DMA_REQUEST_P3_2	Decimal encoding of DMA channel read/write
	nDMA_REQ_VIEW_0	DMA interrupt request multiplexed (see CONF_DMA_VIEW_0)
	nIRQ	ARM slow interrupt
	INT3n	MCSI Tx interrupt
	nDMA_REQ_VIEW_2	DMA interrupt request multiplexed (see CONF_DMA_VIEW_2)
	MUX_VIEW_21	Moved from reduction layer (see CONF_MUX_VIEW21)
CONF_DEBUG_SEL_ TST 22	DMA_REQUEST_P3_3	Decimal encoding of DMA channel read/write
	nDMA_REQ_VIEW_1	DMA interrupt request multiplexed (see CONF_DMA_VIEW_1)
	ARM_nIRQ_VIEW_0	Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_0)
	ARM_nIRQ_VIEW_2	Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_2)
	IACKn	Interrupt acknowledge

Table 18-139. Observable Signals (continued)

Name Register	Muxed Signals	
	Signal Name	**Description**
	nDMA_REQ_VIEW_3	DMA interrupt request multiplexed (see CONF_DMA_VIEW_3)
	MUX_VIEW_22	Moved from reduction layer (see CONF_MUX_VIEW22)
CONF_DEBUG_SEL_ TST 23	Dcxo_clrz	DRP reset signal
	IRQ14	DMA interrupt
	IACKn	Interrupt acknowledge
	ND_Flash_CLK_REQ	Clock request for data transfer
	DSP_nINT_VIEW_0	DSP interrupt source multiplexed (see CONF_DSP_VIEW_0)
	MUX_VIEW_23	Moved from reduction layer (see CONF_MUX_VIEW23)
CONF_DEBUG_SEL_ TST 24	INT4n	DRP DBB Tx interrupt
	DSP_nINT_VIEW_1	DSP interrupt source multiplexed (see CONF_DSP_VIEW_1)
	MUX_VIEW_24	Moved from reduction layer (see CONF_MUX_VIEW24)
CONF_DEBUG_SEL_ TST 25	Dcxo_sysclken	Clock enable for dcxo
	Drp_dbb_tx_req	DRP Tx interrupt
	ARM_nIRQ_VIEW_1	Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_1)
	MUX_VIEW_25	Moved from reduction layer (see CONF_MUX_VIEW25)
CONF_DEBUG_SEL_ TST 26	Drp_dbb_sysclk	DRP retimed clock
	IACKn	Interrupt acknowledge
	ARM_nIRQ_VIEW_0	Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_0)
	MUX_VIEW_26	Moved from reduction layer (see CONF_MUX_VIEW26)
CONF_DEBUG_SEL_ TST 27	Clk13_en	Clock enable
	DSP_nINT_VIEW_2	DSP interrupt source multiplexed (see CONF_DSP_VIEW_2)
	Drp_dbb_rx_req	DRP Rx interrupt
	MUX_VIEW_27	Moved from reduction layer (see CONF_MUX_VIEW27)
CONF_DEBUG_SEL_ TST_28	ARM_nIRQ_VIEW_2	Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_2)
	DSP_nINT_VIEW_2	DSP interrupt source multiplexed (see CONF_DSP_VIEW_2)
	IACKn	Interrupt acknowledge
	MUX_VIEW_28	Moved from reduction layer (see CONF_MUX_VIEW28)
CONF_DEBUG_SEL_ TST_29	DMA_CLK_REQ	Signal from DMA module when going from idle to active mode
	MUX_VIEW_29	Moved from reduction layer (see CONF_MUX_VIEW29)
CONF_DEBUG_SEL_ TST_30	DSP_nINT_VIEW_0	DSP interrupt source multiplexed (see CONF_DSP_VIEW_0)
	MUX_VIEW_30	Moved from reduction layer (see CONF_MUX_VIEW30)
CONF_DEBUG_SEL_ TST_31	ARM_nIRQ_VIEW_1	Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_1)

Table 18-139. Observable Signals (continued)

Name Register	Muxed Signals	
	Signal Name	**Description**
	MUX_VIEW_31	Moved from reduction layer (see CONF_MUX_VIEW31)
CONF_DEBUG_SEL_ TST_32	ARM_nIRQ_VIEW_0	Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_0)
	MUX_VIEW_32	Moved from reduction layer (see CONF_MUX_VIEW32)
CONF_DEBUG_SEL_ TST 33	DMA_REQ_V0	DMA request valid
	DSP_nINT_VIEW_0	DSP interrupt source multiplexed (see CONF_DSP_VIEW_0)
	ARM_nIRQ_VIEW_1	Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_1)
	MUX_VIEW_33	Moved from reduction layer (see CONF_MUX_VIEW33)
CONF_DEBUG_SEL_ TST 34	DMA_REQ_S0	DMA request stall
	DSP_nINT_VIEW_1	DSP interrupt source multiplexed (see CONF_DSP_VIEW_1)
	DPLL_CLKOUT	DPLL clock out
	MUX_VIEW_34	Moved from reduction layer (see CONF_MUX_VIEW34)
CONF_DEBUG_SEL_ TST 35	DMA_REQ_V3	DMA request valid
	INT10n	DMA Interrupt
	nDMA_REQ_VIEW_0	DMA interrupt request multiplexed (see CONF_DMA_VIEW_0)
	nIRQ	ARM slow interrupt
	DSP_CLKOUT	DSP clock out
	MUX_VIEW_35	Moved from reduction layer (see CONF_MUX_VIEW35)
CONF_DEBUG_SEL_ TST 36	DMA_REQ_S3	DMA request stall
	IRQ4	TPU frame interrupt
	nDMA_REQ_VIEW_1	DMA interrupt request multiplexed (see CONF_DMA_VIEW_1)
	Drp_dbb_sinterrupt	DRP general-purpose interrupt
	MCLK	ARM clock in debug signal
	MUX_VIEW_36	Moved from reduction layer (see CONF_MUX_VIEW36)

18.4.4 Debug Module Registers

Table 18-140 lists the base address and address space for the LOCOSTO debug module instance. Table 18-141 provides a summary of the debug module registers. Table 18-142 through Table 18-232 describe the individual registers.

Table 18-140. Instance Summary

Module Name	Base Address	Size
Debug	0xFFFE 9200	2K bytes

18.4.4.1 Reduction Layer Debug Module

This section describes the registers used in the reduction layer debug module. These registers are listed in Table 18-141.

Table 18-141. Debug Module Registers Summary (Reduction Layer Only)

Register Name	Type	Register Width (Bits)	Offset
CONF_DSP_VIEW_0	R/W	8	0x41
CONF_DSP_VIEW_1	R/W	8	0x42
CONF_DSP_VIEW_2	R/W	8	0x43
CONF_DMA_VIEW_0	R/W	8	0x44
CONF_DMA_VIEW_1	R/W	8	0x45
CONF_DMA_VIEW_2	R/W	8	0x46
CONF_DMA_VIEW_3	R/W	8	0x47
CONF_ARM_VIEW_0	R/W	8	0x48
CONF_ARM_VIEW_1	R/W	8	0x49
CONF_ARM_VIEW_2	R/W	8	0x4A
CONF_DMA_REQ_0	R/W	8	0x4B
CONF_DMA_REQ_1	R/W	8	0x4C
CONF_DMA_REQ_2	R/W	8	0x4D
CONF_DMA_REQ_3	R/W	8	0x4E
CONF_DMA_REQ_S	R/W	8	0x4F
CONF_DMA_REQ_V	R/W	8	0x50
CONF_MUX_VIEW0	R/W	8	0x51
CONF_MUX_VIEW1	R/W	8	0x52
CONF_MUX_VIEW2	R/W	8	0x53
CONF_MUX_VIEW3	R/W	8	0x54
CONF_MUX_VIEW4	R/W	8	0x55
CONF_MUX_VIEW5	R/W	8	0x56
CONF_MUX_VIEW6	R/W	8	0x57
CONF_MUX_VIEW7	R/W	8	0x58
CONF_MUX_VIEW8	R/W	8	0x59
CONF_MUX_VIEW9	R/W	8	0x5A
CONF_MUX_VIEW10	R/W	8	0x5B
CONF_MUX_VIEW11	R/W	8	0x5C
CONF_MUX_VIEW12	R/W	8	0x5D
CONF_MUX_VIEW13	R/W	8	0x5E
CONF_MUX_VIEW14	R/W	8	0x5F
CONF_MUX_VIEW15	R/W	8	0x60
CONF_MUX_VIEW16	R/W	8	0x61
CONF_MUX_VIEW17	R/W	8	0x62
CONF_MUX_VIEW18	R/W	8	0x63
CONF_MUX_VIEW19	R/W	8	0x64
CONF_MUX_VIEW20	R/W	8	0x65
CONF_MUX_VIEW21	R/W	8	0x66
CONF_MUX_VIEW22	R/W	8	0x67
CONF_MUX_VIEW23	R/W	8	0x68
CONF_MUX_VIEW24	R/W	8	0x69
CONF_MUX_VIEW25	R/W	8	0x6A
CONF_MUX_VIEW26	R/W	8	0x6B
CONF_MUX_VIEW27	R/W	8	0x6C
CONF_MUX_VIEW28	R/W	8	0x6D
CONF_MUX_VIEW29	R/W	8	0x6E
CONF_MUX_VIEW30	R/W	8	0x6F
CONF_MUX_VIEW31	R/W	8	0x70

Table 18-141. Debug Module Registers Summary (Reduction Layer Only) (continued)

Register Name	Type	Register Width (Bits)	Offset
CONF_MUX_VIEW32	R/W	8	0x71
CONF_MUX_VIEW33	R/W	8	0x72
CONF_MUX_VIEW34	R/W	8	0x73
CONF_MUX_VIEW35	R/W	8	0x74
CONF_MUX_VIEW36	R/W	8	0x75

18.4.4.2 Register Descriptions

Table 18-142 through Table 18-194 describe the individual reduction layer debug module registers.

Table 18-142. CONF_DSP_VIEW_0

Address Offset	0x41		
Physical Address	0xFFFE 9241	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED				MULT			

Bits	Field Name	Description	Type	Reset
7:4	RESERVED		R	0x0000
3:0	MULT	DSP_nINT_VIEW_0: DSP interrupt source (multiplexed) dsp_int(MULT) was chosen from among the 12 DSP interrupt sources.	R/W	0x0

Table 18-143. CONF_DSP_VIEW_1

Address Offset	0x42		
Physical Address	0xFFFE 9242	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED				MULT			

Bits	Field Name	Description	Type	Reset
7:4	RESERVED		R	0x0000
3:0	MULT	DSP_nINT_VIEW_1: DSP interrupt source (multiplexed) dsp_int(MULT) was chosen from among the 12 DSP interrupt sources.	R/W	0x0

Table 18-144. CONF_DSP_VIEW_2

Address Offset	0x43		
Physical Address	0xFFFE 9243	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED				MULT			

Bits	Field Name	Description	Type	Reset
7:4	RESERVED		R	0x0000
3:0	MULT	DSP_nINT_VIEW_2: DSP interrupt source (multiplexed) dsp_int(MULT) was chosen from among the 12 DSP interrupt sources.	R/W	0x0

Table 18-145. CONF_DMA_VIEW_0

Address Offset	0x44		
Physical Address	0xFFFE 9244	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED				MULT			

Bits	Field Name	Description	Type	Reset
7:5	RESERVED		R	0x0000
4:0	MULT	DMA_nREQ_VIEW_0: DMA interrupt request (multiplexed) dma_req(MULT) was chosen from among the 32 DMA interrupt requests.	R/W	0x00

Table 18-146. CONF_DMA_VIEW_1

Address Offset	0x45		
Physical Address	0xFFFE 9245	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED				MULT			

Bits	Field Name	Description	Type	Reset
7:5	RESERVED		R	0x0000
4:0	MULT	DMA_nREQ_VIEW_1: DMA interrupt request (multiplexed) dma_req(MULT) was chosen from among the 32 DMA interrupt requests.	R/W	0x00

Table 18-147. CONF_DMA_VIEW_2

Address Offset	0x46		
Physical Address	0xFFFE 9246	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:5	RESERVED		R	0x0000
4:0	MULT	DMA_nREQ_VIEW_2: DMA interrupt request (multiplexed) dma_req(MULT) was chosen from among the 32 DMA interrupt requests.	R/W	0x00

Table 18-148. CONF_DMA_VIEW_3

Address Offset	0x47		
Physical Address	0xFFFE 9247	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:5	RESERVED		R	0x0000
4:0	MULT	DMA_nREQ_VIEW_3: DMA interrupt request (multiplexed) dma_req(MULT) was chosen from among the 32 DMA interrupt requests.	R/W	0x00

Table 18-149. CONF_ARM_VIEW_0

Address Offset	0x48		
Physical Address	0xFFFE 9248	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:5	RESERVED		R	0x0000
4:0	MULT	ARM_nIRQ_VIEW_0: Internal interrupt to ARM (multiplexed) arm_irq(MULT) was chosen from among the 32 internal interrupts to ARM.	R/W	0x00

Table 18-150. CONF_ARM_VIEW_1

Address Offset	0x49		
Physical Address	0xFFFE 9249	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:5	RESERVED		R	0x0000
4:0	MULT	ARM_nIRQ_VIEW_1: Internal interrupt to ARM (multiplexed) arm_irq(MULT) was chosen from among the 32 internal interrupts to ARM.	R/W	0x00

Table 18-151. CONF_ARM_VIEW_2

Address Offset	0x4A		
Physical Address	0xFFFE 924A	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:5	RESERVED		R	0x0000
4:0	MULT	ARM_nIRQ_VIEW_2: Internal interrupt to ARM (multiplexed) arm_irq(MULT) was chosen from among the 32 internal interrupts to ARM.	R/W	0x00

Table 18-152. CONF_DMA_REQ_0

Address Offset	0x4B		
Physical Address	0xFFFE 924B	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	DMA_REQUEST_Px_0: Decimal encoding of DMA channel read/write. There are five channels, each of which can have read/write. Therefore, 10 combinations are encoded in four lines (multiplexed). 000: DMA_REQUEST_P0_0 001: DMA_REQUEST_P1_0 010: DMA_REQUEST_P2_0 011: DMA_REQUEST_P3_0 100: DMA_REQUEST_P4_0	R/W	0x0

Table 18-153. CONF_DMA_REQ_1

Address Offset	0x4C		
Physical Address	0xFFFE 924C	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	DMA_REQUEST_Px_1: Decimal encoding of DMA channel read/write. There are five channels, each of which can have read/write. Therefore, 10 combinations are encoded in four lines (multiplexed). 000: DMA_REQUEST_P0_1 001: DMA_REQUEST_P1_1 010: DMA_REQUEST_P2_1 011: DMA_REQUEST_P3_1 100: DMA_REQUEST_P4_1	R/W	0x0

Table 18-154. CONF_DMA_R EQ_2

Address Offset	0x4D		
Physical Address	0xFFFE 924D	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	DMA_REQUEST_Px_2: Decimal encoding of DMA channel read/write. There are five channels, each of which can have read/write. Therefore, 10 combinations are encoded in four lines (multiplexed). 000: DMA_REQUEST_P0_2 001: DMA_REQUEST_P1_2 010: DMA_REQUEST_P2_2 011: DMA_REQUEST_P3_2 100: DMA_REQUEST_P4_2	R/W	0x0

Table 18-155. CONF_DMA_REQ_3

Address Offset	0x4E		
Physical Address	0xFFFE 924E	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	DMA_REQUEST_Px_3: Decimal encoding of DMA channel read/write. There are five channels, each of which can have read/write. Therefore, 10 combinations are encoded in four lines (multiplexed). 000: DMA_REQUEST_P0_3 001: DMA_REQUEST_P1_3 010: DMA_REQUEST_P2_3 011: DMA_REQUEST_P3_3 100: DMA_REQUEST_P4_3	R/W	0x0

Table 18-156. CONF_DMA_REQ_S

Address Offset	0x4F		
Physical Address	0xFFFE 924F	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	DMA_REQ_Sx: DMA request stall (multiplexed) DMA_REQ_S(MULT) was chosen from among the five DMA request stall.	R/W	0x0

Table 18-157. CONF_DMA_REQ_V

Address Offset	0x50		
Physical Address	0xFFFE 9250	**Instance**	Debug
Description			
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	DMA_REQ_Vx: DMA request valid (multiplexed) DMA_REQ_V(MULT) was chosen from among the five DMA request valid.	R/W	0x0

Table 18-158. CONF_MUX_VIEW0

Address Offset	0x51		
Physical Address	0xFFFE 9251	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_0: Moved to multiplexer layer (multiplexed) 000: Drp_dbb_tx_req: DRP Tx interrupt 001: DTS(31): DRP digital test bus 010: Drp_dbb_sysclk: DRP retimed clock 011: DTS(15): DRP digital test bus 100: USIM_TX_STATE(0:0): Indicates state of Tx state-machine 101: Dbb_drp_mcmdm(0:0): DRP2 OCP memory signals	R/W	0x0

Table 18-159. CONF_MUX_VIEW1

Address Offset	0x52		
Physical Address	0xFFFE 9252	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_1: Moved to multiplexer layer (multiplexed) 000: Drp_dbb_sinterrupt: DRP general-purpose interrupt 001: DTS(30) : DRP digital test bus 010: DTS(30) : DRP digital test bus 011: DTS(14) : DRP digital test bus 100: USIM_TX_STATE(1:1) : Indicates state of Tx state-machine 101: Dbb_drp_mcmdm(1:1) : DRP2 OCP memory signals	R/W	0x0

Table 18-160. CONF_MUX_VIEW2

Address Offset	0x53		
Physical Address	0xFFFE 9253	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_2: Moved to multiplexer layer (multiplexed) 000: Drp_dbb_rx_req: DRP Rx interrupt 001: DTS(29) : DRP digital test bus 010: Drp_dbb_rx_req: DRP Rx interrupt 011: DTS(13) : DRP digital test bus 100: USIM_RX_STATE(0:0) : Indicates state of Rx state-machine 101: Dbb_drp_mcmdm(2:2) : DRP(2:2) OCP memory signals	R/W	0x0

Table 18-161. CONF_MUX_VIEW3

Address Offset	0x54		
Physical Address	0xFFFE 9254	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_3: Moved to multiplexer layer (multiplexed) 000: Drp_dbb_scmdaccpt(2:2): Read/write command accepted (DRP to DBB - OCP(2:2)) 001: DTS(28) : DRP digital test bus 010: Clk13_en: clock enable 011: DTS(12) : DRP digital test bus 100: USIM_RX_STATE(1:1): Indicates state of Rx state-machine	R/W	0x0

Table 18-162. CONF_MUX_VIEW4

Address Offset	0x55		
Physical Address	0xFFFE 9255	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_4: Moved to multiplexer layer (multiplexed) 000: Drp_dbb_scmdaccpt(1:1): Read/write command 001: Accepted (DRP to DBB - OCP(1:1)) 010: Dcxo_sysclken: Clock enable for dcxo 011: DTS(11): DRP digital test bus 100: USIM_STATE(0:0): Indicates state of the USIM state-machine	R/W	0x0

Table 18-163. CONF_MUX_VIEW5

Address Offset	0x56		
Physical Address	0xFFFE 9256	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_5: Moved to multiplexer layer (multiplexed) 000: APC_Start_Internal: Internal APC start signal (not TSPACT) 001: DTS(26): DRP digital test bus 010: Dcxo_sleepz : Sleep signal for dcxo 011: DTS(10): DRP digital test bus 100: USIM_STATE(1:1): Indicates state of the USIM state-machine	R/W	0x0

Table 18-164. CONF_MUX_VIEW6

Address Offset	0x57		
Physical Address	0xFFFE 9257	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_6: Moved to multiplexer layer (multiplexed) 000: TSPACT(1:1): GSM synchronous signal bit (1:1) 001: DTS(25): DRP digital test bus 010: Dcxo_clrz: DRP reset signal 011: DTS(09): DRP digital test bus 100: USIM_STATE(2:2): Indicates state of the USIM state-machine	R/W	0x0

Table 18-165. CONF_MUX_VIEW7

Address Offset	0x58		
Physical Address	0xFFFE 9258	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_7: Moved to multiplexer layer (multiplexed) 000: TSPACT(0:0): GSM synchronous signal bit (0:0) 001: DTS(24): DRP digital test bus 010: TSPACT(0:0): GSM synchronous signal bit (0:0) 011: DTS(08): DRP digital test bus 100: USIM_STATE(3:3): Indicates state of the USIM state-machine	R/W	0x0

Table 18-166. CONF_MUX_VIEW8

Address Offset	0x59		
Physical Address	0xFFFE 9259	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_8: Moved to multiplexer layer (multiplexed) 00: KB_STATE(0:0) : Key board internal state machine 01: DTS(23) : DRP digital test bus 10: USIM_START_BIT : USIM start bit 11: DTS(07) : DRP digital test bus	R/W	0x0

Table 18-167. CONF_MUX_VIEW9

Address Offset	0x5A		
Physical Address	0xFFFE 925A	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_9: Moved to multiplexer layer (multiplexed) 000: KB_STATE(1:1) : Key board internal state machine 001: DTS(22) : DRP digital test bus 010: GEA_WORKING : GEA module is working 011: DTS(06) : DRP digital test bus 100: Dbb_drp_srespm(0:0) : DRP2 OCP memory signals	R/W	0x0

Table 18-168. CONF_MUX_VIEW10

Address Offset	0x5B		
Physical Address	0xFFFE 925B	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_10: Moved to multiplexer layer (multiplexed) 000: KB_STATE(2:2) : Key board internal state machine 001: DTS(21) : DRP digital test bus 010: GEA_DL_nUL : GEA module is processing downlink not uplink 011: DTS(05) : DRP digital test bus 100: Dbb_drp_srespm(1:1) : DRP2 OCP memory signals	R/W	0x0

Table 18-169. CONF_MUX_VIEW11

Address Offset	0x5C		
Physical Address	0xFFFE 925C	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_11: Moved to multiplexer layer (multiplexed) 000: KB_STATE(3:3) : Key board internal state machine 001: DTS(20) : DRP digital test bus 010: PP_CLK_EN: Parallel port clock enable 011: DTS(04) : DRP digital test bus 100: Reserv Dbb_drp_scmdaccptm : DRP2 OCP memory signals ed	R/W	0x0

Table 18-170. CONF_MUX_VIEW12

Address Offset	0x5D		
Physical Address	0xFFFE 925D	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_12: Moved to multiplexer layer (multiplexed) 00: INT10n: DMA interrupt 01: DTS(19) : DRP digital test bus 10: PP_CLK_IN: Parallel port clock input 11: DTS(03) : DRP digital test bus	R/W	0x0

Table 18-171. CONF_MUX_VIEW13

Address Offset	0x5E		
Physical Address	0xFFFE 925E	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_13: Moved to multiplexer layer (multiplexed) 00: ND_Flash_CLK_REQ: Clock request for data transfer 01: DTS(08) : DRP digital test bus 10: DTS(09) : DRP digital test bus 11: mcu_nstrobe : MCU strobe	R/W	0x0

Table 18-172. CONF_MUX_VIEW14

Address Offset	0x5F		
Physical Address	0xFFFE 925F	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_14: Moved to multiplexer layer (multiplexed) 000: ND_Flash_State(0:0) : Indicates state of the NAND Flash state-machine 001: DTS(07) : DRP digital test bus 010: DTS(08) : DRP digital test bus 011: Dbb_drp_mcmd1(0:0) : (0:0)00: idle 100: dsp_cs(0:0) : DSP chip-select for different modules in wrapper 101: Dbb_drp_mcmd2(0:0) : (0:0)00: idle	R/W	0x0

Table 18-173. CONF_MUX_VIEW15

Address Offset	0x60		
Physical Address	0xFFFE 9260	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_15: Moved to multiplexer layer (multiplexed) 000: ND_Flash_State(1:1) : Indicates state of the NAND Flash state-machine 001: DTS(06) : DRP digital test bus 010: DTS(07) : DRP digital test bus 011: Dbb_drp_mcmd1(1:1) : DBB to DRP - OCP(1:1) 100: dsp_cs(1:1) : DSP chip-select for different modules in wrapper 101: Dbb_drp_mcmd2(1:1) : DBB to DRP - OCP2	R/W	0x0

Table 18-174. CONF_MUX_VIEW16

Address Offset	0x61		
Physical Address	0xFFFE 9261	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_16: Moved to multiplexer layer (multiplexed) 000: ND_Flash_State(2:2) : Indicates state of the NAND Flash state-machine 001: DTS(05) : DRP digital test bus 010: DTS(06) : DRP digital test bus 011: Dbb_drp_mcmd1(2:2) : DBB to DRP - OCP1 100: dsp_cs(2:2) : DSP chip-select for different modules in wrapper 101: Dbb_drp_mcmd2(2:2) : DBB to DRP - OCP(2:2)	R/W	0x0

Table 18-175. CONF_MUX_VIEW17

Address Offset	0x62		
Physical Address	0xFFFE 9262	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_17: Moved to multiplexer layer (multiplexed) 000: ND_Flash_State(3:3) : Indicates state of the NAND Flash state-machine 001: DTS(04) : DRP digital test bus 010: DTS(05) : DRP digital test bus 011: Drp_dbb_sresp1(0:0) : (0:0)0: idle 100: dsp_cs(3:3) : DSP chip-select for different modules in wrapper 101: Drp_dbb_sresp2(0:0) : (0:0)0: idle	R/W	0x0

Table 18-176. CONF_MUX_VIEW18

Address Offset	0x63		
Physical Address	0xFFFE 9263	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_18: Moved to multiplexer layer (multiplexed) 000: ND_Flash_State(4:4) : Indicates state of the NAND Flash state-machine 001: DTS(03) : DRP digital test bus 010: DTS(04) : DRP digital test bus 011: Drp_dbb_sresp1(1:1) : DRP to DBB OCP(1:1) 100: dsp_cs(4:4) : DSP chip-select for different modules in wrapper 101: Drp_dbb_sresp2(1:1) : DRP to DBB OCP2	R/W	0x0

Table 18-177. CONF_MUX_VIEW19

Address Offset	0x64		
Physical Address	0xFFFE 9264	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_19: Moved to multiplexer layer (multiplexed) 00: DSP read signal 01: DRP digital test bus 10: DRP digital test bus	R/W	0x0

Table 18-178. CONF_MUX_VIEW20

Address Offset	0x65		
Physical Address	0xFFFE 9265	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_20: Moved to multiplexer layer (multiplexed) 00: Dsp_nready : DSP ready signal 01: DTS(01) : DRP digital test bus 10: DTS(02) : DRP digital test bus	R/W	0x0

Table 18-179. CONF_MUX_VIEW21

Address Offset	0x66		
Physical Address	0xFFFE 9266	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_21: Moved to multiplexer layer (multiplexed) 00: Dsp_nstrobe : DSP strobe signal 01: DTS(00) : DRP digital test bus 10: DTS(01) : DRP digital test bus	R/W	0x0

Table 18-180. CONF_MUX_VIEW22

Address Offset	0x67		
Physical Address	0xFFFE 9267	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_22: Moved to multiplexer layer (multiplexed) 00: Drp_dbb_scmdaccpt(2:2) : Read/write command accepted (DRP to DBB - OCP(2:2)) 01: TPU_FRAME_IT: TPU frame interrupt 10: DTS(00:00) : DRP digital test bus	R/W	0x0

Table 18-181. CONF_MUX_VIEW23

Address Offset	0x68		
Physical Address	0xFFFE 9268	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_23: Moved to multiplexer layer (multiplexed) 00: DTS(14) : DRP digital test bus 01: DTS(15) : DRP digital test bus 10: Dbb_drp_mcmd2(0:0) : (0:0)00: idle 11: mcu_cs(1:1) : MCU chip-select for different modules in wrapper	R/W	0x0

Table 18-182. CONF_MUX_VIEW24

Address Offset	0x69		
Physical Address	0xFFFE 9269	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_24: Moved to multiplexer layer (multiplexed) 00: DTS(13) : DRP digital test bus 01: DTS(14) : DRP digital test bus	R/W	0x0

Bits	Field Name	Description	Type	Reset
		10: Dbb_drp_mcmd2(0:0) : (0:0)00: idle 11: mcu_cs(2:2) : MCU chip-select for different modules in wrapper		

Table 18-183. CONF_MUX_VIEW25

Address Offset	0x6A		
Physical Address	0xFFFE 926A	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_25: Moved to multiplexer layer (multiplexed) 00: DTS(12) : DRP digital test bus 01: DTS(12) : DRP digital test bus 10: Dbb_drp_mcmd2(1:1) : DBB to DRP - OCP2 11: mcu_cs(3:3) : MCU chip-select for different modules in wrapper	R/W	0x0

Table 18-184. CONF_MUX_VIEW26

Address Offset	0x6B		
Physical Address	0xFFFE 926B	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_26: Moved to multiplexer layer (multiplexed) 00: DTS(11) : DRP digital test bus 01: DTS(12) : DRP digital test bus 10: Dbb_drp_mcmd2(2:2) : DBB to DRP - OCP(2:2) 11: mcu_cs(4:4) : MCU chip-select for different modules in wrapper	R/W	0x0

Table 18-185. CONF_MUX_VIEW27

Address Offset	0x6C				
Physical Address	0xFFFE 926C	**Instance**	Debug		
Description	Reduction layer debug module register				
Type	R/W				
Write Latency	Not relevant				

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_27: Moved to multiplexer layer (multiplexed) 00: DTS(10) : DRP digital test bus 01: DTS(11) : DRP digital test bus 10: Drp_dbb_sresp2(0:0) : (0:0)0: idle 11: mcu_rnw : MCU read signal	R/W	0x0

Table 18-186. CONF_MUX_VIEW28

Address Offset	0x6D				
Physical Address	0xFFFE 926D	**Instance**	Debug		
Description	Reduction layer debug module register				
Type	R/W				
Write Latency	Not relevant				

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_28: Moved to multiplexer layer (multiplexed) 00: DTS(09) : DRP digital test bus 01: DTS(10) : DRP digital test bus 10: Drp_dbb_sresp2(1:1) : DRP to DBB OCP2 11: mcu_nready : MCU ready signal	R/W	0x0

Table 18-187. CONF_MUX_VIEW29

Address Offset	0x6E				
Physical Address	0xFFFE 926E	**Instance**	Debug		
Description	Reduction layer debug module register				
Type	R/W				
Write Latency	Not relevant				

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_29: Moved to multiplexer layer (multiplexed) 00: DTS(18) : DRP digital test bus 01: XF : DSP external flag 10: TSPACT(2) : GSM synchronous signal bit (2:2)	R/W	0x0

Bits	Field Name	Description	Type	Reset
		11: APC_Start : Start APC signal		

Table 18-188. CONF_MUX_VIEW30

Address Offset	0x6F		
Physical Address	0xFFFE 926F	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_30: Moved to multiplexer layer (multiplexed) 00: DTS(17:17) : DRP digital test bus 01: Drp_dbb_txdma_req: DRP data Tx DMA request 10: TSPACT(4:4) : GSM synchronous signal bit (4:4)	R/W	0x0

Table 18-189. CONF_MUX_VIEW31

Address Offset	0x70		
Physical Address	0xFFFE 9270	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED							MULT

Bits	Field Name	Description	Type	Reset
7:1	RESERVED		R	0x0000
0	MULT	MUX_VIEW_31: Moved to multiplexer layer (multiplexed) 0: DTS(16:16) : DRP digital test bus 1: TPU_FRAME_IT: TPU frame interrupt	R/W	0x0

Table 18-190. CONF_MUX_VIEW32

Address Offset	0x71		
Physical Address	0xFFFE 9271	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MULT		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_32: Moved to multiplexer layer (multiplexed) 000: DTS(15:15) : DRP digital test bus 001: mcu_cs(0:0) : MCU chip-select for different modules in wrapper	R/W	0x0

Bits	Field Name	Description	Type	Reset
		010: mcu_nstrobe : MCU strobe		
		011: TPU_FRAME_IT: TPU frame interrupt		
		100: APC_Start_Internal : Internal APC start signal (not TSPACT)		

Table 18-191. CONF_MUX_VIEW33

Address Offset	0x72		
Physical Address	0xFFFE 9272	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_33: Moved to multiplexer layer (multiplexed)	R/W	0x0
		00: TSPACT(0) : GSM synchronous signal bit (1:0)		
		01: DTS(02) : DRP digital test bus		
		10: TSPACT(1) : GSM synchronous signal bit (1:0)		
		11: slicer_en : ULPD state machine output		

Table 18-192. CONF_MUX_VIEW34

Address Offset	0x73		
Physical Address	0xFFFE 9273	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_34: Moved to multiplexer layer (multiplexed)	R/W	0x0
		00: Drp_dbb_rx_req: DRP Rx interrupt		
		01: DTS(01) : DRP digital test bus		
		10: APC_Start_Internal : Internal APC start signal (not TSPACT)		

Table 18-193. CONF_MUX_VIEW35

Address Offset	0x74		
Physical Address	0xFFFE 9274	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MULT	MUX_VIEW_35: Moved to multiplexer layer (multiplexed) 000: DRPTOPTESTCLKOUT(1): DRP multiuse debug clock output 001: DTS(00) : DRP digital test bus 010: DRPTOPTESTCLKOUT(0): DRP multiuse debug clock output 011: TSPACT(3) : GSM synchronous signal bit (3:3) 100: tcxo_en : ULPD state machine output 101: APC_BG_EN : APC bandgap enabel now	R/W	0x0

Table 18-194. CONF_MUX_VIEW36

Address Offset	0x75		
Physical Address	0xFFFE 9275	**Instance**	Debug
Description	Reduction layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MULT	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x0000
1:0	MULT	MUX_VIEW_36: Moved to multiplexer layer (multiplexed) 00: Drp_dbb_scmdaccpt(1:1) : Read/write command accepted (DRP to DBB - OCP(1:1)) 01: TSPACT(5) : GSM synchronous signal bit (5:5) 10: rf_en : ULPD state machine output	R/W	0x0

18.4.4.3 Multiplexer Layer Debug Module

This section describes the registers used in the multiplexer layer debug module. These registers are listed in Table 18-195.

Table 18-195. Debug Module Registers Summary (Multiplexer Layer Only)

Register Name	Type	Register Width (Bits)	Offset
CONF_DEBUG_SEL_TST_0	R/W	8	0x00
CONF_DEBUG_SEL_TST_1	R/W	8	0x01
CONF_DEBUG_SEL_TST_2	R/W	8	0x02
CONF_DEBUG_SEL_TST_3	R/W	8	0x03
CONF_DEBUG_SEL_TST_4	R/W	8	0x04
CONF_DEBUG_SEL_TST_5	R/W	8	0x05
CONF_DEBUG_SEL_TST_6	R/W	8	0x06
CONF_DEBUG_SEL_TST_7	R/W	8	0x07
CONF_DEBUG_SEL_TST_8	R/W	8	0x08
CONF_DEBUG_SEL_TST_9	R/W	8	0x09
CONF_DEBUG_SEL_TST_10	R/W	8	0x0A
CONF_DEBUG_SEL_TST_11	R/W	8	0x0B
CONF_DEBUG_SEL_TST_12	R/W	8	0x0C
CONF_DEBUG_SEL_TST_13	R/W	8	0x0D
CONF_DEBUG_SEL_TST_14	R/W	8	0x0E

Table 18-195. Debug Module Registers Summary (Multiplexer Layer Only) (continued)

Register Name	Type	Register Width (Bits)	Offset
CONF_DEBUG_SEL_TST_15	R/W	8	0x0F
CONF_DEBUG_SEL_TST_16	R/W	8	0x10
CONF_DEBUG_SEL_TST_17	R/W	8	0x11
CONF_DEBUG_SEL_TST_18	R/W	8	0x12
CONF_DEBUG_SEL_TST_19	R/W	8	0x13
CONF_DEBUG_SEL_TST_20	R/W	8	0x14
CONF_DEBUG_SEL_TST_21	R/W	8	0x15
CONF_DEBUG_SEL_TST_22	R/W	8	0x16
CONF_DEBUG_SEL_TST_23	R/W	8	0x17
CONF_DEBUG_SEL_TST_24	R/W	8	0x18
CONF_DEBUG_SEL_TST_25	R/W	8	0x19
CONF_DEBUG_SEL_TST_26	R/W	8	0x1A
CONF_DEBUG_SEL_TST_27	R/W	8	0x1B
CONF_DEBUG_SEL_TST_28	R/W	8	0x1C
CONF_DEBUG_SEL_TST_29	R/W	8	0x1D
CONF_DEBUG_SEL_TST_30	R/W	8	0x1E
CONF_DEBUG_SEL_TST_31	R/W	8	0x1F
CONF_DEBUG_SEL_TST_32	R/W	8	0x20
CONF_DEBUG_SEL_TST_33	R/W	8	0x21
CONF_DEBUG_SEL_TST_34	R/W	8	0x22
CONF_DEBUG_SEL_TST_35	R/W	8	0x23
CONF_DEBUG_SEL_TST_36	R/W	8	0x24

18.4.4.4 Registers Description

Table 18-196 through Table 18-232 describe the individual multiplexer layer debug module registers.

Table 18-196. CONF_DEBUG_SEL_TST_0

Address Offset	0x00		
Physical Address	0xFFFE 9200	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MODE	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_0 port: 000: Dsp_nstrobe : DSP strobe signal 001: DMA_REQUEST_P0_0: Decimal encoding of DMA channel read/write 010: DMA_REQUEST_P1_0: Decimal encoding of DMA channel read/write 011: MAS_0 : Memory access size 100: XDI_O_0 : DSP XIO data line 0 101: DMA_REQUEST_Px_0: Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_0)	R/W	0x0

Bits	Field Name	Description	Type	Reset
		110: MUX_VIEW_0: Moved from reduction layer (see CONF_MUX_VIEW0)		

Table 18-197. CONF_DEBUG_SEL_TST_1

Address Offset	0x01		
Physical Address	0xFFFE 9201	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_1 port: 000: Dsp_rnw : DSP read signal 001: DMA_REQUEST_P0_1: Decimal encoding of DMA channel read/write 010: DMA_REQUEST_P1_1: Decimal encoding of DMA channel read/write 011: MAS_1 : Memory access size 100: XDI_O_1 : DSP XIO data line 1 101: DMA_REQUEST_Px_1: Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_1) 110: MUX_VIEW_1: Moved from reduction layer (see CONF_MUX_VIEW1)	R/W	0x0

Table 18-198. CONF_DEBUG_SEL_TST_2

Address Offset	0x02		
Physical Address	0xFFFE 9202	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_2 port: 000: Dsp_nready : DSP ready signal 001: DMA_REQUEST_P0_2: Decimal encoding of DMA channel read/write 010: DMA_REQUEST_P1_2: Decimal encoding of DMA channel read/write 011: nOPC : Opcode fetch 100: XDI_O_2 : DSP XIO data line 2 101: DMA_REQUEST_Px_2: Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_2) 110: MUX_VIEW_2: Moved from reduction layer (see CONF_MUX_VIEW2)	R/W	0x0

Table 18-199. CONF_DEBUG_SEL_TST_3

Address Offset	0x03		
Physical Address	0xFFFE 9203	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_3 port: 000: dsp_cs_0:DSP chip-select for different modules in wrapper 001: DMA_REQUEST_P0_3: Decimal encoding of DMA channel read/write 010: DMA_REQUEST_P1_3: Decimal encoding of DMA channel read/write 011: nWAIT: Wait between DMA 100: XDI_O_3 : DSP XIO data line 4 101: DMA_REQUEST_Px_3: Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_3) 110: MUX_VIEW_3: Moved from reduction layer (see CONF_MUX_VIEW3)	R/W	0x0

Table 18-200. CONF_DEBUG_SEL_TST_4

Address Offset	0x04		
Physical Address	0xFFFE 9204	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_4 port: 000: dsp_cs_1 : DSP chip-select for different modules in wrapper 001: DMA_REQ_S0: DMA request stall 010: DMA_REQ_S1: DMA request stall 011: mcu_cs_0 : MCU chip-select for different modules in wrapper 100: XDI_O_4 : DSP XIO data line 4 101: DMA_REQ_Sx: DMA request stall	R/W	0x0

Table 18-201. CONF_DEBUG_SEL_TST_5

Address Offset	0x05		
Physical Address	0xFFFE 9205	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_5 port: 000: dsp_cs_2 : DSP chip-select for different modules in wrapper 001: DMA_REQ_V0: DMA request valid 010: DMA_REQ_V1: DMA request valid 011: mcu_cs_1 : MCU chip-select for different modules in wrapper 100: XDI_O_5 : DSP XIO data line 5 101: DMA_REQ_Vx: DMA request valid multiplexed (see CONF_DMA_REQ_V) 110: MUX_VIEW_5: Moved from reduction layer (see CONF_MUX_VIEW5)	R/W	0x0

Table 18-202. CONF_DEBUG_SEL_TST_6

Address Offset	0x06		
Physical Address	0xFFFE 9206	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_6 port: 000: dsp_cs_3 : DSP chip-select for different modules in wrapper 001: DMA_REQUEST_P3_0: Decimal encoding of DMA channel read/write 010: DMA_REQUEST_P4_0: Decimal encoding of DMA channel read/write 011: mcu_cs_2 : MCU chip-select for different modules in wrapper 100: XDI_O_6 : DSP XIO data line 6 101: DMA_REQUEST_Px_0: Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_0) 110: MUX_VIEW_6: Moved from reduction layer (see CONF_MUX_VIEW6)	R/W	0x0

Table 18-203. CONF_DEBUG_SEL_TST_7

Address Offset	0x07		
Physical Address	0xFFFE 9207	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_7 port: 000: Reserved 001: DMA_REQUEST_P3_1: Decimal encoding of DMA channel read/write 010: DMA_REQUEST_P4_1: Decimal encoding of DMA channel read/write 011: mcu_cs_3 : MCU chip-select for different modules in wrapper 100: XDI_O_7 : DSP XIO data line 7 101: DMA_REQUEST_Px_1: Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_1) 110: MUX_VIEW_7: Moved from reduction layer (see CONF_MUX_VIEW7)	R/W	0x0

Table 18-204. CONF_DEBUG_SEL_TST_8

Address Offset	0x08		
Physical Address	0xFFFE 9208	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_8 port: 000: DSP_nINT_VIEW_0: DSP interrupt source multiplexed (see CONF_DSP_VIEW_0) 001: DMA_REQUEST_P3_2: Decimal encoding of DMA channel read/write 010: DMA_REQUEST_P4_2: Decimal encoding of DMA channel read/write 011: mcu_cs_4 : MCU chip-select for different modules in wrapper 100: X_IOSTRBN : DSP XIO strobe 101: DMA_REQUEST_Px_2: Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_2) 110: ARM_nIRQ_VIEW_0: Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_0) 111: MUX_VIEW_8: Moved from reduction layer (see CONF_MUX_VIEW8)	R/W	0x0

Table 18-205. CONF_DEBUG_SEL_TST_9

Address Offset	0x09		
Physical Address	0xFFFE 9209	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_9 port: 000: DSP_nINT_VIEW_1: DSP interrupt source multiplexed (see CONF_DSP_VIEW_1) 001: DMA_REQUEST_P3_3: Decimal encoding of DMA channel read/write 010: DMA_REQUEST_P4_3: Decimal encoding of DMA channel read/write 011: mcu_nready : MCU ready signal 100: XIO_nREADY_MEM : Ready signal from peripheral on XIO space to DSP 101: DMA_REQUEST_Px_3: Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_3) 110: ARM_nIRQ_VIEW_1: Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_1) 111: MUX_VIEW_9: Moved from reduction layer (see CONF_MUX_VIEW9)	R/W	0x0

Table 18-206. CONF_DEBUG_SEL_TST_10

Address Offset	0x0A		
Physical Address	0xFFFE 920A	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_10 port: 000: Drp_dbb_rxdma_req: DRP data Rx DMA request 001: DMA_REQ_S3: DMA request stall 010: DMA_REQ_S4: DMA request stall 011: mcu_nstrobe : MCU strobe 100: Tout : DSP timer out 101: DMA_REQ_Sx: DMA request stall multiplexed (see CONF_DMA_REQ_S) 110: nDMA_REQ_VIEW_0: DMA interrupt request multiplexed (see CONF_DMA_VIEW_0) 111: MUX_VIEW_10: Moved from reduction layer (see CONF_MUX_VIEW10)	R/W	0x0

Table 18-207. CONF_DEBUG_SEL_TST_11

Address Offset	0x0B		
Physical Address	0xFFFE 920B	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_11 port: 000: INT10n: DMA interrupt 001: DMA_REQ_V3: DMA request valid 010: DMA_REQ_V4: DMA request valid 011: mcu_rnw : MCU read signal 100: XF : DSP external flag 101: DMA_REQ_Vx: DMA request valid multiplexed (see CONF_DMA_REQ_V) 110: nDMA_REQ_VIEW_1: DMA interrupt request multiplexed (see CONF_DMA_VIEW_1) 111: MUX_VIEW_11: Moved from reduction layer (see CONF_MUX_VIEW11)	R/W	0x0

Table 18-208. CONF_DEBUG_SEL_TST_12

Address Offset	0x0C		
Physical Address	0xFFFE 920C	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_12 port: 000: IRQ4: TPU frame interrupt 001: Drp_dbb_rxdma_req: DRP data Rx DMA request 010: IACKn: Interrupt acknowledge 011: nMREQ: ARM not memory request 100: Bridge_en : Bridge Enable 101: ARM_nIRQ_VIEW_1: Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_1) 110: MUX_VIEW_12: Moved from reduction layer (see CONF_MUX_VIEW12)	R/W	0x0

Table 18-209. CONF_DEBUG_SEL_TST_13

Address Offset	0x0D		
Physical Address	0xFFFE 920D	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_13 port: 000: DMA_REQUEST_P0_0: Decimal encoding of DMA channel read/write 001: Dsp_nstrobe : DSP strobe signal 010: DSP_nINT_VIEW_0: DSP interrupt source multiplexed (see CONF_DSP_VIEW_0) 011: nDMA_REQ_VIEW_0: DMA interrupt request multiplexed (see CONF_DMA_VIEW_0) 100: Dbb_drp_mcmd1_0 : 000: idle 101: X_A_0 : DSP XIO Address Line 0 110: DMA_REQUEST_Px_0: Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_0) 111: MUX_VIEW_13: Moved from reduction layer (see CONF_MUX_VIEW13)	R/W	0x0

Table 18-210. CONF_DEBUG_SEL_TST_14

Address Offset	0x0E		
Physical Address	0xFFFE 920E	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_14 port: 000: DMA_REQUEST_P0_1: Decimal encoding of DMA channel read/write 001: Dsp_rnw : DSP read signal 010: DSP_nINT_VIEW_1: DSP interrupt source multiplexed (see CONF_DSP_VIEW_1) 011: nDMA_REQ_VIEW_1: DMA interrupt request multiplexed (see CONF_DMA_VIEW_1) 100: Dbb_drp_mcmd1_1 : DBB to DRP - OCP1 101: X_A_1 : DSP XIO address line 1 110: DMA_REQUEST_Px_1: Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_1) 111: MUX_VIEW_14: Moved from reduction layer (see CONF_MUX_VIEW14)	R/W	0x0

Table 18-211. CONF_DEBUG_SEL_TST_15

Address Offset	0x0F		
Physical Address	0xFFFE 920F	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_15 port: 000: DMA_REQUEST_P0_2: Decimal encoding of DMA channel read/write 001: Dsp_nready : DSP ready signal 010: Drp_dbb_rxdma_req: DRP data Rx DMA request 011: INT10n: DMA interrupt 100: Dbb_drp_mcmd1_2 : DBB to DRP - OCP1 101: X_A_2 : DSP XIO address line 2 110: DMA_REQUEST_Px_2: Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_2) 111: MUX_VIEW_15: Moved from reduction layer (see CONF_MUX_VIEW15)	R/W	0x0

Table 18-212. CONF_DEBUG_SEL_TST_16

Address Offset	0x10		
Physical Address	0xFFFE 9210	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_16 port: 000: DMA_REQUEST_P0_3: Decimal encoding of DMA channel read/write 001: Dsp_nready : DSP ready signal 010: INT10n: DMA interrupt 011: IRQ14: DMA interrupt 100: Dbb_drp_mcmd1_2 : DBB to DRP - OCP1 101: X_A_2 : DSP XIO address line 2 110: DMA_REQUEST_Px_3: Decimal encoding of DMA channel read/write multiplexed (see CONF_DMA_REQ_3) 111: MUX_VIEW_16: Moved from reduction layer (see CONF_MUX_VIEW16)	R/W	0x0

Table 18-213. CONF_DEBUG_SEL_TST_17

Address Offset	0x11		
Physical Address	0xFFFE 9211	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_17 port: 000: nDMA_REQ_VIEW_0: DMA interrupt request multiplexed (see CONF_DMA_VIEW_0) 001: dsp_cs_1 : DSP chip-select for different modules in wrapper 010: IRQ4: TPU frame interrupt 011: ARM_nIRQ_VIEW_0: Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_0) 100: Dbb_drp_mcmd2_1 : DBB to DRP - OCP2 101: X_A_4 : DSP XIO address line 4 110: DMA_REQ_Sx: DMA request stall multiplexed (see CONF_DMA_REQ_S) 111: MUX_VIEW_17: Moved from reduction layer (see CONF_MUX_VIEW17)	R/W	0x0

Table 18-214. CONF_DEBUG_SEL_TST_18

Address Offset	0x12		
Physical Address	0xFFFE 9212	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_18 port: 000: nDMA_REQ_VIEW_1: DMA interrupt request multiplexed (see CONF_DMA_VIEW_1) 001: dsp_cs_2 : DSP chip-select for different modules in wrapper 010: IRQ14: DMA interrupt 011: ARM_nIRQ_VIEW_1: Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_1) 100: Dbb_drp_mcmd2_2 : DBB to DRP - OCP2 101: X_A_5 : DSP XIO address line 5 110: DMA_REQ_Vx: DMA request valid multiplexed (see CONF_DMA_REQ_V) 111: MUX_VIEW_18: Moved from reduction layer (see CONF_MUX_VIEW18)	R/W	0x0

Table 18-215. CONF_DEBUG_SEL_TST_19

Address Offset	0x13		
Physical Address	0xFFFE 9213	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MODE	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_19 port: 000: DMA_REQUEST_P3_0: Decimal encoding of DMA channel read/write 001: dsp_cs_3 : DSP chip-select for different modules in wrapper 010: INT4n: DRP DBB Tx interrupt 011: DSP_nINT_VIEW_0: DSP interrupt source multiplexed (see CONF_DSP_VIEW_0) 100: Drp_dbb_rx_req: DRP Rx interrupt 101: X_A_6 : DSP XIO address line 6 110: nDMA_REQ_VIEW_0: DMA interrupt request multiplexed (see CONF_DMA_VIEW_0) 111: MUX_VIEW_19: Moved from reduction layer (see CONF_MUX_VIEW19)	R/W	0x0

Table 18-216. CONF_DEBUG_SEL_TST_20

Address Offset	0x14		
Physical Address	0xFFFE 9214	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED						MODE	

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_20 port: 000: DMA_REQUEST_P3_1: Decimal encoding of DMA channel read/write 001: dsp_cs_4 : DSP chip-select for different modules in wrapper 010: nFIQ: ARM fast interrupt 011: DSP_nINT_VIEW_1: DSP interrupt source multiplexed (see CONF_DSP_VIEW_1) 100: Drp_dbb_scmdaccpt1: Read/write command accepted (DRP to DBB - OCP1) 101: NMIIT: DSP nonmaskable interrupt 110: nDMA_REQ_VIEW_1: DMA interrupt request multiplexed (see CONF_DMA_VIEW_1) 111: MUX_VIEW_20: Moved from reduction layer (see CONF_MUX_VIEW20)	R/W	0x0

Table 18-217. CONF_DEBUG_SEL_TST_21

Address Offset	0x15		
Physical Address	0xFFFE 9215	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_21 port: 000: DMA_REQUEST_P3_2: Decimal encoding of DMA channel read/write 001: nDMA_REQ_VIEW_0: DMA interrupt request multiplexed (see CONF_DMA_VIEW_0) 010: nIRQ: ARM slow interrupt 011: LCD_FIFO_Empty : LCD FIFO Empty 100: Drp_dbb_scmdaccpt2 : Read/write command accepted (DRP to DBB - OCP2) 101: INT3n: MCSI Tx interrupt 110: nDMA_REQ_VIEW_2: DMA interrupt request multiplexed (see CONF_DMA_VIEW_2) 111: MUX_VIEW_21: Moved from reduction layer (see CONF_MUX_VIEW21)	R/W	0x0

Table 18-218. CONF_DEBUG_SEL_TST_22

Address Offset	0x16		
Physical Address	0xFFFE 9216	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_22 port: 000: DMA_REQUEST_P3_3: Decimal encoding of DMA channel read/write 001: nDMA_REQ_VIEW_1: DMA interrupt request multiplexed (see CONF_DMA_VIEW_1) 010: ARM_nIRQ_VIEW_0: Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_0) 011: LCD_FIFO_Full : LCD FIFO full 100: ARM_nIRQ_VIEW_2: Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_2) 101: IACKn: Interrupt acknowledge 110: nDMA_REQ_VIEW_3: DMA interrupt request multiplexed (see CONF_DMA_VIEW_3) 111: MUX_VIEW_22: Moved from reduction layer (see CONF_MUX_VIEW22)	R/W	0x0

Table 18-219. CONF_DEBUG_SEL_TST_23

Address Offset	0x17		
Physical Address	0xFFFE 9217	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_23 port: 000: Dcxo_clrz: DRP reset signal 001: IRQ14: DMA interrupt 010: IACKn: Interrupt acknowledge 011: ND_Flash_CLK_REQ: Clock request for data transfer 100: Drp_dbb_sresp1_0 : 00: idle 101: DSP_nINT_VIEW_0: DSP interrupt source multiplexed (see CONF_DSP_VIEW_0) 110: MUX_VIEW_23: Moved from reduction layer (see CONF_MUX_VIEW23)	R/W	0x0

Table 18-220. CONF_DEBUG_SEL_TST_24

Address Offset	0x18		
Physical Address	0xFFFE 9218	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_24 port: 000: Reserved 001: INT4n: DRP DBB Tx interrupt 010: dsp_cs_0 : DSP chip-select for different modules in wrapper 011: ND_Flash_State_0 : Indicates state of the NAND Flash state-machine 100: Drp_dbb_sresp1_1 : DRP to DBB OCP1 101: DSP_nINT_VIEW_1: DSP interrupt source multiplexed (see CONF_DSP_VIEW_1) 110: MUX_VIEW_24: Moved from reduction layer (see CONF_MUX_VIEW24)	R/W	0x0

Table 18-221. CONF_DEBUG_SEL_TST_25

Address Offset	0x19		
Physical Address	0xFFFE 9219	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_25 port: 000: Dcxo_sysclken: Clock enable for dcxo 001: Bridge_en : Bridge enable 010: dsp_cs_1 : DSP chip-select for different modules in wrapper 011: ND_Flash_State_1 : Indicates state of the NAND Flash state-machine 100: Drp_dbb_sresp2_0 : 00: idle 101: Drp_dbb_tx_req: DRP Tx interrupt 110: ARM_nIRQ_VIEW_1: Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_1) 111: MUX_VIEW_25: Moved from reduction layer (see CONF_MUX_VIEW25)	R/W	0x0

Table 18-222. CONF_DEBUG_SEL_TST_26

Address Offset	0x1A		
Physical Address	0xFFFE 921A	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_26 port: 000: Drp_dbb_sysclk: DRP retimed clock 001: IACKn: Interrupt acknowledge 010: dsp_cs_2 : DSP chip-select for different modules in wrapper 011: ND_Flash_State_2 : Indicates state of the NAND Flash state-machine 100: Drp_dbb_sresp2_1 : DRP to DBB OCP2 101: Drp_dbb_sinterrupt : DRP general-purpose interrupt 110: ARM_nIRQ_VIEW_0: Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_0) 111: MUX_VIEW_26: Moved from reduction layer (see CONF_MUX_VIEW26)	R/W	0x0

Table 18-223. CONF_DEBUG_SEL_TST_27

Address Offset	0x1B		
Physical Address	0xFFFE 921B	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_27 port: 000: Clk13_en: Clock enable 001: DSP_nINT_VIEW_2: DSP interrupt source multiplexed (see CONF_DSP_VIEW_2) 010: dsp_cs_3 : DSP chip-select for different modules in wrapper 011: ND_Flash_State_3 : Indicates state of the NAND Flash state-machine 100: IMIF_CCS_0 : chip-select for boot ROM 101: Drp_dbb_rx_req: DRP Rx interrupt 110: Reserved 111: MUX_VIEW_27: Moved from reduction layer (see CONF_MUX_VIEW27)	R/W	0x0

Table 18-224. CONF_DEBUG_SEL_TST_28

Address Offset	0x1C		
Physical Address	0xFFFE 921C	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_28 port: 000: Reserved 001: ARM_nIRQ_VIEW_2: Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_2) 010: dsp_cs_4 : DSP chip-select for different modules in wrapper 011: ND_Flash_State_4 : Indicates state of the NAND Flash state-machine 100: IMIF_CCS_1 : chip-select for boot ROM 101: DSP_nINT_VIEW_2: DSP interrupt source multiplexed (see CONF_DSP_VIEW_2) 110: IACKn: Interrupt acknowledge 111: MUX_VIEW_28: Moved from reduction layer (see CONF_MUX_VIEW28)	R/W	0x0

Table 18-225. CONF_DEBUG_SEL_TST_29

Address Offset	0x1D		
Physical Address	0xFFFE 921D	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_29 port: 000: TPU_Wait : TPU wait 001: Dsp_nstrobe : DSP strobe signal 010: FIFO_FULL : Camera FIFO FULL 011: DMA_CLK_REQ: Signal from DMA module when going from idle to active mode 100: IMIF_CCS_2 : chip-select for boot ROM 101: MUX_VIEW_29: Moved from reduction layer (see CONF_MUX_VIEW29)	R/W	0x0

Table 18-226. CONF_DEBUG_SEL_TST_30

Address Offset	0x1E		
Physical Address	0xFFFE 921E	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_30 port: 000: TPU_Idle : TPU idle 001: Dsp_rnw : DSP read signal 010: FIFO_NotEmpty : Camera FIFO not empty 011: DMA_RHEA_nStrobe : Strobe signal from DMA to Rhea bus 100: IMIF_CCS_3 : Chip-select for boot ROM 101: DSP_nINT_VIEW_0: DSP interrupt source multiplexed (see CONF_DSP_VIEW_0) 110: MUX_VIEW_30: Moved from reduction layer (see CONF_MUX_VIEW30)	R/W	0x0

Table 18-227. CONF_DEBUG_SEL_TST_31

Address Offset	0x1F		
Physical Address	0xFFFE 921F	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_31 port: 000: ARM_nIRQ_VIEW_1 : Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_1) 001: Dsp_nready : DSP ready signal 010: FIFO_OF : Camera FIFO overflow 011: DMA_RHEA_nReady : Ready signal from Rhea to DMA 100: IMIF_CS6_0 : chip-select for MCU fast internal memory 101: Dsp_nstrobe : DSP strobe signal 110: MUX_VIEW_31: Moved from reduction layer (see CONF_MUX_VIEW31)	R/W	0x0

Table 18-228. CONF_DEBUG_SEL_TST_32

Address Offset	0x20		
Physical Address	0xFFFE 9220	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_32 port: 000: ARM_nIRQ_VIEW_0: Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_0) 001: APC_Start_Internal : Internal APC start signal (not TSPACT) 010: FIFO_UF : Camera FIFO underflow 011: Arbitrer_nWait : Arbitrated wait between DMA 100: Dsp_nready : DSP ready signal 101: MUX_VIEW_32: Moved from reduction layer (see CONF_MUX_VIEW32)	R/W	0x0

Table 18-229. CONF_DEBUG_SEL_TST_33

Address Offset	0x21		
Physical Address	0xFFFE 9221	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_33 port: 000: DMA_REQ_V0: DMA request valid 001: DSP_nINT_VIEW_0: DSP interrupt source multiplexed (see CONF_DSP_VIEW_0) 010: PU_Wait : TPU wait 011: state_machine_0 : Camera state machine 100: ARM_nIRQ_VIEW_1: Internal interrupt to ARM multiplexed (see CONF_ARM_VIEW_1) 101: dsp_cs_1 : DSP chip-select for different modules in wrapper 110: BRIDGE_CLK : Bridge clock 111: MUX_VIEW_33: Moved from reduction layer (see CONF_MUX_VIEW33)	R/W	0x0

Table 18-230. CONF_DEBUG_SEL_TST_34

Address Offset	0x22		
Physical Address	0xFFFE 9222	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_34 port: 000: DMA_REQ_S0: DMA request stall 001: DSP_nINT_VIEW_1: DSP interrupt source multiplexed (see CONF_DSP_VIEW_1) 010: TPU_Idle : TPU idle 011: state_machine_1 : Camera state machine 100: nFIQ : ARM fast interrupts 101: dsp_cs_2 : DSP chip-select for different modules in wrapper 110: DPLL_CLKOUT: DPLL clock out 111: MUX_VIEW_34: Moved from reduction layer (see CONF_MUX_VIEW34)	R/W	0x0

Table 18-231. CONF_DEBUG_SEL_TST_35

Address Offset	0x23		
Physical Address	0xFFFE 9223	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_35 port: 000: DMA_REQ_V3: DMA request valid 001: INT10n: DMA interrupt 010: nDMA_REQ_VIEW_0: DMA interrupt request multiplexed (see CONF_DMA_VIEW_0) 011: state_machine_2 : Camera state machine 100: nIRQ: ARM slow interrupt 101: dsp_cs_3 : DSP chip-select for different modules in wrapper 110: DSP_CLKOUT: DSP clock out 111: MUX_VIEW_35: Moved from reduction layer (see CONF_MUX_VIEW35)	R/W	0x0

Table 18-232. CONF_DEBUG_SEL_TST_36

Address Offset	0x24		
Physical Address	0xFFFE 9224	**Instance**	Debug
Description	Multiplexer layer debug module register		
Type	R/W		
Write Latency	Not relevant		

7	6	5	4	3	2	1	0
RESERVED					MODE		

Bits	Field Name	Description	Type	Reset
7:3	RESERVED		R	0x0000
2:0	MODE	Select mode for test_port_36 port: 000: DMA_REQ_S3: DMA request stall 001: IRQ4: TPU frame interrupt 010: nDMA_REQ_VIEW_1: DMA interrupt request multiplexed (see CONF_DMA_VIEW_1) 100: Drp_dbb_s interrupt: DRP general-purpose interrupt 101: dsp_cs_4 : DSP chip-select for different modules in wrapper 110: MCLK: ARM clock in debug signal 111: MUX_VIEW_36: Moved from reduction layer (see CONF_MUX_VIEW36)	R/W	0x0

UART/IrDA

This chapter describes the function, operation, and configuration of the universal asynchronous receiver/transmitter (UART)/infrared data association (IrDA) modules on the LOCOSTO device.

Topic **Page**

19.1 UART Overview

The LOCOSTO device contains a universal asynchronous receiver/transmitter (UART) module shared by the microprocessor unit (MPU) and the digital signal processor (DSP) using a TIPB/OCP static switch. Two modes, UART and IrDA, are selectable through a set of configuration registers. The UART module connects an external modem device through a standard wired interface or an infrared data association (IrDA) module. The two functions can be supported dynamically with additional general-purpose inputs/outputs (GPIOs) and software control.

19.1.1 Main Features

The following features are common to both modes (UART and IrDA):

- Selectable UART/IrDA modes
- Dual 64-entry first-ins, first-outs (FIFOs) for received and transmitted data payload
- Programmable and selectable transmit and receive FIFO trigger levels for direct memory access (DMA) and interrupt generation
- Programmable sleep mode
- Complete status reporting capabilities in normal and sleep modes
- Frequency prescaler values from 0 to 16,383 to generate the appropriate baud rates
- One clock reference of 52-, 48-, or 13-MHz for baud setting
- Two DMA requests and one interrupt request to the MPU, and one to the DSP
- Idle and wake-up power management

19.1.1.1 UART Features

The key features of the UART in modem mode are as follows:

- 16C750 compatibility
- Baud rate from 300 bits/s to 3.6864M bits/s
- Auto baud between 1200 bits/s and 115.2K bits/s
- Configurable data format:
  - Data bit: 5, 6, 7, or 8 bits
  - Parity bit: even, odd, none
  - Stop-bit: 1, 1.5, 2 bit(s)
- Hardware flow control RTS/CTS
- Software flow control using XON/XOFF characters
- Line break detection and generation
- False-start bit detection
- Fully prioritized interrupt system controls

19.1.1.2 UART/IrDA Features

The key features of the UART in IrDA mode are as follows:

- Slow infrared (SIR 115.2K baud), medium infrared (MIR 0.576M baud), and fast infrared (FIR 4.0M baud) operations
- Frame formatting: addition of variable beginning-of-frame (BOF) characters and end-of-frame (EOF) characters
- Asynchronous transparency (automatic insertion of break character)
- Uplink/downlink CRC generation/detection
- Framing error, cyclic redundancy check (CRC) error, illegal symbol (FIR), and abort pattern (SIR, MIR) detection
- 8-entry status FIFO (with selectable trigger levels) available to monitor frame-length and frame errors

19.1.2 Signals and I/O Descriptions

Figure 19-1 and Table 19-1 describe the UART in terms of I/Os and the interaction with other modules in the LOCOSTO device.

Figure 19-1. UART/IrDA Overview

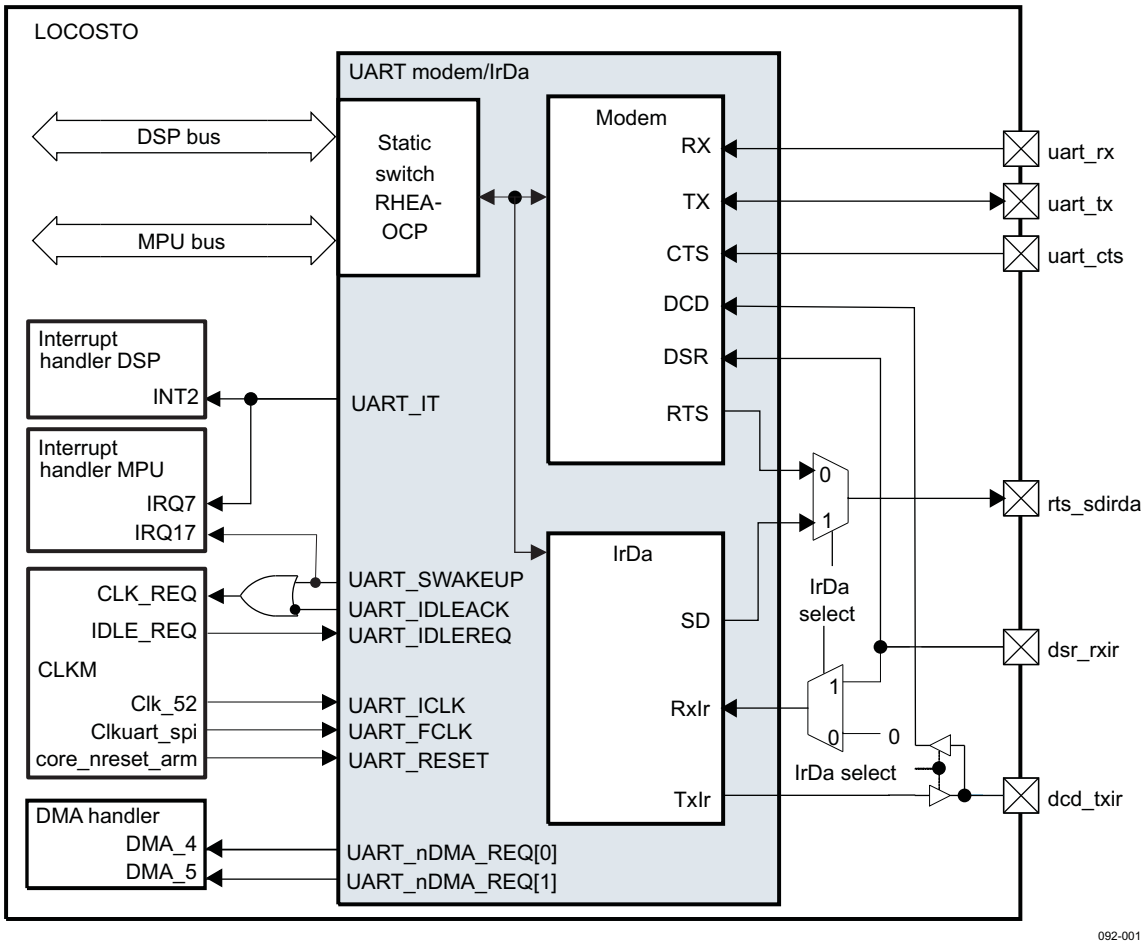


Table 19-1. I/O Description

Signal Name	Input/Output	Description	Reset Value
uart_rx	I	Serial data input	Unknown
uart_tx	O	Serial data output	1
uart_cts	I	Clear-to-send (CTS): Active-low modem status signal. Reading bit 4 of the modem status register (MSR) checks the condition of uart_cts. Reading bit 0 of the MSR indicates that uart_cts input has changed state. uart_cts is used in auto-CTS mode to control the transmitter.	Unknown

Table 19-1. I/O Description (continued)

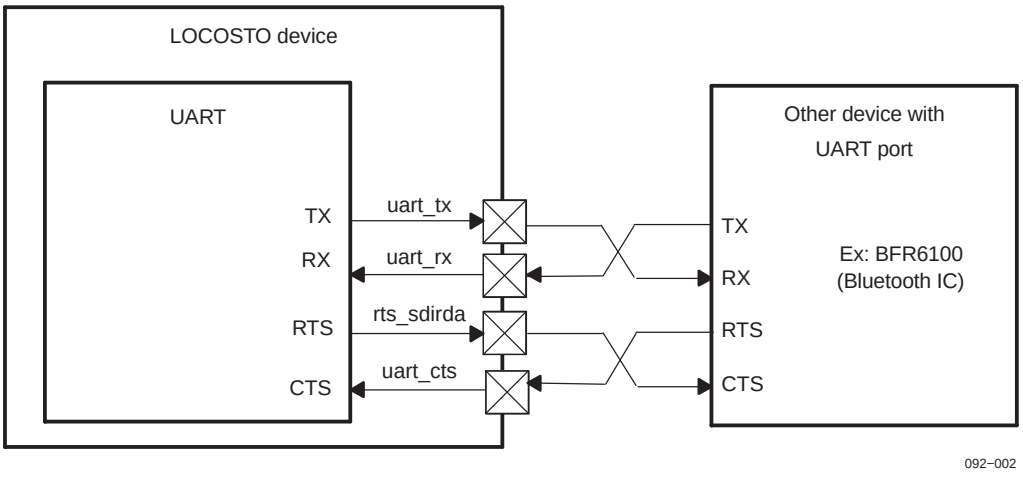
Signal Name	Input/Output	Description	Reset Value
rts_sdirda	O	In UART mode: Request-to-send (RTS). When active (low), RTS informs the modem or data set that the UART is ready to receive data. RTS is set to the active level by setting the RTS modem control register bit and is set to the inactive (high) level as a result of a master reset, during loop mode operations, or by clearing bit MCR[1]. In auto-RTS mode, RTS is set to the inactive level by the receiver threshold control logic. In IrDA mode: SD mode is used to configure the transceivers. Infrared shutdown is used to shut down an external IR transceiver using software. SD mode can also be used to configure the IR transceiver for SIR/MIR/FIR modes in conjunction with the TX input pin of the transceiver device. The SD pinout is an inverted value of ACREG[6].	1
dsr_rxir	I	In UART mode: Data set ready is an active-low modem status signal. Its condition can be checked by reading bit 5 of the modem status register. Bit 1 of the modem status register indicates that DSR has changed levels since the last read from the modem status register. In IrDA mode: Serial data input	Unknown
dcd_txir	I/O	In UART mode: Data carrier detect is an active-low modem status signal. Its condition can be checked by reading bit 7 of the modem status register. Bit 3 of the modem status register indicates that DCD has changed levels since the last read from the modem status register. In IrDA mode: Serial data output	Unknown

19.1.3 UART IrDA Environment

19.1.3.1 System Using UART Communication with Hardware Handshake

The UART can be easily connected to the UART port of an external IC, as shown in Figure 19-2.

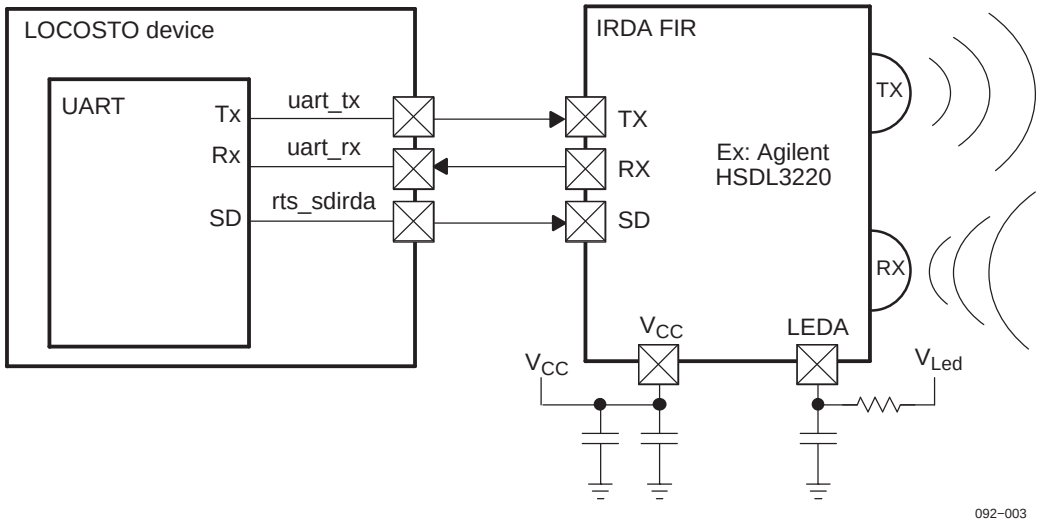
Figure 19-2. UART Connection with Hardware Handshake



19.1.3.2 System Using IrDA Communication

Figure 19-3 shows an IrDA connection with an external device for transmitting and receiving.

Figure 19-3. IrDA Connection



Note: When UART mode is selected (rts_sdirda as an UART signal), a standard GPIO can be used to drive the SD signal.

19.1.4 UART Protocol and Data Format

19.1.4.1 UART Mode

The UART device can operate in three modes:

- UART 16x mode (≤ 230.4K bits/s)
- UART 16x mode with autobauding (≥ 1200 bits/s and ≤ 115.2K bits/s)
- UART 13x mode (≥ 460.8K bits/s)

CAUTION

To be used as a UART, the operating mode must be programmed appropriately in the mode definition register 1 (MDR1) to select one of the three modes. See Section 19.2.2, *Mode Selection*.

The value of MDR1 that selects the mode must not be changed during normal operation.

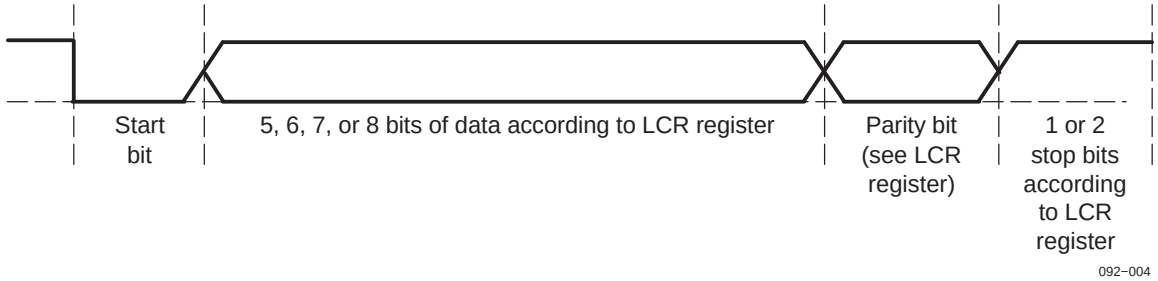
The UART uses a wired interface for serial communication with a remote device. The UART is functionally compatible with the TL16C750 UART, and is also functionally compatible with earlier designs such as the TL16C550.

Three parameters must be set up for a correct protocol between two devices (see Section 19.2.2.2, *UART Mode*, for more information):

- Character length
- Number of stop-bits
- Parity bit

Figure 19-4 shows the setups and the total size of a message.

Figure 19-4. UART Frame Data Format



19.1.4.2 IrDA Mode

The UART device in IrDA mode operates in three modes:

- Slow infrared (SIR ≤ 115.2K baud)
- Medium infrared (MIR 0.576M baud)
- Fast infrared (FIR 4.0M baud)

CAUTION

To be used as an IrDA, the operating mode must be programmed appropriately in the MDR1 to select one of the three modes. See Section 19.2.2, *Mode Selection*.

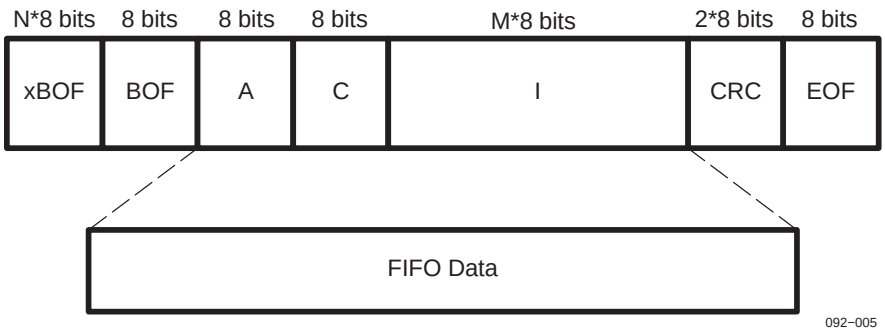
The value of MDR1 that selects the mode must not be changed during normal operation.

19.1.4.2.1 SIR Mode Data Formatting

This section gives specific instructions for SIR mode programming.

In SIR mode, data transfer occurs between the local host and peripheral devices at speeds of up to 115,200 baud. A SIR transmit frame begins with start flags (a single 0xC0, multiple 0xC0, or a single 0xC0 preceded by a number of 0xFF flags), followed by frame data (CRC-16), and ends with a stop flag (0xC1), as shown in Figure 19-5.

Figure 19-5. IrDA SIR Frame Format



When multiple start flags are required, BLR[6] is used to select whether a 0xC0 or 0xFF start pattern is used.

The SIR TX state-machine attaches start flags, CRC-16, and stop flags. It checks outgoing data to establish whether data transparency is required.

SIR transparency is carried out as explained in Section 19.1.4.2.1.1, *Asynchronous Transparency*.

The SIR receive state-machine recovers the receive clock, removes the start flags, removes any transparency from the incoming data, and determines the frame boundary with reception of the stop flag. It also checks for errors such as frame abort (0x7D character followed immediately by a 0xC1 stop flag, without transparency), CRC error, and frame length. At the EOF reception, the local host reads the line status register (LSR) to determine any possible errors in the received frame.

Note: Data can be transferred both ways by the module, but when the device is transmitting, the IR Rx circuitry is automatically disabled by hardware. See the auxiliary control register (see Table 19-64) bit 5 for a description of the logical operation. This applies to all three modes: SIR, MIR, and FIR.

The infrared output in SIR mode can be either 1.6 µs or 3/16th encoding, selected by the PULSE_TYPE bit of the auxiliary control register (ACREG[7]). In 1.6-µs encoding, the infrared pulse width is 1.6 µs; and in 3/16th encoding, the infrared pulse width is 3/16th of a bit duration (1/baud-rate).

The transmitting device must send at least two start flags at the start of each frame for back-to-back frames.

Note: Reception supports variable-length stop bits.

The CRC is applied on the address (A), control (C), and information (I) bytes.

19.1.4.2.1.1 Asynchronous Transparency

Before transmitting a byte, the UART IrDA controller examines each byte of the payload and the CRC field (between BOF and EOF). The following occurs for each byte equal to 0xC0 (BOF), 0xC1 (EOF), or 0x7D (control escape).

19.1.4.2.1.2 In Transmission

- Inserts a control escape (CE) byte preceding the byte.
- Complements bit 5 of the byte (XORs the byte with 0x20).

The byte sent for CRC computation is the initial byte written in the TX FIFO (before the XOR with 0x20).

19.1.4.2.1.3 In Reception

For the A, C, I, and CRC field:

- Compares the byte to the CE byte; if they are not equal, the UART/IrDA controller sends the byte to the CRC detector, and stores it in the RX FIFO.
- If the byte equals the CE byte, the UART/IrDA controller discards the CE byte.
- Complements bit 5 of the byte following the CE byte.
- Sends the complemented byte to the CRC detector and stores it in the RX FIFO.

19.1.4.2.1.4 Abort Sequence

The transmitter can prematurely close a frame. The transmitter aborts by sending the following sequence: 0x7DC1. The abort pattern closes the frame without a CRC field or an ending flag.

The receiver treats a frame as an aborted frame when a 0x7D character, followed immediately by a 0xC1 character, is received without transparency.

19.1.4.2.1.5 Pulse Shaping

SIR mode supports both the 3/16th and the 1.6 µs pulse duration methods.

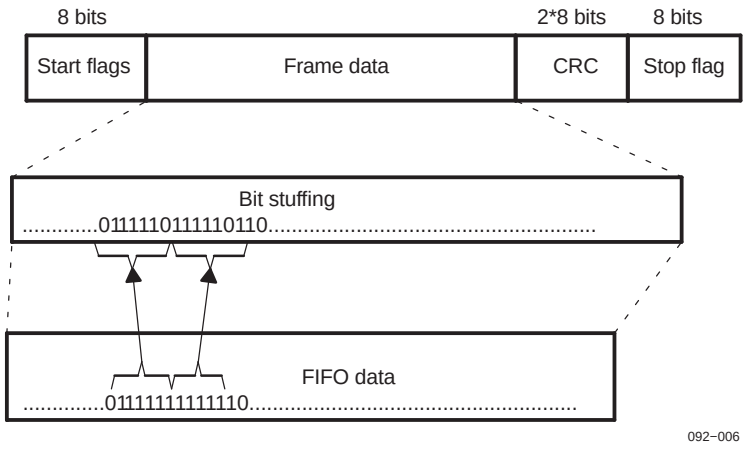
19.1.4.2.1.6 IR Address Checking

In all IR modes, if address checking has been enabled, only the frames intended for the device are written to the RX FIFO. This prevents the reception of frames not meant for this device in a multipoint infrared environment. It is possible to program two frame addresses that the UART IrDA receives with XON1/ADDR1 and XON2/ADDR2 registers.

19.1.4.2.2 MIR Mode

In MIR mode, data transfer occurs between the local host and peripheral devices at 0.576M bits/s or 1.152M bits/s. An MIR transmit frame begins with at least two start flags followed by a frame data and CRC-16, and then ends with a stop flag (see Figure 19-6).

Figure 19-6. IrDA MIR Frame Format



During transmission, the MIR TX state-machine attaches start flags, CRC-16, and stop flags. It also looks for five consecutive 1s in the frame data and automatically inserts a 0 after five consecutive 1s (this is called bit-stuffing).

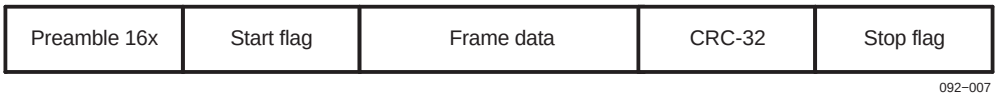
On receive, the MIR RX state-machine recovers the receive clock, removes the start flags, destuffs the incoming data, and determines the frame boundary with reception of the stop flag. It also checks for errors such as frame abort, CRC error, and frame length. At the EOF reception, the local host reads the LSR to determine possible errors of the received frame.

Data can be transferred both ways by the UART, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware.

19.1.4.2.3 FIR Mode

In FIR mode, data transfer occurs between the local host and peripheral devices at 4M bits/s. An FIR frame starts with a preamble, followed by a start flag, frame data, and CRC-32, and then ends with a stop flag (see Figure 19-7).

Figure 19-7. IrDA Frame Format



On transmit, the FIR transmit state-machine attaches the preamble, start flag, CRC-32, and stop flag. It also encodes the transmit data into 4PPM format and generates the serial infrared interaction pulse (SIP).

On receive, the FIR receive state-machine recovers the receive clock, removes the start flag, decodes the 4PPM incoming data, and determines the frame boundary with the reception of a stop flag. It also checks for errors such as illegal symbols, CRC errors, and frame-length errors. At the EOF reception, the local host reads the LSR to determine possible errors of the received frame.

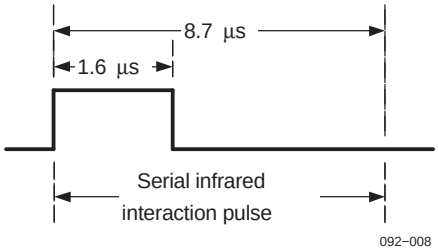
Data can be transferred both ways by the UART, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware.

19.1.4.2.3.1 SIP Generation

In MIR and FIR operation modes, the transmitter must send an SIP at least once every 500 ms. The purpose of the SIP is to inform slow devices (operating in SIR mode) that the medium is occupied.

Figure 19-8 shows the SIP pulse.

Figure 19-8. SIP Pulse



19.1.5 Clocking, Reset, and Power-Management Scheme

19.1.5.1 Clocks

The UART has a functional clock and an interface domains clock. Table 19-2 lists the characteristics of these clocks.

- The UART operates from a single and variable functional clock of 52, 48, or 13 MHz generated by APLL through the clock and reset management (CLKM) module (see Chapter 6, *Power, Reset, and Clock Management*, for more information). The clock is fed in the baud rate generator and is programmably divided to reach a maximum baud rate of 3.6M bits/s when the clock equals to 48 MHz.
- The interface clock, UART_ICLK, triggers access to the UART controller registers through the static switch for reading or writing from the MPU or DSP (see Chapter 6, *Power, Reset, and Clock Management*, for more information).

Table 19-2. UART Clocks

Type	Name	Source	Frequency	Description
Functional	UART_FCLK	ClkUART_SPI	52, 48, or 13 MHz	Clock into the baud generator
Interface	UART_ICLK	Clk_52	52 MHz	Clock bus to interface the UART module with the MPU and DSP

19.1.5.2 Power Management and Power Domain

At the system level, power reduction techniques can be applied by shutting down certain internal clock and power domains of the device. The UART supports an idle request protocol. This protocol is used at system level to shut down clocks of the UART in a clean and controlled manner (see Section 19.2.2.5, *UART Power Management*, and Section 19.2.2.7, *IrDA Mode Power Management*).

Three embedded power-saving features are available (see Section 19.2.2.5, *UART Power Management*, and Section 19.2.2.7, *IrDA Mode Power Management*):

- Sleep mode: The module clock and baud rate clock (from the functional clock) are stopped internally. Because most registers are clocked using these clocks, power consumption is greatly reduced.
- Autoidle: This feature enables gating of the interface clock (open-core protocol [OCP] clock) when no activity is performed on the TIPB/OCP static switch.
- Idle mode: This mode enables the CLKM module to switch off the functional and interface clocks of the module. This mode is used when the device switches to deep-sleep mode.

19.1.5.3 Hardware and Software Resets

The UART has hardware and software resets. Table 19-3 lists the characteristics of each reset.

- To perform global reset or hardware reset of the UART, activate the ARM_nRST signal (see Chapter 6, *Power, Reset, and Clock Management*, for more information).
- To perform software reset, set up a set of configuration registers. For more information, see Section 19.3.1.1, *UART Software Reset*.

Table 19-3. UART Resets

Type	Name	Source	Activation	Domain
Hardware	UART_RESET	ARM_nRST	0	Global reset
Software	See Section 19.3.1.1, *UART Software Reset*	Register programming	1	-

19.1.6 Hardware Requests

19.1.6.1 DMA Requests

UART DMA request mapping is shown in Table 19-4.

Table 19-4. UART DMA Requests

DMA Request Name	Mapping	Description
UART_nDMA_REQ[0]	DMA_4	UART module DMA request
UART_nDMA_REQ[1]	DMA_5	UART module DMA request

19.1.6.2 Interrupt Requests

Table 19-5 lists mapping of interrupts generated by the UART module and their destinations. One interrupt, UART_IT, is generated by the UART module to drive one interrupt on the MPU and the DSP.

Table 19-5. UART Module Interrupt Mapping

Interrupt Name	Mapping	Domain	Description
UART_IT	IRQ_7	MPU	UART module interrupt to MPU interrupt handler
UART_SWAKEUP	IRQ_17	MPU	Wake-up signal used to request clock from CLKM
UART_IT	INT_2	DSP	UART module interrupt to DSP interrupt handler

19.2 Functional Description

This section provides a functional description of the UART submodules.

19.2.1 UART Block Diagram Overview

The UART consists of four primary submodules (see Figure 19-9):

- Mode selection: Two distinct modes, UART and IrDA, are available. A set of registers lets you specify which mode is in use and configure parameters, such as baud rate, flow control, and interrupt.
- FIFO management: Two 64-byte receive and transmit FIFOs for DMA transfer and interrupt generation on programmable threshold.
- Interrupt management: During UART and IrDA modes, a set of several prioritized interrupts is available.
- Clock and baud rate generator: One clock is supplied to the module; the baud rate is generated by division of the UART_FCLK using programmable registers.

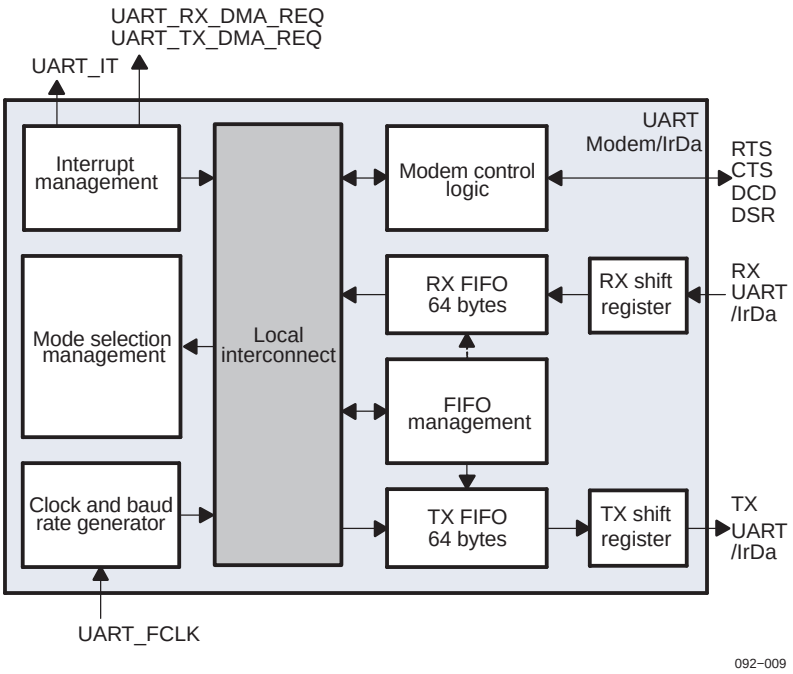
CAUTION

- The software must carefully control the accessibility of each UART peripheral by the desired host.
- DMA transfers from/to the UART must be complete before updating the peripheral ownership.
- The static switch must be properly set up before UART register can be accessed. The result is a time-out error generated by the host bridge.

Two registers control accessibility from the host to a statically shared peripheral (see Chapter 5, *Interconnect*):

- UART_SSW_MPU_CONF in the MPU peripheral address space
- UART_SSW_DSP_CONF in the DSP peripheral address space

Figure 19-9. UART Block Diagram



19.2.2 Mode Selection

Each mode (UART and IrDA) has submodes. Table 19-6 lists the modes of operation programmable through the mode-definition register, MDRI[2:0].

Table 19-6. UART Mode Selection

Value	Mode
0x0	UART 16x mode
0x1	SIR mode
0x2	UART 16x autobaud
0x3	UART 13x mode
0x4	MIR mode
0x5	FIR mode
0x6	Reserved

19.2.2.1 Register Access Mode

19.2.2.1.1 Operational and Configuration Modes

Register access depends on the register access mode. These register access modes are not correlated with functional mode selection. Three modes are available:

- operational_mode
- configuration_mode_A
- configuration_ mode_B

Operational_mode is selected when the function is active; serial data transfer can be performed in this mode.

Configuration_mode_A and configuration_mode_B are used during module initialization. These modes enable access to the configuration registers, which are hidden in operational_mode. These modes are used when the module is inactive (no serial data transfer processed) and only during initialization or reconfiguration of the module.

Register access mode is determined by the LCR register value (see Table 19-7).

Table 19-7. UART/IrDA Register Access Mode Programming (through LCR)

Mode Name	Condition
configuration_mode_A	LCR[7] = 0x1 and LCR[7:0] ! = 0xBF
configuration_mode_B	LCR[7:0] = 0xBF
operational_ mode	LCR[7] = 0x0

For a list of the registers available in each mode, see Section 19.4.2.2, *Register Summary*.

19.2.2.1.2 Register Access Submode

In each access register mode (operational_mode or configuration_mode_A/B), some register accesses are conditional to the programming of a submode. These registers are identified in Section 19.4.2.2, *Register Summary*. The submodes are MSR_SPR, TCR_TLR, and XOFF (see Table 19-8 through Table 19-10).

Table 19-8. configuration_mode_A Mode Summary

Mode Name	Condition
MSR_SPR	EFR[4] = 0x0 or MCR[6] = 0x0

Table 19-8. configuration_mode_A Mode Summary (continued)

Mode Name	Condition
TCR_TLR	EFR[4] = 0x1 and MCR[6] = 0x1

Table 19-9. configuration_mode_B Mode Summary

Mode Name	Condition
TCR_TLR	EFR[4] = 0x1 and MCR[6] = 0x1
XOFF	EFR[4] = 0x0 or MCR[6] = 0x0

Table 19-10. Suboperational_mode Mode Summary

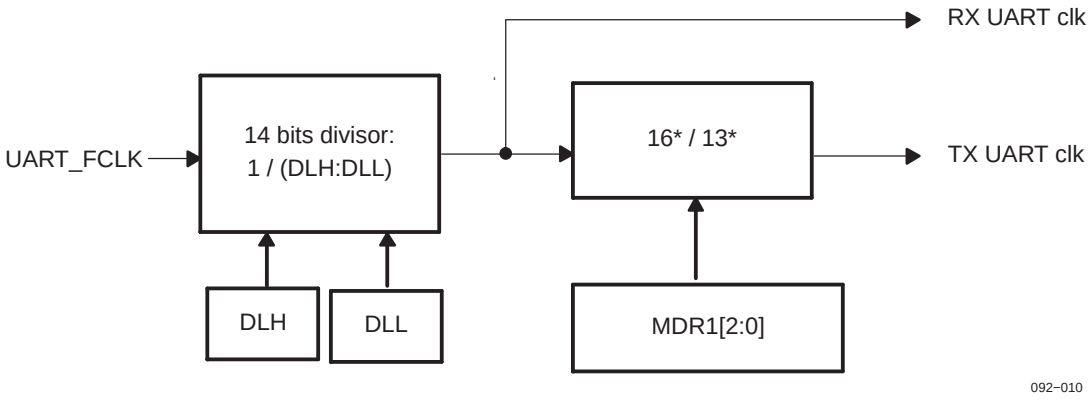
Mode Name	Condition
MSR_SPR	EFR[4] = 0x0 or MCR[6] = 0x0
TCR_TLR	EFR[4] = 0x1 and MCR[6] = 0x1

19.2.2.2 UART Mode

19.2.2.2.1 Clock and Baud Rate Generator

The UART function contains a programmable baud generator and a set of fixed dividers that divide the clock input down to the expected baud rate. Figure 19-10 shows the baud rate generator and associated controls.

Figure 19-10. UART Baud Rate Generator



CAUTION

It is recommended that MODE_SELECT = DISABLE (MDR1[2:0] = 111) before trying to initialize or modify clock parameter controls (DLH, DLL). If this recommendation is not observed, the module can behave unpredicatably.

19.2.2.2.2 Choosing the Divisor Value

- UART 16x mode: divisor value = operating frequency/(16x baud rate)
- UART 13x mode: divisor value = operating frequency/(13x baud rate)

Table 19-11 lists the UART baud rate settings for a 48-MHz clock.

Table 19-11. UART Baud Rate Settings (48-MHz Clock)

Baud Rate	Baud Multiple	DLH, DLL (Decimal)	DLH, DLL (Hex)	Actual Baud	Rate Error (%)
0.3 kb/s	16x	10000	0x27, 0x10	0.3 kb/s	0
0.6 kb/s	16x	5000	0x13, 0x88	0.6 kb/s	0
1.2 kb/s	16x	2500	0x09, 0xC4	1.2 kb/s	0
2.4 kb/s	16x	1250	0x04, 0xE2	2.4 kb/s	0
4.8 kb/s	16x	625	0x02, 0x71	4.8 kb/s	0
9.6 kb/s	16x	312	0x01, 0x38	9.6153 kb/s	+0.16
14.4 kb/s	16x	208	0x00, 0xD0	14.423 kb/s	+0.16
19.2 kb/s	16x	156	0x00, 0x9C	19.231 kb/s	+0.16
28.8 kb/s	16x	104	0x00, 0x68	28.846 kb/s	+0.16
38.4 kb/s	16x	78	0x00, 0x4E	38.462 kb/s	+0.16
57.6 kb/s	16x	52	0x00, 0x34	57.692 kb/s	+0.16
115.2 kb/s	16x	26	0x00, 0x1A	115.38 kb/s	+0.16
230.4 kb/s	16x	13	0x00, 0x0D	230.77 kb/s	+0.16
460.8 kb/s	13x	8	0x00, 0x08	461.54 kb/s	+0.16
921.6 kb/s	13x	4	0x00, 0x04	923.08 kb/s	+0.16
1.8342 Mb/s	13x	2	0x00, 0x02	1.8462Mb/s	+0.16
3.6864 Mb/s	13x	1	0x00, 0x01	3.6923 Mb/s	+0.16

19.2.2.3 UART Data Formatting

The UART can use hardware flow control to manage transmission/reception. Hardware flow control significantly reduces software overhead and increases system efficiency by automatically controlling serial data flow using RTS output and CTS input signals.

The UART modem is enhanced with an autobauding functionality that, in control mode, allows automatic setting of the speed, the number of bits per character, and the parity.

19.2.2.3.1 Frame Formatting

When autobauding is not used, frame format attributes must be defined in the LCR register (see Figure 19-4). Table 19-12 lists the parity modes and their programmable methods.

- Character length is specified with the LCR[1:0] CHAR_LENGTH register field.
- The number of stop-bits is specified with the LCR[2] NB_STOP register bit.
- The parity bit is programmed using the LCR[5:3] (PARITY_EN, PARITY_TYPE_1, and PARITY_TYPE_2) register bits.

Table 19-12. UART Parity Bit Encoding

LCR[3]	LCR[4]	LCR[5]	Parity
0	N/A	N/A	No parity
1	0	0	Odd parity
1	1	0	Even parity
1	0	1	Forced 1
1	1	1	Forced 0

19.2.2.3.2 Hardware Flow Control

Hardware flow control is composed of auto-CTS and auto-RTS, which can be enabled/disabled independently by programming EFR[7:6]. With auto-CTS, CTS must be active before the module can transmit data. Auto-RTS activates the RTS output only when there is enough room in the RX FIFO to receive data; auto-RTS deactivates the RTS output only when the RX FIFO is sufficiently full.

The HALT and RESTORE trigger levels in the TCR determine the levels at which RTS is activated/deactivated. If both auto-CTS and auto-RTS are enabled, data transmission does not occur unless the receiver FIFO has empty space. Thus, overrun errors are eliminated during hardware flow control. If both auto-CTS and auto-RTS are not enabled, overrun errors occur if the transmit data rate exceeds the receive FIFO latency.

19.2.2.3.3 Auto-RTS

Auto-RTS data flow control originates in the receiver block. The receiver FIFO trigger levels used in auto-RTS are stored in the TCR. RTS is active if the RX FIFO level is below the HALT trigger level in TCR[3:0]. When the receiver FIFO HALT trigger level is reached, RTS is deasserted. The sending device (for example, another UART) may send an additional byte after the trigger level is reached, because it may not recognize the deassertion of RTS until it has begun sending the additional byte.

RTS is automatically reasserted when the receiver FIFO reaches the RESUME trigger level programmed by TCR[7:4]. This reassertion requests the sending device to resume transmission. In this case, RTS is an active-low signal.

19.2.2.3.4 Auto-CTS

The transmitter circuitry checks CTS before sending the next data byte. When CTS is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, CTS must be deasserted before the middle of the last stop-bit being sent. Auto-CTS reduces interrupts to the host system. When auto-CTS flow control is enabled, the CTS state changes need not trigger host interrupts, because the device automatically controls its own transmitter. Without auto-CTS, the transmitter sends any data present in the transmit FIFO, and a receiver overrun error can occur. In this case, CTS is an active-low signal.

19.2.2.3.5 Autobauding Modes

In autobauding mode, the UART can extract transfer characteristics (speed, length, and parity) from an AT command (ASCII code). These characteristics are used to receive data following an at (AT) command and to send data.

Here are valid AT commands:

- AT
- DATA <CR>
- at DATA <CR>
- A/
- a/

Line break during the acquisition of the sequence AT is not recognized and echo functionality is not implemented in the hardware.

A/ and a/ are not used to extract characteristics, but they must be recognized because of their special meaning. Either A/ or a/ is used to instruct the software to repeat the last received AT command; therefore, an a/ always comes after an AT and transfer characteristics are not expected to change between an AT and an a/.

When a valid AT is received, AT and all subsequent data are saved into RX FIFO, including final <CR> (0x0D), and the autobaud state-machine waits for the next valid AT command. If an a/ (A/) command is received, the a/ (A/) is saved into RX FIFO and the state-machine waits for the next valid AT command.

On the first successful detection of the baud rate, the UART activates an interrupt to signify that the AT (upper or lower case) sequence is detected. The autobauding status register (UASR) reflects the correct settings for the baud rate that has been detected. The interrupt activity can continue in this fashion each time a subsequent character is received. Therefore, it is recommended that the software enable the RHR interrupt when using the autobauding mode.

The following settings are detected in autobauding mode with a module clock of 48 MHz:

- Speed: 115.2K baud, 57.6K baud, 38.4K baud, 28.8K baud, 19.2K baud, 14.4K baud, 9.6K baud, 4.8K baud, 2.4K baud, or 1.2K baud
- Length: 7 or 8 bits
- Parity: odd, even, or space

The combination of 7-bit character + space parity is not supported.

The autobauding mode is selected when MDR1[2:0] = 010. In UART autobauding mode, DLL, DLH, LCR[5:0] settings are not used; instead, UASR is updated with the configuration detected by the autobauding logic.

19.2.2.3.6 Autobauding Status Register

When the UART is in autobauding mode, the UASR, instead of the LCR, DLL, and DLH registers, is used to set up transmission according to the characteristics of the previous reception.

To reset the autobauding hardware (to start a new AT detection) or to set the UART in standard mode (with no autobaud), MDR1[2:0] must be set to reset state 111, and then to the UART in autobauding mode 010 or to the UART in standard mode 000.

Usage limitation:

- Only 7- and 8-bit characters (5- and 6-bit not supported)
- 7-bit character with space parity not supported
- Baud rate between 1200 and 115,200 baud/s (10 possibilities)

19.2.2.3.7 Error Detection

When the LSR is read, LSR[4:2] reflects the error bits (BI: break condition, FE: framing error, PE: parity error) of the character at the top of the RX FIFO (next character to be read). Therefore, reading the LSR and then reading the RHR identifies errors in a character.

Reading RHR updates (BI, FE, PE) (see Table 19-13 for the UART mode interrupts).

LSR[7] is set when there is an error anywhere in the RX FIFO, and is cleared only when no errors remain in the RX FIFO.

Note: Reading the LSR does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the RHR.

Reading the LSR clears the OE bit if it is set (see Table 19-13 for the UART mode interrupts). Overrun occurs during receive if the RX state-machine tries to write data into the RX FIFO when it is full. When overrun occurs, the device interrupts the local host with IIR[3] and discards the remaining portion of the frame. Overrun also causes an internal flag to be set, which disables further reception. Before the next frame can be received, the system (local host) must:

- Reset the RX FIFO.
- Read the RESUME register (this clears the internal flag).

19.2.2.3.8 Time-out and Break Conditions

19.2.2.3.8.1 Time-out Counter

An RX idle condition is detected when the receiver line, RX, has been high for a time equivalent to 4x programmed word length + 12 bits. The receiver line is sampled midway through each bit. For sleep mode, the counter is reset when there is activity on the RX line. For the time-out interrupt, the counter counts only when there is data in the RX FIFO, and the count is reset when there is activity on the RX line or when the RHR is read.

19.2.2.3.8.2 Break Condition

When a break condition occurs, the TX line is pulled low. A break condition is activated by setting LCR[6]. The break condition is not aligned on the word stream (that is, a break condition can occur in the middle of a character). The only way to send a break condition on a full character is by using the following procedure:

1. Reset transmit FIFO (if enabled).
2. Wait until the transmit shift register becomes empty (LSR[6] = 1).
3. Take a guard time according to stop-bit definition.
4. Set LCR[6] to 1.

Break condition is asserted as long as LCR[6] is set to 1.

The time-out counter and break condition apply only to UART operation, not to the IrDA modes of operation.

19.2.2.4 UART Mode Interrupt Management

The UART function generates interrupts. All interrupts can be enabled/disabled by writing to the appropriate bit in the interrupt enable register (IER). The interrupt status of the device can be checked at any time by reading the interrupt identification register (IIR). The UART and IrDA modes have different interrupts in the UART/IrDA module and, therefore, have different IER and IIR mappings according to the selected mode.

In UART modes, there are seven possible interrupts. These interrupts are prioritized to six levels. When an interrupt is generated, the IIR indicates that an interrupt is pending by bringing IIR[0] to 0 and provides the type of interrupt through IIR[5_UnicodeEncodeError_1]. Table 19-13 summarizes the interrupt control functions.

Table 19-13. UART Mode Interrupts

IIR[5_UnicodeEncodeError_0]	Priority Level	Interrupt Type	Interrupt Source	Reset Method
0 0 0 0 0 1	None	None	None	None
0 0 0 1 1 0	1	Receiver line status	OE, FE, PE, or BI errors in characters in the RX FIFO	FE, PE, BI: Read RHR. OE: Read LSR.
0 0 1 1 0 0	2	RX time-out	Stale data in RX FIFO	Read RHR.
0 0 0 1 0 0	2	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read RHR until interrupt condition disappears.
0 0 0 0 1 0	3	THR interrupt	TFE (THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR until interrupt condition disappears.
0 0 0 0 0 0	4	Modem status	MSR[1:0]/ = 0	Read MSR.
0 1 0 0 0 0	5	XOFF interrupt/special character interrupt	Receive XOFF characters(s)/ special character	Receive XON character(s), if XOFF interrupt/read of IIR, if special character interrupt.

Table 19-13. UART Mode Interrupts (continued)

IIR[5_UnicodeEncodeError_0]	Priority Level	Interrupt Type	Interrupt Source	Reset Method
1 0 0 0 0 0	6	CTS, RTS	RTS pin or CTS pin change state from active (low) to inactive (high)	Read IIR.

For the receiver-line status interrupt, the RX_FIFO_STS bit (LSR[7]) generates the interrupt.

For the XOFF interrupt, if an XOFF flow character detection causes the interrupt, the interrupt is cleared by an XON flow character detection. If special character detection causes the interrupt, the interrupt is cleared by a read of the IIR.

19.2.2.5 UART Power Management

In addition to active mode, the UART/IrDA modules support the following power-saving modes:

19.2.2.5.1 Sleep Mode

In UART modes, sleep mode is enabled by writing 1 to IER[4] (when EFR[4] = 1). Sleep mode is entered when:

- The serial data input line, RX is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- There are no interrupts pending except THR interrupts.

Sleep mode is a good way to reduce power consumption of the UART module, but this state can be achieved only when the UART is set in UART mode. Therefore, even if the UART has no functional key role, it must be initialized in a functional mode to take advantage of sleep mode.

In sleep mode, the module clock and baud rate clock (from the functional clock) are stopped internally. Because most registers are clocked using these clocks, power consumption is greatly reduced. The module wakes up when any change is detected on the RX line, if data is written to the TX FIFO, and when there is a change in the state of the modem input pins.

Note: There must be no writing to the divisor latches, DLL and DLH, to set the baud clock, BCLK, while in sleep mode. It is advisable to disable sleep mode using IER[4] before writing to DLL or DLH.

19.2.2.5.2 Autoidle Feature

Even if the autoidle feature is not a true power-saving state in the state-machine, this feature can gate off the interface clock (OCP clock) when no activity is performed on the interface TIPB. The autoidle feature is enabled by setting the AUTOIDLE bit (SYSC[0]) to 1.

19.2.2.5.3 Idle Mode

Idle mode is different from the autoidle feature. Idle mode enables the CLKM to switch off the functional and interface clocks of the module. This mode is used when the chip switches to deep-sleep mode.

When the CLKM issues an idle request through the UART_IDLEREQ line, the UART goes into idle mode. The clocks can be removed from the module. Depending on the setting of the IDLEMODE field of the system configuration register (SYSC), the module has different behaviors:

1. If the IDLEMODE field is set to no-idle mode (SYSC[4:3] = 01), the UART does not go into idle mode.
2. If the IDLEMODE field is set to force-idle mode (SYSC[4:3] = 00), the UART goes into idle mode and acknowledges the request unconditionally. In this mode, the UART does not execute any transactions.

3. If the IDLEMODE field is set to smart-idle mode(SYSC[4:3] = 10), the UART evaluates its internal capability to have the functional and interface clocks switched off. When there is no internal activity, the request signal is acknowledged through the UART_IDLEACK signal and the UART enters sleep mode.

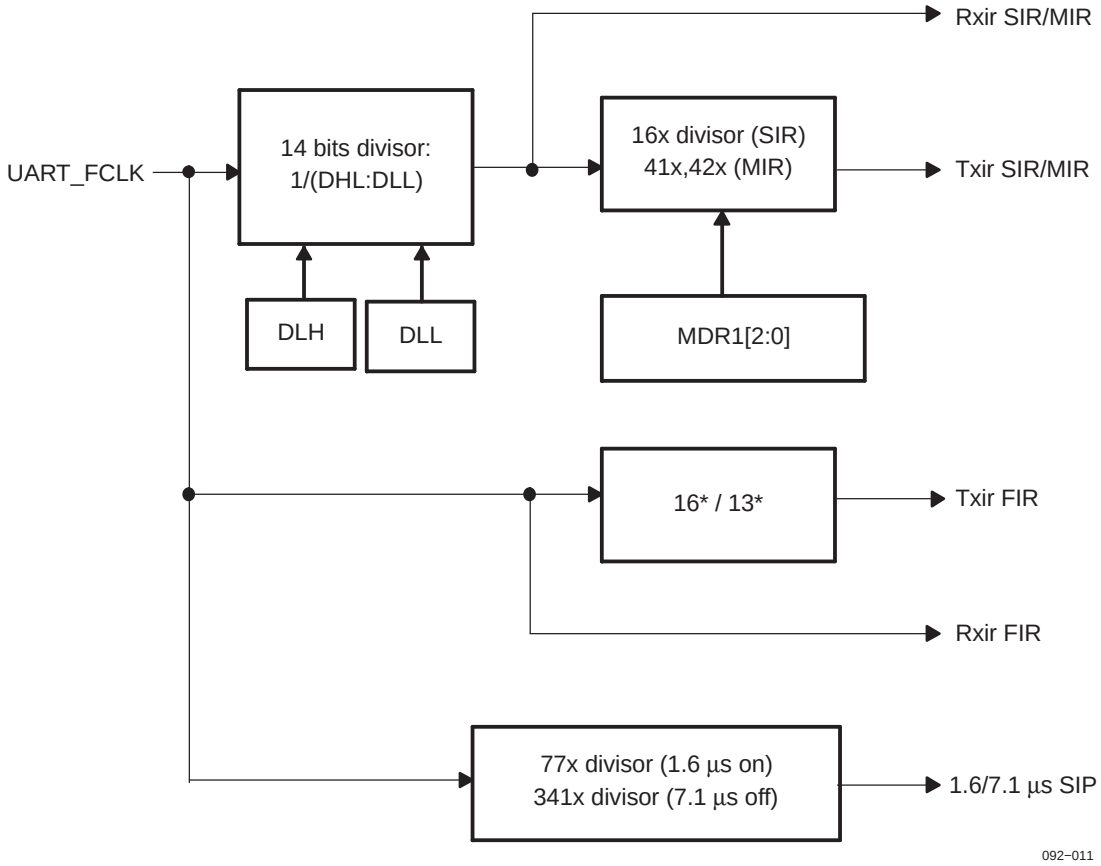
When the IDLEMODE field is set to either force-idle or smart-idle mode, the LOCOSTO device can enter deep-sleep state, and it can be awakened by the UART. To wake up the device, the ENAWAKEUP bit of SYSC[2] must be set.

19.2.2.6 IrDA Mode

19.2.2.6.1 Clock and Baud Rate Generator

The IrDA function contains a programmable baud generator and a set of fixed dividers that divide the clock input down to the expected baud rate. Figure 19-11 shows the IrDA baud rate generator and associated controls.

Figure 19-11. IrDA Baud Rate Generator



CAUTION

It is recommended that MODE_SELECT = DISABLE (MDR1[2:0] = 111) before trying to initialize or modify clock parameter controls (DLH, DLL). If this recommendation is not observed, the module can behave unpredictably.

19.2.2.6.2 Choosing the Appropriate Divisor Value

- SIR mode: divisor value = operating frequency/(16x baud rate)
- MIR mode: divisor value = operating frequency/(41x/42x baud rate)
- FIR mode: divisor value = none

19.2.2.6.3 IrDA Data Formatting

Methods described in this section apply to all IrDA modes (SIR, MIR, and FIR).

19.2.2.6.4 IrDA Reception Control

Data can be transferred both ways by the module, but when the device is transmitting, the IrRX circuitry is automatically disabled by hardware. The IrRx input operation can be disabled with the DIS_IR_RX bit of the auxiliary control register (ACREG[5]).

19.2.2.6.5 IR Address Checking

In all IR modes, if address checking is enabled, only frames intended for the device are written to the RX FIFO. This avoids receiving frames not meant for this device in a multipoint infrared environment. It is possible to program two frame addresses that the UART IrDA receives, with XON1/ADDR1 and XON2/ADDR2 registers.

- Address1 checking is done by setting EFR[0] to 1.
- Address2 checking is done by setting EFR[1] to 1.
- Setting EFR[1:0] to 0 disables all address-checking operations.
- If both bits are set, the incoming frame is checked for both private and public addresses.
- If address checking is disabled, all received frames are written into the reception FIFO.

19.2.2.6.6 Frame Closing

Two methods can be used to terminate a transmission frame:

- Frame-length method: This method is selected when MDR1[7] = 0. The local host writes the frame-length value to the TXFLH and TXFLL registers. The device automatically attaches end flags to the frame when the number of bytes transmitted equals the frame-length value.
- Set-EOT bit method: This method is selected when MDR1[7] = 1. The local host writes 1 to ACREG[0] (EOT bit) just before it writes the last byte to the TX FIFO. When writing the last byte to the TX FIFO, the device internally sets the tag bit for that particular character in the TX FIFO. As the TX state-machine reads data from the TX FIFO, it uses this tag-bit information to attach end flags and to correctly terminate the frame.

19.2.2.6.7 Store and Controlled Transmission

First, the local host writes data into the TX FIFO. Then, it writes a part or a whole frame. By writing 1 to ACREG[2], the local host defers the start of transmission. SCT is enabled when MDR1[5] = 1. This method of transmission is different from the normal mode, where transmission of data starts immediately after data is written to the TX FIFO. SCT is useful for sending short frames without TX underrun.

19.2.2.6.8 Error Detection

When the LSR is read, LSR[4:2] reflects the error bits [FL, CRC, ABORT] of the frame at the top of the STATUS FIFO (the next frame status to be read). An error is triggered by an interrupt (see Table 19-14). STATUS FIFO must be read until it is empty (maximum of eight reads required).

19.2.2.6.9 Underrun During Transmission

Underrun in transmission occurs when the TX FIFO becomes empty before the EOF is transmitted. When underrun occurs, the device closes the frame with end flags but attaches an incorrect CRC value. The receiving device detects a CRC error and discards the frame; it can then ask for a retransmission.

Underrun also causes an internal flag to be set, which disables further transmission. Before the next frame can be transmitted, the local host must do the following:

- Reset the TX FIFO.
- Read the RESUME register (this clears the internal flag).

This functionality can be disabled with ACREG[4], compensated by the extension of the stop-bit in transmission in case the TX FIFO is empty.

19.2.2.6.10 Overrun During Receive

Overrun occurs during receive if the RX state-machine tries to write data into the RX FIFO when it is full. When overrun occurs, the device interrupts the local host with IIR[3] and discards the remaining portion of the frame.

Overrun also causes an internal flag to be set, which disables further reception. Before the next frame can be received, the local host must do the following:

- Reset the RX FIFO.
- Read the RESUME register (this clears the internal flag).

19.2.2.6.11 Status FIFO

In IrDA modes, a status FIFO records the received frame status. When a complete frame is received, the length of the frame and the error bits associated with the frame are written into the status FIFO.

Frame length and error status can be read by reading SFREGL/H and SFLSR. Reading SFLSR causes the read pointer to be incremented. The status FIFO is eight entries deep and, therefore, can hold the status of eight frames.

The local host uses the frame-length information to locate the frame boundary in the received frame data. The local host can screen bad frames using the error status information and can later request the sender to resend only the bad frames.

This status FIFO can be used effectively in DMA because the local host must be interrupted only when the programmed status FIFO trigger level is reached, not each time a frame is received.

19.2.2.6.12 SIR Data Formatting

This section gives specific instructions for SIR mode programming.

19.2.2.6.13 Abort Sequence

The transmitter can prematurely close a frame. The transmitter aborts by sending the following sequence: 0x7DC1. The abort pattern closes the frame without a CRC field or an ending flag.

It is possible to abort a transmission frame by programming the ABORT_EN bit of the auxiliary control register (ACREG [1]).

When this bit is set to 1, 0x7D and 0xC1 are transmitted and the frame is not terminated with CRC or stop flags.

The receiver treats a frame as an aborted frame when a 0x7D character, followed immediately by a 0xC1 character, is received without transparency.

CAUTION

If transmit FIFO is not empty and MDR1[5] = 1, UART IrDA starts a new transfer with data of the previous frame when an abort frame has been sent. Therefore, TX FIFO must be reset before sending an abort frame.

19.2.2.6.14 Pulse Shaping

SIR mode supports both the 3/16th or the 1.6-µs pulse duration methods. ACREG[7] selects the pulse width method in transmit mode.

19.2.2.6.15 MIR and FIR Data Formatting

This section gives common instructions for MIR and FIR mode programming.

At the EOF reception, the local host reads the LSR to detect possible errors in the received frame. When the SIP_MODE bit of MDR1 equals 1 (MDR1[6]), the TX state-machine always sends one SIP at the end of a transmission frame.

But when MDR1[6] = 0, the transmission of the SIP depends on the SEND_SIP bit of the auxiliary control register (ACREG[3]). The local host can set ACREG[3] at least once every 500 ms. The advantage of this approach over the default approach is that the TX state-machine does not need to send the SIP at the end of each frame. This can reduce the overhead required.

19.2.2.6.16 IrDA Mode Interrupt Management

The IrDA function generates interrupts. All interrupts can be enabled/disabled by writing to the appropriate bit in the IER. The interrupt status of the device can be checked at any time by reading the IIR. The UART and IrDA modes have different interrupts in the UART/IrDA module and, therefore, have different IER and IIR mappings according to the selected mode.

In IrDA modes, there are eight possible interrupts (see Table 19-14). The interrupt line is activated when one of the eight interrupts is generated.

Table 19-14. IrDA Mode Interrupts

IIR Bit	Interrupt Type	Interrupt Source	Interrupt Reset Method
0	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read RHR until interrupt condition disappears.
1	THR interrupt	TFE (THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR until interrupt condition disappears.
2	Last byte in RX FIFO	Last byte of frame in RX FIFO	Read IIR.
3	RX overrun	Write to RHR when RX FIFO full	Read RESUME register.
4	Status FIFO interrupt	Status FIFO triggers level reached	Read STATUS FIFO.
5	TX status	1. THR empty before EOF sent. Last bit of transmission of the IrDA frame has occurred but with an underrun error. or 2. Transmission of the last bit of the IrDA frame is finished successfully.	1. Read RESUME register. or 2. Read IIR.
6	Receiver line status interrupt	CRC, abort, or frame-length error is written into STATUS FIFO.	Read STATUS FIFO. (Read until empty; maximum of eight reads required.)
7	Received EOF	Received end-of-frame.	Read IIR.

19.2.2.7 IrDA Mode Power Management

19.2.2.7.1 Sleep Mode

In IrDA modes, sleep mode is enabled by writing 1 to MDR1[3]. Sleep mode is entered when:

- The serial data input line, IRRX is idle.
- The TX FIFO and TX shift registers are empty.
- The RX FIFO is empty.
- The only pending interrupts are THR interrupts.

The module wakes up when a change is detected on the IRRX line or if data is written to the TX FIFO.

19.2.2.7.2 Autoidle Mode and Idle Mode

Autoidle and idle mode behavior are the same as for the UART. See Section 19.2.2.5, *UART Power Management*.

19.2.3 FIFO Management

19.2.3.1 Receiver/Transmitter

The receiver section consists of the receiver holding register (RHR) and the receiver shift register. The RHR is a 64-byte FIFO. The receiver shift register receives serial data from RX input. This data is converted to parallel data and moved to the RHR. If the FIFO is disabled, location 0 of the FIFO is used to store the single data character.

Note: If an overflow occurs, the data in the RHR is not overwritten.

The transmitter section consists of the transmit holding register (THR) and the transmit shift register. The THR is a 64-byte FIFO. The local host writes data to the THR. This data is placed into the transmit shift register, where it is shifted out serially on the TX output.

Note: If the FIFO is disabled, location 0 of the FIFO is used to store the data.

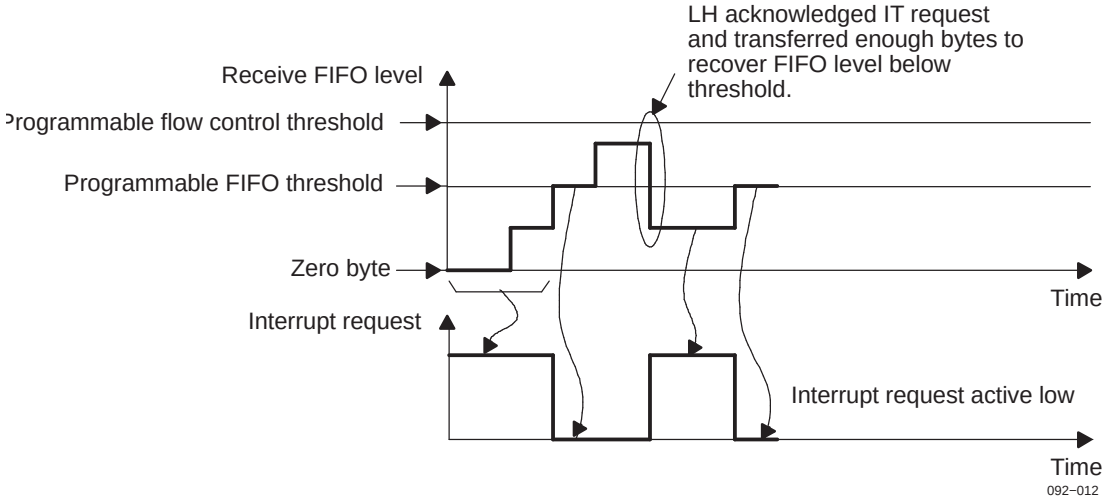
19.2.3.2 FIFO Interrupt Mode

In FIFO interrupt mode (FIFO control register FCR[0] = 1), when relevant interrupts are enabled by the IER, an interrupt signal UART_IP informs the DSP or MPU of the receiver and transmitter status. These interrupts are raised when receive/transmit FIFO thresholds (respectively, TLR[7:4] and TLR[3:0], or FCR[7:6] and FCR[5:4]) are reached. The interrupt signals instruct the actual host to transfer data to the destination (from the UART in receive mode and/or from any source to the UART FIFO in transmit mode).

When UART flow control is enabled along with interrupt capabilities, ensure that the UART flow control FIFO threshold (TCR[3:0]) is greater than or equal to the receive FIFO threshold.

Figure 19-12 and Figure 19-13 show receive and transmit operations, respectively, with the associated interrupt generation.

Figure 19-12. Receive FIFO Interrupt Request Generation



In receive mode, no interrupt is generated until receive FIFO reaches its threshold. Once low, the interrupt can be deasserted only when the local host has handled enough bytes to make the FIFO level below threshold. The flow control threshold is set at a higher value than the FIFO threshold. Table 19-15 resumes the registers to program the threshold value.

Receive threshold programming with access to the TCR register: TCR[7:4]: RX_FIFO_TRIG_START field and TCR[3:0]: RX_FIFO_TRIG_ HALT field.

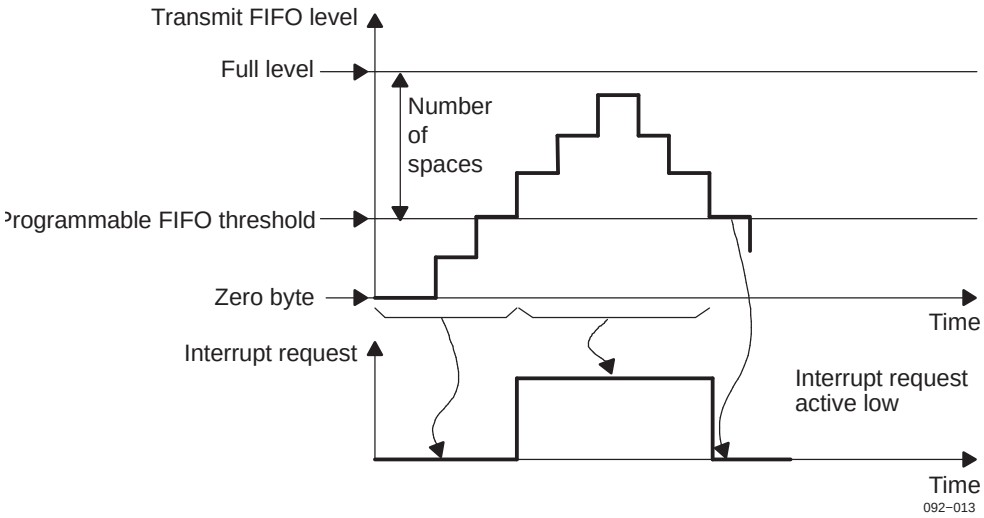
- Trigger levels from 0_UnicodeEncodeError_60 bytes are available with a granularity of 4 (trigger level = 4 x [4-bit register value]).
- The programmer must ensure that TCR[3:0] > TCR[7:4] whenever auto-RTS is enabled, to avoid improper operation of the device.
- In FIFO interrupt mode with flow control, the programmer must also ensure that the trigger level to halt transmission is greater than or equal to the receive FIFO trigger level (either TLR[7:4] or FCR[7:6]); otherwise, the FIFO operation stalls. In FIFO DMA mode with flow control, this concept does not exist, because a DMA request is sent each time a byte is received.

Table 19-15. RX FIFO Trigger Level Setting Summary

SCR[7]	TLR[7:4](1)	RX FIFO Trigger Level
0	= 0 0 0 0	Defined by FCR[7:6] (8, 16, 56, or 60 characters)
0	≠ 0 0 0 0	Defined by TLR[7:4] (from 4 to 60 characters, with a granularity of 4 characters)
1	Value	Defined by the concatenated value of TLR[7:4] and FCR[7:6] (from 1 to 63 characters, with a granularity of 1 character)

(1) The combination of TLR[7:4]=0000 and FCR[7:6]=00 (all zeros) is not supported (requires a minimum of one space). All 0s result in unpredictable behavior.

Figure 19-13. Transmit FIFO Interrupt Request Generation



In transmit mode, an interrupt request is automatically asserted when the TX FIFO is empty. This request is deasserted when the TX FIFO crosses the threshold level. The interrupt line is deasserted until a sufficient number of elements has been transmitted to go below the TX FIFO threshold. Table 19-16 lists the TX FIFO trigger level settings.

Table 19-16. TX FIFO Trigger Level Setting Summary

SCR[6]	TLR[3:0]	TX FIFO Trigger Level
0	= 0 0 0 0	Defined by FCR[5:4] (8, 16, 32, or 56 spaces)
0	≠ 0 0 0 0	Defined by TLR[3:0] (from 4 to 60 spaces, with a granularity of 4 spaces)
1	Value	Defined by the concatenated value of TLR[3:0] and FCR[5:4] (from 1 to 63 spaces, with a granularity of 1 space).

19.2.3.3 FIFO DMA Modes of Operation

There are four modes of DMA operation, DMA modes 0, 1, 2, and 3. They can be selected in the following manner:

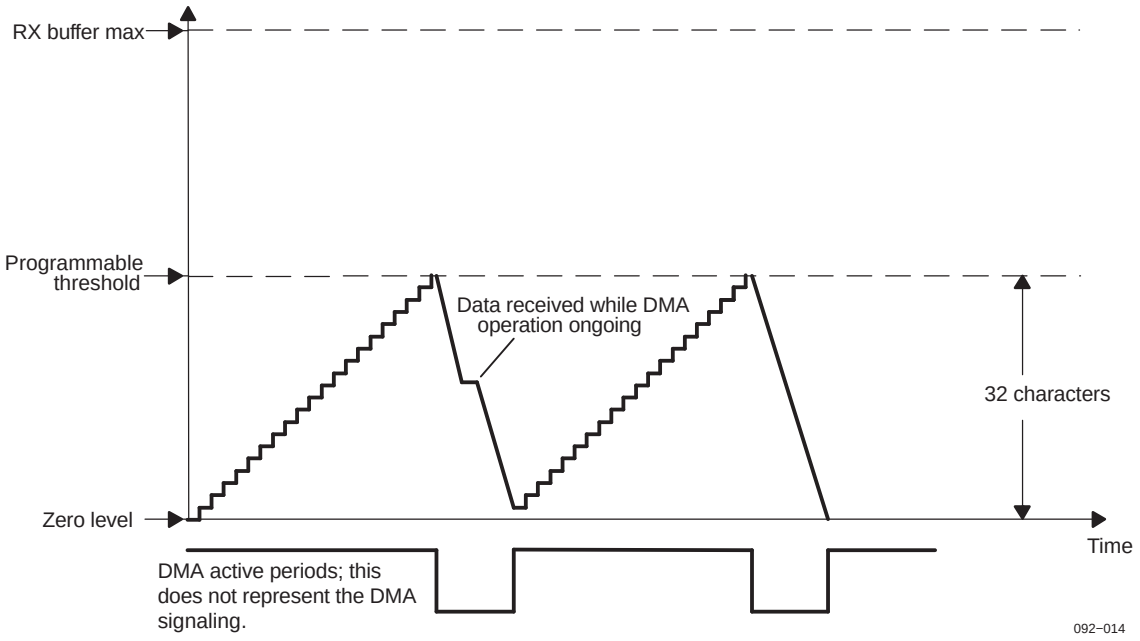
- If no DMA operation is desired, set SCR[0] to 1 and SCR[2:1] to 00 (FCR[3] is discarded).
- If DMA mode 1 is desired, set SCR[0] to 0 and FCR[3] to 1, or set SCR[0] to 1 and SCR[2:1] to 01 (FCR[3] is discarded).
- If DMA mode 2 or 3 is desired, set SCR[0] to 1 and SCR[2:1] to 10 or 11, respectively.

Note: When DMA mode 0 has been programmed, the signals associated with DMA operation are not active.

If the FIFO are disabled (FCR[0] = 0), DMA occurs in single-character transfers.

Figure 19-14 and Figure 19-15 show different types of DMA requests supported in DMA modes 1, 2, and 3.

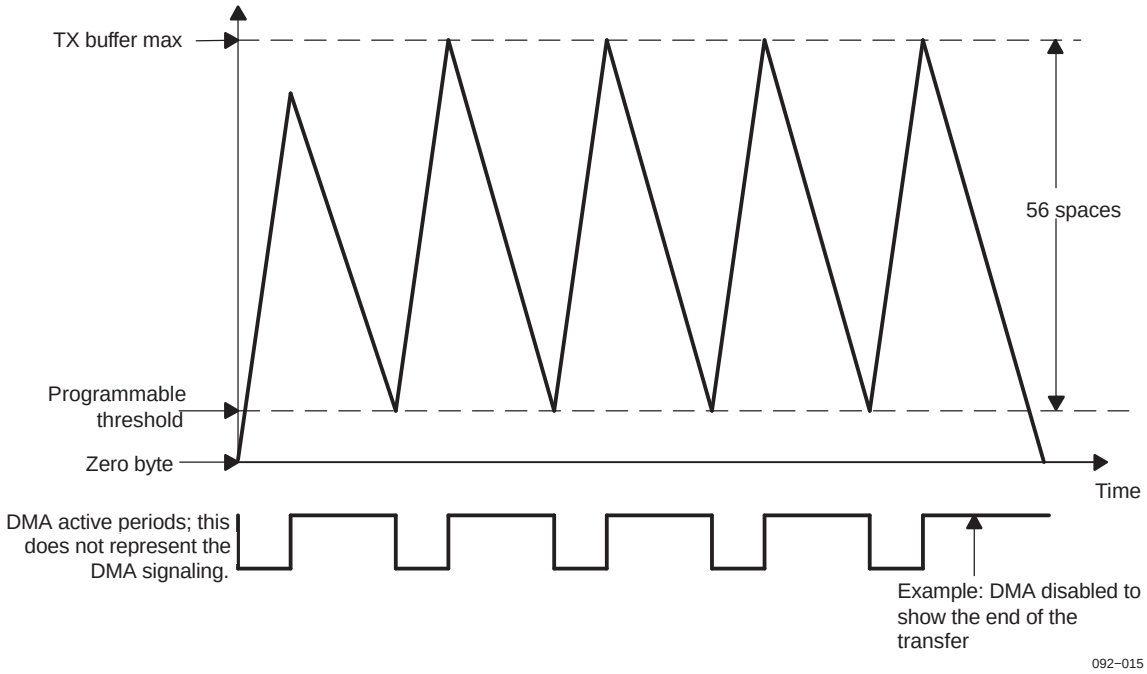
Figure 19-14. Receive FIFO DMA Request Generation



In receive mode, a DMA request is generated when the receive FIFO reaches the threshold level defined in the trigger level register (TLR). This request is deasserted when the number of bytes defined by the threshold level has been read by the system DMA (sDMA).

In transmit mode, a DMA request is automatically asserted when the transmit FIFO is empty. This request is deasserted when the number of bytes defined by the number of spaces in the TLR has been written by the sDMA. If an insufficient number of characters is written, the DMA request remains active.

Figure 19-15. Transmit FIFO DMA Request Generation



The DMA request is again asserted if the FIFO can receive the number of bytes defined by the TLR.

The threshold can be programmed in a number of ways. Figure 19-15 shows a DMA transfer that operates with a space setting of 56 that could arise because of the auto settings in the FCR[5:4] or if the TLR[3:0] is concatenated with the FCR[5:4].

The setting of 56 spaces in the UART/IrDA module must correspond to settings of the sDMA so that the buffer does not overflow (program the DMA request size of the local host controller to equal value the number of spaces in the UART/IrDA module).

Figure 19-16 provides another example with eight spaces to show the buffer level crossing the space threshold. The local host DMA controller settings must correspond with those of the UART/IrDA module.

Figure 19-16. Transmit FIFO DMA Request Generation (Eight Spaces)

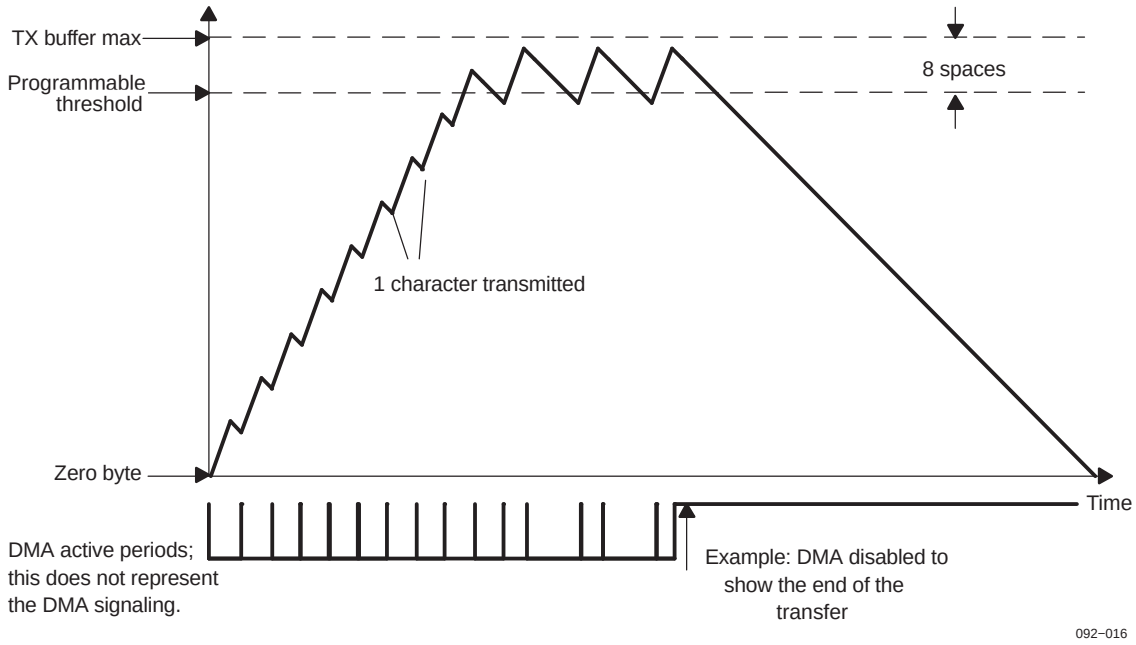
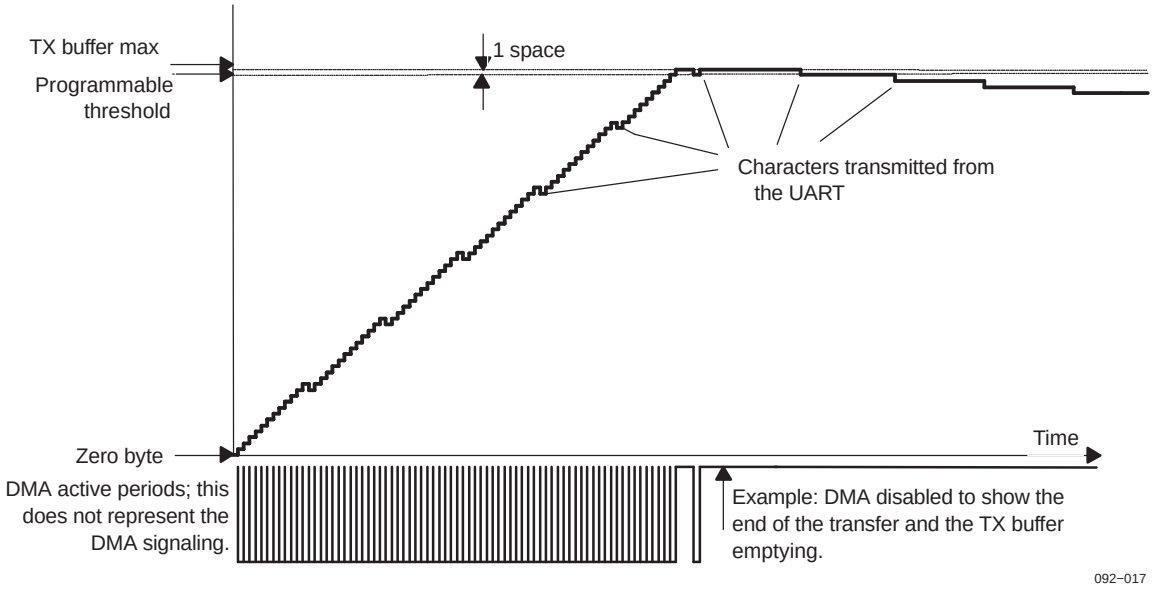


Figure 19-17 shows the setting of one space that uses the DMA for each transfer of one character to the transmit buffer. The buffer is filled at a faster rate than the baud rate transmits data to the TX pin. Eventually, the buffer is completely full and the DMA operations stop transferring data to the transmit buffer. On two occasions, the buffer holds the maximum amount of data words; shortly after this, the DMA is disabled to show the slower transmission of the data words to the TX pin. Eventually, the buffer is emptied at the rate specified by the baud rate settings of the DLL and DLH registers. The DMA settings must correspond to the system local host DMA controller settings to ensure the correct operation of this logic.

Figure 19-17. Transmit FIFO DMA Request Generation (One Space)



19.3 Programming Model

This section provides an overview of the UART programming.

19.3.1 UART Configuration Example

This section outlines the programming stages to operate one UART with FIFO, interrupt, and no DMA capabilities. This operation is a 3-step procedure that ensures quick start of these modules (it does not cover every UART module feature). The first stage covers software reset of the module (interrupts, status, and controls). The second stage deals with FIFO configuration and enable. The last stage deals with the baud rate data and stop configuration. The programming sequences are described in the following sections.

19.3.1.1 UART Software Reset

Goal:

To clear the IER and MCR registers, remove any UART breaks (LCR[6] = 0), and put the module in reset (MDR1[2:0] = 0x07).

Procedure:

To write into the IER and MCR, EFR[4] must first be set to 1.

To access the enhanced feature register (EFR), 0xBF must first be written to the LCR. Thus:

1. LCR = 0xBF: First, write to the LCR.
2. EFR[4] = 1: When LCR = 0xBF, enable the EFR.
3. LCR[7] = 0: Access to IER and MCR is allowed.
4. IER = 0x00: Disable interrupt.
5. MCR = 0x00: Force control signals inactive.
6. LCR[6] = 0: UART breaks are removed.
7. MDR1 = 0x07: UART is in reset or disabled.

Alternately, SYSC[1] can be set to 1 to instigate a hardware reset from the generic synchronous reset module. The reset progress can be monitored by SYSS[0]. Once complete, the preceding sequence must ensure that the UART module is in the equivalent disabled mode with reference to MDR1[2:0].

19.3.1.2 UART FIFO Configuration

Goal:

To set the trigger level for the halt/restore (the TCR), set the trigger level for the transmit/receive (the TLR), and configure the FIFO (the FCR).

Procedure:

To write into the TLR and TCR, both EFR[4] and MCR[6] must be set to 1.

To write into the FCR, EFR[4] must be set to 1. EFR[4] = 1 was done in Section 19.3.1.1; therefore, only a simple write to MCR[6] is necessary.

1. MCR[6] = 1
2. Set TCR, TLR, and FCR to the desired value.

Here, accesses to the TCR, TLR, and FCR must be disabled to avoid further undesired writes to these registers:

1. LCR = 0xBF; provides access to EFR.
2. EFR[4] = 0
3. LCR[7] = 0
4. MCR[6] = 0

19.3.1.3 Baud Rate Delay and Stop Configuration

Goal:

To configure UART data and stop (the LCR), baud rate (the DLH and DLL), and enable UART operation. If interrupt capability is added, configuration must be added immediately before UART enable.

Procedure:

1. Set LCR to desired value.
2. LCR[7] to 1: Gives access to DLH and DLL registers.
3. Set DLH and DLL.
4. LCR[7] = 0: Removes access to DLH and DLL registers.
5. Set IER to desired value: Sets interrupts.
6. MDR1[2:0] = 0: Enables UART without autobauding.

The UART is operational.

19.4 Register Manual

This section contains the register manual.

19.4.1 UART/IrDA Register Instance Summary

Table 19-17 and Table 19-18 show the base address and address spaces for the UART/IrDA module instances when accessed by the MPU or the DSP.

Table 19-17. Instance Summary for MPU

Module Name	MPU Base Address	Size
UART(1)	0xFFFF 7000	2K bytes

(1) The UART is a TIBP strobe 0 peripheral; therefore, it is accessible by both the MPU and the DMA controllers.

Table 19-18. Instance Summary for DSP

Module Name	DSP Base Address	Size
UART	0x7900	256 bytes

19.4.2 Module Register Mapping Summary

19.4.2.1 Operational Mode and Configuration Mode

Register access depends on the register access mode. The register access modes are not correlated with the functional mode selection. Three modes are available: operational_mode, configuration_mode_A, and configuration_ mode_B. Operational_mode is selected when the function is active; serial data transfer can be performed in this mode. Configuration_mode_A and configuration_ mode_B are used during module initialization. These last two modes enable access to configuration registers, which are hidden in operational_mode. These modes are used when the module is inactive (no serial data transfer processed) and only during initialization or reconfiguration of the module.

The register access mode is determined by the LCR value (see Table 19-20).

CAUTION

UART registers are limited to 8-LSB and 16-bit data access; 32-bit data access is not allowed and can corrupt register contents.

Table 19-19 lists the UART/IrDA registers by address offset and their respective configuration modes.

CAUTION

To determine the physical address of the register, use Base Address + Register Offset where S = 1 for DSP access and S = 2 for MPU access; that is, LSR(UART) physical address for MPU access is 0xFFFF 7000 + (0x05 * 2) = 0xFFFF 700A. DSP access is 0x7900 + (0x05 * 1) = 0x7905.

Table 19-19. UART/IrDA Register Overview

Address Offset	Registers					
	Configuration_Mode_A		Configuration_Mode_B		Operational_Mode	
	Read	Write	Read	Write	Read	Write
0x00	DLL	DLL	DLL	DLL	RHR	RHR
0x01 * S	DLH	DLH	DLH	DLH	IER (UART)/ IER (IrDA)	IER (UART)/ IER (IrDA)
0x02 * S	IIR (UART)/ IIR (IrDA)	FCR	EFR/ EFR[4] (IrDA)	EFR/ EFR[4] (IrDA)	IIR (UART)/ IIR (IrDA)	FCR (UART)/ FCR (IrDA)
0x03 * S	LCR (UART)/ LCR[7] (IrDA)	LCR (UART)/ LCR[7] (IrDA)	LCR (UART)/ LCR[7] (IrDA)	LCR (UART)/ LCR[7] (IrDA)	LCR (UART)/ LCR[7] (IrDA)	LCR (UART)/ LCR[7] (IrDA)
0x04 * S	MCR (UART)	MCR (UART)	XON1 (UART)/ ADDR1 (IrDA)	XON1 (UART)/ ADDR1 (IrDA)	MCR (UART)	MCR (UART)
0x05 * S	LSR (UART)/ LSR (IrDA)	-	XON2 (UART)/ ADDR2 (IrDA)	XON2 (UART)/ ADDR2 (IrDA)	LSR (UART)/ LSR (IrDA)	_UnicodeEncodeError_
0x06 * S	MSR (UART)/ TCR	TCR	XOFF1 (UART)/ TCR	XOFF1 (UART)/ TCR	MSR (UART)/ TCR	TCR
0x07 * S	TLR/SPR	TLR/SPR	TLR/XOFF2 (UART)	TLR/XOFF2 (UART)	TLR/SPR	TLR/SPR
0x08 * S	MDR1[2:0]-UART	MDR1[2:0]-UART	MDR1[2:0]-UART	MDR1[2:0]-UART	MDR1[2:0]-UART	MDR1[2:0]-UART
	MDR (IrDA)	MDR (IrDA)	MDR (IrDA)	MDR (IrDA)	MDR (IrDA)	MDR (IrDA)
0x09 * S	MDR2 (UART)	MDR2 (UART)	MDR2 (UART)	MDR2 (UART)	MDR2 (UART)	MDR2 (UART)
	MDR2 (IrDA)	MDR2 (IrDA)	MDR2 (IrDA)	MDR2 (IrDA)	MDR2 (IrDA)	MDR2 (IrDA)
0x0A * S	SFLSR (IrDA)	TXFLL (IrDA)	SFLSR (IrDA)	TXFLL (IrDA)	SFLSR (IrDA)	TXFLL (IrDA)
0x0B * S	RESUME (IrDA)	TXFLH (IrDA)	RESUME (IrDA)	TXFLH (IrDA)	RESUME (IrDA)	TXFLH (IrDA)
0x0C * S	SFREGL (IrDA)	RXFLL (IrDA)	SFREGL (IrDA)	RXFLL (IrDA)	SFREGL (IrDA)	RXFLL (IrDA)
0x0D * S	SFREGH (IrDA)	RXFLH (IrDA)	SFREGH (IrDA)	RXFLH (IrDA)	SFREGH (IrDA)	RXFLH (IrDA)
0x0E * S	UASR	-	UASR	-	BLR (IrDA)	BLR (IrDA)
0x0F * S	_UnicodeEncodeError_	-	-	-	ACREG (IrDA)	ACREG (IrDA)
0x10 * S	SCR	SCR	SCR	SCR	SCR	SCR
0x11 * S	SSR	-	SSR	-	SSR	-
0x12 * S	_UnicodeEncodeError_	-	-	-	EBLR (IrDA)	EBLR (IrDA)
0x14 * S	MVR	-	MVR	-	MVR	-
0x15 * S	SYSC	SYSC	SYSC	SYSC	SYSC	SYSC
0x16 * S	SYSS	-	SYSS	-	SYSS	-
0x17 * S	_UnicodeEncodeError_	-	-	-	-	-

Table 19-19. UART/IrDA Register Overview (continued)

Address Offset	Registers					
	Configuration_Mode_A		Configuration_Mode_B		Operational_Mode	
	Read	Write	Read	Write	Read	Write
0x18 * S	_UnicodeEncode Error_	-	-	-	-	-

- When REGISTER_NAME notation applies, the register is used for both UART and IrDA and has the same meaning.
- When REGISTER_NAME(UART) notation applies, the register is used for UART only.
- When REGISTER_NAME(IrDA) notation applies, the register is used for IrDA only.
- When REGISTER_NAME1(UART)/REGISTER_NAME2(IrDA) notation applies, register REGISTER_NAME1(UART) is used for UART only, and register REGISTER_NAME2(IrDA) is used for IrDA only. If REGISTER_ NAME1 equals REGISTER_NAME2, the register has a different meaning for the UART and IrDA features.

  For example: XOFF1(UART)/TCR means that XOFF1 is used for UART only, although the TCR is used for both UART and IrDA features (the use of either XOFF1 and TCR in UART mode is defined by the submodes).
- When REGISTER_NAME[m:n](X) notation is applied in the table, only register bits numbered m to n apply to feature X.

Table 19-20. UART/IrDA Register Access Mode Programming (through LCR)

Mode Name	Condition
configuration_mode_A	LCR[7] = 0x1 and LCR[7:0]! = 0xBF
configuration_mode_B	LCR[7] = 0x0 and LCR[7:0] = 0xBF
operational_mode	LCR[7] = 0x0

19.4.2.2 Register Summary

Table 19-21 through Table 19-32 list the register summaries for the configuration and operational modes of the UART.

Table 19-21. UART Register Summary for configuration_mode_A Mode Active(1)

Register Name	Type	Register Width (Bits)	Address Offset
DLL	RW	16	0x00
DLH	RW	16	0x01 * S
IIR	R	16	0x02 * S
FCR	W	16	0x02 * S
LCR	RW	16	0x03 * S
MCR	RW	16	0x04 * S
LSR	R	16	0x05 * S
See Table 19-22 and either Table 19-23 or Table 19-24.			0x06 * S
See Table 19-22 and either Table 19-23 or Table 19-24.			0x07 * S
MDR1	RW	16	0x08 * S
MDR2	RW	16	0x09 * S
SFLSR	R	16	0x0A * S
TXFLL	W	16	0x0A * S
RESUME	R	16	0x0B * S
TXFLH	W	16	0x0B * S

(1) Condition: LCR[7]=0x1 and LCR[7..0]!=0xBF

Table 19-21. UART Register Summary for configuration_mode_A Mode Active (continued)

Register Name	Type	Register Width (Bits)	Address Offset
RXFLL	W	16	0x0C * S
SFREGL	R	16	0x0C * S
RXFLH	W	16	0x0D * S
SFREGH	R	16	0x0D * S
UASR	R	16	0x0E * S
SCR	RW	16	0x10 * S
SSR	R	16	0x11 * S
MVR	R	16	0x14 * S
SYSC	RW	16	0x15 * S
SYSS	R	16	0x16 * S

Table 19-22. UART Subconfiguration_Mode_A Mode Summary

Mode Name	Condition
MSR_SPR	(EFR[4]=0x0 or MCR[6]=0x0)
TCR_TLR	EFR[4]=0x1 and MCR[6]=0x1

Table 19-23. UART Register Summary for Subconfiguration_Mode_A Mode: MSR_SPR Mode Active(1)

Register Name	Type	Register Width (Bits)	Address Offset
MSR_SPR	R	16	0x06 * S
TCR_TLR	RW	16	0x07 * S

(1) Condition: (EFR[4]=0x0 or MCR[6]=0x0)

Table 19-24. UART Register Summary for Subconfiguration_Mode_A Mode: TCR_TLR Mode Active(1)

Register Name	Type	Register Width (Bits)	Address Offset
TCR	RW	16	0x06 * S
TLR	RW	16	0x07 * S

(1) Condition: EFR[4]=0x1 and MCR[6]=0x1

Table 19-25. UART Register Summary for configuration_mode_B Mode Active(1)

Register Name	Type	Register Width (Bits)	Address Offset
DLL	RW	16	0x00
DLH	RW	16	0x01 * S
EFR	RW	16	0x02 * S
LCR	RW	16	0x03 * S
XON1/ADDR1	RW	16	0x04 * S
XON2/ADDR2	RW	16	0x05 * S
See Table 19-26 and either Table 19-27 or Table 19-28.			0x06 * S
See Table 19-26 and either Table 19-27 or Table 19-28.			0x07 * S
MDR1	RW	16	0x08 * S
MDR2	RW	16	0x09 * S
SFLSR	R	16	0x0A * S

(1) Condition: LCR[7]=0x1 and LCR[7..0]=0xBF

Table 19-25. UART Register Summary for configuration_mode_B Mode Active (continued)

Register Name	Type	Register Width (Bits)	Address Offset
TXFLL	W	16	0x0A * S
RESUME	R	16	0x0B * S
TXFLH	W	16	0x0B * S
RXFLL	W	16	0x0C * S
SFREGL	R	16	0x0C * S
RXFLH	W	16	0x0D * S
SFREGH	R	16	0x0D * S
UASR	R	16	0x0E * S
SCR	RW	16	0x10 * S
SSR	R	16	0x11 * S
MVR	R	16	0x14 * S
SYSC	RW	16	0x15 * S
SYSS	R	16	0x16 * S

Table 19-26. UART Subconfiguration_mode_B Mode Summary

Mode Name	Condition
TCR_TLR	EFR[4]=0x1 and MCR[6]=0x1
XOFF	(EFR[4]=0x0 or MCR[6]=0x0)

Table 19-27. UART Register Summary for Subconfiguration_mode_B Mode: TCR_TLR Mode Active(1)

Register Name	Type	Register Width (Bits)	Address Offset
TCR	RW	16	0x06 * S
TLR	RW	16	0x07 * S

(1) Condition: EFR[4]=0x1 and MCR[6]=0x1

Table 19-28. UART Register Summary for Subconfiguration_mode_B Mode: XOFF Mode Active(1)

Register Name	Type	Register Width (Bits)	Address Offset
XOFF1	RW	16	0x06 * S
XOFF2	RW	16	0x07 * S

(1) Condition: EFR[4]=0x0 and MCR[6]=0x0

Table 19-29. UART Register Summary for operational_mode Mode Active(1)

Register Name	Type	Register Width (Bits)	Address Offset
RHR	R	16	0x00
THR	W	16	0x00
IER	RW	16	0x01 * S
IIR	R	16	0x02 * S
FCR	W	16	0x02 * S
LCR	RW	16	0x03 * S
MCR	RW	16	0x04 * S
LSR	R	16	0x05 * S
See Table 19-30 and either Table 19-31 or Table 19-32.			0x06 * S

(1) Condition: LCR[7]=0x0

Table 19-29. UART Register Summary for operational_mode Mode Active (continued)

Register Name	Type	Register Width (Bits)	Address Offset
See Table 19-30 and either Table 19-31 or Table 19-32.			0x07 * S
MDR1	RW	16	0x08 * S
MDR2	RW	16	0x09 * S
SFLSR	R	16	0x0A * S
TXFLL	W	16	0x0A * S
RESUME	R	16	0x0B * S
TXFLH	W	16	0x0B * S
RXFLL	W	16	0x0C * S
SFREGL	R	16	0x0C * S
RXFLH	W	16	0x0D * S
SFREGH	R	16	0x0D * S
BLR	RW	16	0x0E * S
ACREG	RW	16	0x0F * S
SCR	RW	16	0x10 * S
SSR	R	16	0x11 * S
EBLR	RW	16	0x12 * S
MVR	R	16	0x14 * S
SYSC	RW	16	0x15 * S
SYSS	R	16	0x16 * S

Table 19-30. UART Suboperational_mode Mode Summary

Mode Name	Condition
MSR_SPR	(EFR[4]=0x0 or MCR[6]=0x0)
TCR_TLR	EFR[4]=0x1 and MCR[6]=0x1

Table 19-31. UART Register Summary for Suboperational_mode Mode: MSR_SPR Mode Active(1)

Register Name	Type	Register Width (Bits)	Address Offset
MSR	R	16	0x06 * S
TLRSPR	RW	16	0x07 * S

(1) Condition: EFR[4]=0x0 and MCR[6]=0x0

Table 19-32. UART Register Summary for Suboperational_mode Mode: TCR_TLR Mode Active(1)

Register Name	Type	Register Width (Bits)	Address Offset
TCR	RW	16	0x06 * S
TLR	RW	16	0x07 * S

(1) Condition: EFR[4]=0x1 and MCR[6]=0x1

19.5 UART Register Descriptions

Table 19-33 through Table 19-70 describe the UART registers.

Table 19-33. DLL

Address Offset	0x00
Instance	UART
Description	Divisor latches low
	This register, combined with DLH, stores the 14-bit divisor for generation of the baud clock in the baud rate generator. DLH stores the most significant part of the divisor. DLL stores the least significant part of the divisor.
	Note: DLL and DLH can be written to only before sleep mode is enabled (that is, before IER[4] is set).
Type	RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CLOCK_LSB							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:0	CLOCK_LSB	Stores the 8-bit LSB divisor value.	RW	0x00

Table 19-34. RHR

Address Offset	0x00
Instance	UART
Description	Receive holding register
	The receiver section consists of the receiver holding register (RHR) and the receiver shift register. The RHR is a 64-byte FIFO. The receiver shift register receives serial data from RX input. The data is converted to parallel data and moved to the RHR. If the FIFO is disabled, location 0 of the FIFO stores the single data character.
	Note: If an overflow occurs, the data in the RHR is not overwritten.
Type	R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RHR							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:0	RHR	Receive holding register	R	Undefined

Table 19-35. THR

Address Offset	0x00
Instance	UART
Description	Transmit holding register
	The transmitter section consists of the transmit holding register (THR) and the transmit shift register. The transmit holding register is a 64-byte FIFO. The LH writes data to the THR. The data is placed into the transmit shift register, where it is shifted out serially on the TX output. If the FIFO is disabled, location 0 of the FIFO stores the data.
Type	W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								THR							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:0	THR	Transmit holding register	W	Undefined

Table 19-36. IER

Address Offset	0x01 * S		
		Instance	UART
			LCR[7] = 0x1 and LCR[7:0]! = 0xBF
Description	Interrupt enable register		
Type	RW		

IER Register - UART Bit Field Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CTS_IT	RTS_IT	XOFF_IT	SLEEP_MODE	MODEM_STS_IT	LINE_STS_IT	THR_IT	RHR_IT

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7	CTS_IT	Can be written only when EFR[4] = 1 0x0: Disables the nCTS interrupt. 0x1: Enables the nCTS interrupt.	RW	0
6	RTS_IT	Can be written only when EFR[4] = 1 0x0: Disables the interrupt. 0x1: Enables the nRTS interrupt.	RW	0
5	XOFF_IT	Can be written only when EFR[4] = 1 0x0: Disables the XOFF interrupt. 0x1: Enables the XOFF interrupt.	RW	0
4	SLEEP_MODE	Can be written only when EFR[4] = 1 0x0: Disables sleep mode. 0x1: Enables sleep mode (stop baud rate clock when the module is inactive).	RW	0
3	MODEM_STS_IT	0x0: Disables the modem status register interrupt. 0x1: Enables the modem status register interrupt.	RW	0
2	LINE_STS_IT	0x0: Disables the receiver line status interrupt. 0x1: Enables the receiver line status interrupt.	RW	0
1	THR_IT	0x0: Disables the THR interrupt. 0x1: Enables the THR interrupt.	RW	0
0	RHR_IT	0x0: Disables the RHR interrupt and time-out interrupt. 0x1: Enables the RHR interrupt and time-out interrupt.	RW	0

IER Register - IrDA Bit Field Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EOF_IT	LINE_STS_IT_I	TX_STATUS_IT	STS_FIFO_ TRIG_IT	RX_OVERRUN_IT	LAST_RX_	THR_IT	RHR_IT

Bits	Field Name	Description		Type	Reset
15:8	Reserved	Reserved		R	Undefined
7	EOF_IT			RW	0
		0x0:	Disables the received EOF interrupt.		
		0x1:	Enables the received EOF interrupt.		
6	LINE_STS_IT_I			RW	0
		0x0:	Disables the receiver line status interrupt.		
		0x1:	Disables the receiver line status interrupt.		
5	TX_STATUS_IT	TX_STATUS_IT interrupt reflects two possible conditions. The MDR2[0] must be read to determine the status in the event of this interrupt.		RW	0
		0x0:	Disables the TX status interrupt.		
		0x1:	Enables the TX status interrupt.		
4	STS_FIFO_ TRIG_IT			RW	0
		0x0:	Disables status FIFO trigger level interrupt.		
		0x1:	Enables status FIFO trigger level interrupt.		
3	RX_OVERRUN_IT			RW	0
		0x0:	Disables the RX overrun interrupt.		
		0x1:	Enables the RX overrun interrupt.		
2	LAST_RX_ BYTE IT			RW	0
		0x0:	Disables the last byte of frame in RX FIFO interrupt.		
		0x1:	Enables the last byte of frame in RX FIFO interrupt.		
1	THR_IT			RW	0
		0x0:	Enables the last byte of frame in RX FIFO interrupt.		
		0x1:	Enables the THR interrupt.		
0	RHR_IT			RW	0
		0x0:	Disables the RHR interrupt and time-out interrupt.		
		0x1:	Enables the RHR interrupt and time-out interrupt.		

Table 19-37. DLH

Address Offset	0x01 * S		
		Instance	UART
Description	Divisor latches high		
	These two registers store the 14-bit divisor for generation of the baud clock in the baud rate generator. DLH stores the most significant part of the divisor. DLL stores the least significant part of the divisor.		
	Note: DLL and DLH can be written to only before sleep mode is enabled (that is, before IER[4] is set).		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CLOCK_MSB							

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Reserved	R	Undefined
5:0	CLOCK_MSB	Stores the 6-bit MSB divisor value.	RW	0x00

Table 19-38. FCR

Address Offset	0x02 * S		
		Instance	UART
Description	FIFO control register		
Type	W		

FCR Register—Normal Bit Field Details: EFR[4] = 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RX_FIFO_TRIG		Reserved		DMA_MODE	TX_FIFO_CLEAR	RX_FIFO_CLEAR	FIFO_EN

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:6	RX_FIFO_TRIG	Sets the trigger level for the RX FIFO:If SCR[7] = 0 and TLR[7:4] ≠ 0000: 00: 8 characters 01: 16 characters 10: 56 characters 11: 60 characters. RX_FIFO_TRIG is not considered. If SCR[7] = 1, RX_FIFO_TRIG is 2 LSB of the trigger level (1–63 on 6 bits) with the granularity 1.	W	0x0
5:4	Reserved	Reserved	R	Undefined
3	DMA_MODE	This register is considered if SCR[0] = 0. 0x0: DMA_MODE 0 (no DMA) 0x1: DMA_MODE 1 (UART_nDMA_REQ[0] in TX, UART_nDMA_REQ[1] in RX)	W	0
2	TX_FIFO_CLEAR	0x0: No change 0x1: Clears the transmit FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.	W	0
1	RX_FIFO_CLEAR	0x0: No change 0x1: Clears the receive FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.	W	0
0	FIFO_EN	0x0: Disables the transmit and receive FIFOs. The transmit and receive holding registers are 1-byte FIFOs. 0x1: Enables the transmit and receive FIFOs. The transmit and receive holding registers are 64-byte FIFOs.	W	0

FCR Register - Enhanced Bit Field Details: EFR[4] = 1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RX_FIFO_TRIG		TX_FIFO_TRIG		DMA_MODE	TX_FIFO_CLEAR	RX_FIFO_CLEAR	FIFO_EN

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:6	RX_FIFO_TRIG	Sets the trigger level for the RX FIFO:If SCR[7] = 0 and TLR[7:4] = 0000: If SCR[7] = 0 and TLR[7:4] = 0000, RX_FIFO_TRIG is not considered. If SCR[7] = 1, RX_FIFO_TRIG is 2 LSB of the trigger level (1_UnicodeEncodeError_63 on 6 bits) with the granularity 1. 0x0: 8 characters 0x1: 16 characters 0x2: 56 characters 0x3: 60 characters	W	0x0
5:4	TX_FIFO_TRIG	Sets the trigger level for the TX FIFO: If SCR[6] = 0 and TLR[3:0] = 0000: TX_FIFO_TRIG can be written only if EFR[4] = 1. If SCR[6] = 0 and TLR[3:0] = 0000, TX_FIFO_TRIG not considered. If SCR[6] = 1, TX_FIFO_TRIG is 2 LSB of the trigger level (1_UnicodeEncodeError_63 on 6 bits) with the granularity 1. 0x0: 8 spaces 0x1: 16 spaces 0x2: 32 spaces 0x3: 56 spaces	W	0x0
3	DMA_MODE	Can be changed only when the baud clock is not running (DLL and DLH set to 0). This register is considered if SCR[0] = 0. 0x0: DMA_MODE 0 (no DMA) 0x1: DMA_MODE 1 (UART_nDMA_REQ[0] in TX, UART_nDMA_REQ[1] in RX)	W	0
2	TX_FIFO_CLEAR	Can be changed only when the baud clock is not running (DLL and DLH set to 0). 0x0: No change 0x1: Clears the transmit FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.	W	0
1	RX_FIFO_CLEAR	Can be changed only when the baud clock is not running (DLL and DLH set to 0). 0x0: No change 0x1: Clears the receive FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.	W	0
0	FIFO_EN	Can be changed only when the baud clock is not running (DLL and DLH set to 0). 0x0: Disables the transmit and receive FIFOs. The transmit and receive holding registers are 1-byte FIFOs. 0x1: Enables the transmit and receive FIFOs. The transmit and receive holding registers are 64-byte FIFOs.	W	0

Table 19-39. IIR

Address Offset	0x02 * S		
		Instance	UART
Description	Interrupt identification register		
	The IIR is a read-only register that provides the source of the interrupt in a prioritized manner.		
	Note: An interrupt source can be flagged only if enabled in the IER.		
Type	R		

IIR Register--UART Bit Field Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FCR_MIRROR		IT_TYPE					IT_PENDING

Bits	Field Name	Description		Type	Reset
15:8	Reserved	Reserved		R	Undefined
7:6	FCR_MIRROR	Mirror the contents of FCR[0] on both bits.		R	0x0
5:1	IT_TYPE	Seven possible interrupts listed below in UART mode; other combination never occurred.		R	0x0
		0x0:	Modem interrupt. Priority = 4		
		0x1:	THR interrupt. Priority = 3		
		0x2:	RHR interrupt. Priority = 2		
		0x3:	Receiver line status error. Priority = 1		
		0x6:	RX time-out. Priority = 2		
		0x8:	Xoff/special character. Priority = 5		
		0x10:	CTS, RTS change state from active (low) to inactive (high). Priority = 6		
0	IT_PENDING			R	1
		0x0:	An interrupt is pending.		
		0x1:	No interrupt is pending.		

IIR Register--IrDA Bit Field Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EOF_IT	LINE_STS_IT	TX_STATUS_IT	STS_FIFO_IT	RX_OE_IT	RX_FIFO_LB_IT	THR_IT	RHR_IT

Bits	Field Name	Description		Type	Reset
15:8	Reserved	Reserved		R	Undefined
7	EOF_IT			R	0
		0x0:	Received EOF interrupt inactive		
		0x1:	Received EOF interrupt active		
6	LINE_STS_IT			R	0
		0x0:	Receiver line status interrupt inactive		
		0x1:	Receiver line status interrupt active		
5	TX_STATUS_IT			R	0
		0x0:	TX status interrupt inactive		

Bits	Field Name	Description		Type	Reset
		0x1:	TX status interrupt active		
4	STS_FIFO_IT			R	0
		0x0:	Status FIFO trigger level interrupt inactive		
		0x1:	Status FIFO trigger level interrupt active		
3	RX_OE_IT			R	0
		0x0:	RX overrun interrupt inactive		
		0x1:	RX overrun interrupt active		
2	RX_FIFO_LB_IT		Receive FIFO last byte interrupt	R	0
		0x0:	Last byte of frame in RX FIFO interrupt inactive		
		0x1:	Last byte of frame in RX FIFO interrupt active		
1	THR_IT			R	0
		0x0:	THR interrupt inactive		
		0x1:	THR interrupt active		
0	RHR_IT			R	0
		0x0:	RHR interrupt inactive		
		0x1:	RHR interrupt active		

Table 19-40. EFR

Address Offset 0x02 * S

Instance UART

Description Enhanced feature register

This register enables or disables enhanced features. Most enhanced functions apply only to UART modes, but EFR[4] enables write accesses to FCR[5:4], the TX trigger level, which is also used in IrDA modes.

Type RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								AUTO_CTS_	AUTO_RTS_	SPEC_CHAR	ENHANCED_	SW_FLOW_ CONTROL			

Bits	Field Name	Description		Type	Reset
15:8	Reserved	Reserved		R	Undefined
7	AUTO_CTS_ EN	Auto-CTS enable bit (UART mode only)		RW	0
		0x0:	Normal operation		
		0x1:	Auto-CTS flow control is enabled; that is, transmission is halted when the nCTS pin is high (inactive).		
6	AUTO_RTS_ EN	Auto-RTS enable bit (UART mode only)		RW	0
		0x0:	Normal operation		
		0x1:	Auto-RTS flow control is enabled; that is, the nRTS pin goes high (inactive) when the receiver FIFO HALT trigger level, TCR[3:0], is reached and goes low (active) when the receiver FIFO RESTORE transmission trigger level is reached.		
5	SPEC_CHAR	(UART mode only) Special character detect		RW	0
		0x0:	Normal operation		
		0x1:	Special character detect enable. Received data is compared with XOFF2 data. If a match occurs, the received data is transferred to RX FIFO and IIR bit 4 is set to 1 to indicate that a special character has been detected.		
4	ENHANCED_ EN	Enhanced functions write enable bit		RW	0
		0x0:	Disables writing to IER bits 4_UnicodeEncodeError_7, FCR bits 4_UnicodeEncodeError_5, and MCR bits 5_UnicodeEncodeError_7.		

Bits	Field Name	Description	Type	Reset
		0x1: Enables writing to IER bits 4_UnicodeEncodeError_7, FCR bits 4_UnicodeEncodeError_5, and MCR bits 5_UnicodeEncodeError_7.		
3:0	SW_FLOW_CONTROL	These bits must be kept at 0x0.	RW	0x0

Bit3	Bit2	Bit1	Bit0	Tx,Rx software flow controls
0	0X	X		No transmit flow control
1	0X	X		Transmit XON1, XOFF1
0	1X	X		Transmit XON2, XOFF2
1	1X	X		Transmit XON1, XON2: XOFF1, XOFF2(1)
X	X0	0		No receive flow control
X	X1	0		Receiver compares XON1, XOFF1
X	X0	1		Receiver compares XON2, XOFF2
X	X1	1		Receiver compares XON1, XON2: XOFF1, XOFF2†

(1) In these cases, the XON1 and XON2 characters or the XOFF1 and XOFF2 characters must be transmitted/received sequentially with XON1/XOFF1 followed by XON2/XOFF2.

Table 19-41. LCR

Address Offset 0x03 * S

Instance UART

Description Line control register

LCR[6:0] define parameters of the transmission and reception for UART mode. LCR[7] is used to put the module in operational mode or Configuration_Mode_A/B.

Type

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DIV_EN	BREAK_EN	PARITY_TYPE2	PARITY_TYPE1	PARITY_EN	NB_STOP	CHAR_LENGTH	

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7	DIV_EN		RW	0
		0x0: Operational mode		
		0x1: Divisor latch enable; puts the module in Configuration_Mode_A/B. Allows access to DLL, DLH, and other registers (refer to register mapping).		
		Configuration_Mode_B: LCR[7:0] = 0xBF. Otherwise, Configuration_Mode_A		
6	BREAK_EN	Break control bit (UART mode only) **Note:** When LCR[6] is set to 1, the TX line is forced to 0 and remains in this state as long as LCR[6] = 1.	RW	0
		0x0: Normal operating condition		
		0x1: Forces the transmitter output to go low to alert the communication terminal. TX line is forced to 0 and remains in this state as long as BREAK_EN = 1.		
5	PARITY_TYPE2	Selects the forced parity format (if LCR[3] = 1). UART mode only. If LCR[5] = 1 and LCR[4] = 0, the parity bit is forced to 1 in the transmitted and received data. If LCR[5] = 1 and LCR[4] = 1, the parity bit is forced to 0 in the transmitted and received data.	RW	0
4	PARITY_TYPE1	UART mode only	RW	0
		0x0: Odd parity is generated (if LCR[3] = 1).		
		0x1: Even parity is generated (if LCR[3] = 1).		

Bits	Field Name	Description	Type	Reset
3	PARITY_EN	UART mode only 0x0: No parity 0x1: A parity bit is generated during transmission, and the receiver checks for received parity.	RW	0
2	NB_STOP	Number of stop bits (UART mode only) 0x0: 1 stop bit (word length = 5, 6, 7, 8) 0x1: 1.5 stop bits (word length = 5)1 _UnicodeEncodeError_ 2 stop bits (word length = 6, 7, 8)	RW	0
1:0	CHAR_LENGTH	Specifies the word length to be transmitted or received. UART mode only. 0x0: 5 bits 0x1: 6 bits 0x2: 7 bits 0x3: 8 bits	RW	0x0

Table 19-42. MCR

Address Offset	0x04 * S		
		Instance	UART
Description	Modem control register (UART mode only)		
	MCR[3:0] controls the interface with the modem, data set, or peripheral device that is emulating the modem.		
Type	RW		

MCR Register—Enhanced Bit Field Details: EFR[4] = 1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TCR_TLR	XON_EN	LOOPBACK_EN	CD_STS_CH	RI_STS_CH	RTS	DTR

Bits	Field Name	Description	Type	Reset
15:7	Reserved	Reserved	R	Undefined
6	TCR_TLR	0x0: No action 0x1: Enables access to the TCR and TLR registers.	RW	0
5	XON_EN	0x0: Disables XON—any function. 0x1: Enables XON—any function.	RW	0
4	LOOPBACK_EN	0x0: Normal operating mode 0x1: Enables local loopback mode (internal). In this mode, the MCR[3:0] signals are looped back into MSR[7:4]. The transmit output is looped back to the receive input internally.	RW	0
3	CD_STS_CH	0x0: In loopback, forces DCD input high and IRQ outputs to inactive state. 0x1: In loopback, forces DCD input low and IRQ outputs to inactive state.	RW	0
2	RI_STS_CH	0x0: In loopback, forces nRI input inactive (high). 0x1: In loopback, forces nRI input active (low).	RW	0

Bits	Field Name	Description	Type	Reset
1	RTS	In loopback, controls MSR[4]. If auto-RTS is enabled, the nRTS output is controlled by hardware flow control. 0x0: Forces nRTS output to inactive (high). 0x1: Forces nRTS output to active (low).	RW	0
0	DTR	0x0: Forces DTR output to inactive (high). 0x1: Forces DTR output to active (low).	RW	0

MCR Register--Normal Bit Field Details: EFR[4] = 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TCR_TLR	XON_EN	LOOPBACK_EN	CD_STS_CH	RI_STS_CH	RTS	Reserved

Bits	Field Name	Description	Type	Reset
15:7	Reserved	Reserved	R	Undefined
6	TCR_TLR	0x0: No action 0x1: Enables access to the TCR and TLR registers.	R	0
5	XON_EN	0x0: Disables XON—any function. 0x1: Enables XON—any function,	RW	0
4	LOOPBACK_EN	0x0: Normal operating mode 0x1: Enables local loopback mode (internal). In this mode the MCR[3:0] signals are looped back into MSR[7:4]. The transmit output is looped back to the receive input internally.	RW	0
3	CD_STS_CH	0x0: In loopback, forces IRQ outputs to inactive state. 0x1: In loopback, forces IRQ outputs to inactive state.	RW	0
2	RI_STS_CH	0x0: In loopback, forces nRI input inactive (high). 0x1: In loopback, forces nRI input active (low)	RW	0
1	RTS	In loopback, controls MSR[4]. If auto-RTS is enabled, the nRTS output is controlled by hardware flow control. 0x0: Forces nRTS output to inactive (high). 0x1: Forces nRTS output to active (low).	RW	0
0	Reserved	Reserved	R	Undefined

Table 19-43. XON1/ADDR1

Address Offset	0x04 * S		
		Instance	UART
Description	UART mode: XON1 character; IrDA mode: ADDR1 address		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								XON_WORD1							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:0	XON_WORD1	Stores the 8-bit XON1 character in UART modes and ADDR1 address 1 for IrDA modes.	RW	0x00

Table 19-44. LSR

Address Offset	0x05 * S		
		Instance	UART
Description	Line status register		
Type	R		

LSR Register—UART Bit Field Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RX_FIFO_STS	TX_SR_E	TX_FIFO_E	RX_BI	RX_FE	RX_PE	RX_OE	RX_FIFO_E

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7	RX_FIFO_STS		R	0
		0x0: Normal operation		
		0x1: At least one parity error, framing error, or break indication in the RX FIFO. Bit 7 is cleared when no errors are present in the RX FIFO.		
6	TX_SR_E		R	1
		0x0: Transmitter hold (TX FIFO) and shift registers are not empty.		
		0x1: Transmitter hold (TX FIFO) and shift registers are empty.		
5	TX_FIFO_E		R	1
		0x0: Transmit hold register (TX FIFO) is not empty.		
		0x1: Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.		
4	RX_BI		R	0
		0x0: No break condition		
		0x1: A break was detected while the data being read from the RX FIFO was being received (that is, RX input was low for one character + 1 bit time frame).		
3	RX_FE		R	0
		0x0: No framing error in data read from RX FIFO		
		0x1: Framing error occurred in data read from RX FIFO (received data did not have a valid stop bit).		
2	RX_PE		R	0
		0x0: No parity error in data read from RX FIFO		
		0x1: Parity error in data read from RX FIFO		
1	RX_OE		R	0
		0x0: No overrun error		

Bits	Field Name	Description	Type	Reset
		0x1: Overrun error has occurred. Set when the character held in the receive shift register is not transferred to the RX FIFO. This case can occur only when receive FIFO is full.		
0	RX_FIFO_E		R	0
		0x0: No data in the receive FIFO		
		0x1: At least one data character in the RX FIFO		

LSR Register—IrDA Bit Field Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								THR_EMPTY	STS_FIFO_FULL	RX_LB	FTL	ABORT	CRC	STS_FIFO_E	RX_FIFO_E

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7	THR_EMPTY		R	1
		0x0: Transmit holding register (TX FIFO) is not empty.		
		0x1: Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.		
6	STS_FIFO_FULL		R	0
		0x0: Status FIFO not full		
		0x1: Status FIFO full		
5	RX_LB	Receive last byte	R	0
		0x0: The RX FIFO (RHR) does not contain the last byte of the frame to be read.		
		0x1: The RX FIFO (RHR) contains the last byte of the frame to be read. This bit is set only when the last byte of a frame is available to be read. It is used to determine the frame boundary. It is cleared on a single read of the LSR register.		
4	FTL	Frame too long	R	0
		0x0: No frame-too-long error in frame		
		0x1: Frame-too-long error in the frame at the top of the STATUS FIFO (next character to be read). This bit is set to 1 when a frame exceeding the maximum length (set by RXFLH and RXFLL registers) is received. When this error is detected, current frame reception is terminated. Reception is stopped until the next START flag is detected.		
3	ABORT		R	0
		0x0: No abort pattern error in frame		
		0x1: Abort pattern received. SIR and MIR: abort pattern. FIR: Illegal symbol.		
2	CRC		R	0
		0x0: No CRC error in frame		
		0x1: CRC error in the frame at the top of the STATUS FIFO (next character to be read).		
1	STS_FIFO_E		R	1
		0x0: Status FIFO not empty		
		0x1: Status FIFO empty		
0	RX_FIFO_E		R	1

Bits	Field Name	Description	Type	Reset
		0x0: No data in the receive FIFO		
		0x1: At least one data character in the RX FIFO		

Table 19-45. XON2/ADDR2

Address Offset 0x05 * S

Instance UART

Description

Type RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								XON_WORD2							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:0	XON_WORD2	Stores the 8-bit XON2 character in UART modes and ADDR2 address 2 for IrDA modes.	RW	0x00

Table 19-46. XOFF1

Address Offset 0x06 * S

Instance UART

Description UART mode XOFF1 character

Type RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								XOFF_WORD1							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:0	XOFF_WORD1	Stores the 8-bit XOFF1 character used in UART modes.	RW	0x00

Table 19-47. TCR

Address Offset 0x06 * S

Instance UART

Description Transmission control register

This register stores the receive FIFO threshold levels to start/stop transmission during hardware flow control.

Type RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RX_FIFO_TRIG_ START				RX_FIFO_TRIG_ HALT			

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:4	RX_FIFO_TRIG_ START	RX FIFO trigger level to RESTORE transmission (0 _UnicodeEncodeError_ 60)	RW	0x0
3:0	RX_FIFO_TRIG_ HALT	RX FIFO trigger level to HALT transmission (0 _UnicodeEncodeError_ 60)	RW	0xF

Table 19-48. MSR

Address Offset 0x06 * S

Instance UART

Description Modem status register (UART mode only)

Table 19-48. MSR (continued)

This register provides information about the current state of the control lines from the modem, data set, or peripheral device to the LH. It also indicates when a control input from the modem changes state.

Type R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								NCD_STS	NRI_STS	NDSR_STS	NCTS_STS	DCD_STS	RI_STS	DSR_STS	CTS_STS

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7	NCD_STS	In loopback mode, it is equivalent to MCR[3].	R	-
6	NRI_STS	This bit is the complement of the nRI input. In loopback mode, it is equivalent to MCR[2].	R	-
5	NDSR_STS	In loopback mode, it is equivalent to MCR[0].	R	-
4	NCTS_STS	This bit is the complement of the nCTS input. In loopback mode, it is equivalent to MCR[1].	R	-
3	DCD_STS	0x1: Indicates that DCD input (or MCR[3] in loopback) has changed state. Cleared on a read.	R	0
2	RI_STS	Indicates that nRI input (or MCR[2] in loopback) has changed state from low to high. Cleared on a read.	R	0
1	DSR_STS	0x1: Indicates that DSR input (or MCR[0] in loopback) has changed state. Cleared on a read.	R	0
0	CTS_STS	0x1: Indicates that nCTS input (or MCR[1] in loopback) has changed state. Cleared on a read.	R	0

Table 19-49. SPR

Address Offset 0x07 * S

Instance UART

Description Scratchpad register

This read/write register does not control the module. It is intended as a scratchpad register to be used by the programmer to hold temporary data.

Type RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SPR_WORD							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:0	SPR_WORD	Scratchpad register	RW	0x00

Table 19-50. XOFF2

Address Offset 0x07 * S

Instance UART

Description UART mode XOFF2 character

Type RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								XOFF_WORD2							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:0	XOFF_WORD2	Stores the 8-bit XOFF2 character in used in UART modes.	RW	0x00

Table 19-51. TLR

Address Offset 0x07 * S

Instance UART

Description Trigger-level register

This register stores the programmable transmit and receive FIFO trigger levels used for DMA and IRQ generation.

Type RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RX_FIFO_TRIG_				TX_FIFO_TRIG_			

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:4	RX_FIFO_TRIG_DMA	Receive FIFO trigger level	RW	0x00
3:0	TX_FIFO_TRIG_DMA	Transmit FIFO trigger level	RW	0x00

Table 19-52. MDR1

Address Offset 0x08 * S

Instance UART

Description Mode-definition register 1

The mode of operation can be programmed by writing to MDR1[2:0]; therefore, the MDR1 must be programmed on startup after configuration of the configuration registers (DLL, DLH, LCR). The value of MDR1[2:0] must not be changed during normal operation.

Type RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FRAME_END_	SIP_MODE	SCT	SET_TXIR	IR_SLEEP	MODE_SELECT		

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7	FRAME_END_MODE		RW	0
		0x0: Frame-length method		
		0x1: Set EOT bit method.		
6	SIP_MODE	MIR/FIR modes only. IrDA only.	RW	0
		0x0: Manual SIP mode: SIP is generated with the control of ACREG[3].		
		0x1: Automatic SIP mode: SIP is generated after each transmission.		
5	SCT	Store and control the transmission. IrDA only.	RW	0

Bits	Field Name	Description	Type	Reset
		0x0: Starts the infrared transmission when a value is written to THR. 0x1: Starts the Infrared transmission with the control of ACREG[2]. **Note**: Before starting a transmission, there must be no reception ongoing.		
4	SET_TXIR	Used to configure the infrared transceiver. IrDA only. 0x0: No action 0x1: IRTX pin output is forced high.	RW	0
3	IR_SLEEP	0x0: IrDA sleep mode disabled 0x1: IrDA sleep mode enabled	RW	0
2:0	MODE_SELECT	UART/IrDA mode selection 0x0: UART 16x mode 0x1: SIR mode 0x2: UART 16x auto-baud 0x3: UART 13x mode 0x4: MIR mode 0x5: FIR mode 0x6: Reserved 0x7: Disable (default state)	RW	0x7

Table 19-53. MDR2

Address Offset 0x09 * S

Instance UART

Description Mode-definition register 2

MDR2[0] describes the status of the interrupt in IIR[5]. The IRTX_UNDERRUN bit must be read after an IIR[5] TX_STATUS_IT interrupt occurs. Bits [2:1] of this register set the trigger level for the frame status FIFO (8 entries) and must be programmed before the mode is programmed in MDR1[2:0].

Type RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													STS_FIFO_TRIG		IRTX_UNDERRUN

Bits	Field Name	Description	Type	Reset
15:3	Reserved	Reserved	R	Undefined
2:1	STS_FIFO_TRIG	IrDA mode only Frame status FIFO threshold select: 0x0: 1 entry 0x1: 4 entries 0x2: 7 entries 0x3: 8 entries	RW	0x0
0	IRTX_UNDERRUN	0x0: The last bit of the frame was transmitted successfully without error.	R	0

Bits	Field Name	Description	Type	Reset
		0x1: An underrun has occurred. The last bit of the frame was transmitted but with an underrun error. The bit is reset to 0 when the RESUME register is read.		

Table 19-54. TXFLL

Address Offset 0x0A * S

Instance UART

Description Transmit frame-length register low

IrDA mode only

The registers TXFLL and TXFLH hold the 13-bit transmit frame length (expressed in bytes). TXFLL holds the least-significant bits (LSBs) and TXFLH holds the most-significant bits (MSBs). The frame-length value is used if the frame-length method of frame closing is used.

Type W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TXFLL							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:0	TXFLL	LSB register used to specify the frame length	W	0x00

Table 19-55. SFLSR

Address Offset 0x0A * S

Instance UART

Description Status FIFO line status register

IrDA mode only

Reading this register effectively reads frame status information from the status FIFO (this register does not physically exist). Reading this register increments the status FIFO read pointer (SFREGL and SFREGH must be read first).

Type R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											OE_ERROR	FTL_ERROR	ABORT_DETECT	CRC_ERROR	Reserved

Bits	Field Name	Description	Type	Reset
15:5	Reserved	Reserved	R	Undefined
4	OE_ERROR		R	-
		0x1: Overrun error in RX FIFO when frame at top of RX FIFO was received. Top of RX FIFO = next frame to be read from RX FIFO.		
3	FTL_ERROR	Frame-too-long error	R	-
		0x1: Frame-length too long error in frame at top of RX FIFO		
2	ABORT_DETECT		R	-
		0x1: Abort pattern detected in frame at top of RX FIFO		
1	CRC_ERROR	0x1: CRC error in frame at top of RX FIFO	R	-
0	Reserved	Reserved	R	0

Table 19-56. RESUME

Address Offset	0x0B * S
Instance	UART
Description	IR-IrDA mode only This register is used to clear internal flags, which halt transmission/reception when an underrun/overrun error occurs. Reading this register resumes the halted operation. This register does not physically exist and reads always as 0x00.
Type	R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RESUME							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:0	RESUME	Dummy read to restart the TX or RX	R	0x00

Table 19-57. TXFLH

Address Offset	0x0B * S
Instance	UART
Description	Transmit frame-length register low IrDA mode only The registers TXFLL and TXFLH hold the 13-bit transmit frame length (expressed in bytes). TXFLL holds the LSBs and TXFLH holds the MSBs. The frame-length value is used if the frame-length method of frame closing is used.
Type	W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											TXFLH				

Bits	Field Name	Description	Type	Reset
15:5	Reserved	Reserved	R	Undefined
4:0	TXFLH	MSB register used to specify the frame length	W	0x00

Table 19-58. RXFLL

Address Offset	0x0C * S
Instance	UART
Description	Received frame-length register low The registers RXFLL and RXFLH hold the 12-bit receive maximum frame length. RXFLL holds the LSBs, and RXFLH holds the MSBs. If the intended maximum receive frame length is n bytes, program RXFLL and RXFLH to be n + 3 in SIR or MIR modes and n + 6 in FIR mode (+3 and +6 are caused by frame format with CRC and stop flag; there are 2 bytes associated with the FIR stop flag).
Type	W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RXFLL							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:0	RXFLL	LSB register used to specify the frame length in reception	W	0x00

Table 19-59. SFREGL

Address Offset	0x0C * S
Instance	UART
Description	Status FIFO register low IrDA mode only. The frame lengths of received frames are written into the status FIFO. This information can be read by reading the SFREGL and SFREGH registers (these registers do not physically exist). The LSBs are read from SFREGL, and the MSBs are read from SFREGH. Reading these registers does not alter the status FIFO read pointer. These registers must be read before the pointer is incremented by reading the SFLSR.
Type	R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SFREGL							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:0	SFREGL	LSB part of the frame length	W	0x–

Table 19-60. SFREGH

Address Offset	0x0D * S
Instance	UART
Description	Status FIFO register high IrDA mode only. The frame lengths of received frames are written into the status FIFO. This information can be read by reading the SFREGL and SFREGH registers (these registers do not physically exist). The LSBs are read from SFREGL, and the MSBs are read from SFREGH. Reading these registers does not alter the status FIFO read pointer. These registers must be read before the pointer is incremented by reading the SFLSR.
Type	R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												SFREGH			

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	R	Undefined
3:0	SFREGH	MSB register used to specify the frame length in reception	W	0x0

Table 19-61. RXFLH

Address Offset	0x0D * S
Instance	UART
Description	Received frame-length register high IrDA mode only. The registers RXFLL and RXFLH hold the 12-bit receive maximum frame length. RXFLL holds the LSBs, and RXFLH holds the MSBs. If the intended maximum receive frame length is n bytes, program RXFLL and RXFLH to be n + 3 in SIR or MIR modes and n + 6 in FIR mode (+3 and +6 are caused by frame format with CRC and stop flag; there are 2 bytes associated with the FIR stop flag).
Type	W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												RXFLH			

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	R	Undefined
3:0	RXFLH	MSB register used to specify the frame length in reception	W	0x0

Table 19-62. BLR

Address Offset	0x0E * S
	Instance UART
Description	BOF control register
	IrDA mode only.
	BLR[6] is used to select whether 0xC0 or 0xFF start patterns are to be used, when multiple start flags are required in SIR mode. If only one start flag is required, this is always 0xC0. If n start flags are required, either (n_UnicodeEncodeError_1) 0xC0 or (n_UnicodeEncodeError_1) 0xFF flags are sent, followed by a single 0xC0 flag (immediately preceding the first data byte).
Type	RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								STS_FIFO_RESET	XBOF_TYPE	Reserved					

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7	STS_FIFO_RESET	Status FIFO reset. This bit is self-clearing.	RW	0
6	XBOF_TYPE	SIR xBOF select 0xFF 0xC0	RW	1
5:0	Reserved	Read returns 0.	R	0X0

Table 19-63. UASR

Address Offset	0x0E * S
	Instance UART
Description	UART autobauding status register
	UART autobauding mode only.
	This status register returns the speed, the number of bits by characters, and the type of the parity in UART autobauding mode.
	In autobauding mode, the input frequency of the UART modem must be fixed to 48 MHz. Any other module clock frequency results in an incorrect baud rate
Type	R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PARITY_TYPE		BIT_BY_CHAR	SPEED				

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:6	PARITY_TYPE		R	0x0
		0x0: No parity identified		
		0x1: Parity space		
		0x2: Even parity		
		0x3: Odd parity		

Bits	Field Name	Description	Type	Reset
5	BIT_BY_CHAR		R	0
		0x0: 7-bit character identified		
		0x1: 8-bit character identified		
4:0	SPEED	Used to report the speed identified	R	0x00
		0x0: No speed identified		
		0x1: 115,200 baud		
		0x2: 57,600 baud		
		0x3: 38,400 baud		
		0x4: 28,800 baud		
		0x5: 19,200 baud		
		0x6: 14,400 baud		
		0x7: 9600 baud		
		0x8: 4800 baud		
		0x9: 2400 baud		
		0xA: 1200 baud		

Table 19-64. ACREG

Address Offset 0x0F * S

Instance UART

Description Auxiliary control register

IrDA mode only

Type RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PULSE_TYPE	SD_MOD	DIS_IR_RX	DIS_TX_UNDER	SEND_SIP	SCTX_EN	ABORT_EN	EOT_EN

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7	PULSE_TYPE	SIR pulse-width select:	RW	0
		0x0: 3/16 of baud-rate pulse width		
		0x1: 1.6 µs		
6	SD_MOD	Primary output used to configure transceivers. Connected to the SD/MODE input pin of IrDA transceivers.	RW	0
		0x0: SD pin is set to high.		
		0x1: SD pin is set to low.		
5	DIS_IR_RX		RW	0
		0x0: Normal operation (RX input automatically disabled during transmit but enabled outside of transmit operation).		
		0x1: Disables RX input (permanent state; independent of transmit).		
4	DIS_TX_UNDER RUN RW 0		RW	0
		0x0: Long stop bits cannot be transmitted. TX underrun is enabled.		
		0x1: Long stop bits can be transmitted. TX underrun is disabled.		
3	SEND_SIP	MIR/FIR modes only. Send serial infrared interaction pulse (SIP).	RW	0

Bits	Field Name	Description	Type	Reset
		If this bit is set during an MIR/FIR transmission, the SIP is sent at the end of it. This bit is automatically cleared at the end of the SIP transmission. 0x0: No action 0x1: Send SIP pulse.		
2	SCTX_EN	Store and control TX start. When MDR1[5] = 1 and the LH writes 1 to this bit, the TX state-machine starts frame transmission. This bit is self-clearing.	RW	0
1	ABORT_EN	Frame abort. The LH can intentionally abort transmission of a frame by writing 1 to this bit. Neither the end flag nor the CRC bits are appended to the frame.	RW	0
0	EOT_EN	EOT (end of transmission) bit. The LH writes 1 to this bit just before it writes the last byte to the TX FIFO in the set-EOT bit frame-closing method. This bit is automatically cleared when the LH writes to the THR (TX FIFO).	RW	0

Table 19-65. SCR

Address Offset 0x10 * S

Instance UART

Description Supplementary control register

Type RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RX_TRIG_ GRANU1	TX_TRIG_ GRANU1	DSR_IT	RX_CTS_DSR_ WAKE_UP_ ENABLE	TX_EMPTY_ CTL I	DMA_MODE_2	DMA_MODE_ CTL	

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7	RX_TRIG_ GRANU1	0x0: Disables the granularity of 1 for TRIGGER RX level. 0x1: Enables the granularity of 1 for TRIGGER RX level.	RW	0
6	TX_TRIG_ GRANU1	0x0: Disables the granularity of 1 for TRIGGER TX level. 0x1: Enables the granularity of 1 for trigger TX level.	RW	0
5	DSR_IT	0x0: Disables DSR* interrupt. 0x1: Enables DSR* interrupt.”	RW	0
4	RX_CTS_DSR_ WAKE_UP_ ENABLE	Enable/disable wake-up interrupt. 0x0: Disables the wake-up interrupt and clears SSR[1]. 0x1: Waits for falling edge on pins RX, CTS, or DSR to generate an interrupt	R	0
3	TX_EMPTY_ CTL I		RW	0

Bits	Field Name	Description		Type	Reset
		0x0:	Normal mode for THR interrupt (see Table 19-13 for details of UART mode interrupts).		
		0x1:	The THR interrupt is generated when TX FIFO and TX shift registers are empty.		
2	DMA_MODE_2		Used to specify the DMA mode valid if SCR[0] = 1	RW	0x0
		0x0:	DMA mode 0 (no DMA)		
		0x1:	DMA mode 1 (UART_nDMA_REQ[0] in TX, UART_nDMA_REQ[1] in RX)		
		0x2:	DMA mode 2 (UART_nDMA_REQ[0] in RX)		
		0x3:	DMA mode 3 (UART_nDMA_REQ[0] in TX)		
1	DMA_MODE_ CTL			RW	0
		0x0:	The DMA_MODE is set with FCR[3].		
		0x1:	The DMA_MODE is set with SCR[2:1].		

Table 19-66. SSR

Address Offset 0x11 * S

Instance UART

Description Supplementary status register

Type R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														RX_CTS_DSR_ WAKE_UP_STS	TX_FIFO_FULL

Bits	Field Name	Description		Type	Reset
15:2	Reserved	Reserved		R	Undefined
1	RX_CTS_DSR_ WAKE_UP_STS	0x0:	No falling edge event on CTS and DSR	R	0
		0x1:	A falling edge occurred on CTS or DSR.		
0	TX_FIFO_FULL			R	0
		0x0:	TX FIFO is not full.		
		0x1:	TX FIFO is full.		

Table 19-67. EBLR

Address Offset 0x12 * S

Instance UART

Description BOF length register

IR-IrDA mode only

In IR-IrDA SIR operation, this register specifies the number of BOF + xBOFs to transmit. The value set in this register must take into account the BOF character; therefore, to send only one BOF with no XBOF, this register must be set to 1. To send one BOF with N XBOF, this register must be set to N+1. The value 0 sends 1 BOF plus 255 XBOF.

In IR-IrDA MIR mode, this register specifies the number of additional start flags (MIR protocol mandates a minimum of 2 start flags).

Type RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EBLR							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:0	EBLR	IR-IrDA mode: This register can define up to 176 xBOFs, the maximum required by IrDA specification. 0x0: Feature disabled 0x1: Generate RX_STOP interrupt after receiving 1 zero bit. 0xFF: Generate RX_STOP interrupt after receiving 255 zero bits.	RW	0x00

Table 19-68. MVR

Address Offset	0x14 * S
Instance	UART
Description	Module version register The reset value is fixed by hardware and corresponds to the RTL revision of this module. A reset has no effect on the value returned _UnicodeEncodeError_ UART/IrDA SIR only module is revision 1.x (WMU_012_1 specification). _UnicodeEncodeError_ UART/IrDA with SIR, MIR, and FIR support is revision 2.x (WMU_012_2 specification). _UnicodeEncodeError_ UART/IrDA with SIR, MIR, and FIR support is revision 3.x (this specification). For example: MVR = 0x30 => version 3.0, MVR = 0x38 => version 3.8.
Type	R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MAJOR_REV				MINOR_REV			

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	Undefined
7:4	MAJOR_REV	Major revision number of the module	R	See (1)
3:0	MINOR_REV	Minor revision number of the module	R	See (1)

(1) TI Internal data.

Table 19-69. SYSC

Address Offset	0x15 * S
Instance	UART
Description	System configuration register The autoidle bit controls a power-saving technique to reduce the logic power consumption of the OCP interface. When the feature is enabled, the clock is gated off until an OCP command for this device is detected. When the software reset bit is set high, it causes a full device reset.
Type	RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											IDLEMODE		ENAWAKEUP	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
15:5	Reserved	Reserved	R	Undefined
4:3	IDLEMODE	Power-management req/ack control 0x0: Force-idle. An idle request is acknowledged unconditionally.	RW	0X0

Bits	Field Name	Description	Type	Reset
		0x1: No-idle. An idle request is never acknowledged.		
		0x2: Smart-idle. Acknowledgement of an idle request is given based on the internal activity of the module.		
		0x3: Reserved		
2	ENAWAKEUP	Wake-up feature control	RW	0
		0x0: Wakeup is disabled.		
		0x1: Wakeup is enabled.		
1	SOFTRESET	Software reset. Set this bit to 1 to trigger a module reset. This bit is automatically reset by the hardware. Reads return 0.	RW	0
		0x0: Normal mode		
		0x1: The module is reset.		
0	AUTOIDLE	Internal OCP clock-gating strategy	RW	0
		0x0: Clock is running.		
		0x1: Automatic interface clock-gating strategy is applied based on the interface activity.		

Table 19-70. SYSS

Address Offset	0x16 * S		
		Instance	UART
Description	System status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RESETDONE

Bits	Field Name	Description	Type	Reset
15:1	Reserved	Reserved	R	Undefined
0	RESETDONE	Internal reset monitoring	R	0
		0x0: Internal module reset is ongoing.		
		0x1: Reset complete		

I2C Module

This chapter describes the I²C modules of the Locosto device.

Topic **Page**

20.1 I²C Overview

The I²C peripheral provides an interface between a local host (LH), such as a microprocessor unit (MPU) or digital signal processor (DSP) and any I²C-bus-compatible device that connects through the I²C serial bus. External components attached to the I²C bus can serially transmit/receive up to 8-bit data to/from the LH device through the two-wire I²C interface.

The I²C controllers of the Locosto device do not support the multimaster mode. Each I²C device (including the I²C controllers) is recognized by a unique address and can operate as either transmitter or receiver, according to the function of the device. In addition to being a transmitter or receiver, a device connected to the I²C bus can also be considered as master or slave when performing data transfers. Note that a master device is the device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. During the transfer, any device addressed by this master is considered a slave.

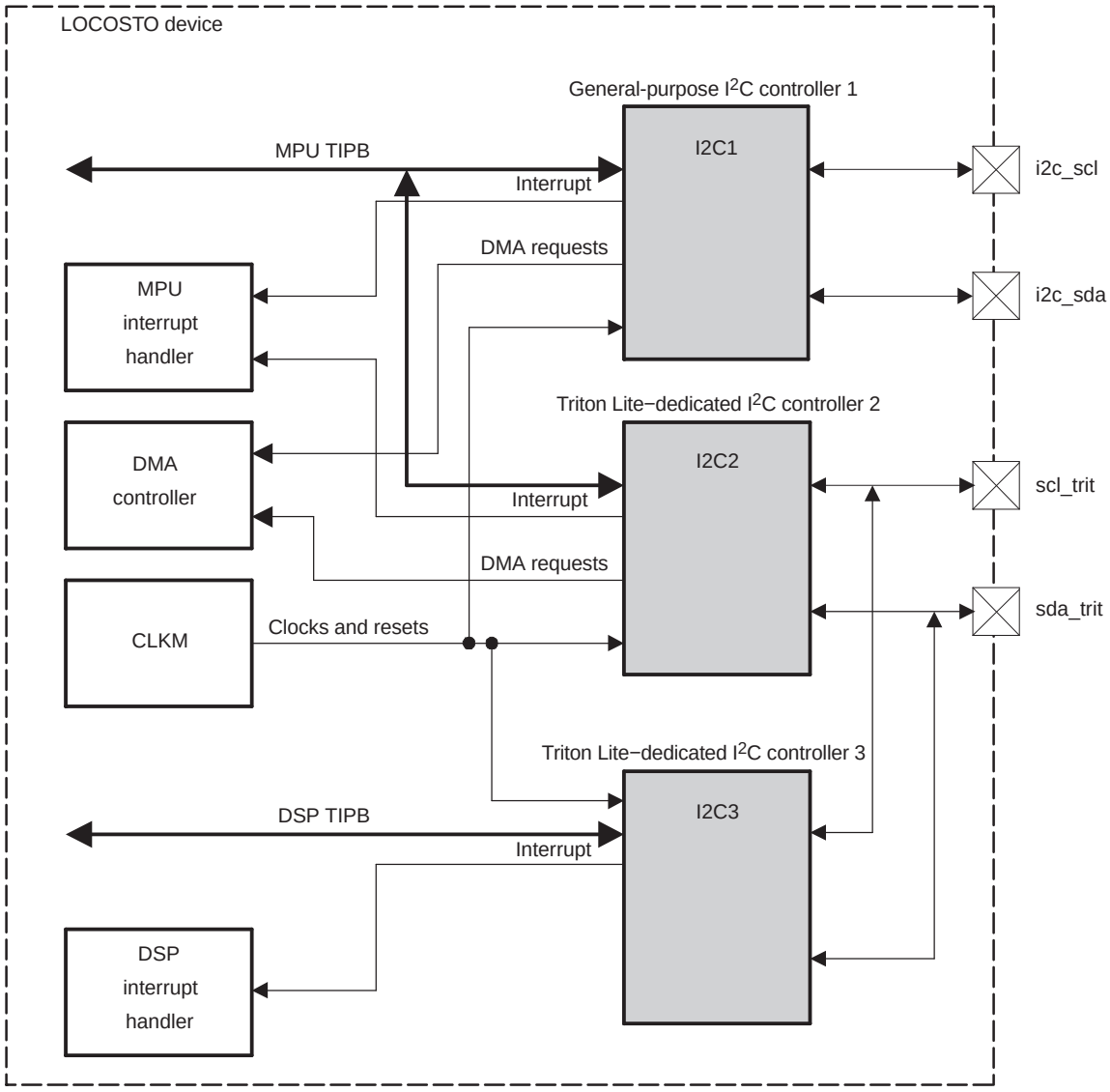
This I²C peripheral supports any slave or master I²C-compatible device. The Locosto device contains three separate I²C controllers:

- Two I²C controllers are accessed only from the MPU (I2C1 and I2C2 instances)
- One I²C controller is accessed only from the DSP (I2C3 instance)

  The I2C1 instance is intended for general purposes, whereas the I2C2 and I2C3 instances share the same I²C bus and they are mainly dedicated for supporting Triton Lite ABB chip. However, additional slave-only I²C devices can be attached to the shared-I²C bus controlled by the I2C2 or I2C3 controllers.

Figure 20-1 shows the three I²C controllers implemented in the Locosto device.

Figure 20-1. I²C System Overview



The main features of the I²C controller are:

- Compliant with Philips I²C specification version 2.1 (refer to I²C Bus Specification Version 2.1)
- Supports both standard mode (up to 100K bits/s) and fast mode (up to 400K bits/s)
- 7-bit and 10-bit device addressing modes
- General call
- Start/restart/stop
- Master transmitter/slave receiver mode
- Master receiver/slave transmitter mode
- Combined master transmit/receive and receive/transmit mode (refer to I²C Bus Specification Version 2.1)
- Transmit/receive burst buffer of two 16-bit words
- Module enable/disable capability
- Programmable clock generation
- 16-bit wide access to maximize bus throughput
- Designed for low power
- Two DMA channels per I²C controller (for I2C1 and I2C2 instances only)

- Wide interrupt capability

Note: Features Not Supported

The present I²C modules do not support:

- High-speed (HS) mode for transfer up to 3.4M bits/s
- C-bus compatibility mode
- Multimaster transmitter / multimaster receiver modes

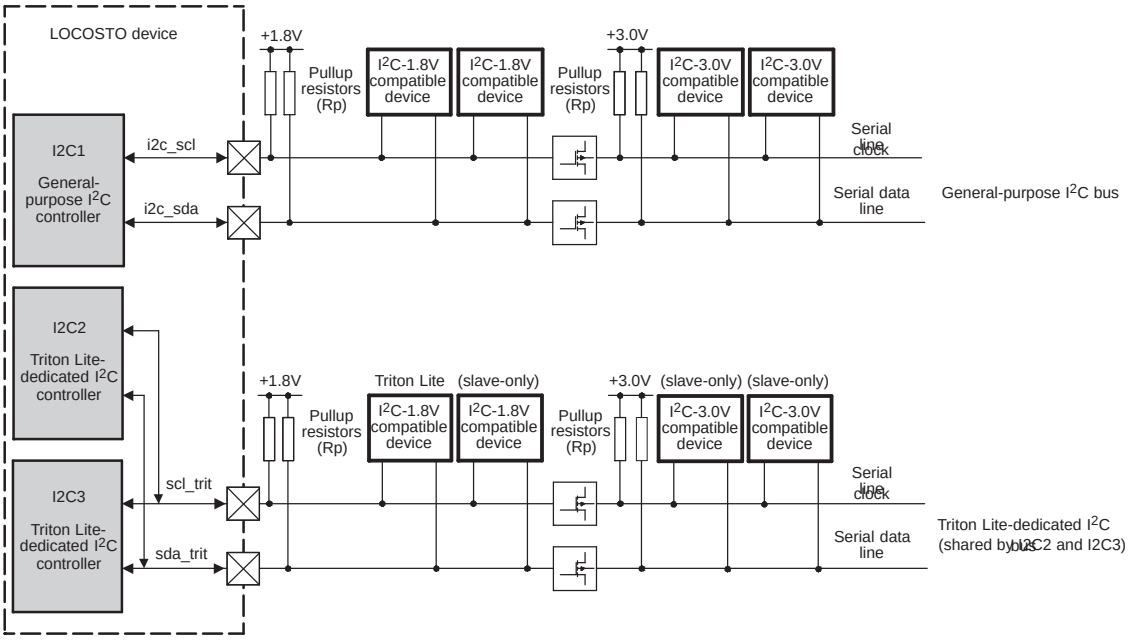
20.2 I²C Environment

Data is communicated between an I²C controller and I²C devices connected to it with the I²C bus by the serial data line (i2c_sda or trit_sda) and the serial clock line (i2c_scl or trit_scl).

These two wires carry information between the Locosto device and other devices connected to the I²C bus. Both serial data and serial clock lines are bidirectional pins. They must be connected to a positive supply voltage by a pullup resistor. When the bus is free, both pins are high. The driver of these two pins has an open drain to perform the required wired-AND function.

Figure 20-2 shows typical connections between the I²C controllers of the Locosto device and I²C devices.

Figure 20-2. Locosto I²C Controllers and I²C Devices Typical Connections



20.2.1 I²C Functional Interfaces

Table 20-1 shows the pins associated with the I²C interface.

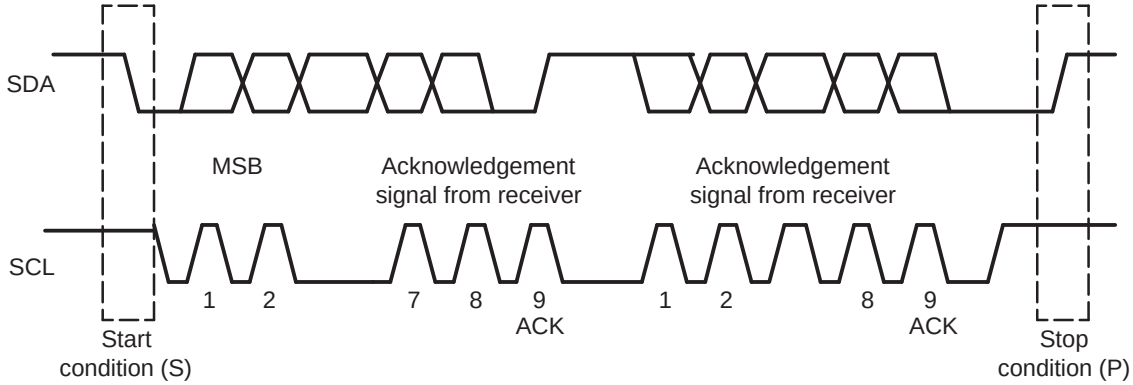
Table 20-1. Signal Pads

Name	Type	Reset Value	Description
i2c_scl	In/Out (OD)	Input	Serial clock line of I2C1 instance. Open-drain output buffer. Requires external pullup resistor (Rp).
i2c_sda	In/Out (OD)	Input	Serial data line of I2C1 instance. Open-drain output buffer. Requires external pullup resistor (Rp).
scl_trit	In/Out (OD)	Input	Serial clock line shared by I2C2 and I2C3 instances. Open-drain output buffer. Requires external pullup resistor (Rp).
sda_trit	In/Out (OD)	Input	Serial data line shared by I2C2 and I2C3 instances. Open-drain output buffer. Requires external pullup resistor (Rp).

20.2.1.1 Serial Data Formats

Each I²C controller operates in 16-bit word data format (byte-write access is supported for the last access). Each byte transmitted on the SDA line is 8 bits long. The number of bytes that can be transmitted or received is unrestricted. The data is transferred with the most significant bit (MSB) first. Each byte is followed by an acknowledge bit from the I²C module, if the module is in receiver mode. Figure 20-3 shows a typical I²C communication format. The I²C controller supports endianness.

Figure 20-3. I²C Data Transfer

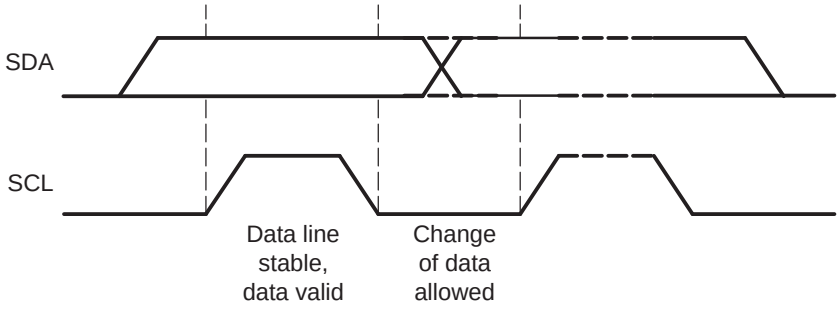


20.2.1.2 Data Validity

The data on the SDA line (i2c_sda for the I2C1 module, and sda_trit for the I2C2 and I2C3 modules) must be stable during the clock-high period. The high and low data states can only change when the clock signal on the SCL line (i2c_scl for the I2C1 module, and scl_trit for the I2C2 and I2C3 modules) is low.

Figure 20-4 shows an example of data validity requirements.

Figure 20-4. Bit Transfer on the I²C Bus



20.2.1.3 START and STOP Conditions

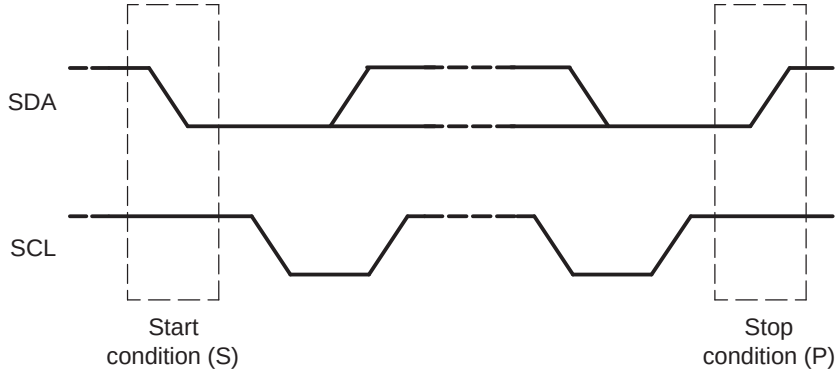
The I²C module generates START and STOP conditions when it is configured as a master.

- The START condition is a high-to-low transition on the SDA line while SCL is high.
- The STOP condition is a low-to-high transition on the SDA line while SCL is high.

The bus is considered to be busy after the START condition (I2Cn.I2C_STAT[12] BB = 1) and free after the STOP condition (I2Cn.I2C_STAT[12] BB=0).

Figure 20-5 shows the START and STOP condition waveforms.

Figure 20-5. START and STOP Conditions

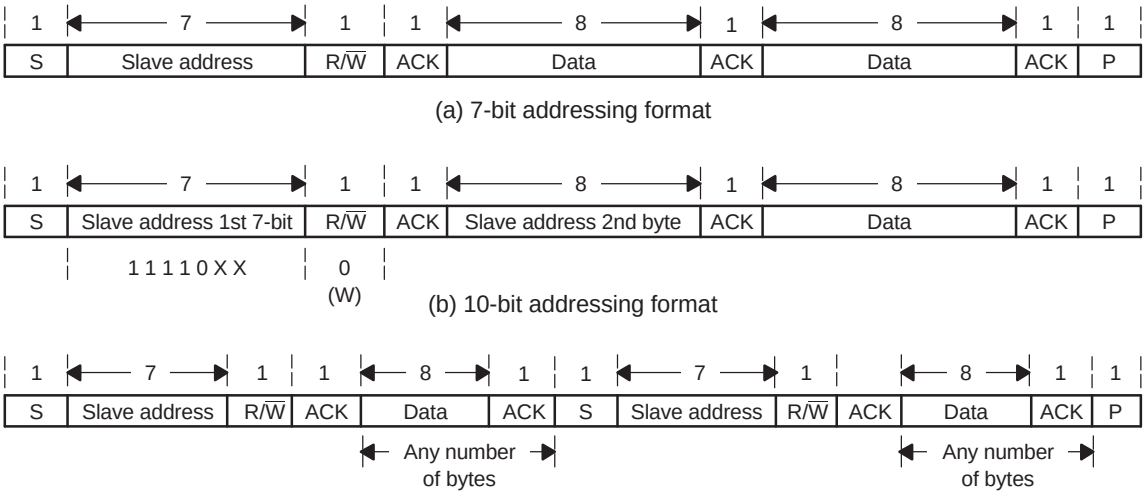


20.2.1.4 Addressing

The I²C module supports two data formats, as shown in Figure 20-6:

- 7-bit/10-bit addressing format
- 7-bit/10-bit addressing format with repeated start condition

Figure 20-6. I²C Data Transfer Formats



Between a START (S) and a STOP (P) condition, the words always consist of 8 bits. In the acknowledge mode, an extra acknowledgement bit is inserted after each byte.

The first word after a START condition is reserved for addressing:

- In the 7-bit addressing format, the address is composed of 7 MSB slave address bits and 1 LSB R/W bit.
- In the 10-bit addressing format, the address is composed of a first byte with the pattern 11110XXY, where XX represents the two MSB bits of the 10-bit address and Y is the R/W bit:
  - If the R/W bit is zero, the next byte contains the last eight bits of the slave address.
  - If the R/W bit is one, the next byte contains data transmitted from slave to master.

The least significant bit R/W of the address byte indicates the direction of transmission of the following data bytes. If R/W is 0, the master writes data into the selected slave; if it is 1, the master reads data out of the slave.

20.2.1.4.1 Master Transmitter

In this mode, data assembled in one of the previously described data formats is shifted on the SDA in synch with the self-generated clock pulses on the SCL. The clock pulses are inhibited and SCL held low when the intervention of the processor is required after a byte has been transmitted.

20.2.1.4.2 Master Receiver

This mode can only be entered from the master transmitter mode. Regardless of the address format (see Figure 20-6 (a), (b), and (c)), the master receiver is entered after the slave address byte and the R/W bit have been transmitted, if the R/W bit is high. Serial data bits received on bus line SDA are shifted in synch with the self-generated clock pulses on SCL. The clock pulses are inhibited and SCL is held low when the intervention of the processor is required after a byte has been transmitted. At the end of a transfer, the master receiver generates the stop condition.

20.2.1.4.3 Slave Transmitter

This mode can only be entered from the slave receiver mode. Regardless of the address format (see Figure 20-6 (a), (b), and (c)), the slave transmitter mode is entered if the slave address byte is the same as its own address and bit R/W has been transmitted, if the R/W bit is high. The slave transmitter shifts the serial data out on data line SDA in synch with the clock pulses that are generated by the master device. The slave transmitter does not generate the clock, but it can hold clock line SCL low while intervention of the LH is required.

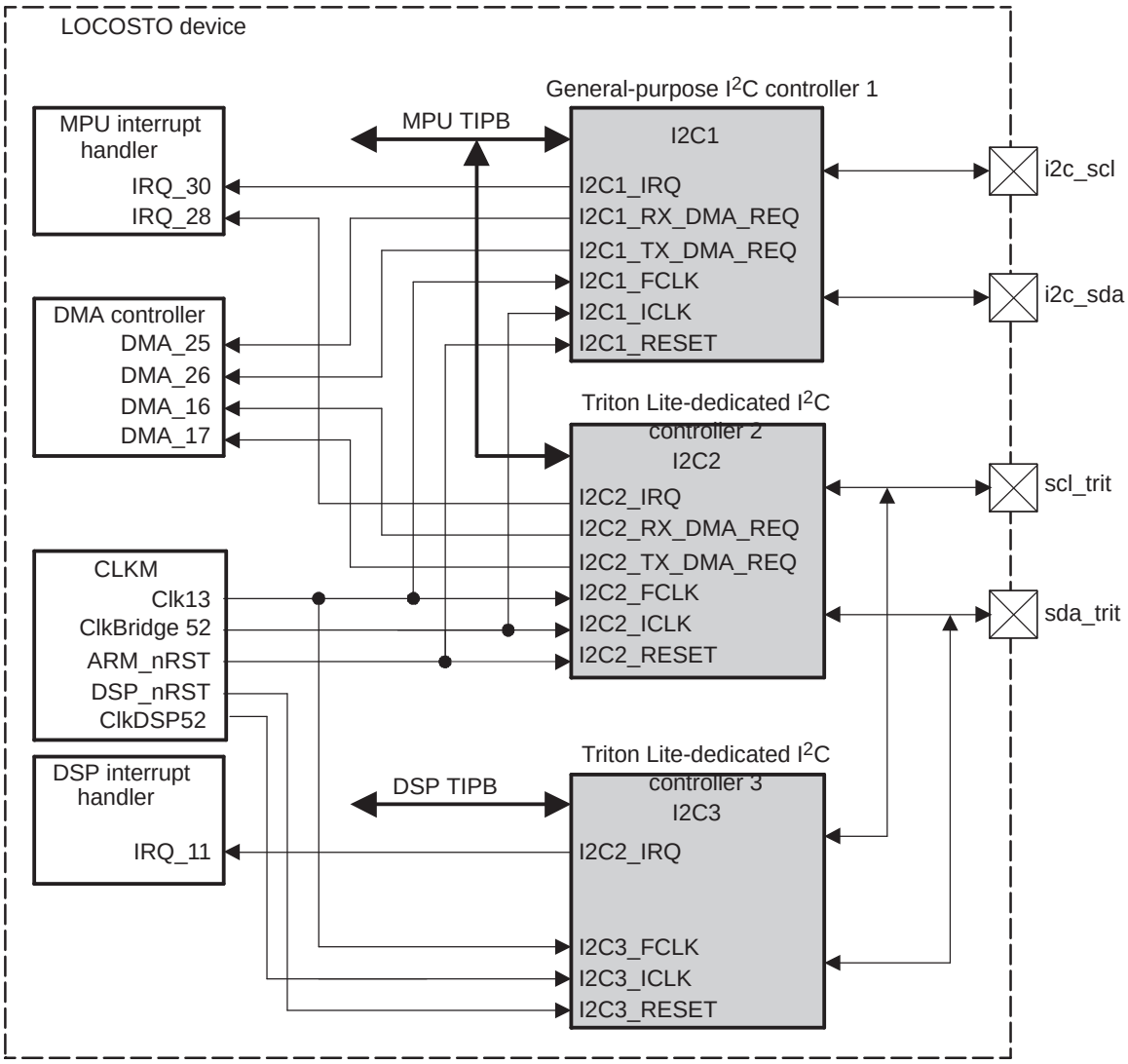
20.2.1.4.4 Slave Receiver

In this mode, serial data bits received on the bus line SDA (i2c_sda for the I2C1 module, sda_trit for the I2C2 and I2C3 modules) are shifted-in in synchronization with the clock pulses on SCL (i2c_scl for the I2C1 module, scl_trit for the I2C2 and I2C3 modules) that are generated by the master device. The slave receiver does not generate the clock, but it can hold clock line SCL low while intervention of the LH is required following the reception of a byte.

20.3 I²C Integration

Figure 20-7 illustrates the integration of the three I²C modules in the Locosto device.

Figure 20-7. I²C Module Integration



20.3.1 Clocking, Reset, and Power Management Scheme

20.3.1.1 I²C Clocks

Each I2Cn module (where n = 1, 2, or 3) is clocked with an independent functional clock (I2Cn_FCLK) and an interface clock (I2Cn_ICLK).

- The functional clock (I2Cn_FCLK) is a fixed 13-MHz clock provided by the CLKM module (Clk13 output signal). It is processed by the I²C prescaler block to produce the internal sampling clock (I2Cn_INTERNAL_CLK). The prescaler consists of I2Cn.I2C_PSC[7:0] PSC field, which is used for dividing down the functional clock to obtain an internal sampling clock with a frequency value of I2Cn_FCLK/(value of the I2Cn.I2C_PSC[7:0] PSC field + 1). The I²C module is operated with an internal ~13 MHz clock.

- The interface clock (I2Cn_ICLK) is used to trigger access to the I²C configuration interface (the I2C1 and I2C2 modules are accessed by the MPU only, whereas the I2C3 module is accessed from the DSP only). Its source is the CLKM module (ClkBridge52 signal for the I2C1 and I2C2 modules, and the ClkDSP52 signal for the I2C3 module).

Table 20-2 lists each I²C functional clock.

Table 20-2. I²C Clocks

	Frequency	Name	Source
I2C1 Functional clock	13 MHz	I2C1_FCLK	Clk13
I2C2 Functional clock	13 MHz	I2C2_FCLK	Clk13
I2C3 Functional clock	13 MHz	I2C3_FCLK	Clk13
I2C1 Interface clock	See *Chapter 6, Power, Reset, and Clock Management*	I2C1_ICLK	ClkBridge52
I2C2 Interface clock	See *Chapter 6, Power, Reset, and Clock Management*	I2C2_ICLK	ClkBridge52
I2C3 Interface clock	See *Chapter 6, Power, Reset, and Clock Management*	I2C3_ICLK	ClkDSP52
I2C1 Internal clock	13/(PSC+1)MHz (1)	I2C1_INTERNAL_CLK	Prescaler block
I2C2 Internal clock	13/(PSC+1)MHz (1)	I2C2_INTERNAL_CLK	Prescaler block
I2C3 Internal clock	13/(PSC+1)MHz (1)	I2C3_INTERNAL_CLK	Prescaler block

(1) PSC: I2Cn.I2C_PSC[7:0] PSC field, where n = 1, 2, or 3

20.3.1.2 Module Resets

20.3.1.2.1 Hardware Reset

The I2C1 and I2C2 modules are reset by hardware with an activation of the ARM_nRST signal, whereas the I2C3 module is reset by hardware with an activation of the DSP_nRST signal. These hardware reset signals have a global reset action on these modules.

20.3.1.2.2 Software Reset

The I2Cn.I2C_SYSC[1] SRST bit functions as a software reset for the I2Cn module (where n = 1, 2 or 3). The software reset is triggered by writing a 1 to this bit. The software reset action on the module is equivalent to a hardware reset.

Note that the I2Cn.I2C_CON[15] I2C_EN bit can hold the I2Cn module (where n = 1, 2, or 3) in reset after the device reset has been released. When the system bus reset (hardware reset) is removed, I2Cn.I2C_CON[15] I2C_EN bit (where n = 1, 2, or 3) remains set to 0. The functional part of the I2Cn module is held in a reset state while the I2Cn.I2C_CON[15] I2C_EN bit (where n = 1, 2, or 3) is 0 and all configuration registers can be accessed.

Note: When the I2Cn module is held in reset by clearing the I2Cn.I2C_CON[15] I2C_EN bit to 0, the RX and TX FIFOs are cleared, the I2Cn.I2C_IE, I2Cn.I2C_BUF, I2Cn.I2C_CNT, I2Cn.I2C_CON, I2Cn.I2C_OA, I2Cn.I2C_SA, I2Cn.I2C_PSC, I2Cn.I2C_SCLL and I2Cn.I2C_SCLH registers are not reset and keep their programmed values.

20.3.2 Hardware Requests

The I2C1 and I2C2 modules issue requests to the local host with interrupts and DMA events, whereas the I2C3 module issue requests to the local host with interrupts only.

20.3.2.1 DMA Requests

Each of the I2C1 and I2C2 modules generates two DMA requests, read (I2C1_RX_DMA_REQ and I2C2_RX_DMA_REQ) and write (I2C1_TX_DMA_REQ and I2C2_TX_DMA_REQ), which can be used by the DMA controller to synchronously read received data from the I2Cn.I2C_DATA (with n = 1, 2) and write transmitted data to the I2Cn.I2C_DATA register (with n = 1, 2). The DMA read and write requests are generated, but the bits I2C_STAT[RRDY] and I2C_STAT[XRDY] are not set.

The DMA request signals (I2Cn_TX_DMA_REQ and I2Cn_RX_DMA_REQ, where n = 1, 2) are activated for every new 16-bit word to be read or written in the FIFOs.

Table 20-3 lists all the DMA events for the I2C1 module and I2C2 modules.

Table 20-3. I2C1 and I2C2 DMA Events

Attributes	Values	Name	Mapping	Comments
I2C1				
DMA request	2	I2C1_TX_DMA_REQ	DMA_26	Destination: DMA controller
		I2C1_RX_DMA_REQ	DMA_25	
I2C2				
DMA request	2	I2C2_TX_DMA_REQ	DMA_17	Destination: DMA controller
		I2C2_RX_DMA_REQ	DMA_16	

20.3.2.2 Interrupt Requests

Each of the three I²C modules owns one interrupt line (see Table 20-4). The I²C module generates five types of interrupts: no-acknowledge, general call, registers ready-for-access, receive, and transmit. When the interrupt signal is activated, the local host must read the I2Cn.I2C_STAT register (where n = 1, 2, or 3) to define the type of the interrupt, process the request, and write a 1 into this register to clear the interrupt flag.

These five interrupts are accompanied by five interrupt masks and flags defined in the I2Cn.I2C_IE and I2Cn.I2C_STAT registers, respectively (where n = 1, 2, or 3).

- No-ACKnowledge interrupt (NACK), generated when the master I²C does not receive an acknowledgement from the receiver.
- General call interrupt (GC), generated when the device detects the general call address. The general call address is composed of all zeros (applies to both 7-bit and 10-bit addressing modes).
- Registers ready-for-access interrupt (ARDY), generated by the I²C when the previously programmed address, data, and command have been performed and the status bits have been updated. This interrupt is used to let the LH know that the I²C registers are ready for access.
- Receive interrupt/status (RRDY), generated when there is received data in the I2Cn.I2C_DATA register ready to be read by the LH. The LH can poll this bit to read the received data from the I2Cn.I2C_DATA register.
- Transmit interrupt/status (XRDY), generated when the LH needs to put more data in the I2Cn.I2C_DATA register after the transmitted data has been shifted out on the SDA pin. The LH can poll this bit to write the next transmitted data into the I2Cn.I2C_DATA register.

Table 20-4. I²C Interrupts

Attributes	Values	Name	Mapping	Comments
I2C1				
Interrupt request	1	I2C1_IRQ	IRQ_30	Destination is MPU interrupt handler. See (1)

(1) For more information about interrupts, see Chapter 12, *Interrupt Handlers*.

Table 20-4. I²C Interrupts (continued)

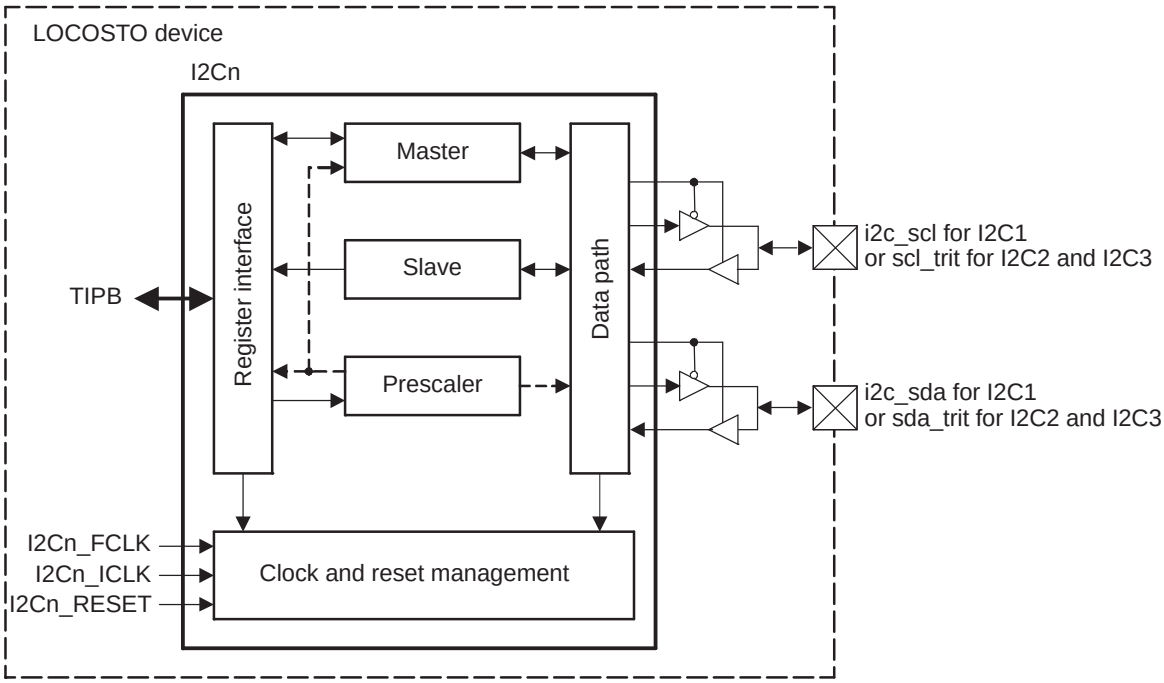
Attributes	Values	Name	Mapping	Comments
I2C1				
I2C2				
Interrupt request	1	I2C2_IRQ	IRQ_28	Destination is MPU interrupt handler. See (1)
I2C3				
Interrupt request	1	I2C3_IRQ	IRQ_11	Destination is DSP interrupt handler. See (1)

20.4 Functional Description

20.4.1 Block diagram

Figure 20-8 shows the modules included within the I²C module.

Figure 20-8. I²C Block Diagram



20.4.1.1 Transmit Mode

The transmit mode is available for master or slave (master or slave mode configurable with the I2Cn.I2C_CON[10] MST bit). This mode is configured by setting the I2Cn.I2C_CON[9] TRX bit to 1.

The MPU fills the data to transmit via the I²C controller I2C1 or I2C2 by writing it into the I2Cn.I2C_DATA register (where n = 1 or 2). The transmitter can write new data into this register while the I2Cn.I2C_STAT[4] XRDY is set to 1.

In a same way, the DSP fills the data to transmit via the I²C controller I2C3 by writing it into the I2C3.I2C_DATA register. The transmitter can write new data into this register while the I2C3.I2C_STAT[4] XRDY is set to 1.

- A master transmitter requests new data if the I2Cn.I2C_CNT register value is different from 0.
- A slave transmitter requests new data if a read is performed by the external master.

Note: Dummy Transmitter Values

The transmitter requests two bytes to be written in the I2Cn.I2C_DATA register, even if only a single byte is needed. In this case, the other byte must be filled with a dummy 0x00 value that is transmitted over the I²C line.

20.4.1.2 Receive Mode

This mode is available for master or slave. It is configured by setting the I2Cn.I2C_CON[9] TRX bit to 0. The MPU or the DSP can read new data from the I2Cn.I2C_DATA register when the I2Cn.I2C_STAT[3] RRDY bit is set to 1. Each time this bit is set to 1, the I²C module generates an interrupt to the MPU or the DSP if the interrupt is enabled (the I2Cn I2C_IE[3] RRDY_IE bit).

Note: Clearing the I2Cn.I2C_STAT[3] RRDY bit

In interrupt mode, the MPU or the DSP must read the received data in the I2Cn.I2C_DATA register, and then clear the I2Cn.I2C_STAT[3] RRDY bit in the interrupt routine in order to receive a new interrupt.

The I2Cn.I2C_STAT[3] RRDY is not set to 1 if the DMA receive mode is enabled (the I2Cn.I2C_BUF[15] RDMA_EN, where n = 1 or 2). Instead, a DMA receive request is generated to the DMA.

Note: The DMA receive mode is not available for the I2C3 instance.

20.4.1.3 Data Format and FIFO

The FIFO size is 2×16 bits. Bytes within a word (16 bits) are stored and read in little endian format (I2Cn.I2C_CON[14] BE = 0) or big endian format (I2Cn.I2C_CON[14] BE = 1)

When read, the I2Cn.I2C_DATA register contains the data packet (1 or 2 bytes). This register must be accessed 16-bit wide by the MPU or the DSP. When reading an odd number of received bytes, the upper byte (with I2Cn.I2C_CON[14] BE = 0) or the lower byte (with I2Cn.I2C_CON[14] BE = 1) of the last access always reads as 0x00. The MPU or the DSP must check the I2Cn.I2C_STAT[15] SBD status bit in order to flush this null byte.

When written, this register contains the byte(s) value(s) to transmit over the I^2C data line (1or 2 bytes). This register must be accessed 16-bit-wide, except for the last byte in case of an odd number of bytes to transmit. The last byte of the data packet can be written using a byte write access or a 16-bit-wide access. In the 16-bit-wide access, only the relevant byte is transmitted by the module based on the byte counter (I2Cn.I2C_CNT register). This feature can be useful for DMA access supported by the I2C1 and I2C2 modules, which supports only one word size per channel.

In SYSTEST loop back mode (I2Cn.I2C_SYSTEST[13:12] TMODE field = 0b11), this register is also the entry/receive point for the data.

Note: A read access with an empty buffer returns the previous read data value. A write access with a full buffer is ignored. In both events, the FIFO pointers are not updated.

20.4.1.4 Clocking

The three I^2C modules use a functional clock (I2Cn_FCLK) provided by the CLKM module. If required, this clock is divided by a prescaler value (I2Cn.I2C_PSC register) to obtain a sampling clock (I2Cn_INTERNAL_CLK). This sampling clock is used to create the SCL external clock by configuring the I2Cn.I2C_SCLH and I2Cn.I2C_SCLL registers. See Figure 20-9.

The I2Cn_INTERNAL_CLK clock must be calculated by taking into account the t_{HIGH} and t_{LOW} values:

- I2Cn_INTERNAL_CLK time period =(t_{LOW} or t_{HIGH}) /(I2Cn.I2C_SCLL or I2Cn.I2C_SCLH + 7) (when I2Cn.I2C_PSC = 0)
- I2Cn_INTERNAL_CLK = (t_{LOW} or t_{HIGH})/(I2Cn.I2C_SCLL or I2Cn.I2C_SCLH + 6) (when I2Cn.I2C_PSC = 1)
- I2Cn_INTERNAL_CLK time period = (t_{LOW} or t_{HIGH})/(I2Cn.I2C_SCLL or I2Cn.I2C_SCLH + 5) (when I2Cn.I2C_PSC > 1)

Figure 20-9. I²C Clocking

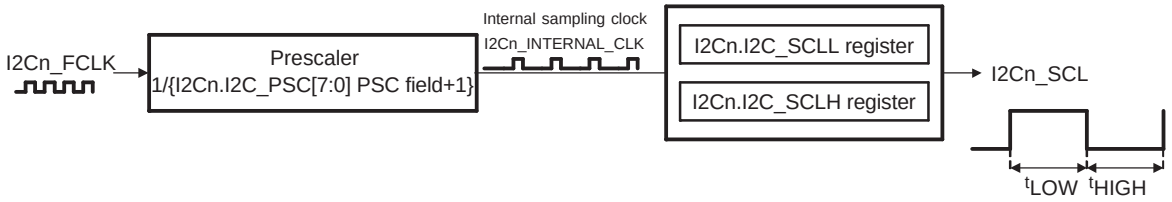


Table 20-5 gives the minimum values to program in the I2Cn.I2C_SCLL and I2Cn.I2C_SCLH registers for the maximum I²C bit rates in standard and fast modes.

Table 20-5. Register Values for Maximum I²C Bit Rates in Standard and Fast Modes

	Standard mode			Fast mode			
I2Cn_FCLK frequency (MHz)	13			13			
I2Cn.I2C_PSC[7:0] PSC field value	**0**	**1**		**0**		**1**	
I2Cn_INTERNAL_CLK frequency (MHz)	13	6.5		13		6.5	
I2Cn.I2C_SCLL[7:0] SCLL field minimum value	**58**	**27**	**26**	**9**	**10**	**2**	**3**
I2Cn.I2C_SCLH[7:0] SCLH field minimum value	**58**	**26**	**27**	**10**	**9**	**3**	**2**
Maximum noise filter period (ns) (See (1))	77	154		77		154	
Maximum bit rate (K bps)	100			400			

(1) noise filter period = 1/(I2Cn_INTERNAL_CLK frequency)

20.5 Programming Model

20.5.1 Main Program

20.5.1.1 Module Configuration Before Enabling the Module

1. Configure the I^2C prescaler for the I2Cn module (the I2Cn_FCLK is provided at 13 MHz) to obtain an I^2C internal clock (I2Cn_INTERNAL_CLK) of 13MHz or 6.5 MHz (see Table 20-5). (Such that I2Cn.I2C_PSC[7:0] PSC field value + 1 = x, where x = I2Cn_FCLK / y and y is the desired I2C_INTERNAL_CLK frequency value).
2. Program the I2Cn.I2C_SCLL[7:0] SCLL and I2Cn.I2Cn.I2C_SCLH[7:0] SCLH for the I2Cn module to obtain a baud rate of 100K bps or 400K bps. These values depend on the frequency of the I^2C internal clock (I2Cn_INTERNAL_CLK).
3. In slave mode, configure the I^2C own address of the I2Cn module (I2Cn.I2C_OA[9:0] OA field).
4. Take the I2Cn module out of reset (I2Cn.I2C_CON[15] I2C_EN bit = 1).

Note: The I2C1 and I2C2 modules are accessed only from the MPU, whereas the I2C3 module is accessed only from the DSP.

20.5.1.2 Initialization Procedure

1. Configure the I2Cn.I2C_CON register (endianness, master/slave mode, transmit/receive mode) of the I2Cn module.
2. If using interrupt for transmit/receive data, enable interrupt masks by setting the desired bits in the I2Cn.I2C_IE register of the I2Cn module to 1.
3. If using DMA for transmit/receive data, enable the DMA request generation by setting the desired bits in the I2Cn.I2C_BUF register (set the I2Cn.I2C_BUF[15] RDMA_EN bit to 1 to enable the DMA request generation for receive data, and set the I2Cn.I2C_BUF[7] XDMA_EN to enable the DMA request generation for transmit data) and program the DMA controller.

Note: The DMA mode is not supported by the I2C3 module.

20.5.1.3 Configure Slave Address and Data Counter Registers

In master mode, configure the slave address in the I2Cn.I2C_SA register and the number of bytes associated with the transfer in the I2Cn.I2C_CNT register, for the I2Cn module.

20.5.1.4 Initiate a Transfer

Poll the I2Cn.I2C_STAT[12] BB (bus busy) bit . If it is cleared to 0 (bus not busy), configure START/STOP (I2Cn.I2C_CON[0] STT and I2Cn.I2C_CON[1] STP bits) condition to initiate a transfer, for the I2Cn module.

20.5.1.5 Poll Receive Data

Poll the I2Cn.I2C_STAT[3] RRDY bit; use the RRDY interrupt, or use the DMA to read the receive data in the data receive register (I2Cn.I2C_DATA register), for the I2Cn module.

Note: The DMA mode is not supported by the I2C3 module.

20.5.1.6 Poll Transmit Data

Poll the I2Cn.I2C_STAT[4] XRDY bit; use the XRDY interrupt, or use the DMA to write data into the data transmit register (I2CnI2C_DATA), for the I2Cn module.

Note: The DMA mode is not supported by the I2C3 module.

20.5.2 Interrupt Subroutine Sequence

1. Test for no-acknowledge and resolve accordingly.
2. Test for register access ready and resolve accordingly.
3. Test for receive data and resolve accordingly.
4. Test for transmit data and resolve accordingly.

20.5.3 Flow Diagrams

Figure 20-10 through Figure 20-18 should help users program the different modes of the I^2C controller.

Figure 20-10. Setup Procedure

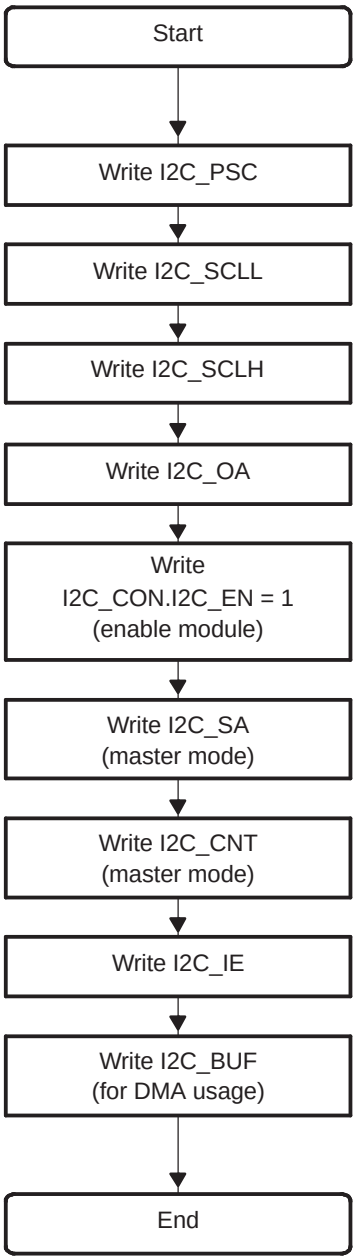
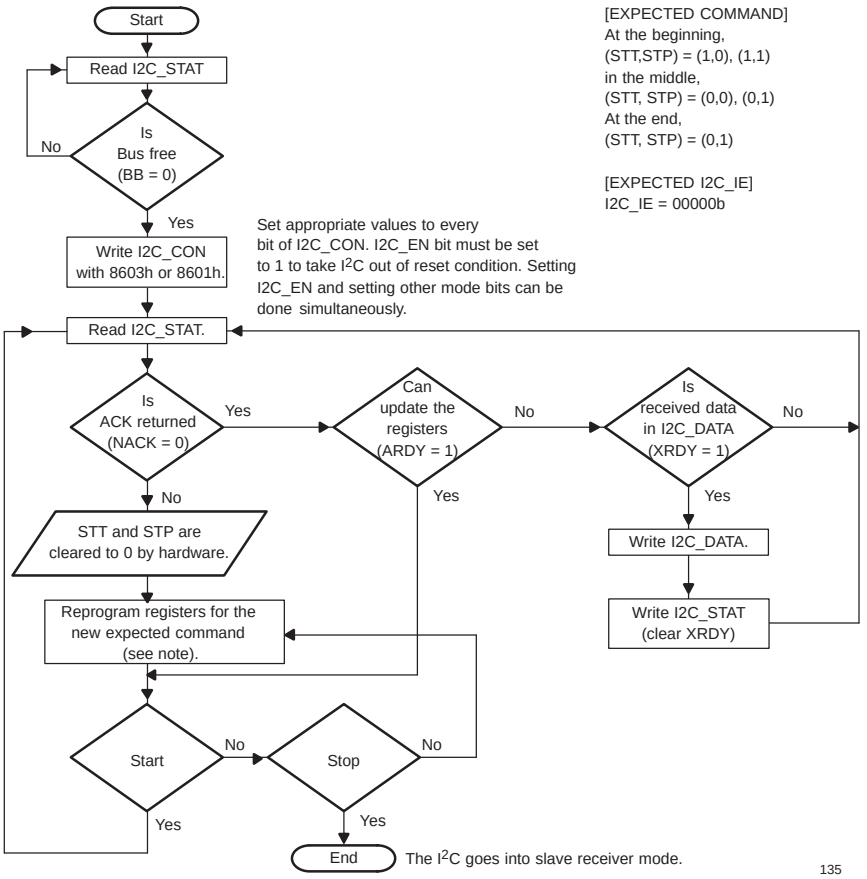
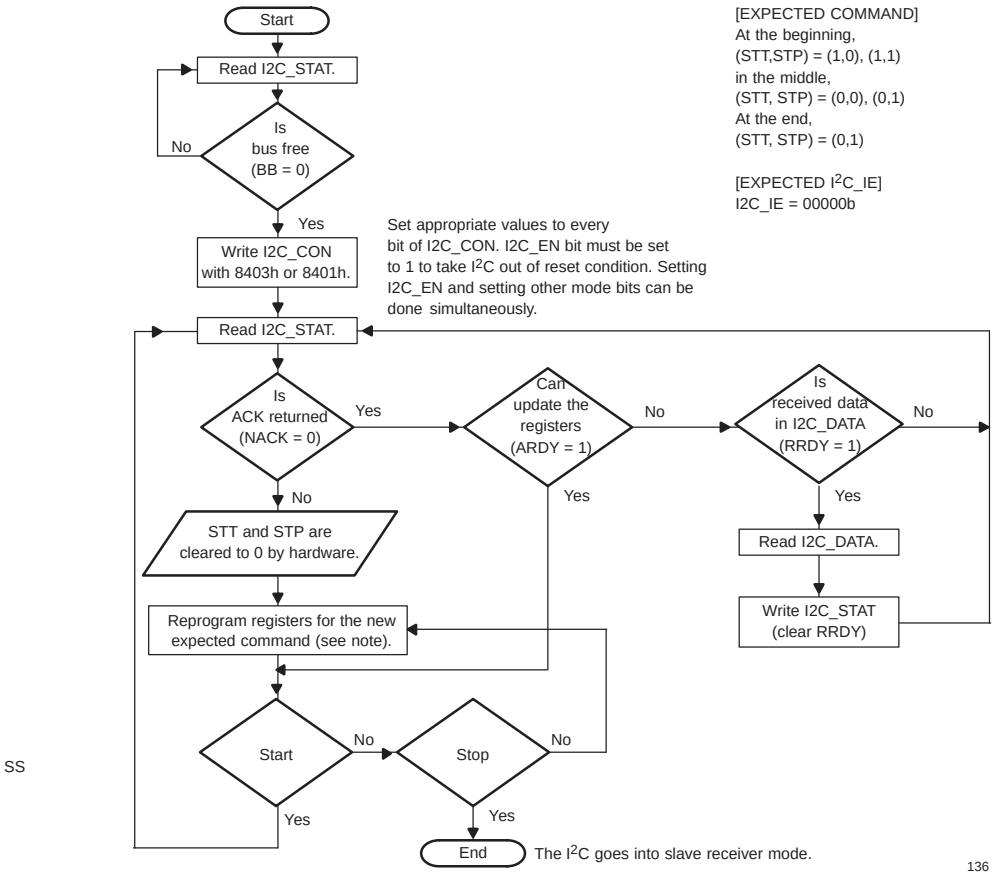


Figure 20-11. Master Transmitter Mode, Polling



Note: Reprogram registers means: I2Cn.I2C_CON[11] STB bit and/or I2Cn.I2C_CON[10] MST bit and/or I2Cn.I2C_SA[9:0] SA field and/or I2Cn.I2C_CNT register and/or I2Cn.I2C_CON[0] STT bit and/or I2Cn.I2C_CON[1] STP bit.

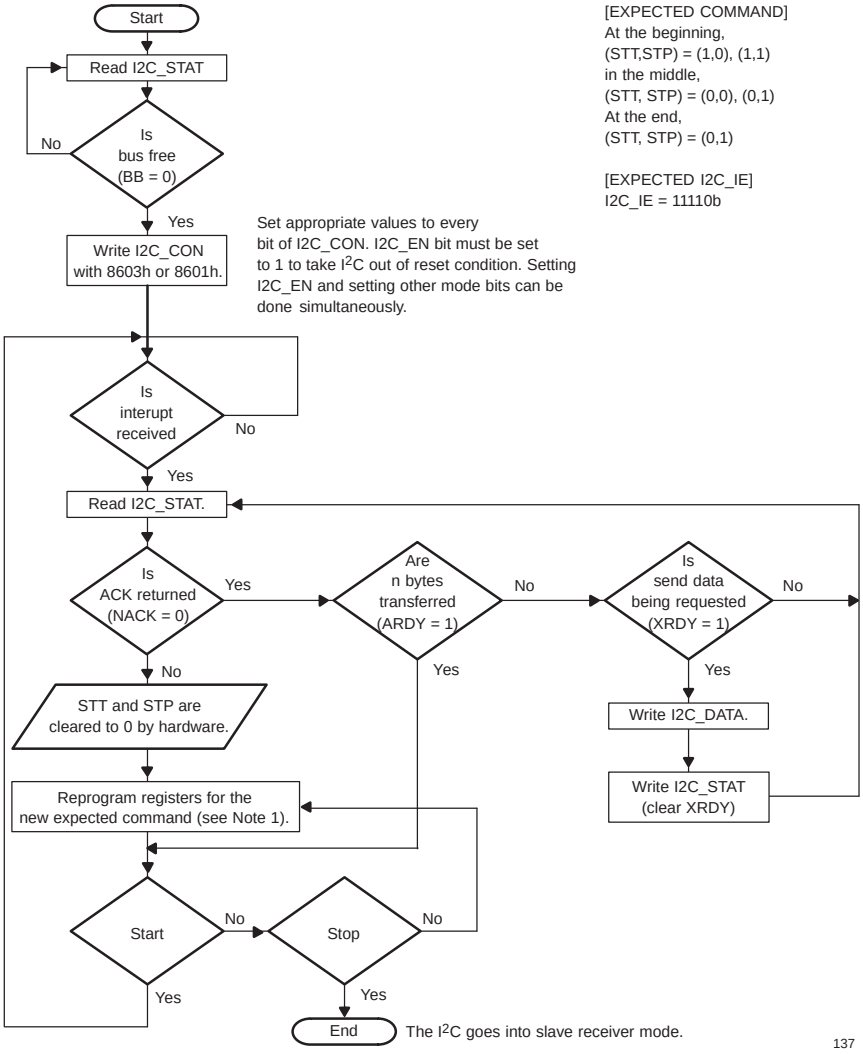
Figure 20-12. Master Receiver Mode, Polling



SS

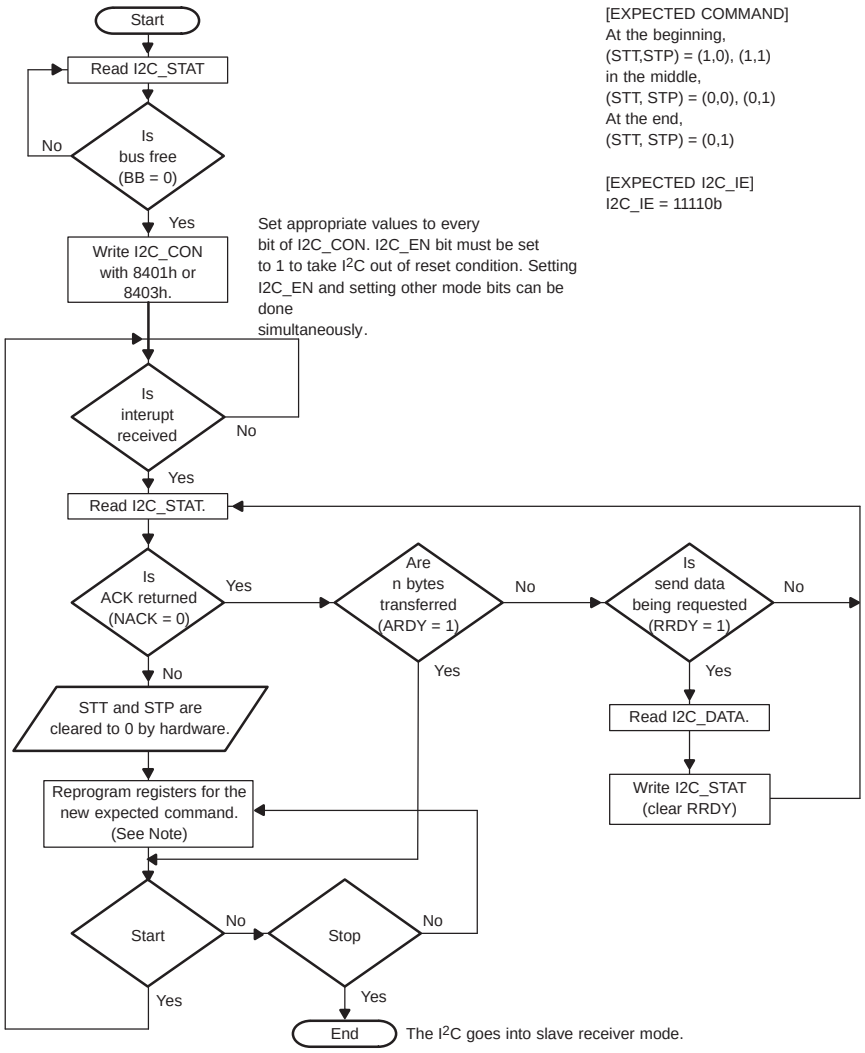
Note: Reprogram registers means: I2Cn.I2C_CON[11] STB bit and/or I2Cn.I2C_CON[10] MST bit and/or I2Cn.I2C_SA[9:0] SA field and/or I2Cn.I2C_CNT register and/or I2Cn.I2C_CON[0] STT bit and/or I2Cn.I2C_CON[1] STP bit.

Figure 20-13. Master Transmitter Mode, Interrupt



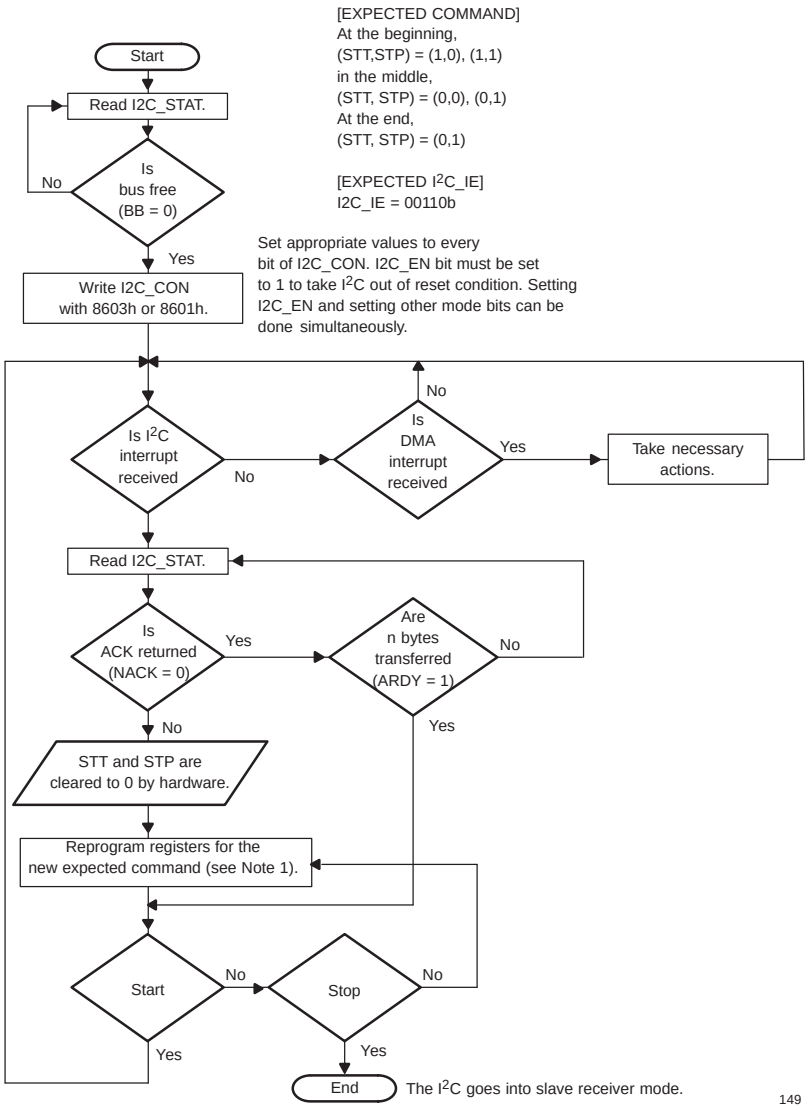
Note: Reprogram registers means: I2Cn.I2C_CON[11] STB bit and/or I2Cn.I2C_CON[10] MST bit and/or I2Cn.I2C_SA[9:0] SA field and/or I2Cn.I2C_CNT register and/or I2Cn.I2C_CON[0] STT bit and/or I2Cn.I2C_CON[1] STP bit.

Figure 20-14. Master Receiver Mode, Interrupt



Note: Reprogram registers means: I2Cn.I2C_CON[11] STB bit and/or I2Cn.I2C_CON[10] MST bit and/or I2Cn.I2C_SA[9:0] SA field and/or I2Cn.I2C_CNT register and/or I2Cn.I2C_CON[0] STT bit and/or I2Cn.I2C_CON[1] STP bit.

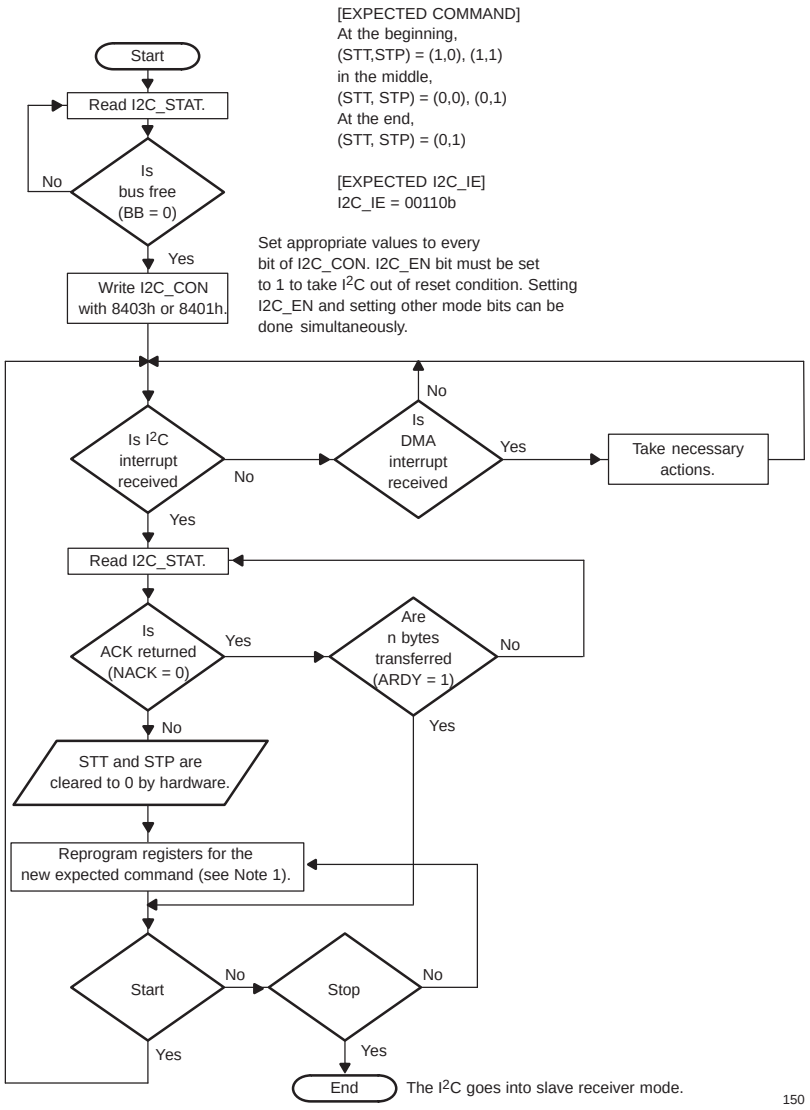
Figure 20-15. Master Transmitter Mode, DMA



Notes:

1. Reprogram registers means: I2Cn.I2C_CON[11] STB bit and/or I2Cn.I2C_CON[10] MST bit and/or I2Cn.I2C_SA[9:0] SA field and/or I2Cn.I2C_CNT register and/or I2Cn.I2C_CON[0] STT bit and/or I2Cn.I2C_CON[1] STP bit.
2. The DMA mode is not supported by the I2C3 module.

Figure 20-16. Master Receiver Mode, DMA



Notes:

1. Reprogram registers means: I2Cn.I2C_CON[11] STB bit and/or I2Cn.I2C_CON[10] MST bit and/or I2Cn.I2C_SA[9:0] SA field and/or I2Cn.I2C_CNT register and/or I2Cn.I2C_CON[0] STT bit and/or I2Cn.I2C_CON[1] STP bit.
2. The DMA mode is not supported by the I2C3 module.

Figure 20-17. Slave Transmitter/Receiver Mode, Polling

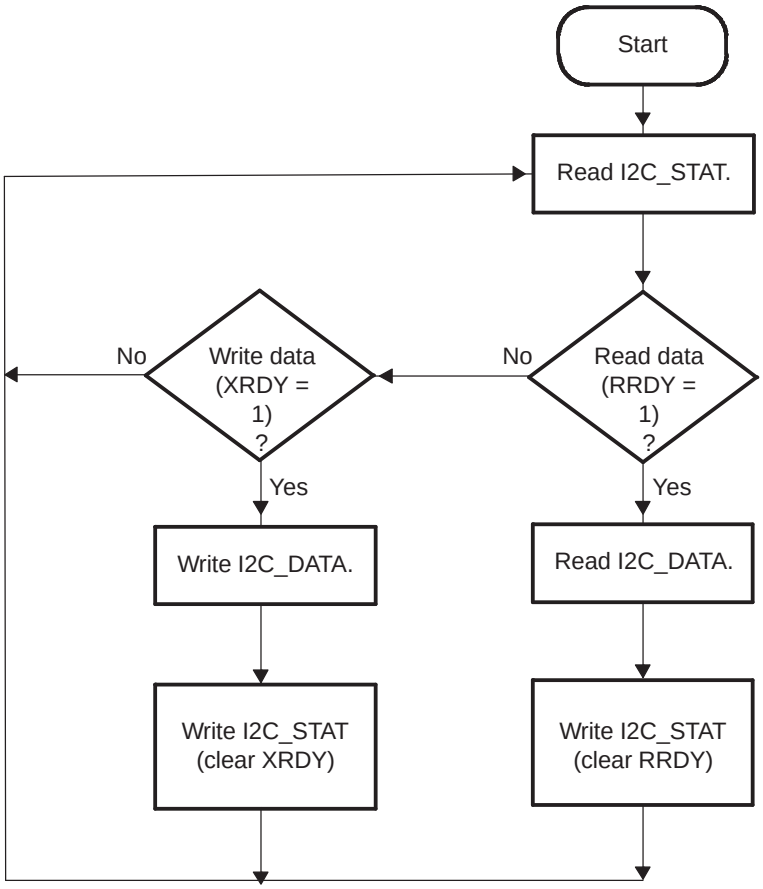
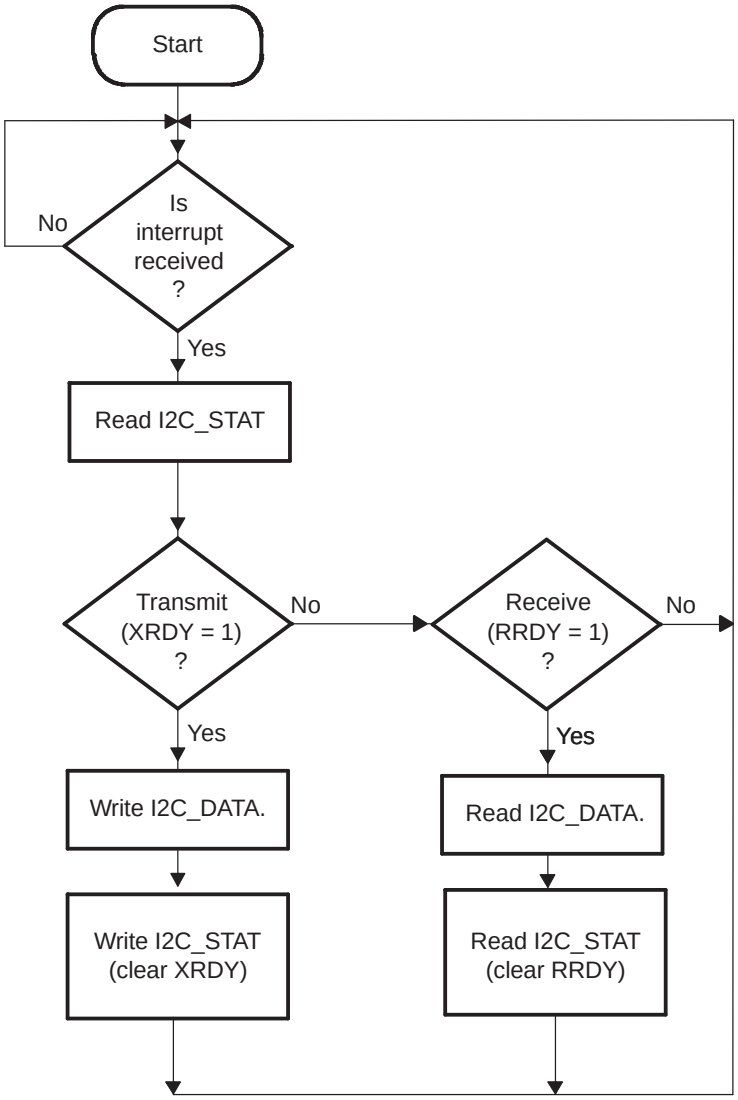


Figure 20-18. Slave Transmitter/Receiver Mode, Interrupt



20.6 I²C Register Manual

CAUTION

I²C registers are limited to 16-bit data accesses. 32-bit and 8-bit data accesses are not allowed and could corrupt registers content.

CAUTION

In order to determine the physical address of the register, use Base Address + Register Offset, where S = 2 for I2C1 and I2C2 registers accessed by the MPU only, and S = 1 for I2C3 registers accessed by the DSP only.

20.6.1 Instance Summary

Table 20-6 lists the base address and the address space of each I²C module instance of the Locosto device.

Table 20-6. Instance Summary

Module Name	MPU Base Address (hex)	DSP Base Address (hex)	Size
I2C1	0xFFFF B800	No access from the DSP	2K bytes
I2C2	0xFFFF C800	No access from the DSP	2K bytes
I2C3	No access from the MPU	0x7880	256 bytes

20.6.2 Register Mapping Summary

Table 20-7 summarizes the I²C register and their associated offset address for each I²C register.

Table 20-7. I²C Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset
I2C_REV	R	16	0x00 * S
I2C_IE	RW	16	0x01 * S
I2C_STAT	R	16	0x02 * S
I2C_SYSS	R	16	0x04 * S
I2C_BUF	RW	16	0x05 * S(1)
I2C_CNT	RW	16	0x06 * S
I2C_DATA	RW	16	0x07 * S
I2C_SYSC	RW	16	0x08 * S
I2C_CON	RW	16	0x09 * S
I2C_OA	RW	16	0x0A * S
I2C_SA	RW	16	0x0B * S
I2C_PSC	RW	16	0x0C * S
I2C_SCLL	RW	16	0x0D * S
I2C_SCLH	RW	16	0x0E * S
I2C_SYSTEST	RW	16	0x0F * S

(1) This register is not available for the I2C3 module as it does not support the DMA mode.

20.6.3 Registers Description

Table 20-8 through Table 20-22 describe each of the I²C registers within the module instance.

Table 20-8. I²C Module Revision Register (I2C_REV)

Address Offset	0x00 * S			
Physical Address	MPU	0xFFFF B800	**Instance**	I2C1
	MPU	0xFFFF C800		I2C2
	DSP	0x7880		I2C3
Description	Module revision register			
	This register contains the hard-coded revision number of the module.			
Type	R			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-								REV							

Bits	Field Name	Description	Type	Reset
15:8	-	Reserved	R	0x00
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision Examples: 0x10 for revision 1.0	R	See (1)

(1) TI internal data. A reset has no effect on the value returned.

Table 20-9. I²C Interrupt Enable Register (I2C_IE)

Address Offset	0x01 * S			
Physical Address	MPU	0xFFFF B802	**Instance**	I2C1
	MPU	0xFFFF C802		I2C2
	DSP	0x7881		I2C3
Description	I²C interrupt enable register			
	This register controls the interrupts mask/unmask function.			
Type	RW			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-										GC_IE	XRDY_IE	RRDY_IE	ARDY_IE	NACK_IE	-

Bits	Field Name	Description	Type	Reset
15:6	-	Reserved	R	0x000
5	GC_IE	General call interrupt enable	RW	0
		Mask or unmask the interrupt signaled by bit in I2C_STAT[GC]		
		0x0: General call interrupt disabled		
		0x1: General call interrupt enabled		
4	XRDY_IE	Transmit data ready interrupt enable	RW	0
		Mask or unmask the interrupt signaled by bit I2C_STAT[XRDY]		
		0x0: Transmit data ready interrupt disabled		
		0x1: Transmit data ready interrupt enabled		
3	RRDY_IE	Receive data ready interrupt enable	RW	0
		Mask or unmask the interrupt signaled by bit I2C_STAT[RRDY]		
		0x0: Receive data ready interrupt disabled		
		0x1: Receive data ready interrupt enabled		

Bits	Field Name	Description	Type	Reset
2	ARDY_IE	Register access ready interrupt enable Mask or unmask the interrupt signaled by bit I2C_STAT[ARDY] 0x0: Register access ready interrupt disabled 0x1: Register access ready interrupt enabled	RW	0
1	NACK_IE	No acknowledgement interrupt enable Mask or unmask the interrupt signaled by bit I2C_STAT[NACK] 0x0: Not Acknowledge interrupt disabled 0x1: Not Acknowledge interrupt enabled	RW	0
0	-	Reserved. This bit must be cleared for a proper behavior	RW	0

Table 20-10. I²C Status Register (I2C_STAT)

Address Offset	0x02 * S			
Physical Address	MPU	0xFFFF B804	**Instance**	I2C1
	MPU	0xFFFF C804		I2C2
	DSP	0x7882		I2C3
Description	I²C status register			
	This register provides core status information for interrupt handling, and other I²C control management.			
Type	R			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SBD	-		BB	ROVR	XUDF	AAS	-			GC	XRDY	RRDY	ARDY	NACK	-

Bits	Field Name	Description	Type	Reset
15	SBD	Single byte data status 0x0: No action 0x1: Single valid byte in last 16-bit data read	R	0
14:13	-	Reserved	R	0x0
12	BB	Bus busy status 0x0: Bus is free 0x1: Bus is occupied	R	0
11	ROVR	Receive overrun status. This read-only bit indicates whether the receiver has experienced overrun. Overrun occurs when the shift register is full and the receive buffer is full. An overrun condition does not result in a data loss; the peripheral is just holding the bus (low state on the serial clock line) and prevent others bytes from being received. 0x0: Normal operation 0x1: Receiver overrun	R	0
10	XUDF	Transmit underflow status. This read-only bit indicates whether the transmitter has experienced underflow. In the master transmit mode, underflow occurs when the shift register is empty, the transmit buffer is empty, and there are still some bytes to transmit (I2Cn.I2C_CNT[15:0] DCOUNT not equal to 0) 0x0: Normal operation 0x1: Transmit underflow	R	0
9	AAS	Address recognized as slave status 0x0: No action 0x1: Address device recognized	R	0
8:6	-	Reserved	R	0x0
5	GC	General call status. Set to 1 by core when General Call address detected and interrupt signaled to the MPU. Write 1 to clear.	RW	0

Bits	Field Name	Description	Type	Reset
		0x0: No action		
		0x1: General call address detected		
4	XRDY	Transmit data ready status	RW	0
		Set to 1 by core when transmitter and when new data is requested. When set to 1 by core, an interrupt is signaled to the MPU.		
		0x0: Transmit buffer full (or receiver mode)		
		0x1: Transmit data ready (for write)		
3	RRDY	Receive data ready status	RW	0
		When set to 1 by core when receiver mode, a new data can be read. When set to 1 by core, an interrupt is signaled to the MPU.		
		0x0: Receive buffer empty		
		0x1: Receive data available (for read)		
2	ARDY	Register Access ready status	RW	0
		When set to 1, it indicates that previously access has been performed and registers are ready to be accessed again. An interrupt is signaled to the MPU.		
		0x0: No action		
		0x1: Access ready		
1	NACK	No acknowledgement status	RW	0
		Bit is set when no acknowledge has been received, an interrupt is signaled to the MPU. Write 1 to clear this bit.		
		0x0: Normal operation		
		0x1: No acknowledge detected		
0	-	Reserved	RW	0

Table 20-11. I²C System Status Register (I2C_SYSS)

Address Offset	0x04 * S			
Physical Address	MPU	0xFFFF B808	**Instance**	I2C1
	MPU	0xFFFF C808		I2C2
	DSP	0x7884		I2C3
Description	I²C system status register			
	This register provides status information about the module, excluding interrupt status information.			
Type	R			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-															RDONE

Bits	Field Name	Description	Type	Reset
15:1	-	Reserved	R	0x00
0	RDONE	Reset done	R	0
		0x0: Internal module reset is ongoing.		
		0x1: Reset completed		

Table 20-12. I²C Buffer Configuration Register (I2C_BUF)

Address Offset	0x05 * S			
Physical Address	MPU	0xFFFF B80A	**Instance**	I2C1
	MPU	0xFFFF C80A		I2C2
Description	I²C buffer configuration register			
	This register enables the DMA transfer.			

Table 20-12. I²C Buffer Configuration Register (I2C_BUF) (continued)

Type	RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDMA_EN								XDMA_EN							

Bits	Field Name	Description	Type	Reset
15	RDMA_EN	Receive DMA channel enable	RW	0
		0x0: Receive DMA channel disabled		
		0x1: Receive DMA channel enabled		
14:8		Reserved	R	0x00
7	XDMA_EN	Transmit DMA channel enable	RW	0
		0x0: Transmit DMA channel disabled		
		0x1: Transmit DMA channel enabled		
6:0		Reserved	R	0x00

Note: The DMA mode is not supported by the I2C3 module.

Table 20-13. I²C Data Counter Register (I2C_CNT)

Address Offset	0x06 * S			
Physical Address	MPU	0xFFFF B80C	Instance	I2C1
	MPU	0xFFFF C80C		I2C2
	DSP	0x7886		I2C3
Description	Data counter register			
	This read/write register is used to control the number of bytes in the I²C data payload.			
Type	RW			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCOUNT															

Bits	Field Name	Description	Type	Reset
15:0	DCOUNT	Data count (master mode only)	RW	0x0000

Table 20-14. I²C Data Access Register (I2C_DATA)

Address Offset	0x07 * S			
Physical Address	MPU	0xFFFF B80E	Instance	I2C1
	MPU	0xFFFF C80E		I2C2
	DSP	0x7887		I2C3
Description	Data access register			
	This register is the entry point for the local host to read data from or write data to the FIFO buffer.			
Type	RW			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Bits	Field Name	Description	Type	Reset
15:0	DATA	Transmit/receive FIFO data	RW	0x0000

Table 20-15. I²C System Configuration Register (I2C_SYSC)

Address Offset	0x08 * S			
Physical Address	MPU	0xFFFF B810	**Instance**	I2C1
	MPU	0xFFFF C810		I2C2
	DSP	0x7888		I2C3
Description	System configuration register			
	This register controls the various parameters of the OCP interface.			
Type	RW			

1	0
SRST	

Bits	Field Name	Description	Type	Reset
15:2		Reserved	R	0x0000
1	SRST	Software reset	RW	0
		This bit is automatically reset by the hardware. During reads, it always returns 0.		
		0x0: Normal mode		
		0x1: The module is reset.		
0		Reserved	R	0

Table 20-16. I²C Configuration Register (I2C_CON)

Address Offset	0x09 * S			
Physical Address	MPU	0xFFFF B812	**Instance**	I2C1
	MPU	0xFFFF C812		I2C2
	DSP	0x7889		I2C3
Description	I²C configuration register			
	This register controls the functional features.			
	Caution: during an active transfer phase (STT has been set to 1), no modification must be done in this register. Changing it may result in unpredictable behavior.			
Type	RW			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C_EN	BE	-		STB	MST	TRX	XSA	-	XOA	-				STP	STT

Bits	Field Name	Description	Type	Reset
15	I2C_EN	Module enable	RW	0
		0x0: Controller in reset. FIFO is cleared and status bits are set to their default value.		
		0x1: Module enabled		
14	BE	Big endian mode	RW	0
		0x0: Little endian mode		
		LS byte is transmitted first in transmit mode. In receive mode, the first byte is stored in the LS byte position.		
		0x1: Big endian mode		
13:12	-	Reserved	R	0x0
11	STB	Start byte mode (master mode only)	RW	0
		0x0: Normal mode		

Bits	Field Name	Description	Type	Reset
		0x1: Start byte mode		
10	MST	Master/slave mode	RW	0
		0x0: Slave mode		
		0x1: Master mode		
9	TRX	Transmitter/receiver mode (master mode only)	RW	0
		0x0: Receiver mode		
		0x1: Transmitter mode		
8	XSA	Expand slave address	RW	0
		0x0: 7-bit slave address mode		
		0x1: 10-bit slave address mode		
7	-	Reserved. This bit must be cleared for a proper behavior.	RW	0
		0x0 No action		
6	XOA	Expand Own Address	RW	0
		0x0: 7-bit own address mode		
		0x1: 10-bit own address mode		
5:2	-	Reserved	R	0x00
1	STP	Stop condition (master mode only)	RW	0
		0x0: No action or stop condition detected		
		0x1: Stop condition queried		
0	STT	Start condition (master mode only)	RW	0
		0x0 No action or start condition generated		
		0x1 Start		

Table 20-17. I²C Own Address Register (I2C_OA)

Address Offset	0x0A * S			
Physical Address	MPU	0xFFFF B814	**Instance**	I2C1
	MPU	0xFFFF C814		I2C2
	DSP	0x788A		I2C3
Description	I²C own address register			
	This register is used to specify the I²C module's 7-bit or 10-bit address.			
Type	RW			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-								OA							

Bits	Field Name	Description	Type	Reset
15:10	-	Reserved	R	0x00
9:0	OA	Own address	RW	0x000

Table 20-18. I²C Slave Address Register (I2C_SA)

Address Offset	0x0B * S			
Physical Address	MPU	0xFFFF B816	Instance	I2C1
	MPU	0xFFFF C816		I2C2
	DSP	0x788B		I2C3
Description	I²C slave address register			
	This register is used to specify the addressed I²C module's 7-bit or 10-bit address.			
Type	RW			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-								SA							

Bits	Field Name	Description	Type	Reset
15:10	-	Reserved	R	0x00
9:0	SA	Slave address	RW	0x3FF

Table 20-19. I²C Clock Prescaler Register (I2C_PSC)

Address Offset	0x0C * S			
Physical Address	MPU	0xFFFF B818	**Instance**	I2C1
	MPU	0xFFFF C818		I2C2
	DSP	0x788C		I2C3
Description	I²C clock prescaler register			
	This register is used to specify the internal clocking of the I²C peripheral core.			
	The core logic is sampled at the clock rate of the system clock for the module divided by (PSC+1).			
Type	RW			

7	6	5	4	3	2	1	0
PSC							

Bits	Field Name	Description	Type	Reset
15:8		Reserved	R	0x00
7:0	PSC	Fast/standard mode prescale sampling clock divider value	RW	0x00

Table 20-20. I²C SCL Low Time Register (I2C_SCLL)

Address Offset	0x0D * S			
Physical Address	MPU	0xFFFF B81A	**Instance**	I2C1
	MPU	0xFFFF C81A		I2C2
	DSP	0x788D		I2C3
Description	I²C SCL low time register			
	This register is used to determine the SCL low time value in master mode.			
Type	RW			

7	6	5	4	3	2	1	0
SCLL							

Bits	Field Name	Description	Type	Reset
15:8		Reserved	R	0x00
7:0	SCLL	Fast/standard mode SCL low time (master mode only)	RW	0x00

Table 20-21. I²C SCL High Time Register (I2C_SCLH)

Address Offset	0x0E * S			
Physical Address	MPU	0xFFFF B81C	Instance	I2C1
	MPU	0xFFFF C81C		I2C2
	DSP	0x788E		I2C3
Description	I²C SCL high time register			
	This register is used to determine the SCL high time value when it is the master.			
Type	RW			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-								SCLH							

Bits	Field Name	Description	Type	Reset
15:8	-	Reserved	R	0x00
7:0	SCLH	Fast/standard mode SCL high time (master mode only)	RW	0x00

Table 20-22. I²C System Test Register (I2C_SYSTEST)

Address Offset	0x0F * S			
Physical Address	MPU	0xFFFF B81E	**Instance**	I2C1
	MPU	0xFFFF C81E		I2C2
	DSP	0x788F		I2C3
Description	I²C system test register			
	This register is used to facilitate system-level tests by overriding some of the standard functional features of the peripheral.			
	Caution: Never set this register in normal operation mode.			
Type	RW			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ST_EN	FREE	TMODE		SSB	-							SCL_I	SCL_O	SDA_I	SDA_O

Bits	Field Name	Description		Type	Reset
15	ST_EN	System test enable		RW	0
		0x0:	Normal mode		
		0x1:	System test enabled. Allows setting of other system test register bits.		
14	FREE	Free running mode after breakpoint.		RW	0
		0x0:	Stop mode (on breakpoint condition)		
			In master mode, it stops after completion of an ongoing bit transfer		
			.In slave mode, it stops during the phase transfer when one byte is completely transmitted/received.		
		0x1:	Free running mode		
13:12	TMODE	Test mode select		RW	0x0
		0x0:	Functional mode (default)		
		0x1:	Reserved		
		0x2:	Test of SCL counters (I2C_SCLL, I2C_SCLH, I2C_PSC). SCL provides a permanent clock with master mode.		
		0x3:	Loopback mode select + SDA/SCL IO mode select		
11	SSB	Set status bits		RW	0
		0x0:	No action		
		0x1:	Set all interrupt status bits to 1		
10:4	-	Reserved		R	0x00
3	SCL_I	SCL line sense input value. It returns the logical state of the serial clock line (i2cn_scl)		R	0
		0x0:	Read 0 from SCL line		
		0x1:	Read 1 from SCL line		
2	SCL_O	SCL line sense output value. It forces the logical state of the serial clock line (i2cn_scl) in test mode (when ST_EN = 1 and TMODE = b11)		RW	0
		0x0:	Write 0 to SCL line		

Bits	Field Name	Description	Type	Reset
		0x1: Write 1 to SCL line		
1	SDA_I	SDA line sense input value. It returns the logical state of the serial data line (i2cn_sda) 0x0: Read 0 from SDA line 0x1: Read 1 from SDA line	R	0
0	SDA_O	SDA line sense output value. It forces the logical state of the serial data line (i2cn_sda) in test mode (when ST_EN = 1 and TMODE = b11) 0x0: Write 0 to SDA line 0x1: Write 1 to SDA line	RW	0

LCD Interface

This chapter describes the features and functions of the LCD interface module for the LOCOSTO device.

Topic **Page**

21.1 Module Overview

The liquid crystal display (LCD) interface connects an external color graphical controller to the digital baseband (DBB) device (LOCOSTO). This 8-bit parallel interface is compliant with the 8-bit 6800-series and 8086-series parallel interface standard to support a large range of LCDs available on the market.

The configuration and data information are transferred from the microprocessor unit (MPU) to the LCD interface through the TI peripheral bus (TIPB) by 16-bit words up to a rate of 52M words/s. The control and data information are transferred from the LCD interface to the external LCD controller at a speed that is a fraction (1, 2, 4, or 8) of the 13-MHz GSM clock. Data are transferred in 8-bit words.

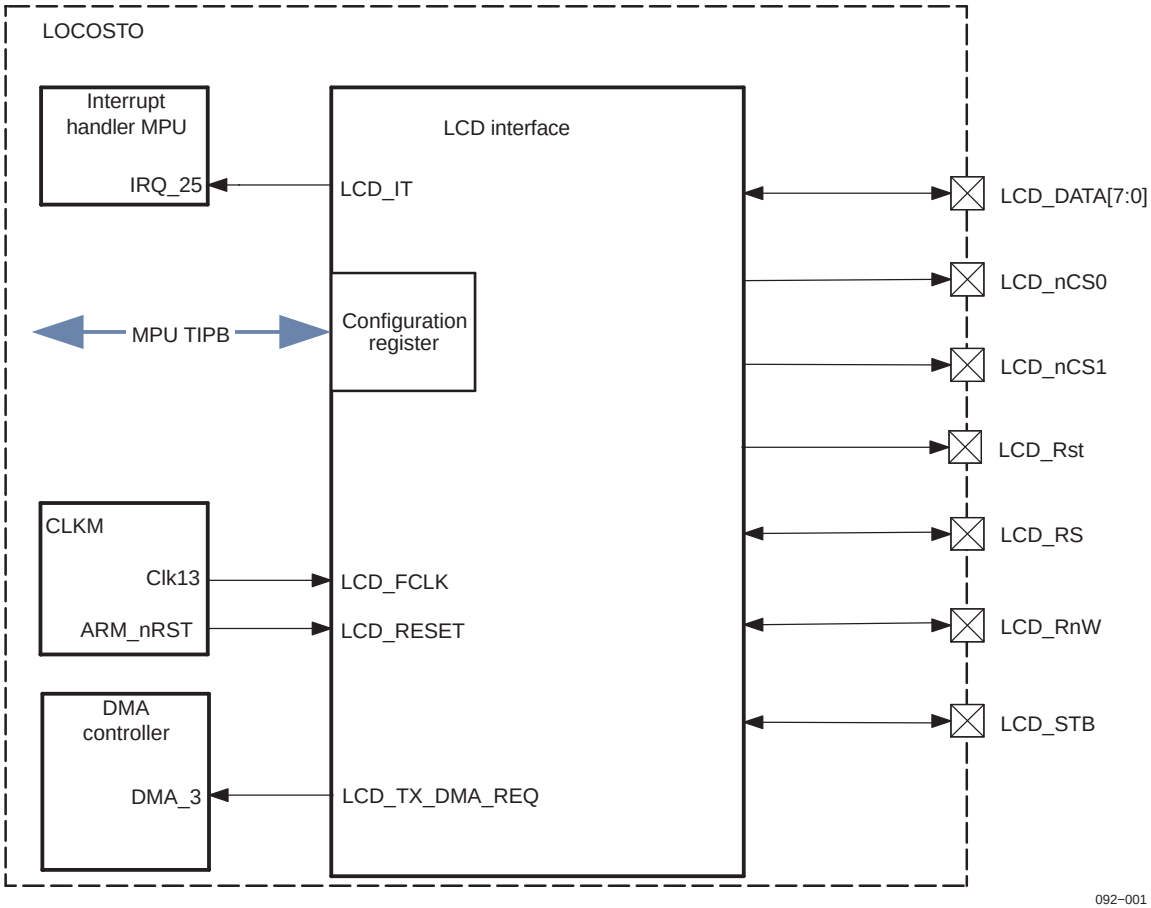
A 2-port first-in, first-out (FIFO) buffer of 128 x 16-bit words adapts the transfer rate between the internal static random access memory (SRAM) and the LCD graphical RAM.

This interface targets LCD devices with the following features:

- Color LCD screens (passive or active matrix)
- 8-bit 6800-series or 8086-series parallel interface
- 16-bit (RGB 656) or 24-bit (RGB 888) color
- Up to QVGA (320 x 240) format

Figure 21-1 shows an LCD overview.

Figure 21-1. LCD Transfer Data Flow



Note: Another alternative for connecting an external or secondary LCD controller is to use the serial port interface (SPI), which is bidirectional and composed of four lines dedicated to the transfer of data to and from external devices.

21.1.1 Main Features

The main features of the LCD interface are:

- Receive and transmit capability
- Two interface modes: 6800 and 8086
- 16-bit word width between memory and the LCD interface
- 52M-words/s transfer rate between memory and the LCD interface
- Speed of data and control transfer is a fraction (1, 2, 4, or 8) of the 13-MHz GSM clock
- Receive and transmit interrupts
- FIFO buffer used for adapting transfer rate
- Two-port FIFO of 128 x 16-bit words
- FIFO used in transmission data or commands
- Register used in reception

21.1.2 Signal and I/O Descriptions

Table 21-1 lists the I/O signals.

Table 21-1. I/O Descriptions

Signal Name	I/O[1]	Description	Value at Reset
LCD_nRst	O	LCD controller reset	0
LCD_STB	O	6800 mode: Strobe enable (by default). Read data from the LCD controller on the strobe falling edge. Send data to the LCD controller on the strobe rising edge. 8086 mode: Read enable (by default). Read data from the LCD controller on the strobe rising edge. Can be used as GPIO.	0
LCD_RnW	O	6800 mode: Read/write control pin. When high: read; when low: write. 8086 mode: Write enable clock. When high: Read When low: Write Send data to the LCD controller on the RnW falling edge. Can be used as GPIO (by default).	0
LCD_nCS[1:0]	O	LCD interface chip-select for chip 0 or 1. Data I/O are available when low. When high, data I/O are in a high impedance state. Bits shared with the GPIO module and used as GPIO by default. These bits are used as outputs for LCD interface.	1
LCD_RS	O	LCD interface register selection (by default), used also as GPIO	0
LCD_DATA[7:0]	I/O	LCD interface I/O data bus (by default)	Unknown

(1) I = Input, O = Output

CAUTION

The 8-bit data bus is shared with the LCD, the NAND flash, and the parallel-camera function and can be selected by one dynamic switch. The bus selection switch is performed dynamically by the chip-configuration control register LCD-CAM-NAND flash. The LCD data bus is selected at reset and outputs a low level on the package pin LCD/NAND flash/CAM bus.

See Chapter 18, *Configuration*, for more information about the LCD-CAM-NAND flash register and the LCD interface pins multiplexing.

21.1.3 LCD Environment

The LCD implementation described in this section is based on one LCD module that embeds the display and the controller. The display is the Philips LPH8754-2 with one thin film transistor (TFT) a-Si AMLCD based on an active matrix of 2.2-inch color 176 x 220.

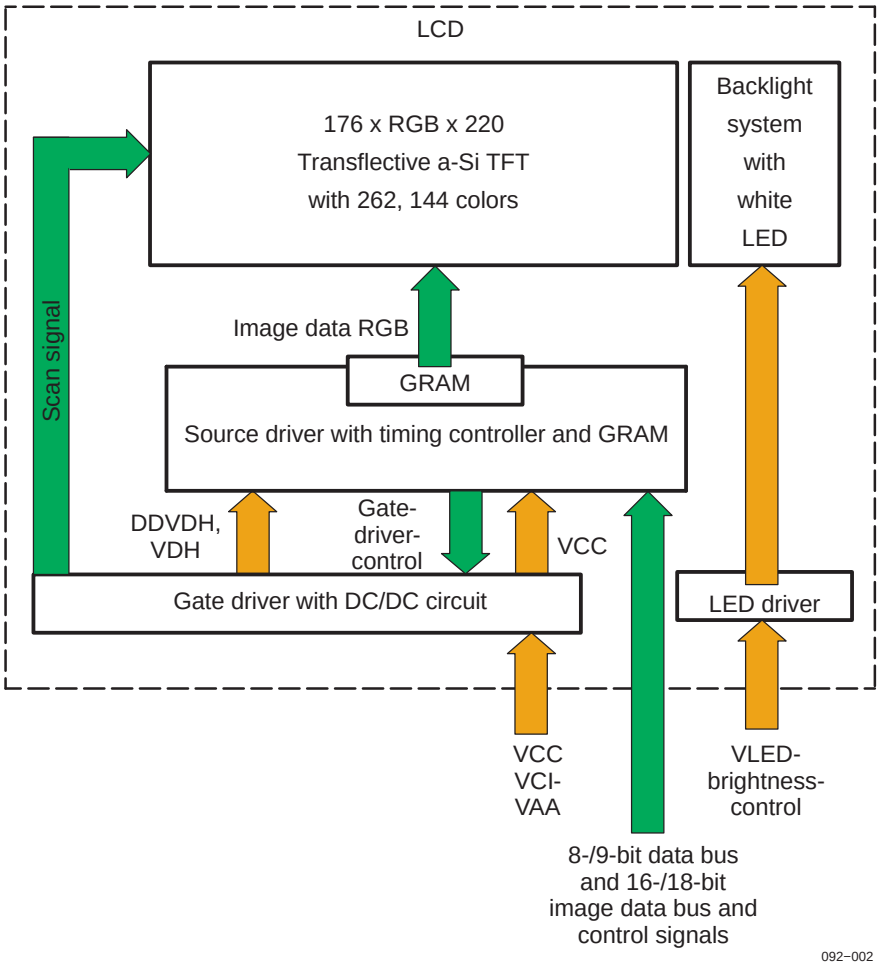
Because of the high pin count, the controller is based on two chip-sets, both produced by Hitachi: the gate IC (HD66774) and the source IC (HD66772). The chip-sets are sometimes named row and column drivers.

Note: For more information on interfacing the LCD controller driver with a different LCD module, see the *LOCOSTO LCD Driver Customization Guidelines Application Note* (SWCA008).

21.1.3.1 LCD Module Block Diagram

Figure 21-2 shows the LCD module block diagram.

Figure 21-2. LCD Module Block Diagram



21.1.3.2 LCD

The LPH8754-2 display has both a microprocessor and a video interface. The microprocessor interface selects and modifies operating conditions. Video data can be written at a frame rate of 60 Hz using the dedicated video interface.

The microprocessor interface for the LPH8754-2 module is the 80-system, and the preferred video interface mode is the RGB. For a description of the interface electrical and timing diagrams, see the Hitachi HD66772 data sheet.

Before images can be displayed, a sequence of commands must be sent by the microprocessor interface. These commands establish the power-on sequence and set the internal control registers. See the LPH8754-2 data sheet for more information on power-on and boot-up sequences.

21.1.3.2.1 LCD Features

- a-Si AMLCD TFT display panel with transflective cell
- Small and thin design of panel and illumination unit
- Interfaces: 80-system, 8-bit parallel bus, and 18-bit parallel RGB video
- Three LED backlights with 0.615-mm light guide
- In-cell diffusive reflector with high gain and high brightness

- High contrast in reflective and transmissive mode

21.1.3.3 LCD Controller Module

The data for display MD[8:1] are written to the values of the chip-select and reset signals in synchronization with the WR, RD, and RS signals. These signals belong to the source driver (HD66772).

21.1.3.3.1 HD66774 IC

The HD66774 is a gate-driver IC for a system with a color-TFT-liquid-crystal dot-matrix graphic display. The IC incorporates a circuit for driving 240 channels of TFT gate-line driving and all power-supply circuits required for liquid crystal displays. This chip-set supports 176 x 220 pixel TFT color LCD panels and offers 262,144 color display capability at low-power consumption of 4.5 mW.

The HD66774 has five internal registers. The data is written to these registers using a gate serial data interface. This interface can be directly connected to the HD66770 or the HD66772 source driver for an automatic serial transfer of instructions. When an instruction is written to the HD6670/772 by the bus from the CPU, it is output from the serial interface of the HD66770/772, and the HD66774 receives the instruction to adjust the settings of one of its internal registers.

The register settings are transferred from the HD66770 or the HD66772 source driver. The interface consists of a chip-select signal (GCS), a transfer clock (GCL), and data input (GDA) lines.

The data transfer starts when the falling edge of the GCS line indicates that the data is to be transferred. The transfer ends when the rising edge of the GCS line indicates the transfer is over. The bits are transferred in 16-bit units, and the data is transferred in the order from MSB to LSB.

For more information on power-on and setting-off sequences or about LCD controller functioning, see the HD66774 data sheets.

21.1.3.3.2 HD66772 IC

The HD66772 source driver is used in combination with the HD66774 gate driver/power-supply IC to display 176RGB x 240-dot graphics on TFT color LCD displays in 262,144 colors. The HD66772 features low-voltage operation (1.8 V minimum) and an internal RAM from which it drives the color images.

The HD66774 features an interface to drive the 240 TFT gate lines and voltage followers to generate the LCD-driving voltage. Because the HD66772 incorporates a circuit that interfaces with the HD66774, it can set instructions for the HD66774. The device supports functions such as an 8-color display function, and standby and sleep modes that allow precise power control by software.

The HD66772 has an 18-bit bus architecture and three registers:

- 16-bit index register (IR)
- 18-bit write-data register (WDR)
- 18-bit read-data register (RDR)

Before the internal operation of the HD66772 starts, control information is temporarily stored in the WDR register to allow high-speed interfacing with a high-performance microcomputer.

These signals, which include the register selection signal (RS), the read-write (R/W) signal, and the internal 16-bit data bus signals (DB15 to DB0), make up the HD66772 instructions. The access to the GRAM uses the internal 18-bit data bus.

The HD66772 is internally initialized by the reset input. Because the busy flag (BF) indicates a busy state (BF = 1) during the reset period, no instruction or GRAM data access from the MPU is acceptable.

For more information on reset, initialization, and functioning of the HD66772, see the HD66772 data sheet.

21.1.3.3.3 Connection Between the LOCOSTO and the LCD Controller

Figure 21-3 shows the main LCD connection.

Figure 21-3. Main LCD Connection

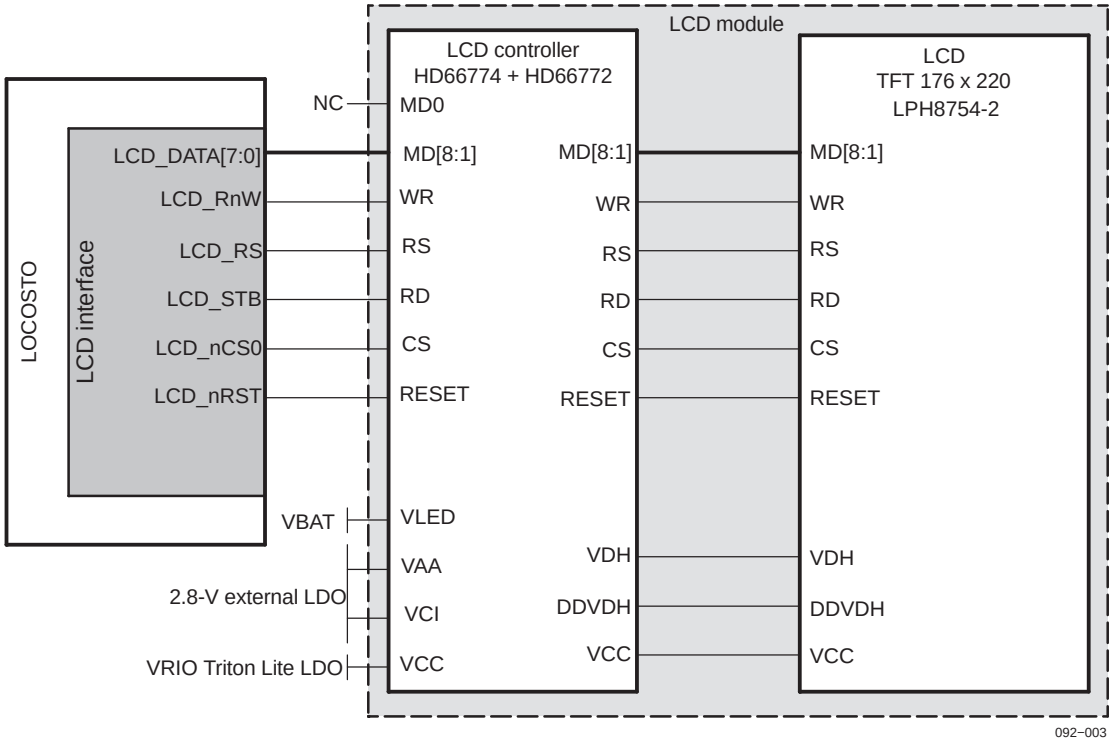


Table 21-2 lists the signals between the LOCOSTO device and the LCD controller.

Table 21-2. Connection Between the LOCOSTO and the LCD Controller

LOCOSTO					HD66774			
Signal	**Ball**	**Mode**	**I/O**	**Power**	**Signal**	**Connector Pin**	**I/O**	**Power**
LCD_DATA	E9	0	O	VDD_IO	MD1	50	I	VRIO
LCD_DATA	B8	0	O		MD2	51	I	
LCD_DATA	C8	0	O		MD3	52	I	
LCD_DATA	E8	0	O		MD4	53	I	
LCD_DATA	B7	0	O		MD5	54	I	
LCD_DATA	D8	0	O		MD6	55	I	
LCD_DATA	C7	0	O		MD7	56	I	
LCD_DATA	B6	0	O		MD8	57	I	
LCD_RnW	F9	0	O		WR	47	I	
LCD_RS	D9	0	O		RS	46	I	
LCD_STB	B10	0	O		RD	48	I	
LCD_nCS0	E10	1	O		CS	45	I	
LCD_nRST	C10	0	O		RESET	58	I	

21.1.4 Clocking, Reset, and Power-Management Scheme

21.1.4.1 Clocks

The LCD interface operates from two clocks:

- Functional clock of 13 MHz (LCD_FCLK) generated by the CLKM module. This clock is fractioned and used for transmission data between the LCD interface and the LCD controller.
  See Chapter 6, *Power, Reset, and Clock Management,* for more information.
- Clock of 52 MHz (LCD_ICLK) is used for reception data between the TIPB and LCD interface.

Table 21-3 describes the LCD interface clocking.

Table 21-3. LCD Interface Clocking

Type	Name	Source	Frequency	Description
Functional	LCD_FCLK	CLKM	13 MHz	Used for data exchange between the LCD interface and LCD controller
Interface	LDC_ICLK	API/TIPB bridge	52 MHz	Used for data exchange between the TIPB and LCD interface

21.1.4.2 Hardware and Software Resets

21.1.4.2.1 Hardware Reset

The ARM_nRST signal performs hardware reset by completely resetting the module, including the registers and the state-machine.

21.1.4.2.2 Software Reset

The SOFT_NRST bit LCD.CNTL_REG[0] performs software reset. By setting SOFT_NRST to 0, the LCD interface is reset. The bit returns high when the LCD interface is reset.

Table 21-4 lists the LCD interface hardware and software resets.

Table 21-4. LCD Interface Hardware and Software Resets

Type	Name	Source	Activation	Domain
Hardware	ARM_nRST	CLKM	0	Global reset
Software	SOFT_NRST	CNTL_REG	0	Global reset

21.1.4.3 Power Management

To minimize power consumption, a power management mode is available. The power management mode automatically configures the LCD_DATA[7:0] bus as output by setting the di_do_sel_ngate bit LCD_CNTL_REG[5] to 0 when the device is in deep sleep. This method prevents the LCD_DATA[7:0] bus from being in a floating state whenever the bus is in input and read capabilities are not supported by the LCD controller.

21.1.5 Hardware Requests

21.1.5.1 Interrupts

The LCD interface has two sources of interrupts, which correspond to FIFO empty and to the completion of a read access on the LCD controller, respectively. In the first case, the interrupt acknowledgment consists of writing data to the FIFO; in the second case, the interrupt acknowledgment consists of reading the data register of the LCD interface (RD_REG).

These two interrupt sources are combined in one AND gate to produce IRQ interruption, which is active at low level. To clear the interrupt line, all interrupt sources must be acknowledged or disabled.

21.1.5.2 DMA Request

The LCD interface generates a DMA request to load its TXFIFO under control of the DMA instead of the MPU. This feature can be enabled or disabled using the CNTL_REG[8] bit.

A DMA request is generated when TXFIFO is empty and the DMA feature is enabled. New DMA requests are generated on FIFO empty if at least the number of words corresponding to the MIN_FRAME_SIZE configuration have been transmitted since the last DMA request.

However, the DMA feature can be enabled before that by again clearing and setting the CNTL_REG[8] bit.

Table 21-5 shows the interrupt and DMA mapping from the LCD interface to the MPU.

Table 21-5. Interrupt and DMA Mapping

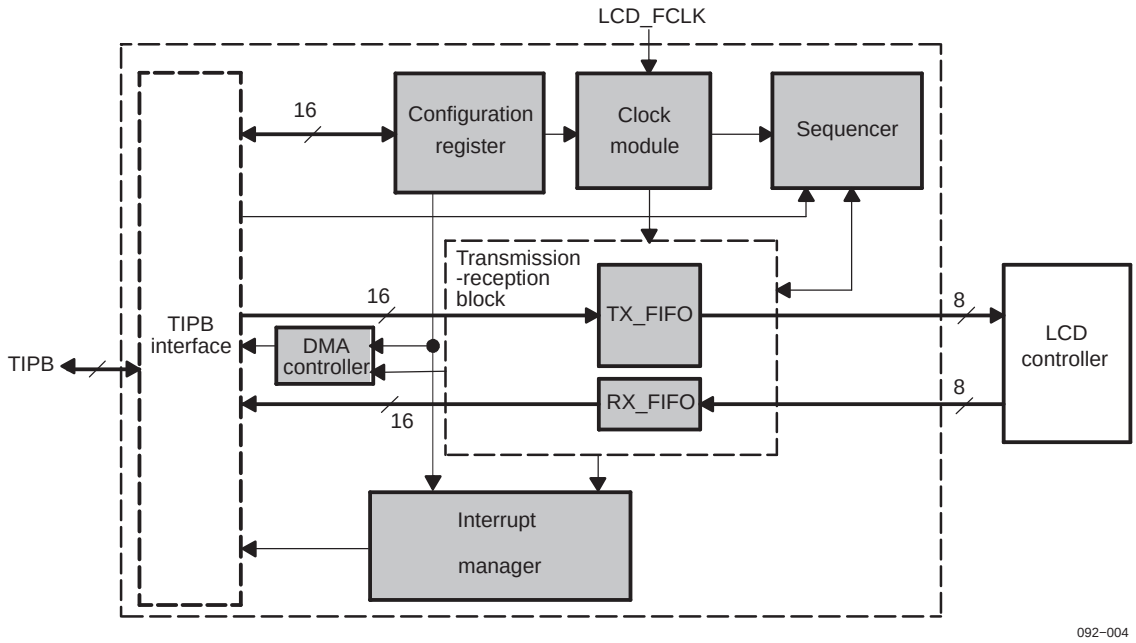
Type	Source Hardware Request Name	Destination Hardware Request Name	Description
Interrupt	LCD_IT	IRQ_25	Interrupt coming from the LCD interface
DMA	LCD_TX_DMA_REQ	DMA_3	Request coming from the LCD interface

21.2 Functional Description

21.2.1 Block Diagram

Figure 21-4 shows the functional description with the LCD interface hardware blocks.

Figure 21-4. LCD Interface Block Diagram



The LCD interface functional block diagram includes the following submodules:

- TIPB interface
- DMA controller
- Clock submodule receiving the 13-MHz GSM clock
- Transmitter/receiver (TX/RX) block composed of transmitter FIFO and receiver register
- Interrupt manager
- Configuration registers
- Sequencer submodule

21.2.1.1 TIPB Interface

This submodule includes the register address decoder, which ensures communication with the TIPB.

21.2.1.2 DMA Controller

Because data transfer is executed by using either a DMA request or interrupt, 1 bit in the control register CNTL_REG enables or disables the DMA transfer mode. When the DMA_EN bit LCD.CNTL_REG[8] is set to 1, the DMA transfer mode is adopted; otherwise, DMA capability is disabled. See the programming guide for more information.

21.2.1.3 Clock Submodule

The clock submodule generates reference clocks for the main sequencer, the transmitter-receiver block and associated timers, and other submodules.

Thus, this module provides the clock (13/N MHz, where N = 1, 2, 4, or 8) used to receive or transmit data from/to the LCD controller.

21.2.1.4 Transmitter/Receiver Sub-block

This submodule consists of the transmitter FIFO and receiver register. Data exchange with this module and the TIPB occurs at 52 MHz. Exchange data with the LCD controller occurs at a fraction of 13 MHz (13/N, where N = 1, 2, 4, or 8).

The FIFO includes the following main features:

- Capacity of 128 x 16 bits
- Two ports, so it can be written to and read at the same time by two different processes
- Flag FIFO empty

The receiver register includes the following main features:

- Capacity of 16 bits
- Flag LCD_READ_EVENT

Two status bits, FIFO_EMPTY_STATUS and FIFO_FULL_STATUS, of the LCD_IF_STS_REG register represent the state of FIFO_TX; the LCD_READ_EVENT_STATUS bit represents the state of the receiver register.

See the programming guide for more information.

21.2.1.5 Interrupt Controller

The interrupts for the LCD interface are mapped on the MPU IRQ line. The interrupt controller is associated with the FIFO and STATUS registers.

The main features include:

- Set flag bits in the status register LCD_IF_STS_REG
- Generation of one interrupt signal to the MPU controller

21.2.1.6 Register Block

The register block gathers the command and status registers accessible in read/write access by the MPU and includes these registers:

- Control register (CNTL_REG)
- LCD controller register (LCD_CNTL_REG)
- LCD interface status register (LCD_IF_STS_REG)
- Write data FIFO register (WR_FIFO)
- Read data register (RD_REG)

See Section 21.4.2 for more information on register descriptions.

21.2.1.7 Sequencer Submodule

The main sequencer is based on a finite state-machine. The reception and transmission state-machines are scheduled, a 52-MHz clock and a fraction (1, 2, 4, or 8) of a 13-MHz clock, respectively.

The LCD interface provides an interface to the MPU to load data from internal SRAM into the display RAM of an external LCD controller. This interface also updates the configuration registers to read status information and graphical data from the LCD controller.

The MPU transfers configuration and data information to the LCD interface through the TIPB using 16-bit words at a rate of 52M words/s. The LCD interface transfers control and data information to the external LCD controller at a speed that is a fraction (1, 2, 4, or 8) of the 13-MHz GSM clock.

A 2-port FIFO buffer of 128 x 16-bit words adapts the transfer rate between the internal SRAM and the LCD graphical RAM. The FIFO size chosen is a tradeoff between the interrupt rate and physical dimensions.

21.2.2 Error Reporting

No flags directly indicate the status of the LCD errors. The only method is to check that the LCD is refreshing itself properly by reading the status register of the LCD controller (for example, HD66774) to see that the raster-row position is updated.

21.3 Programming Model

This section describes the basic operations of the LCD interface.

Note: For complementary information on the setup and dataflow of the LCD in an imaging application, see the *Imaging Application Note* (APN212).

21.3.1 Setup for Typical Configuration(s)

21.3.1.1 LCD Interface Setup

The LCD interface is set up using only one register (CNTL_REG). The parameter settings concern the following:

- Data rate in transmission and reception
- Usage of interrupts
- Mode of transmission 6800 or 8086
- Enabling of the clock and DMA
- Minimum frame size used only in DMA mode
- Number of dummy read cycles that are no operation (NOP) cycles integrated for synchronization and that depend on controller types

See Section 21.4.2, *Register Descriptions*, for more information on the N_DUMMY bit field in the CNTL_REG register.

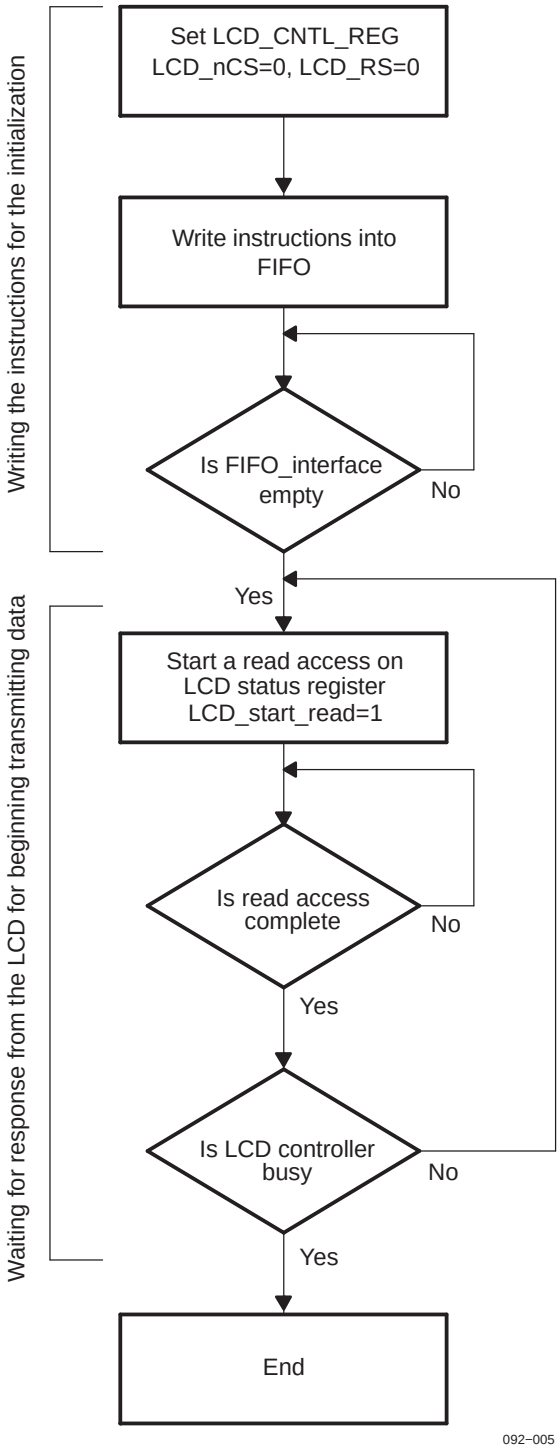
21.3.1.2 LCD Controller Initialization

The LCD can be used only when the LCD controller is selected. To select LCD controller 0, set the LCD_nCS0 bit LCD.LCD_CNTL_REG[0] at 0. LCD controller 1 can be selected if LCD_nCS0 is high and the LCD_nCS1 bit LCD.LCD_CNTL_REG[4] is set to 0.

The first part of Figure 21-5 describes the initialization phase. The instructions sent in this phase permit the establishment of the power sequence and the internal control registers (adjusting the frame frequency, LCD inversion mode, and partial mode). See Section 21.4.2, *Register Descriptions*, for more information.

The second part of Figure 21-5 describes the waiting that occurs before the beginning of transmitting data. It is not used in all types of LCD controllers. Thus, for some LCD controllers, when all of the instructions are sent to the LCD controller, the LCD interface waits for a response from the LCD controller indicating that it is ready to establish communication.

Figure 21-5. LCD Controller Initialization



21.3.1.3 Load LCD Controller Display RAM

Loading the display RAM of the LCD controller requires these two steps:

1. Set the LCD_CNTL_REG register to address display RAM.
2. Write data into the FIFO each time the FIFO becomes empty.

Note: Read access to the LCD controller cannot be performed when the FIFO is not empty.

21.3.2 Operational Mode

This section describes the reset/initialization phase, the write data to the LCD controller, and the read data from the LCD controller.

21.3.2.1 Reset/Initialization Phases

A dedicated nRESET pin controls the reset of the external LCD controller. The LCD_nRESET bit LCD.LCD_CNTL_REG[3] controls the nRESET pin.

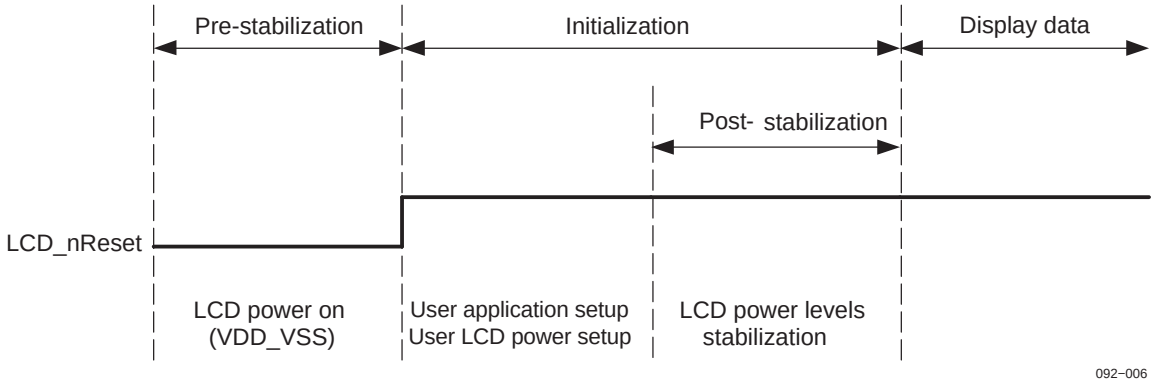
Before the MPU can send display data, the LCD interface must pass two important phases:

- During the pre-stabilization phase, the LCD_nRESET pin must stay low long enough for the LCD power supplies (VDD-VSS) to stabilize. The duration of the pre-stabilization phase depends on the LCD material used. Following the duration, the LCD_nRESET pin is asserted high.
- The initialization phase consists of the MPU providing the user application and LCD power-supply setup by programming the internal instructions of the LCD controller. The initialization phase stops when all LCD power levels are stabilized (post-stabilization phase).

One of the general-purpose timers in the DBB can control durations required by pre-stabilization and post-stabilization phases.

Figure 21-6 shows the LCD reset/initialization phases.

Figure 21-6. LCD Reset/Initialization Phases



21.3.2.2 Write Data to LCD Controller

Data are read from the transmit FIFO of the LCD interface and written to the LCD controller using the following steps:

1. Set the transmit data rate divider (TX_CLOCK_DIV) to LCD controller requirements (fraction [1, 2, 4, or 8] of the 13-MHz GSM clock).
2. Set the RS signal to select either the instruction register set or the embedded display RAM.
3. Load data into the transmit FIFO.

The internal FIFO of the LCD interface is aligned with 16 bits. The external interface with the LCD controller is either 8 bits only or 8/16 bits.

- With an 8-/16-bit LCD controller, the 16-bit (instruction or graphical) data is sent most significant bit (MSB) first.
- With an 8-bit-only LCD controller, two words of 8 bits are required to build 16-bit data.
  - The graphical data of 16 bits is written into two consecutive locations of the graphical RAM (the MSB to the current location and the least significant bit [LSB] to the next location).
  - Instruction data require more attention.

- For 8-bit instructions, the instruction must be duplicated to build a 16-bit word (same MSB and LSB). Instructions are always written to the same address. Another solution uses the NOP instruction as an LSB part with the LCD controller supporting the feature.
- For 16-bit instructions, see the instruction set of the LCD controller for a valid 16-bit instruction, knowing that the MSB is always sent first (most of the time, the MSB is the pseudo-address and the LSB is the meaningful data).

Note: The FIFO is double-port with read and write access. The rates of write and read accesses are different: Write access is more rapid than the read access. During the read access of the FIFO interface, the LCD controller cannot be accessed. TXFIFO reload can be performed under MPU control after one interrupt or under DMA controller control after one DMA request. The data register read of the LCD interface can begin after one interrupt request.

The process of writing data can differ in some LCD controllers. In the LCD module (see the implementation example in Section 21.1.3), RS = 0 is followed by the register selection; thus, the data written in the FIFO is the index of the controller register that we want to access. RS = 1 is then followed by the data we want to write. These data are transferred to the selected register.

Loading graphic data in the graphic RAM of the display takes the following sequence:

- RS = 0 and write index of the WR_GRAM_REG register in the FIFO
- RS = 1 and write display data in the FIFO

21.3.2.3 Read Data From LCD Controller

To read data from the LCD controller, follow these steps:

1. Set the receive data rate divider based on the LCD controller requirements (fraction [1, 2, 4, or 8] of the 13-MHz GSM clock).
2. Set the RS signal to select either the status register or the embedded display RAM.
3. Set the read access activation bit (LCD_start_read).
4. When the read data transfer from the LCD controller is complete (using the interrupt or polling of the read status bit), the MPU can read the data from the received data register.

As for write access, the LCD interface always performs two consecutive 8-bit read accesses to get 16-bit data. This behavior implies some constraints/ limitations on read access operation.

- With an 8-/16-bit LCD controller, data (instruction or graphical) is transferred from the LCD controller to the received data register.
- With an 8-bit-only LCD controller:
  - Two pieces of 8-bit graphical data are transferred from the LCD controller to the received data register (MSB = first data, LSB = second data).
  - The status register is read two times (MSB = first access, LSB = second access) to build 16-bit data of the received data register.

Note: The status register can change between the two read accesses; therefore, the MSB is not necessarily equal to the LSB.

21.4 Register Manual

Table 21-6 summarizes the LCD interface instance.

Table 21-6. Instance Summary

Module Name	MPU Base Address	Size
LCD interface	0xFFFF A000	2 KB

The LCD interface is the TIPB strobe 0 interface; therefore, both the MPU and DMA can access it.

21.4.1 Module Register Mapping Summary

Table 21-7 summarizes the LCD register offset address.

Table 21-7. LCD Register Offset Address

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
CNTL_REG	R/W	16	0x00	0xFFFF A000
LCD_CNTL_REG	R/W	16	0x02	0xFFFF A002
LCD_IF_STS_REG	R	16	0x04	0xFFFF A004
WR_FIFO	W	16	0x06	0xFFFF A006
RD_FIFO	R	16	0x08	0xFFFF A008

21.4.2 Register Descriptions

Table 21-8 through Table 21-12 describe the LCD controller registers.

Table 21-8. CNTL_REG

Address Offset	0x00		
Physical address	0xFFFF A000	**Instance**	LCD
Description	Control register		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N_DUMMY		MIN_FRAME_SIZE		SUSPEND_EN	FLIP_BYTES	MODE	DMA_EN	LCD_READ_EVENT_IT _EN	FIFO_EMPTY_IT_EN	RX_CLOCK_DIV		TX_CLOCK_DIV		CLOCK13_EN	SOFT_NRST

Bits	Field Name	Description	Type	Reset
15:14	N_DUMMY	Number of dummy read cycles R/W 00 00: 0 dummy read cycle 01: 1 dummy read cycle 10: 2 dummy read cycles	R/W	00
13:12	MIN_FRAME_SIZE	Minimum number of transmitted words	R/W	11
11	SUSPEND_EN	0: Transmission never suspended 1: Low the suspension of transmission (used in emulation mode)	R/W	1
10	FLIP_BYTES	0: Read/write MSB first 1: Read/write LSB first	R/W	0
9	MODE	0: Mode 6800 1: Mode 8086	R/W	0
8	DMA_EN	0: Disable DMA capability 1: Enable DMA capability	R/W	0
7	LCD_READ_EVENT_IT _EN	Enable interrupt for received data 0: Interrupt disabled 1: Interrupt enabled	R/W	0

Bits	Field Name	Description	Type	Reset
6	FIFO_EMPTY_IT_EN	Enable FIFO empty interrupt 0: Interrupt disabled 1: Interrupt enabled	R/W	0
5:4	RX_CLOCK_DIV	Receive data rate from LCD controller 00: 13 MHz / 8 01: 13 MHz / 4 10: 13 MHz / 2 11: 13 MHz / 1	R/W	00
3:2	TX_CLOCK_DIV	Transmit data rate from LCD controller 00: 13 MHz / 8 01: 13 MHz / 4 10: 13 MHz / 2 11: 13 MHz / 1	R/W	00
1	CLOCK13_EN	Enable LCD interface module 13-MHz working clock 0: Clock stopped 1: Clock enabled	R/W	0
0	SOFT_NRST	Write 0 to reset the LCD interface module. The bit returns high when the LCD interface has been reset.	R/W	1

Table 21-9. LCD_CNTL_REG

Address Offset	0x002		
Physical address	0xFFFF A002	**Instance**	LCD
Description	LCD controller register		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										di_do_sel_ngate	LCD_nCS1	LCD_nRESET	LCD_start_read	LCD_RS	LCD_nCS0

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Reads return 1.	R	1
5	di_do_sel_ngate	Automatic I/O configuration of the LCD_DATA bus to output when in deep sleep. 0x1: Disable 0x0: Enable	R/W	1
4	LCD_nCS1	Selection of LCD controller 1. If LCD_nCS0 is low, then LCD controller 1 is not selected. If LCD_nCS0 is high, then: 0: LCD controller 1 selected 1: LCD controller 1 not selected	R/W	1
3	LCD_nRESET	Initialization of the LCD controller When this bit is low, the LCD controller is reset.	R/W	0
2	LCD_start_read	Toggle bit used to start a read access on LCD controller. 1: Start read access	R/W	0

Bits	Field Name	Description	Type	Reset
1	LCD_RS	Selection of instruction/data type Active level depends on the controller in use. The most of controller used the following convention: 0: Instruction. The instruction can be the address of the control register that we want to access. 1: Data	R/W	0
0	LCD_nCS0	Selection of LCD controller 0 0: LCD controller 0 is selected. 1: LCD controller 0 is not selected.	R/W	1

Table 21-10. LCD_IF_STS_REG

Address Offset	0x004		
Physical address	0xFFFF A004	**Instance**	LCD
Description	LCD interface status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													LCD_read_event_ status	FIFO_FULL_ STATUS	FIFO_EMPTY_ STATUS

Bits	Field Name	Description	Type	Reset
15:3	Reserved	Reads return 1.	R	1
2	LCD_read_event_ status	0: No read operation/read ongoing 1: Data received from LCD controller This bit is auto-reset to 0 on a read access to the RD_REG register.	R	0
1	FIFO_FULL_ STATUS	0: FIFO is not full. 1: FIFO is full.	R	0
0	FIFO_EMPTY_ STATUS	0: FIFO is not empty. 1: FIFO is empty.	R	1

Table 21-11. WR_FIFO

Address Offset	0x006		
Physical address	0xFFFF A006	**Instance**	LCD
Description	LCD write data FIFO		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_TX															

Bits	Field Name	Description	Type	Reset
15:0	DATA_TX	7:0 = Transmit FIFO register LSBs 15:8 = Transmit FIFO register MSBs	W	Undefined

Table 21-12. RD_REG

Address Offset	0x008		
Physical address	0xFFFF A008	**Instance**	LCD
Description	LCD read data register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_RX															

Bits	Field Name	Description	Type	Reset
15:0	DATA_RX	7:0 = Received data register LSBs 15:8 = Received data register MSBs	R	Undefined

WARNING

This is a register and not a FIFO. Therefore, the CPU must read data before starting a new read access to the LCD controller or data will be lost.

Security Features

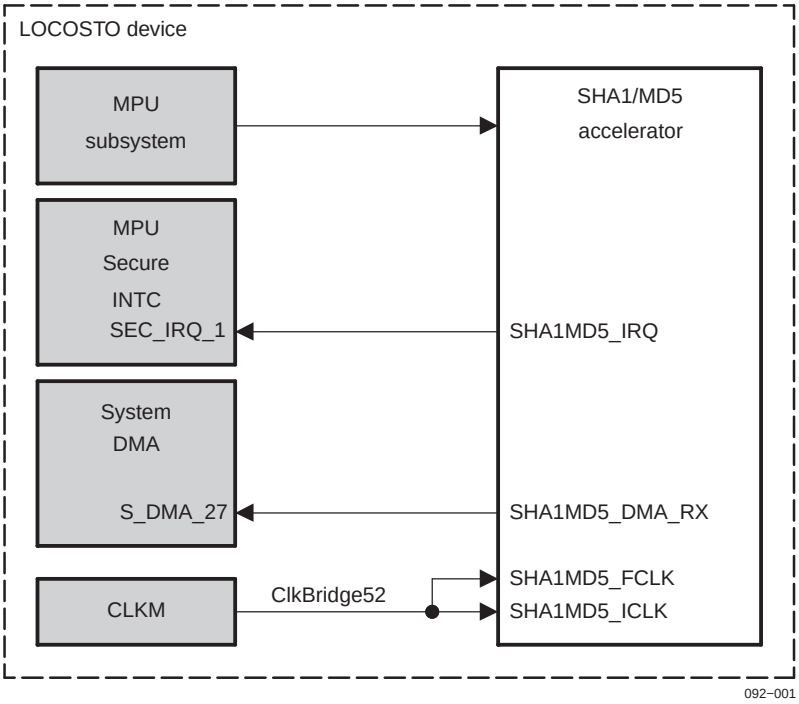
This chapter describes the security features of the LOCOSTO device. All security features described in this chapter are available in both general-purpose (GP) and high-security (HS) devices. Contact your TI representative for more information on high-security devices.

Topic **Page**

22.1 SHA1/MD5 Accelerator

The SHA1/MD5 (SHAM1) module provides hardware accelerated hash functions and can run either the SHA1 algorithm (in compliance with the FIPS 180-1 standard) or the MD5 message digest algorithm developed by Rivest in 1991. The algorithms produce a condensed representation of a message or a data file, called digest or signature, which can then be used to verify the message integrity (see Figure 22-1).

Figure 22-1. SHA1/MD5 Module Overview



The SHA1/MD5 crypto-accelerator includes the following main features:

- Capability to hash up to 128M bytes of data in a single operation and produce a 160-bit signature for the SHA1 or a 128-bit signature for the MD5.
- The SHA1 algorithm can handle input messages up to 235 bits long (see the following note). After padding the message into 512-bit blocks, the processing requires 80 cycles per block for SHA1 and 64 cycles per block for MD5.
- The register bank holds the data input buffer, digest, digest counts, and control/status through a 32-bit access.
- The data input buffer for both SHA1 and MD5 is 512 bits.
- Double-buffering of the input buffer lets the host queue the next hash block when processing the current block.
- DMA (receive) and IRQ channels allow the transfer of 32-bit data. The DMA is used to load the message to digest to the SHA1/MD5; the IRQ signals that the digest is ready.

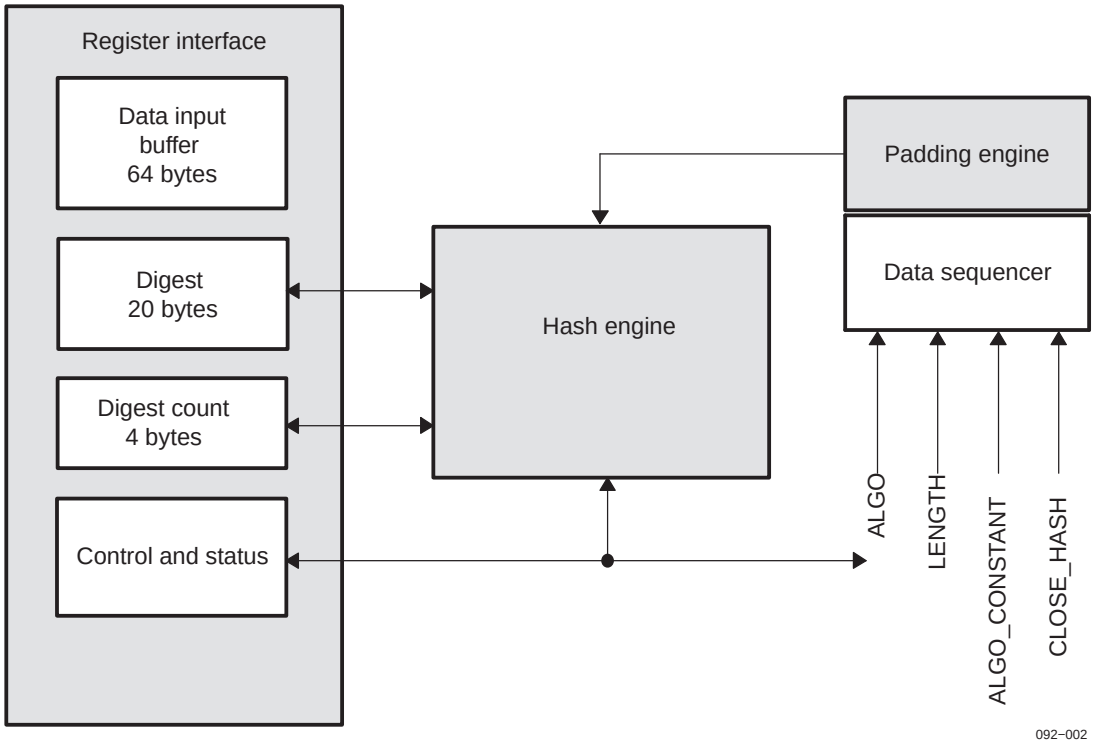
Note: The following restrictions apply to the SHA1 module in regard to the FIPS 180-1 standard:

- A single hash operation cannot process more than 2^{30} bits (2^{27} bytes or 128M bytes). Therefore, messages sized between 2^{27} and 2^{32} bytes must be processed iteratively by the packet of data smaller than 2^{27} bytes.
- The message to hash must be smaller than 2^{35} bits (2^{32} bytes), although the FIPS 180-1 allows up to a 2^{64}-bit message. This is due to the digest count SHAMD5.SHA_DIGCNT register, which is only 32 bits wide.
- The message must be a multiple of bytes (the LENGTH field in SHAMD5.SHA_CTRL must be defined in bytes). The FIPS 180_UnicodeEncodeError_1 does not necessarily require messages to be a multiple of bytes. The size is specified in bits up to 2^{64} bits. For messages that are not multiples of bytes, the padding to the byte boundary must be performed by the software first to get a message with a multiple of bytes. The hash then can be hardware-accelerated by the SHA1/MD5 module.

22.1.1 Architecture Overview

Figure 22-2 shows the module architecture consisting of four primary blocks: the register interface, padding engine, hash engine, and data sequencer.

Figure 22-2. SHA1/MD5 Module Architecture



22.1.1.1 Register Interface

The register interface is a bank of registers written through the interface bus. This register bank holds the data input buffer, digest, digest count, and control/ status.

The data input buffer consists of sixteen 32-bit registers, or a total of 64 bytes. The hash data for both the SHA1 and MD5 algorithms are in blocks of 64 bytes. Thus, the registers provide sufficient storage for one block. Once a block is queued and the hash engine is idle or the processing of the prior block is complete, the data is transferred to a working register bank internal to the hash engine. This empties the input buffer, allowing it to accept the next 64-byte block of data. Double-buffering the input data lets the host queue the next hash block while processing the current block, thus maximizing module use.

The digest consists of five 32-bit registers providing up to 160 bits of storage for a signature, which covers the needs of the SHA1 and MD5 algorithms. The digest count register (SHAM1.SHA_DIGCNT) is a 32-bit register that contains the current number of bytes that have been hashed. In addition, control and status registers (SHAM1.SHA_CTRL and SHAM1.SHA_MASK) contain configuration bits (ALGO, LENGTH, ALGO_CONSTANT, CLOSE_HASH, AUTOIDLE, DMA_EN, IT_EN, and SOFT_RESET) and status bits (OUTPUT_ READY and INPUT_READY) for the hash operation.

22.1.1.2 Padding Engine

The padding engine operates on the input data before it is inserted into the hash engine. A pad is applied to the end of the input data when the hash is to be closed following processing of the entire input packet. In other words, the pad is applied only to the final 64-byte block of data, not to the intermediate blocks.

Padding is performed in accordance with the SHA1 and MD5 standard algorithms.

22.1.1.3 Hash Engine

The hash engine is responsible for data processing. The hash engine receives a 64-byte block of input data and, through a series of nonlinear functions, produces a 160-bit output digest for SHA1 and a 128-bit output digest for MD5.

As defined in the FIPS 180-1 standard, a SHA1 digest is computed iteratively through a sequence of 80 operations, or rounds. The MD5 requires 64 rounds.

The hash engine performs one round/cycle for both SHA1 and MD5. Thus, for each 64-byte input block, 80 cycles are required for a SHA1 hash and 64 cycles for an MD5 hash.

22.1.1.4 Data Sequencer

The data sequencer manages the data flow between the bus interface and the hash engine. In addition, the data sequencer provides a load signal to commence hashing, a sequence counter to step the engine through the individual rounds of the algorithms, and a write strobe to output the final digest.

22.1.2 Integration Features

22.1.2.1 Clocking and Power Management

Clocks

The SHA1/MD5 accelerator operates from the ClkBridge52 clock of the CLKM module, which is used for the SHA1/MD5 configuration interface and the functional interface.

Idle Mode

Functional and interface clock gating for power saving is activated by setting the SHAM1.SHA_MASK[0] AUTOIDLE bit to 1. In this mode, the clocks are active only when registers are accessed (that is, when the interface clock is used) or when a hashing operation is ongoing (that is, when the functional clock is used). When this bit is set to 0, both clocks are free-running.

22.1.2.2 Reset Management

Hardware reset is available from the core_protected_nRST signal, which is issued by the PRRM module.

Software reset is performed by setting the SHAM1.SHA_MASK[1] SOFTRESET bit to 1.

The behavior of the software and hardware resets is the same, except that the software reset bit resets this module without affecting the whole core_protected reset domain.

22.1.2.3 Hardware Requests

DMA Requests

One DMA request is mapped to the DMA controller, as defined in Table 22-1.

Table 22-1. SHA1/MD5 DMA Request

Attributes	Values	Name	Mapping	Comments
DMA request	1	SHA1MD5_RX_ DMA_REQ	DMA_27	Destination: DMA controller

Interrupts

Interrupt request SEC_IRQ_1 is generated when the digest is ready for transmission. The destination is the MPU secure interrupt handler.

22.1.3 SHA1/MD5 Operation Description

This module runs either the SHA1 or MD5 algorithm, depending on the SHAM1.SHA_CTRL[2] ALGO bit value (SHA1 mode when the bit is set to 1, MD5 mode when it is set to 0).

22.1.3.1 SHA1 Mode

22.1.3.1.1 Starting a New Hash

To start a new hash, configure SHAM1.SHA_CTRL by performing the following steps:

1. Set the ALGO (bit 2) to 1 (select SHA1).
2. Set the ALGO_CONSTANT (bit 3) to 1 to initialize all registers from SHAM1.SHA_DIGEST_A through SHAM1.SHA_DIGEST_E (with default value specified by the algorithm [either SHA1 or MD5]) and set the SHAM1.SHA_DIGCNT register to 0.
3. Specify the hash data LENGTH (bit 31:5) to process in bytes.
4. Set CLOSE_HASH (bit 4) to let the SHA engine perform the padding. If SHA is computed in one shot, the length of the message can be any value up to 128M bytes. To process an intermediate SHA digest, CLOSE_HASH is set to 0, in which case the packets to be hashed must be 64 bytes; the last packet must be hashed with the CLOSE_HASH bit set to 1.

   Once the configuration is done, the SHA engine can receive the data to process (the SHAM1.SHA_CTRL[1] INPUT_READY status bit is equal to 1). Data must be written to the 16 x 32-bit SHAM1.SHA_DIN_X registers that provide storage for one 64-byte block of data. Unless the CLOSE_HASH bit is set, all SHAM1.SHA_DIN_X 64-byte input buffers must be filled. Data can be written either by single write accesses to the 16 registers from a processor or by a DMA transfer.

   For a DMA transfer, the SHAM1.SHA_MASK[3] DMA_EN is set to 1 before starting the new hash. Thus, a DMA request is generated each time the SHAM1.SHA_DIN_X 64-byte input buffer is empty.

   Configure the DMA to transfer 16 data of 32-bit each time it is triggered by a DMA request from the module. The destination address for the DMA must be defined in the address mapping of the module SHA1/MD5 with offset 0x1C, and a constant addressing mode must be set. The 16 data written to address 0x1C are sent to the 16 SHAM1.SHA_DIN_X registers ranging from 0x1C through 0x58.

   The module detects that a 64-byte block is available, then it moves the data to a working register space for processing, and asserts the SHAM1.SHA_CTRL[1] INPUT_READY bit to 1. If the SHAM1.SHA_MASK[3] DMA_EN bit is set, a new DMA request triggers a new block transfer; otherwise, the processor polls the SHAM1.SHA_CTRL[1] INPUT_READY bit and writes the 16 data of 32 bits when it equals 1.

   This operation is repeated until the length of the message to hash is reached. Then the SHAM1.SHA_CTRL[0] OUTPUT_READY bit indicates that the hash operation is complete. If the SHAM1.SHA_MASK[2] IT_EN bit is set, an interrupt (active low) is also generated to indicate the hash completion. Writing 1 into the SHAM1.SHA_CTRL[0] OUTPUT_READY bit clears this interrupt.

   The processor can then read the five digest registers A through E that contain the 160-bit SHA1 hash result. If the hash is an intermediate result of a larger hash, then the digest count register must also be read and saved.

WARNING

In the SHA1 mode, the results are in big endian. The first byte of the sum digest is in SHA_DIGEST_A[31:24], the second byte is in SHA_DIGEST_A[23:16], and the last byte of the sum is in SHA_DIGEST_E[7:0].

22.1.3.1.2 Continue a Prior Hash

To continue a prior hash, the SHAM1.SHA_DIGEST_A through SHAM1.SHA_DIGEST_E registers and SHA_DIGCNT register must be restored with the results of the prior hash.

The SHAM1.SHA_CTRL register then must be configured: the SHAM1.SHA_CTRL[3] ALGO_CONSTANT bit is set to 0; the SHAM1.SHA_CTRL[2] ALGO bit is set to 1 (for SHA); the SHAM1.SHA_CTRL[31:5] LENGTH field defines how much data remains to be processed and must be written with the size of the new data packet to be hashed in bytes. If this new data packet is the last one in the input message, then the SHAM1.SHA_CTRL[4] CLOSE_HASH bit must be set to 1.

Hash completion is indicated the same way as for a new hash. The 160-bit result can be read in the SHAM1.SHA_DIGEST_A through SHAM1.SHA_ DIGEST_E registers. The SHAM1.SHA_DIGCNT register returns restored Digest Count + Length when it is read, and the hashing is complete.

22.1.3.1.3 Close a Hash

The amount of data to hash is not necessarily a multiple of 64 bytes. SHAM1.SHA_CTRL[4] CLOSE_HASH bit is set to append padding so that the message size becomes a multiple of 64 bytes. However, the message is necessarily a multiple of bytes. Consequently, a minimum of 9 bytes must be added to the message. Nine bytes is the minimum number of bytes that contains the minimum 65 bits padding specified by the FIPS 180-1.

If the size of the last block of data is less than or equal to 55 bytes, no additional 64-byte block is required. But if the last block of data contains more than 55 bytes, an extra 64-byte block must be added to make the padding as specified by FIPS 180-1. This extra block is added automatically by the hardware, so the module is fed with a 64-byte block of data. On the last block of data, however, appending a pad may result in the creation of an extra 64-byte block.

Data written beyond the total number of bytes specified in the SHAM1.SHA_CTRL[31:5] LENGTH field is ignored. Therefore, if the DMA is used, it does not have to be reprogrammed specifically to handle the last block transfer. The DMA can transfer sixteen 32-bit data for the last block. The data beyond the total number of bytes specified in the LENGTH field is ignored and the padding is done to the remaining data to process as specified by the LENGTH field. Therefore, it neither corrupts the hash result nor stalls the DMA because extra data is acknowledged.

The one or two last blocks that contain the padding are processed as the other blocks. Hash completion is then indicated in the same way as for a new hash, and the 160-bit result can be read in the SHAM1.SHA_DIGEST_A through SHAM1.SHA_DIGEST_E registers. The SHAM1.SHA_DIGCNT register returns restored Digest Count + Length when it is read and hashing is completed.

Assuming a message of 129 bytes, Table 22-2 shows the SHA digest for three passes, and Table 22-3 shows the SHA digest for one pass.

Table 22-2. SHA Digest Processed in Three Passes

	DIGEST(A to E)	SHA_DIGCNT	SHA_CTRL	SHA_DIN_X(0 to 15)
1st pass			Write: LENGTH: 64 ALGO: 1/0 (SHA/MD5) ALGO_CONSTANT: 1 CLOSE_HASH: 0	First 64 bytes of message
2nd pass	Round 1 digest calculation	Write: 64	Write: LENGTH: 64 ALGO: 1/0 (SHA/MD5) ALGO_CONSTANT: 0 CLOSE_HASH: 0	Second 64 bytes of message
3rd pass	Round 2 digest calculation	Write: 128	Write: LENGTH: 1 ALGO: 1/0 (SHA/MD5) ALGO_CONSTANT: 0 CLOSE_HASH: 1	Last byte of message
	Final digest	Read: 129		

If the three passes are not performed in succession, the digest registers must be saved and restored for the next use of the SHA engine. If the rounds are performed consecutively , there is no need to do anything with the digest registers.

Table 22-3. SHA Digest Processed in One Pass

	DIGEST(A to E)	SHA_DIGCNT	SHA_CTRL	SHA_DIN_X(0 to 15
1st pass			Write: LENGTH: 129 ALGO: 1/0 (SHA/ MD5) ALGO_CONSTANT: 1 CLOSE_HASH: 1	First 64 bytes of message
	Round 1 digest calculation			Second 64 bytes of message
	Round 2 digest calculation			Last byte of message
	Final digest	129		

22.1.3.2 MD5 Modes

22.1.3.2.1 Start a New Hash

To start a new hash, perform the following steps:

1. Set the SHAM1.SHA_CTRL[2] ALGO bit to 0 to select the MD5 algorithm.
2. Set the SHAM1.SHA_CTRL[3] ALGO_CONSTANT bit to 1 to initialize all registers from SHAM1.SHA_DIGEST_A through SHAM1.SHA_DIGEST_ D registers (with default values specified by the algorithm) and setting the SHAM1.SHA_DIGCNT register to 0.
3. Specify the SHAM1.SHA_CTRL[31:5] LENGTH field of the hash data to process in bytes.
4. Set the SHAM1.SHA_CTRL[4] CLOSE_HASH bit to let the SHA1/MD5 engine do the padding. If MD5 is computed in one shot, the length of the message can be any value up to 128M bytes. To process an intermediate MD5 digest, CLOSE_HASH is set to 0, in which case packets to be hashed must be 64 bytes; the last packet must be hashed with CLOSE_HASH bit set to 1.

Once the configuration is completed, the hash engine can receive the data to process (SHAM1.SHA_CTRL[1] INPUT_READY bit is equal to 1). Data must be written to the 16 x 32-bit SHAM1.SHA_DIN_X registers that provide storage for one 64-byte block of data. Unless the SHAM1.SHA_CTRL[4] CLOSE_HASH bit is set, the SHAM1.SHA_DIN_X 64-byte input buffer must be completely filled. Data can be written either by single write accesses to the 16 registers from a processor or by a DMA transfer.

To use DMA transfer, the SHAM1.SHA_MASK[3] DMA_EN bit is set to 1 before starting the new hash. This way a DMA request is generated every time the SHAM1.SHA_DIN_X 64-byte input buffer is empty.

The DMA must be configured to transfer 16 data of 32-bits each time it is triggered by a DMA request from the module. The destination address for the DMA must be defined in the address mapping of the module SHA1/MD5 with offset 0x1C, and a constant addressing mode must be set. The 16 data written to address 0x1C are sent to the 16 SHAM1.SHA_DIN_X registers ranging from 0x1C through 0x58.

The module detects that a 64-byte block is available; it moves the data to a working register space for processing and asserts the SHAM1.SHA_CTRL[1] INPUT_READY bit to 1. If the SHAM1.SHA_MASK[3] DMA_EN bit is set to 1, then a new DMA request triggers a new block transfer; otherwise, the processor polls the SHAM1.SHA_CTRL[1] INPUT_READY bit and writes the 16 data of 32 bits when it equals 1.

This operation repeats until the length of the message to hash is reached. Then the SHAM1.SHA_CTRL[0] OUTPUT_READY bit indicates that the hash operation is complete. If the SHAM1.SHA_MASK[2] IT_EN bit is set, an interrupt (active low) is also generated to indicate the hash completion. Writing 1 into the SHAM1.SHA_CTRL[0] OUTPUT_READY bit clears this interrupt.

The processor can then read the four digest registers A through D that contain the 128-bit MD5 hash result. The digest count register must also be read and saved if the hash is an intermediate result of a larger hash.

WARNING

In MD5 mode, the results are in little endian. The first byte of the sum digest is in SHA_DIGEST_A[7:0], the second byte is in SHA_DIGEST_A[15:8], and the last byte of the sum is in SHA_DIGEST_E[31:24].

22.1.3.2.2 Continue a Prior Hash

To continue a prior hash, the SHAM1.SHA_DIGEST_A through SHAM1.SHA_DIGEST_D registers and SHAM1.SHA_DIGCNT registers must be restored with the results of the prior hash. Then, the SHAM1.SHA_CTRL[3] ALGO_CONSTANT bit must be set to 0 and the SHAM1.SHA_CTRL[2] ALGO bit is set to 0 (for MD5). The SHAM1.SHA_CTRL[31:5] LENGTH field defines how much more data is processed and must be written with the size of the new data packet to be hashed in bytes. If this new data packet is the last one in the input message, then the SHAM1.SHA_CTRL[4] CLOSE_HASH bit must be set to 1.

Hash completion is indicated the same way as for a new hash and the 128-bit result can be read in the SHAM1.SHA_DIGEST_A through SHAM1.SHA_ DIGEST_D registers. SHAM1.SHA_DIGCNT returns restored Digest Count + Length after it is read and hashing is complete.

22.1.3.2.3 Close a Hash

The amount of data to hash is not necessarily a multiple of 64 bytes. In this case, the SHAM1.SHA_CTRL[4] CLOSE_HASH bit must be set to append padding so that the message size becomes a multiple of 64 bytes. Refer to the previous SHA1 algorithm for further details on padding.

The module is fed with a 64-byte block of data, as long as enough data is available. On the last block of data, however, a pad is appended, which may result in the creation of an extra 64-byte block.

Data written beyond the total number of bytes specified in the SHAM1.SHA_CTRL[31:5] LENGTH field is ignored. Therefore, if the DMA is used, it does not have to be reprogrammed specifically to handle the last block transfer. The DMA can also transfer sixteen 32-bit data for the last block. The data beyond the total number of bytes specified in the SHAM1.SHA_ CTRL[31:5] LENGTH field is ignored, and the padding is done to the remaining data to process as specified by the LENGTH field. Therefore, it neither corrupts the hash result nor stalls the DMA because extra data is acknowledged.

The one or two last blocks that contain the padding are processed in the same way as the other blocks. Hash completion is then indicated the same way as for a new hash, and the 128-bit result can be read in the SHAM1.SHA_DIGEST_A through SHAM1.SHA_DIGEST_D registers. SHAM1.SHA_DIGCNT returns restored Digest Count + Length when it is read and hashing is completed.

22.1.3.3 Interrupt Generation

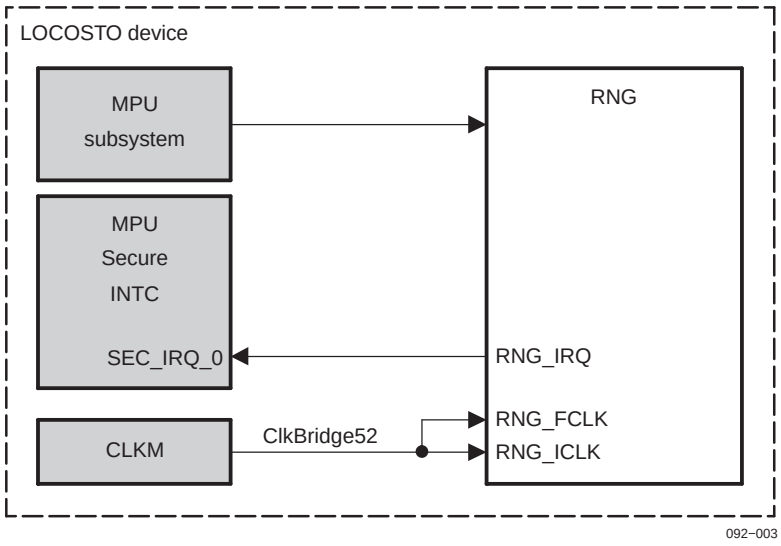
If the SHAM1.SHA_MASK[2] IT_EN bit is set to 1, an interrupt is generated at the completion of the hash with the following steps:

1. Receive last block of data (<= 64 bytes). Number of data bytes defined by SHAM1.SHA_CTRL[31:5] LENGTH field is received in SHAM1.SHA_DIGEST_A through SHAM1.SHA_DIGEST_D registers.
2. Padding is applied to the last block of data, if required.
3. Hash the last block of data (80 cycles in SHA1 mode and 64 cycles in MD5 mode).
4. If required, an extra 64-byte block of data is added to complete the padding.
5. Hash this extra block of data (80 cycles in SHA1 mode and 64 cycles in MD5 mode).
6. An interrupt is generated (active low).

22.2 RNG Accelerator

The hardware random-number generator (RNG1) provides a true, nondeterministic noise source for the purpose of generating keys, initialization vectors, and other requirements. The RNG is built on two ring oscillators that create unpredictable output to feed a complex nonlinear combinatorial circuit. The output is then postprocessed for statistical robustness to meet NIST requirements of FIPS 140-1 and FIPS 186-1 based on ANSI X9.17 Annex C (see Figure 22-3).

Figure 22-3. RNG Module Overview



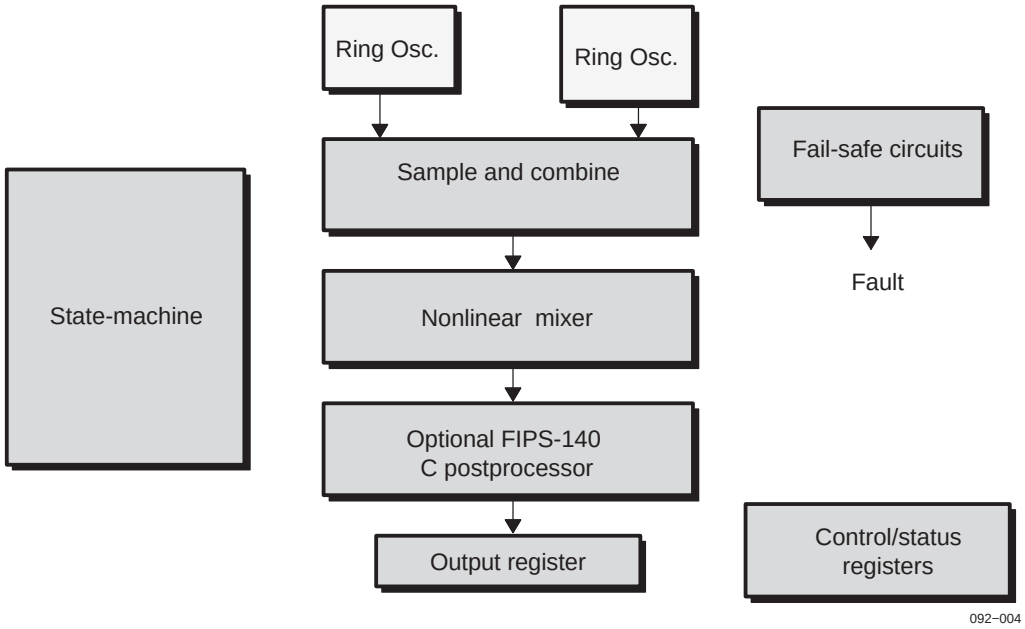
The main features of the RNG follow:

- The RNG is based on two ring oscillators (shot noise) to create entropy. To generate this entropy, the system requires 225 (33554432) system clock cycles to produce the first random output. Then, it takes 160 system clock cycles to produce each subsequent 32-bit random number.
- The RNG architecture is based on four LFSRs followed by a nonlinear entropic hasher.
- The random number is accessible to the applications in a 32-bit read-only register. Once the register is read, the RNG immediately generates a new value, which is then shifted into the output register.
- A busy bit is provided in the status register to ensure that a new random number is generated since the last access.
- The RNG provides a built-in self-test that checks the number of consecutive bits sampled to provide the statical robustness required by FIPS 140.
- The internal power-saving mode is built to carefully manage the entropy previously generated.
- The IRQ channel signals that the data is ready to be transferred

22.2.1 Architecture Overview

The RNG core is designed using dual-shot noise generators that create unpredictable jittering output when asynchronously sampled by the system clock provided to the RNG. The outputs from the shot noise generators feed a complex nonlinear combinatorial circuit (mixer) that produces the final RNG output (see Figure 22-4).

Figure 22-4. RNG Diagram



22.2.2 Integration Features

22.2.2.1 Clocking and Power Management

Clocks

The RNG accelerator operates from the ClkBridge52 clock of the CLKM module, which is used for the RNG configuration interface and the functional interface.

Idle Mode

Functional and interface clock gating for power saving is activated by setting the RNG1.RNG_MASK[0] AUTOIDLE bit to 1. In this mode, clocks are active only when registers are accessed (interface clock is used) or a random number generation is on going (functional clock is used). When this bit is set to 0, both clocks are free-running.

22.2.2.2 Reset Management

Hardware reset is available from the core_protected_nRST signal issued by the PRCM module.

To reset the software, set the RNG1.RNG_MASK[1] SOFTRESET bit to 1.

The behavior of software reset and hardware reset is the same, except that the software reset bit resets this module without affecting the whole core_protected reset domain.

22.2.2.3 Hardware Requests

DMA Requests

No DMA request is generated in this module.

22.2.2.4 Interrupts

An interrupt request, RNG_IRQ (mapping to SEC_IRQ_0) is generated when data is ready for transmission. The destination is the MPU secure interrupt handler.

22.2.3 RNG Operation Description

Operate the RNG mode with the following actions:

- Initial latency after reset

  This ensures that sufficient entropy is built up in the RNG. It takes 2^{25} (33554432) system clock cycles after RNG reset to produce the first random output. Reading the random number output register (RGN1.RNG_OUT) triggers the generation of the next random number and the RNG1.RNG_STAT[0] BUSY bit is set to 1.

- Generate random number

  It only takes 160 system clock cycles to produce each subsequent 32-bit random number once the initial reset latency has elapsed.

  The random numbers are accessible to the application in a 32-bit read-only register (RNG1.RNG_OUT). Once the register is read, the RNG module immediately generates a new value, which is available after 160 system clock cycles and is then shifted into the output register.

- Read random number

  To ensure that a new random number has been generated since the last RNG_OUT register is read, a busy bit is provided in the RNG1.RNG_STAT register. It can be polled when a new random number is required to make sure that the new number is available and can be read in the RNG1.RNG_OUT register.

  Another way of doing this is to enable the RNG interrupt in the with the RNG1.RNG_MASK [2] IT_EN bit. The interrupt is thus generated when the random number generation is complete and can cause an interrupt subroutine of a processor to read this new random number.

- Go into sleep mode

  The safest way to go into sleep mode is to stop reading the last-generated random number. This requires that the input clock be present so that the module can jump to its different state. At this state, the rings and the 24-bit counter (if the RNG1.RNG_MASK[0] AUTOIDLE bit is set to 1) are off. The RNG can be kept in this state indefinitely. As soon as a random number is read, the module wakes up.

  The second safest way to go into the sleep mode is to cut the input clock. The module does not jump to the different states. The entropy is preserved in the system and cutting the clock does not affect the randomness. The clock can be restarted at any time.

  There is no software reset required for these two methods.

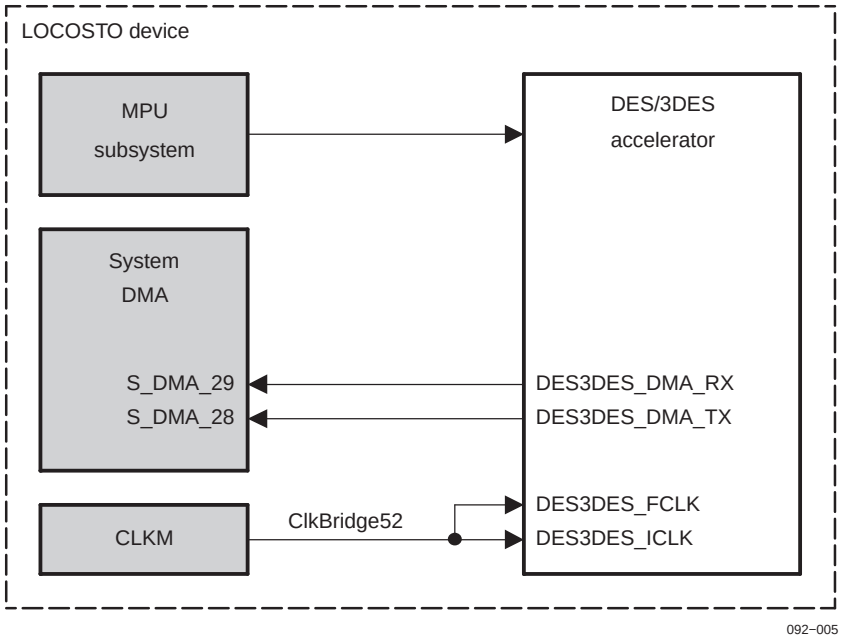
If an application does not need the RNG any more and wants to go into deep sleep mode without waiting, the application can write 0 in the RNG1.RNG_CONFIG[11:6] RESET_COUNT field, and the input system clock can be switched off. If this happens, and if a random number is needed later, then a soft reset is required, as randomness cannot be ensured. The penalty is that 2^{25} cycles are required before the first random number is ready. Therefore, this method must be used as a last resort when RNG is no longer used.

22.3 DES/3DES Accelerator

The DES/3DES security module provides hardware-accelerated data encryption and decryption functions. It can run either the single DES or the triple DES (3DES) algorithm in compliance with FIPS 46-3 standard. It supports ECB (electronic codebook) and CBC (cipher block chaining) modes of operation in hardware (see Figure 22-5).

DES is a symmetric algorithm in that the encryption and decryption keys are identical. Each triple DES encrypt/decrypt operation is a compound of DES encrypt and decrypt operations.

Figure 22-5. DES/3DES Bus System Overview



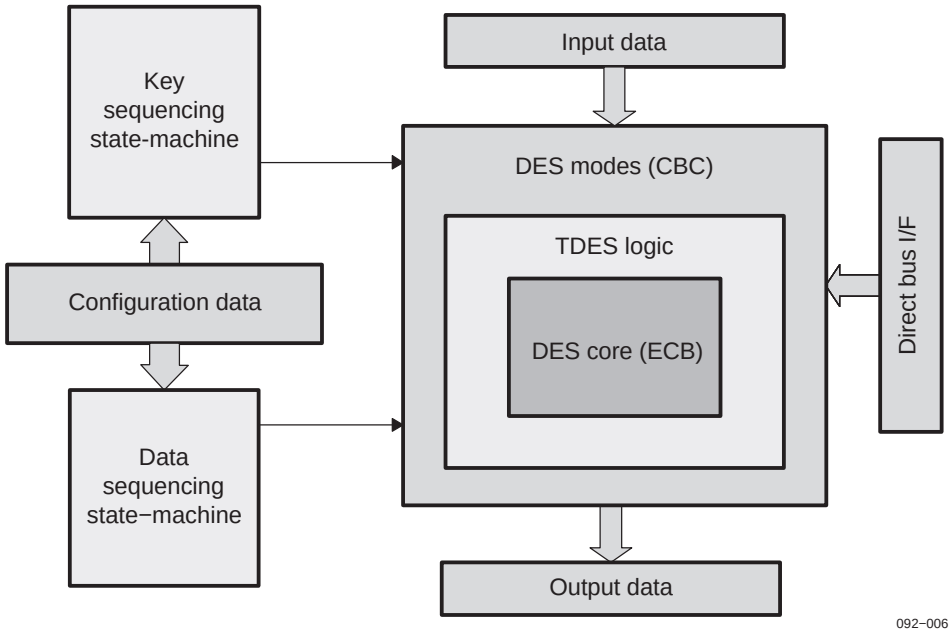
The main features of the DES/3DES crypto accelerator are:

- ECB and CBC modes are supported for encryption and decryption.
- Very fast pipelined architecture: The core executes 16 rounds of DES in 16 clocks cycles and 3DES in 48 clocks cycles per 8-byte data block.
- The embedded key scheduler generates the 16 subkeys from the key input required for the encryption of a single DES block.
- The embedded data sequencer is designed to support continuous DES operations by monitoring the external input buffer until a 64-bit block is available and to save it to the internal working registers of the DES/3DES core.
- For single DES operation, only a 64-bit key is required through hardware direct bus or L4 interface register. For 3DES operation, three 64-bit keys are required through the L4 interface registers only.
- DMA (transmit and receive) channels allow the transfer of 8-byte data.

22.3.1 Architecture Overview

As shown in Figure 22-6, the module architecture consists of primary blocks: the configuration data or register interface, the DES core, the data sequencer, the CBC mode wrapper, the key sequencer, and the TDES wrapper.

Figure 22-6. DES Secure IP Diagram



22.3.1.1 Configuration Data

Configuration and data for the DES/3DES module is provided through a 32-bit wide register interface, which is accessible on the interface bus. Configuration of the engine entails keys, initialization vectors, mode specification, and mask enabling.

The data input buffers consist of two 32-bit registers, or a total of 8 bytes, and are used to deliver packets of data to the DES core. The registers provide sufficient storage for one block. Similarly, the two 32-bit output buffers hold the DES output data until they are read out by the host. The same register addresses are used to access the input buffers on a write and the output buffers on a read. Several status bits provide the flow control to manage the transfers of packets of data. The mask enable register enables or disables input and output DMA requests, free-running clock, loading of the DES keys with the host or through the direct bus interface and, once the configuration is set, starts the transfer and processing of data.

22.3.1.2 DES Core (ECB)

The DES core is an efficient, pipelined implementation of the DES block cipher algorithm. For the DES core, the 16 rounds (iterations) of the DES algorithm are performed in 16 clock cycles. The composition of the DES core is twofold. The main data path operates on the input block, performing the required expansion, permutation, and substitution operations. The key scheduler generates a new round (iteration) key XORed (as part of the cipher function) with the data.

By itself, the implemented DES core provides ECB compliant DES output. The ECB mode is a basic cryptographic method, which transforms the 8-byte data input into 8-byte data output. The analogy to codebook means the same plain text block always produces the same cipher text block for a given key. The ECB core device must encrypt data in integer multiples of 64 bits. If less than 64 bits is available for encryption, then the least significant bits of the unused portion of the input block must be padded before ECB encryption. The corresponding decrypting device must discard these padding bits after decryption of the block of cipher text.

22.3.1.3 Data Sequencer

The data sequencer manages data flow to and from the DES core. Because the DES core does not contain an internal buffer space, it must communicate efficiently with the external data buffers to ensure the highest throughput from the engine. For data input, the data sequencer monitors the input buffer until an 8-byte block is available. If the DES core is idle, it writes this data block to the internal working registers of the DES core, thus clearing the input buffers for the next block.

Upon completion of an encrypt or decrypt operation, the data sequencer writes the DES output data block to the output buffers. If the output buffers are full at the time of completion, the DES core is held until the buffer is cleared. This has the negative effect of stalling the engine, thus reducing throughput. Although the data sequencer is designed to support uninterrupted packet encryption, the host must properly manage the input and output packet buffers to achieve optimum performance.

22.3.1.4 Key Sequencer and TDES Wrapper

The key sequencer is primarily active during TDES encryption, which is accomplished by invoking the DES core three consecutive times, and feeding the output back to the input after each of the first two iterations. Each time the engine is invoked, a new key is entered and the direction of operation is alternated (encrypt/decrypt), as defined in the TDES standard.

22.3.1.5 DES Modes (CBC)

For most applications, you must operate in the cipher-block-chaining (CBC), which provides an error characteristic that is valuable in protecting against fraudulent data alteration. Chaining is a feedback mechanism that links the encryption of the current block with the encryption results from the prior block. In CBC mode, the input data (plain text) is XORed with the previous output block (cipher text) before encryption. Decryption is simply the reverse of this operation. The output data (plain text) is XORed with the previous input block (cipher text) following decryption. An initialization vector provides the XOR feedback for the first block of a packet. The CBC mode produces the same cipher text if the same plain text is encrypted using the same key and IV. Because the CBC mode is a block method of encryption, it must operate on 64-bit data blocks. Partial data blocks (blocks of less than 64 bits) must be padded to a 64-bit boundary.

22.3.2 Integration Features

22.3.2.1 Clocking and Power Management

Clocks

The DES/3DES accelerator operates from the ClkBridge52 clock of the CLKM module, which is used for the DES/3DES configuration interface and the functional interface.

Idle Mode

Functional and interface clock gating for power saving is activated by setting D3D1.DES_MASK[0] AUTOIDLE bit to 1. In this mode, clocks are active only when registers are accessed (when the interface clock is used) or an operation is on going (when the functional clock is used). When this bit is set to 0, both clocks are free-running.

22.3.2.2 Reset Management

Hardware reset is available using the core_protected_nRST signal issued from the PRRM module.

To reset the software, set the D3D1.DES_MASK[1] SOFTRESET bit to 1.

The reset behavior is the same for both the software and hardware, except that the software reset bit resets this module without affecting the whole core_protected reset domain.

22.3.2.3 Hardware Requests

DMA Requests

Two DMA requests are mapped to the DMA controller, as listed in Table 22-4.

Table 22-4. DES/3DES DMA Request

Attributes	Values	Name	Mapping	Comments
DMA request	2	DES3DES_TX_ DMA_REQ	DMA_29	Destination: DMA controller
		DES3DES_RX_ DMA_REQ	DMA_28	

Interrupts

No interrupt request is generated in this module.

22.3.3 DES/3DES Operation Description

The DES/3DES module runs either the DES or the triple DES algorithm, depending on the value of the DES_CTRL[3] TDES bit (DES mode by default when the bit is 0, or in TDES mode when the bit is set to 1).

22.3.3.1 DES Mode

To start a new encryption or decryption using the DES algorithm, the key1, IV, mask, and configuration registers must be written in a specific order.

1. Set the D3D1.DES_MASK[4] DIRECT_BUS_EN bit to 1 if key1 is loaded through the direct bus interface, or set the bit to 0 if key1 is loaded through the host interface. The direct bus interface is used only in the DES mode.
2. To use the DMA features, set the D3D1.DES_MASK[2] DMA_REQ_IN_EN bit and the D3D1.DES_MASK[3] DMA_REQ_OUT_EN bit to 1.
3. Select the free-running clock mode or power-saving mode D3D1.DES_MASK[0] AUTOIDLE bit.

   The D3D.DES_KEY1_L/H registers are then loaded by the host if this option is chosen. If the direct bus option is chosen, key1 is automatically updated by the values present at the direct bus interface.
4. Set the D3D1.DES_CTRL[3] TDES bit to 0 (DES mode).
5. To select the CBC mode, set the D3D1.DES_CTRL[4] CBC bit to 1; to select the ECB mode, set the bit to 0.
6. To encrypt data, set the D3D1.DES_CTRL[2] DIRECTION bit to 1; to decrypt data, set the bit to 0.

   If the CBC mode is considered, then the D3D1.DES_IV_L/H registers are loaded.

   Once the D3D1.DES_CTRL register is written, the DES/3DES module can receive the first block of data. Data is written by the host, which is either a DMA or a processor. The D3D1.DES_CTRL[1] INPUT_READY bit provides the status to the host. When high, this bit indicates that the input buffer is empty and the host is free to write the block of data input in the two 32-bit registers (D3D1.DES_DATA_L/H). When low, this bit indicates that the input buffers have been written.

   When using DMA to write the data, the D3D1.DES_MASK[2] DMA_REQ_IN_EN and D3D1.DES_MASK[3] DMA_REQ_OUT_EN bits must have been set to 1 during the configuration phase. This way, DMA requests can be generated. When D3D1.DES_CTRL is written, the input DMA request is sent to the DMA only when the D3D1.DES_MASK[5] START bit in the DES_MASK register is set high by the processor. It lets the processor first configure the DMA (allocating two DMA channels, one for writing and one for reading) and then control when the transfer of the first data block is triggered. The DMA must be configured to transfer two words of 32 bits each time an input DMA request is sent by the DES/3DES module.

The destination address for the DMA must be defined in the address mapping of the DES/3DES module with a 0x24h offset, and a constant addressing mode must be set. The two data words written to address 0x24h are sent to the D3D1.DES_DATA_L/H registers ranging from 0x24h through 0x28h. Once the first data block is transferred, the module detects that an 8-byte block is available and then moves the data to a working register space for processing. The D3D1.DES_CTRL[1] INPUT_READY bit is asserted low when the input buffers are written. The processing starts, and then the INPUT_READY bit is set high to request the next block of data. With the START bit high, this is transformed into a new DMA request.

It takes 16 clock cycles to process the data and to generate the cipher text or plain text (corresponding to 16 rounds). The OUTPUT_READY status bit (bit 0 of the DES_CTRL register) is then set high. This is translated into an output DMA request to inform the DMA that 8 bytes of processed data are available.

As soon as the OUTPUT_READY bit is asserted to read the output buffer, and if the input buffers are already refilled, a new processing can start. If the output buffers are not read fast enough, the DES/3DES core holds off the processing until the output is read.

If the processor is loading the DES/3DES core and the DMA is not used (DMA requests disabled), the START bit must still be set to 1 because, if in power-saving mode, it lets the clock keep toggling during all the transfers and processing. As soon as the DES_CTRL register is written, the processor can write the first 8 bytes because the INPUT_READY status bit is high. The INPUT_ READY bit is then asserted low to indicate that the input buffers are full. Data are transferred to the working registers. The processing starts, and the INPUT_READY bit is set high to request a new packet. To be able to write the next data packet, the processor must poll the INPUT_READY status bit. To read the result of the processing, the processor must poll the OUTPUT_ READY status bit.

Only for the first data loading, the processor or DMA can write two consecutive packets of 8 bytes each before reading the output result. After that, the write and read operations must be interleaved.

22.3.3.2 Triple DES Mode

Using the triple DES mode is the same process as using the DES mode with the following changes:

- In step 1, load the three keys (key1, key2, and key3) without using the direct bus interface (set D3D1.DES_MASK[4] DIRECT_BUS_EN bit to 0).
- In step 4, the D3D1.DES_CTRL[3] TDES bit must be set to 1 in to use the triple DES mode.

22.3.3.3 General Remarks

The DES/3DES module does not provide any internal padding. If a packet of less than 8 bytes is to be processed, the host or DMA must pad it to generate a 64-bit packet.

During processing, the processor or the DMA may not write into the control registers because the input registers are double-buffered. Consequently, the next DES operation cannot be configured in advance.

22.4 Security Programming Models

22.4.1 Programming Charts

Figure 22-7. SHA1/MD5 Programming Model

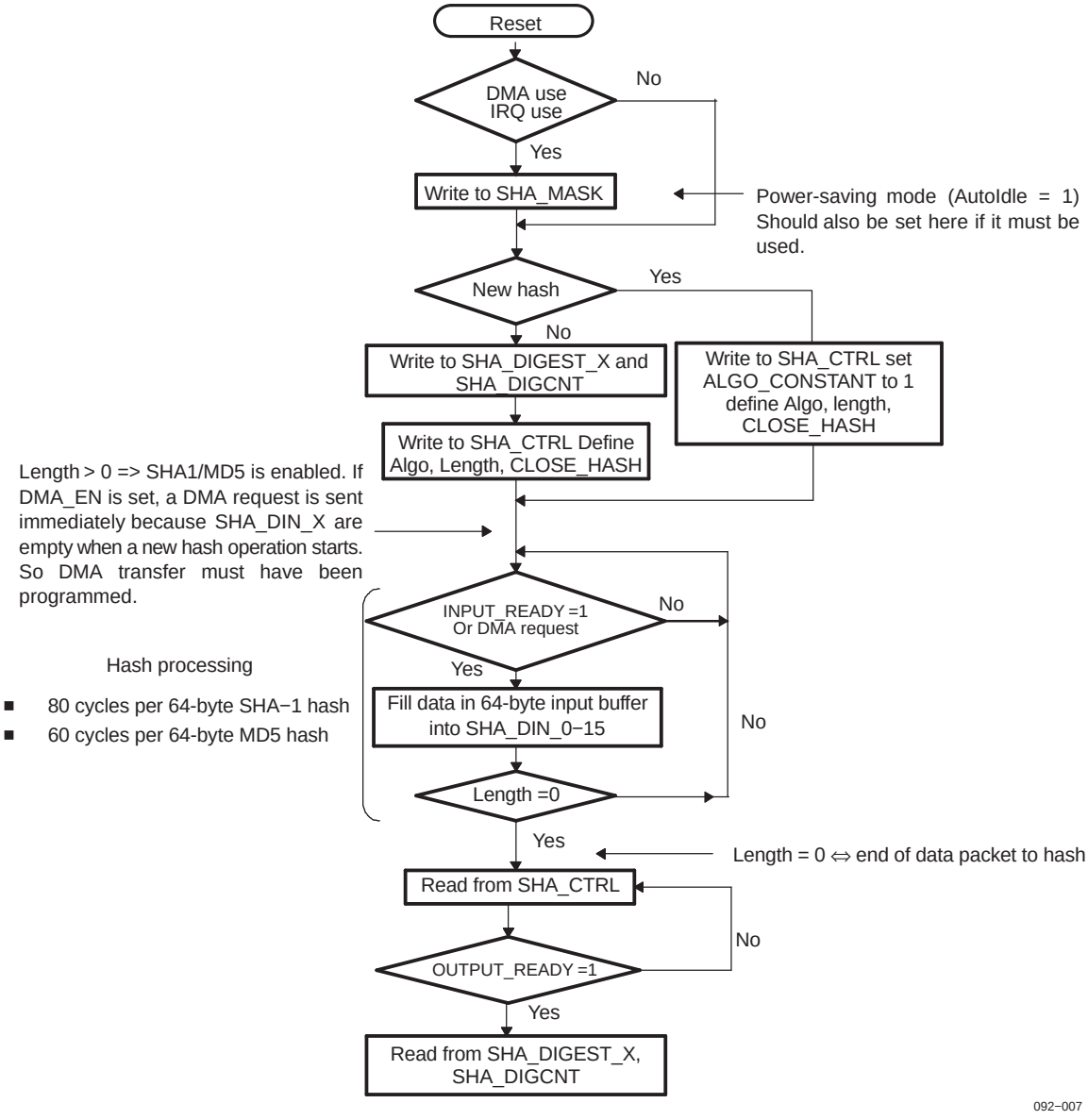


Figure 22-8. RNG Programming Model

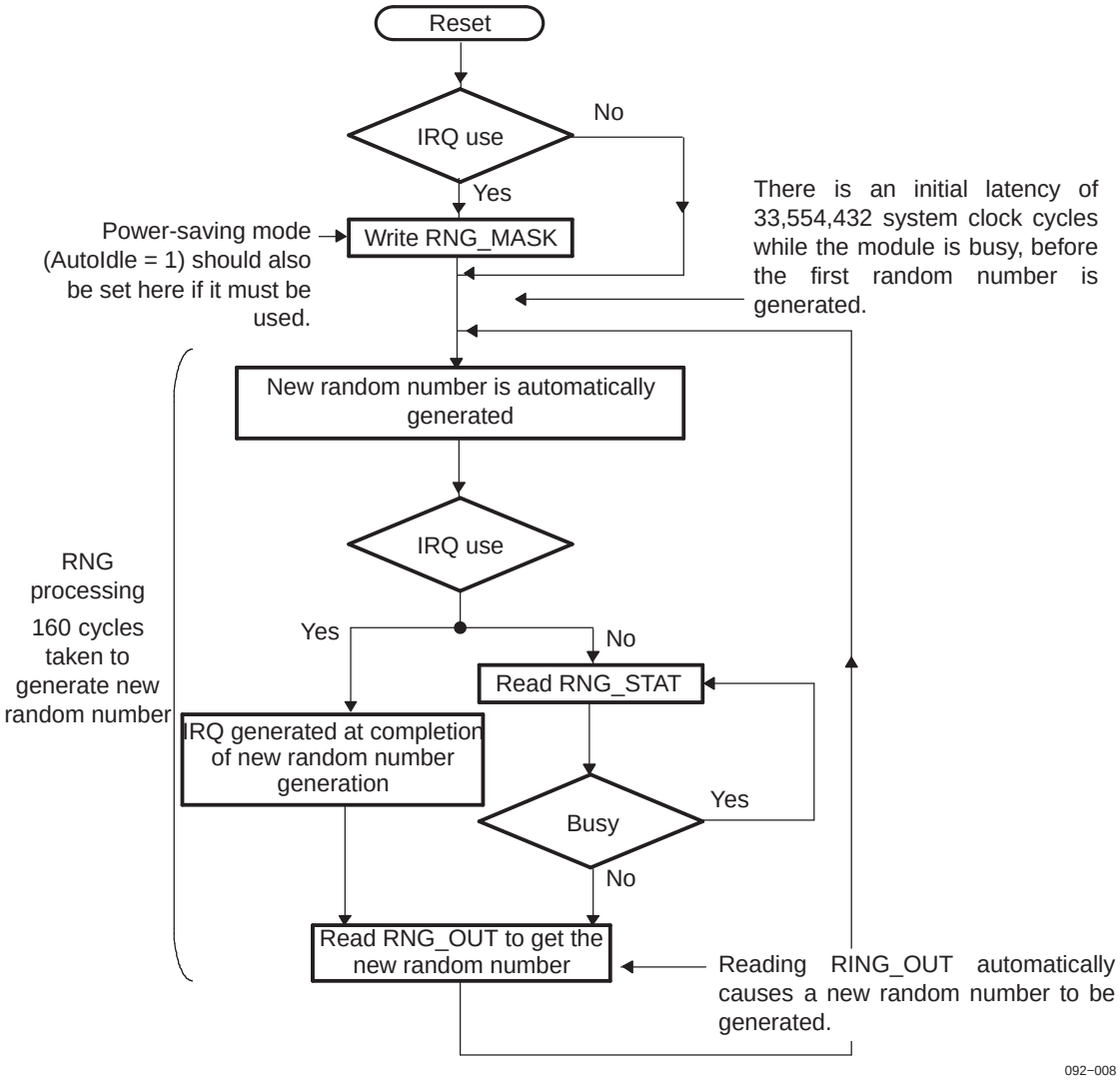
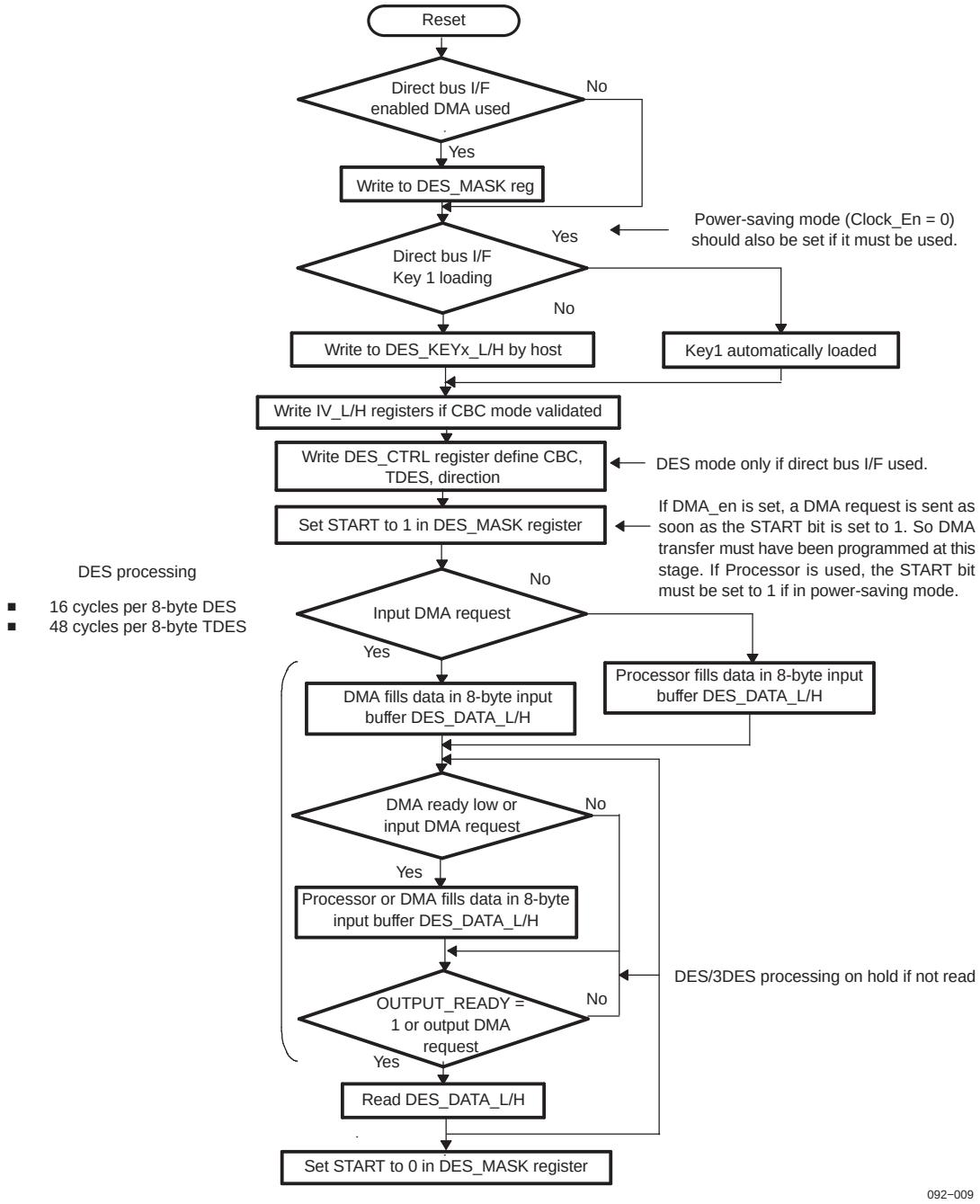


Figure 22-9. DES Programming Model



22.5 Registers

Table 22-5 lists the physical address of the registers of crypto-hardware accelerators.

Table 22-5. Physical Addresses of Crypto Hardware Accelerators

Device Name	Description	Base address (hex)	Size
RNG1	Module	0x09A0 0000	4K bytes
D3D1	Module	0x0990 0000	4K bytes
SHAM1	Module	0x0980 0000	4K bytes

22.5.1 RNG1 Registers

Table 22-6 lists the RNG1 registers. Table 22-7 through Table 22-13 describe the register bits.

CAUTION

RNG1 registers are limited to 32-bit data access. 16-bit and 8-bit accesses are not allowed.

Table 22-6. RNG1 Registers

Name	Description	Type	Size (Bits)	Physical Address
RNG_OUT	RNG output	R	32	0x09A00000
RNG_STAT	RNG status	R	32	0x09A00004
RNG_ALARM	RNG alarm counter	RW	32	0x09A00024
RNG_CONFIG	RNG configuration	RW	32	0x09A00028
RNG_REV	RNG revision	R	32	0x09A0003C
RNG_MASK	RNG mask and reset	RW	32	0x09A00040
RNG_SYSSTATUS	RNG system status	R	32	0x09A00044

Table 22-7. RNG Output Register (RNG_OUT)

Address Offset	0x00		
Physical address	0x09A00000	**Instance**	RNG1
Description	Output register for random number		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_OUT																															

Bits	Field Name	Description	Type	Reset
31:0	RNG_OUT	Output random number generated	R	0x00000000

Table 22-8. RNG Status Register (RNG_STAT)

Address Offset	0x04		
Physical address	0x09A00004	**Instance**	RNG1
Description	Ready/busy status		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															BUSY

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reserved	R	0x00000000
0	BUSY	When 1, module is busy generating a new random number. This bit is automatically cleared to 0 when a valid random number is available in the output register. The interrupt line of the RNG module is directly connected to this bit, which can be seen as the interrupt status bit. The result is an interrupt status bit which does not get cleared by writing a 1 to it as usual (this bit is read-only), but gets cleared by reading the RNG_OUT register.	R	1

Table 22-9. RNG Alarm Counter Register (RNG_ALARM)

Address Offset	0x24		
Physical address	0x09A00024	**Instance**	RNG1
Description	Counter for count reset threshold is met.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								ALARM_ COUNTER							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reserved	R	0x000000
7:0	ALARM_ COUNTER	Counter for alarm, counts number of times consecutive count reset threshold is met.	R	0x00

Table 22-10. RNG Configuration Register (RNG_CONFIG)

Address Offset	0x28		
Physical address	0x09A00028	**Instance**	RNG1
Description	Config register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				RESET_COUNT								RING2_DELAY			RING1_DELAY		

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Reserved	R	0x00000
11:6	RESET_COUNT	Threshold for unchanged consecutive output bits for resetting LFSRs. Bit 11 is set to 1 to start the ring oscillator after reset.	RW	0x20
5:3	RING2_DELAY	Delay for ring2 oscillator	RW	0x3
2:0	RING1_DELAY	Delay for ring1 oscillator	RW	0x4

Note: The ring oscillators must be asynchronous to the system clock. If there is a manufacturing problem and one of the ring oscillators is synchronous to its sample clock (system clock), alarms will occur and the alarm counter will increment. In this event, the delay fields in the register allow you to tune the delay loop of the ring oscillators. The default values are in the middle of the 3-bit range (011 and 100). Selecting lower or higher values makes the ring oscillators slightly slower or faster.

Table 22-11. RNG Revision Register (RNG_REV)

Address Offset	0x3C		
Physical address	0x09A0003C	**Instance**	RNG1
Description	Revision number		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REV							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reserved	R	0x000000
7:0	REV	Revision number	R	0x_UnicodeEncodeError__UnicodeEncodeError_ (This is a value from TI internal data.)

Table 22-12. RNG Mask and Reset Register (RNG_MASK)

Address Offset	0x40		
Physical address	0x09A00040	**Instance**	RNG1
Description	Mask and config register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													IT_EN	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Reserved	R	0x00000000
2	IT_EN	Mask for interrupt (when 1, interrupt is not masked)	RW	0
1	SOFTRESET	When set to 1, a soft reset sequence starts. This bit is automatically cleared when reset is done, and this bit is always 0 when read.	RW	0
0	AUTOIDLE	When 0, clock is free-running. When 1, module is in power-saving mode: clock is running only when module is accessed or generating new random number.	RW	0

Table 22-13. RNG System Status Register (RNG_SYSSTATUS)

Address Offset	0x44		
Physical address	0x09A00044	**Instance**	RNG1
Description	Internal reset monitoring		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															RESETDONE

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reserved. Must be written with 0s.	R	0x00000000
0	RESETDONE	To monitor reset. When 1, the RNG has finished a reset. When 0, the RNG is currently performing a reset.	R	1

22.5.2 D3D1 Registers

Table 22-14 lists the D3D1 registers. Table 22-15 through Table 22-28 describe the register bits.

CAUTION

D3D1 registers are limited to 32-bit data access. 16-bit and 8-bit accesses are not allowed.

Table 22-14. D3D1 Registers

Name	Description	Type	Size (Bits)	Physical Address
DES_KEY3_L	DES key 3 low	RW	32	0x09900000
DES_KEY3_H	DES key 3 high	RW	32	0x09900004
DES_KEY2_L	DES key 2 low	RW	32	0x09900008
DES_KEY2_H	DES key 2 high	RW	32	0x0990000C
DES_KEY1_L	DES key 1 low	RW	32	0x09900010
DES_KEY1_H	DES key 1 high	RW	32	0x09900014
DES_IV_L	DES initialization vector low	RW	32	0x09900018
DES_IV_H	DES initialization vector high	RW	32	0x0990001C
DES_CTRL	DES control	RW	32	0x09900020
DES_DATA_L	DES data low	RW	32	0x09900024
DES_DATA_H	DES data high	RW	32	0x09900028
DES_REV	DES revision	R	32	0x0990002C
DES_MASK	DES mask and reset	RW	32	0x09900030
DES_SYSSTATUS	DES system status	R	32	0x09900034

Table 22-15. DES Key 3 Low Register (DES_KEY3_L)

Address Offset	0x00		
Physical address	0x09900000	**Instance**	D3D1
Description	Holds the key		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DES_KEY3_L																															

Bits	Field Name	Description	Type	Reset
31:0	DES_KEY3_L	Key 3 LSB. The three symmetric keys are written to the six key registers. For single DES operation, only the DES_KEY1_L /H registers must be populated. The other four key registers can be ignored. For TDES operation, all six key registers must be written. When read, this register returns 0x00000000.	RW	0x00000000

Table 22-16. DES Key 3 High Register (DES_KEY3_H)

Address Offset	0x04		
Physical address	0x09900004	**Instance**	D3D1
Description	Holds the key		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DES_KEY3_H																															

Bits	Field Name	Description	Type	Reset
31:0	DES_KEY3_H	Key 3 MSB. The three symmetric keys are written to the six key registers. For single DES operation, only the DES_KEY1_L/H registers must be populated. The other four key registers can be ignored. For TDES operation, all six key registers must be written. When read, this register returns 0x00000000.	RW	0x00000000

Table 22-17. DES Key 2 Low Register (DES_KEY2_L)

Address Offset	0x08		
Physical address	0x09900008	**Instance**	D3D1
Description	Holds the key		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DES_KEY2_L																															

Bits	Field Name	Description	Type	Reset
31:0	DES_KEY2_L	Key 2 LSB. The three symmetric keys are written to the six key registers. For single DES operation, only the DES_KEY1_L/H registers must be populated. The other four key registers can be ignored. For TDES operation, all six key registers must be written. When read, this register returns 0x000000.	RW	0x00000000

Table 22-18. DES Key 2 High Register (DES_KEY2_H)

Address Offset	0x0C		
Physical address	0x0990000C	**Instance**	D3D1
Description	Holds the key		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DES_KEY2_H																															

Bits	Field Name	Description	Type	Reset
31:0	DES_KEY2_H	Key 2 MSB. The three symmetric keys are written to the six key registers. For single DES operation, only the DES_KEY1_L/H registers must be populated. The other four key registers can be ignored. For TDES operation, all six key registers must be written. When read, this register returns 0x000000.	RW	0x00000000

Table 22-19. DES Key 1 Low Register (DES_KEY1_L)

Address Offset	0x10		
Physical address	0x09900010	**Instance**	D3D1
Description	Holds the key		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DES_KEY1_L																															

Bits	Field Name	Description	Type	Reset
31:0	DES_KEY1_L	Key 1 LSB. The three symmetric keys are written to the six key registers. For single DES operation, only the DES_KEY1_L/H registers must be populated. The other four key registers can be ignored. For TDES operation, all six key registers must be written. When read, this register returns 0x000000.	RW	0x00000000

Table 22-20. DES Key 1 High Register (DES_KEY1_H)

Address Offset	0x14		
Physical address	0x09900014	**Instance**	D3D1
Description	Holds the key		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DES_KEY1_H																															

Bits	Field Name	Description	Type	Reset
31:0	DES_KEY1_H	Key 1 MSB. The three symmetric keys are written to the six key registers. For single DES operation, only the DES_KEY1_L/H registers must be populated. The other four key registers can be ignored. For TDES operation, all six key registers must be written. When read, this register returns 0x000000.	RW	0x00000000

Table 22-21. DES Initialization Vector Low Register (DES_IV_L)

Address Offset	0x18		
Physical address	0x09900018	**Instance**	D3D1
Description	For cypher-block-chaining, this register holds an initialization vector.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DES_IV_L																															

Bits	Field Name	Description	Type	Reset
31:0	DES_IV_L	Init vector LSB. For cipher-block-chaining, the two IV registers DES_IV_L and DES_IV_H must be populated. For simple electronic codebook (ECB) mode, this register can be ignored.	RW	0x00000000

Table 22-22. DES Initialization Vector High Register (DES_IV_H)

Address Offset	0x1C		
Physical address	0x0990001C	**Instance**	D3D1
Description	For cypher-block-chaining, this register holds an initialization vector.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DES_IV_H																															

Bits	Field Name	Description	Type	Reset
31:0	DES_IV_H	Init vector MSB. For cipher-block-chaining, the two IV registers DES_IV_L and DES_IV_H must be populated. For simple electronic codebook (ECB) mode, this register can be ignored.	RW	0x00000000

Table 22-23. DES Control Register (DES_CTRL)

Address Offset	0x20		
Physical address	0x09900020	**Instance**	D3D1
Description	Control register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											CBC	TDES	DIRECTION	INPUT_READY	OUTPUT_READY	

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Reserved. Must be written with 0s and ignored when read.	RW	0x0000000
4	CBC	When 1, cypher-block-chaining is used. When 0, electronic codebook is used.	RW	0
3	TDES	When 1, 3DES is selected. By default when this bit is 0, DES is invoked.	RW	0
2	DIRECTION	When 1, an encrypt operation is performed. When 0, a decrypt operation is performed.	RW	0
1	INPUT_READY	When 1, the 8-byte input buffer is empty and the next data can be written. When data input registers are full, it automatically goes to 0. This status bit also participates in the generation of a DMA input request in case a DMA channel is configured to download data.	R	1
0	OUTPUT_READY	When 1, the 8-byte output buffer can be read. Bit is read only. When reading, the bit is set back to 0. If output data is not read, the system holds the processing off until it is read. No new input data can be written to the input data registers. This status bit also participates in the generation of a DMA output request in case a DMA channel is configured to empty the buffer.	R	0

Table 22-24. DES Data Input/Output Low Register (DES_DATA_L)

Address Offset	0x24		
Physical address	0x09900024	**Instance**	D3D1
Description	Data register to read/write encrypted/decrypted data		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DES_DATA_L																															

Bits	Field Name	Description	Type	Reset
31:0	DES_DATA_L	Register holds data (LSB) for encryption/decryption. This 32-bit register is used to write or read encrypted or decrypted data. Large packets are partitioned into 8-byte blocks and sequentially entered into the DES engine, one at a time. Once the process is completed, the encrypted/decrypted 8-byte blocks are read.	RW	0x00000000

Table 22-25. DES Data Input/Output High Register (DES_DATA_H)

Address Offset	0x28		
Physical address	0x09900028	**Instance**	D3D1
Description	Data register to read/write encrypted/decrypted data		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DES_DATA_H																															

Bits	Field Name	Description	Type	Reset
31:0	DES_DATA_H	Register holds data (MSB) for encryption/decryption. This 32-bit register is used to write or read encrypted or decrypted data. Large packets are partitioned into 8-byte blocks and sequentially entered into the DES engine, one at a time. Once the process is completed, the encrypted/decrypted 8-byte blocks are read.	RW	0x00000000

Table 22-26. DES Revision Register(DES_REV)

Address Offset	0x2C		
Physical address	0x0990002C	**Instance**	D3D1
Description	Revision number		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REV							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reserved	RW	0x000000
7:0	REV	Revision number	R	0x_UnicodeEncodeError__UnicodeEncodeError_ (This is a value from TI internal data.)

Table 22-27. DES Mask and Reset Register (DES_MASK)

Address Offset	0x30		
Physical address	0x09900030	**Instance**	D3D1
Description	Mask register		
Type	RW		

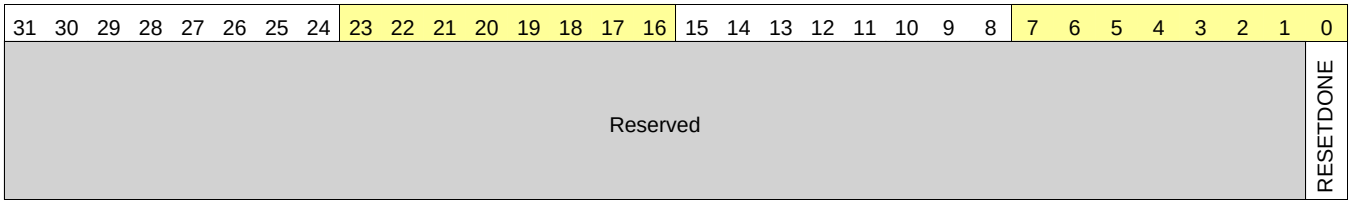
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										START	DIRECT_BUS_EN	DMA_REQ_ OUT_EN	DMA_REQ_ IN_EN	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:6	Reserved	Reserved. Must be written with 0s and must be ignored on a read.	RW	0x0000000
5	START	When 1, input DMA request is sent. This bit has a double functionality when the DMA is requested for uploading/downloading data, or a simple functionality when only the processor is used. When the DMA is used (DMA requests have been enabled) and configured by the processor, the start bit set to 1 allows triggering the input DMA request for the first download of data. It is also used to keep the clock toggling as long as transfers of data and their processing are performed. It is useful in power-saving mode. When the processor is used (DMA requests disabled) the start bit keeps the clock toggling in power-saving mode. This bit must be asserted low by the processor, thus stopping the clock toggling once the full operation is complete.	RW	0
4	DIRECT_BUS_EN	When 1, Key 1 is loaded through the direct bus interface. This bit allows to select the downloading of the three keys from the host interface when 0 or the downloading of a single key from the direct bus interface when 1.	RW	0
3	DMA_REQ_ OUT_EN	When 1, input DMA is not masked. If set to 0, input DMA request is masked. If set to 1, the DMA request indicates when DES_DATA_L/H registers used as input buffers are empty and triggers a DMA transfer to fill those registers.	RW	0
2	DMA_REQ_ IN_EN	When 1, output DMA is not masked. If set to 0, the output DMA request is masked. If set to 1, the DMA request indicates when DES_DATA_L/H registers used as output buffers are full and ready for reading after encryption/decryption completion.	RW	0

Bits	Field Name	Description	Type	Reset
1	SOFTRESET	When 1, a soft reset sequence starts. Release is done automatically. Only hardware reset affects this bit; it is not affected by its own soft reset.	RW	0
0	AUTOIDLE	When 0, clock is free-running. When 1, module is in power-saving mode: clock is running only when module is accessed or an encryption/ decryption is ongoing.	RW	0

Table 22-28. DES System Status Register (DES_SYSSTATUS)

Address Offset	0x34		
Physical address	0x09900034	**Instance**	D3D1
Description	Status of reset		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reserved. Must be written with 0s.	R	0x00000000
0	RESETDONE	Internal reset monitoring. When 1, the DES/3DES has finished a reset. When 0, the DES/3DES is currently performing a reset.	R	1

22.5.3 SHAM1 Registers

Table 22-29 lists the SHA1/MD5 registers. Table 22-30 through Table 22-55 describe the register bits.

CAUTION

SHAM1 registers are limited to 32-bit data access. 16-bit and 8-bit accesses are not allowed.

Table 22-29. SHAM1 Registers

Name	Description	Type	Size (Bits)	Physical Address
SHA_DIGEST_A	SHA hash digest A	RW	32	0x09800000
SHA_DIGEST_B	SHA hash digest B	RW	32	0x09800004
SHA_DIGEST_C	SHA hash digest C	RW	32	0x09800008
SHA_DIGEST_D	SHA hash digest D	RW	32	0x0980000C
SHA_DIGEST_E	SHA hash digest E	RW	32	0x09800010
SHA_DIGCNT	SHA digest counter	RW	32	0x09800014
SHA_CTRL	SHA control	RW	32	0x09800018
SHA_DIN_0	SHA input data 0	RW	32	0x0980001C
SHA_DIN_1	SHA input data 1	RW	32	0x09800020
SHA_DIN_2	SHA input data 2	RW	32	0x09800024
SHA_DIN_3	SHA input data 3	RW	32	0x09800028
SHA_DIN_4	SHA input data 4	RW	32	0x0980002C
SHA_DIN_5	SHA input data 5	RW	32	0x09800030
SHA_DIN_6	SHA input data 6	RW	32	0x09800034
SHA_DIN_7	SHA input data 7	RW	32	0x09800038

Table 22-29. SHAM1 Registers (continued)

Name	Description	Type	Size (Bits)	Physical Address
SHA_DIN_8	SHA input data 8	RW	32	0x0980003C
SHA_DIN_9	SHA input data 9	RW	32	0x09800040
SHA_DIN_10	SHA input data 10	RW	32	0x09800044
SHA_DIN_11	SHA input data 11	RW	32	0x09800048
SHA_DIN_12	SHA input data 12	RW	32	0x0980004C
SHA_DIN_13	SHA input data 13	RW	32	0x09800050
SHA_DIN_14	SHA input data 14	RW	32	0x09800054
SHA_DIN_15	SHA input data 15	RW	32	0x09800058
SHA_REV	SHA revision	R	32	0x0980005C
SHA_MASK	SHA mask and reset	RW	32	0x09800060
SHA_SYSSTATUS	SHA system status	R	32	0x09800064

Table 22-30. SHA Hash Digest A Register (SHA_DIGEST_A)

Address Offset	0x000		
Physical address	0x09800000	**Instance**	SHAM1
Description	Digest A register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIGEST_A																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIGEST_A	Initial digest. It is written to the five hash registers. When the status register indicates that the operation is complete, the resultant hash digest can be read from these registers. When the specified hash length is longer than the 64-byte hash block size, it is not necessary to read and then write the digest registers to begin processing the next block. The digest is automatically updated for the next operation. The SHA1 algorithm uses all five digest registers, A through E, but the MD5 algorithm uses only four digest registers, A through D.	RW	0x00000000

Table 22-31. SHA Hash Digest B Register (SHA_DIGEST_B)

Address Offset	0x004		
Physical address	0x09800004	**Instance**	SHAM1
Description	Digest B register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIGEST_B																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIGEST_B	Initial digest. It is written to the five hash registers. When the status register indicates that the operation is complete, the resultant hash digest can be read from these registers. When the specified hash length is longer than the 64-byte hash block size, it is not necessary to read and then write the digest registers to begin processing the next block. The digest is automatically updated for the next operation. The SHA1 algorithm uses all five digest registers, A through E, but the MD5 algorithm uses only four digest registers, A through D.	RW	0x00000000

Table 22-32. SHA Hash Digest C Register (SHA_DIGEST_C)

Address Offset	0x008		
Physical address	0x09800008	**Instance**	SHAM1
Description	Digest C register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIGEST_C																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIGEST_C	Initial digest. It is written to the five hash registers. When the status register indicates that the operation is complete, the resultant hash digest can be read from these registers. When the specified hash length is longer than the 64-byte hash block size, it is not necessary to read and then write the digest registers to begin processing the next block. The digest is automatically updated for the next operation. The SHA1 algorithm uses all five digest registers, A through E, but the MD5 algorithm uses only four digest registers, A through D.	RW	0x00000000

Table 22-33. SHA Hash Digest D Register (SHA_DIGEST_D)

Address Offset	0x00C		
Physical address	0x09800000	**Instance**	SHAM1
Description	Digest D register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIGEST_D																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIGEST_D	Initial digest. It is written to the five hash registers. When the status register indicates that the operation is complete, the resultant hash digest can be read from these registers. When the specified hash length is longer than the 64-byte hash block size, it is not necessary to read and then write the digest registers to begin processing the next block. The digest is automatically updated for the next operation. The SHA1 algorithm uses all five digest registers, A through E, but the MD5 algorithm uses only four digest registers, A through D.	RW	0x00000000

Table 22-34. SHA Hash Digest E Register (SHA_DIGEST_E)

Address Offset	0x010		
Physical address	0x09800010	**Instance**	SHAM1
Description	Digest E register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIGEST_E																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIGEST_E	Initial digest. It is written to the five hash registers. When the status register indicates that the operation is complete, the resultant hash digest can be read from these registers. When the specified hash length is longer than the 64-byte hash block size, it is not necessary to read and then write the digest registers to begin processing the next block. The digest is automatically updated for the next operation. This register is only used by the SHA1 algorithm.	RW	0x00000000

Table 22-35. SHA Digest Count Register (SHA_DIGCNT)

Address Offset	0x014		
Physical address	0x09800014	**Instance**	SHAM1
Description	SHA digest count register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIGCNT																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIGCNT	Initial digest count. This 32-bit register contains the total number of bytes that have been processed in the current hash operation. When a new hash is started, the SHA_CTRL register ALGO_CONSTANT bit setting to 1 causes the initialization to zero of this register. When a hash is continued (see details in Section 22.1.3) it must be written with a restored digest count in bytes of a prior hash. At the end of the hash operation, the total number of bytes that were processed can be read in this register when the SHA_CTRL register OUTPUT_READY bit indicates that the operation is complete.	RW	0x00000000

Table 22-36. SHA Control Register (SHA_CTRL)

Address Offset	0x018		
Physical address	0x09800018	**Instance**	SHAM1
Description	SHA1 control register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LENGTH																											CLOSE_HASH	ALGO_CONSTANT	ALGO	INPUT_READY	OUTPUT_READY

Bits	Field Name	Description	Type	Reset
31:5	LENGTH	Code the length of message to hash. This field is written with the total size of the data to be hashed, in bytes and up to 128M bytes. For hash lengths longer than 64 bytes, you do not have to update the digest registers after the initial data block; this is done automatically.	RW	0x00000
4	CLOSE_HASH	When 1, the final hash is performed. This bit can be set at the beginning of the hash operation if the hashing can be done in one operation.	RW	0
3	ALGO_CONSTANT	When 1, the default value is used. SHA_DIGCNT is also reset to zero.	RW	0
2	ALGO	When 1, hash is a SHA1. When 0, hash is an MD5.	RW	0
1	INPUT_READY	If 1, the SHA_DIN_X registers are available to input a new 64-byte data.	RW	1
0	OUTPUT_READY	If 1, hash operation is complete. To clear this bit, write 1. This serves as a pending interrupt. Before clearing the interrupt, it is recommended to also clear the length field (field 31-5 in the SHA_CTRL register). This prevents the module from waiting for another message to hash (when there are none) and to go into power-saving mode.	RW	0

Table 22-37. SHA Input Data 0 Register (SHA_DIN_0)

Address Offset	0x01C		
Physical address	0x0980001C	**Instance**	SHAM1
Description	SHA data input register 0		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_0																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_0	Holds message to be hashed. The 64-byte block of data to be hashed is written to these sixteen 32-bit registers covering address offsets 0x1C to 0x58, inclusive. These registers are holding registers that can be written any time the SHA_CTRL register INPUT_ READY bit is set to 1. For hash lengths longer than 64 bytes, multiple blocks of data are written to this register bank. All blocks, except for the last, are required to be 64 bytes in size. If the last block is not 64 bytes long, the CLOSE_HASH control bit must be previously set. It is also possible to write into those sixteen registers using constant addressing mode at address 0x1C. An internal counter increments the address and dispatches the sixteen data into those sixteen SHA_DIN_0 to SHA_DIN_15 registers. This addressing mode is available for DMA transfer purposes.	RW	0x00000000

Table 22-38. SHA Input Data 1 Register (SHA_DIN_1)

Address Offset	0x020		
Physical address	0x09800020	**Instance**	SHAM1
Description	SHA data input register 1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_1																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_1	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-39. SHA Input Data 2 Register (SHA_DIN_2)

Address Offset	0x024		
Physical address	0x09800024	**Instance**	SHAM1
Description	SHA data input register 2		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_2																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_2	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-40. SHA Input Data 3 Register (SHA_DIN_3)

Address Offset	0x028		
Physical address	0x09800028	**Instance**	SHAM1
Description	SHA data input register 3		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_3																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_3	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-41. SHA Input Data 4 Register (SHA_DIN_4)

Address Offset	0x02C		
Physical address	0x0980002C	**Instance**	SHAM1
Description	SHA data input register 4		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_4																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_4	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-42. SHA Input Data 5 Register (SHA_DIN_5)

Address Offset	0x030		
Physical address	0x09800030	**Instance**	SHAM1
Description	SHA data input register 5		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_5																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_5	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-43. SHA Input Data 6 Register (SHA_DIN_6)

Address Offset	0x034		
Physical address	0x09800034	**Instance**	SHAM1
Description	SHA data input register 6		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_6																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_6	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-44. SHA Input Data 7 Register (SHA_DIN_7)

Address Offset	0x038		
Physical address	0x09800038	**Instance**	SHAM1
Description	SHA data input register 7		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_7																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_7	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-45. SHA Input Data 8 Register (SHA_DIN_8)

Address Offset	0x03C		
Physical address	0x0980003C	**Instance**	SHAM1
Description	SHA data input register 8		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_8																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_8	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-46. SHA Input Data 9 Register (SHA_DIN_9)

Address Offset	0x040		
Physical address	0x09800040	**Instance**	SHAM1
Description	SHA data input register 9		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_9																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_9	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-47. SHA Input Data 10 Register (SHA_DIN_10)

Address Offset	0x044		
Physical address	0x09800044	**Instance**	SHAM1
Description	SHA data input register 10		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_10																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_10	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-48. SHA Input Data 11 Register (SHA_DIN_11)

Address Offset	0x048		
Physical address	0x09800048	**Instance**	SHAM1
Description	SHA data input register 11		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_11																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_11	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-49. SHA Input Data 12 Register (SHA_DIN_12)

Address Offset	0x04C		
Physical address	0x0980004C	**Instance**	SHAM1
Description	SHA data input register 12		

Table 22-49. SHA Input Data 12 Register (SHA_DIN_12) (continued)

Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_12																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_12	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-50. SHA Input Data 13 Register (SHA_DIN_13)

Address Offset	0x050		
Physical address	0x09800050	Instance	SHAM1
Description	SHA data input register 13		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_13																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_13	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-51. SHA Input Data 14 Register (SHA_DIN_14)

Address Offset	0x054		
Physical address	0x09800054	Instance	SHAM1
Description	SHA data input register 14		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_14																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_14	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-52. SHA Input Data 15 Register (SHA_DIN_15)

Address Offset	0x058		
Physical address	0x09800058	Instance	SHAM1
Description	SHA data input register 15		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHA_DIN_15																															

Bits	Field Name	Description	Type	Reset
31:0	SHA_DIN_15	Holds message to be hashed (see Table 22-37)	RW	0x00000000

Table 22-53. SHA Revision Register (SHA_REV)

Address Offset	0x05C		
Physical address	0x0980005C	Instance	SHAM1
Description	SHA revision register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REV							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reserved. Read as 0s.	R	0x000000
7:0	REV	Revision number	R	0x_UnicodeEncodeError__UnicodeEncodeError_ (This is a value from TI internal data.)

Table 22-54. SHA Mask and Reset Register (SHA_MASK)

Address Offset	0x060		
Physical address	0x09800060	**Instance**	SHAM1
Description	Contains control bits		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												DMA_EN	IT_EN	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:4	Reserved	Reserved. Must be written with 0s.	RW	0x0000000
3	DMA_EN	Enables DMA when 1. If set to 0, DMA request is masked. If set to 1, DMA request indicates when SHA_DIN_X registers are empty and triggers a DMA transfer to fill those registers.	RW	0
2	IT_EN	Enables interrupt when 1. If set to 0, the interrupt is masked. If set to 1, the interrupt is generated and indicates hash completion.	RW	0
1	SOFTRESET	When a 1 is written to this bit, it initiates a soft reset sequence. This bit is automatically cleared on reset and always 0 when read.		0
0	AUTOIDLE	When 0, clocks are on. When 1, module is in power-saving mode: clock runs only when module is accessed or a hash is ongoing.	RW	0

Table 22-55. SHA System Status Register (SHA_SYSSTATUS)

Address Offset	0x064		
Physical address	0x09800064	**Instance**	SHAM1
Description	Contains RESETDONE bit status		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															RESETDONE

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reserved. Must be written with 0s.	RW	0x00000000
0	RESETDONE	Status of reset. When 1, the SHA1/MD5 has finished a reset. When 0, the SHA1/MD5 is currently performing a reset.	R	1

USB Client

This chapter introduces the universal serial bus (USB) client of the LOCOSTO device.

Topic **Page**

23.1 USB Client Overview

The universal serial bus (USB) device controller supports the implementation of a full-speed (FS) device compatible with the Universal Serial Bus Specification Revision 2.0 and the Universal Serial Bus Specification Revision 1.1.

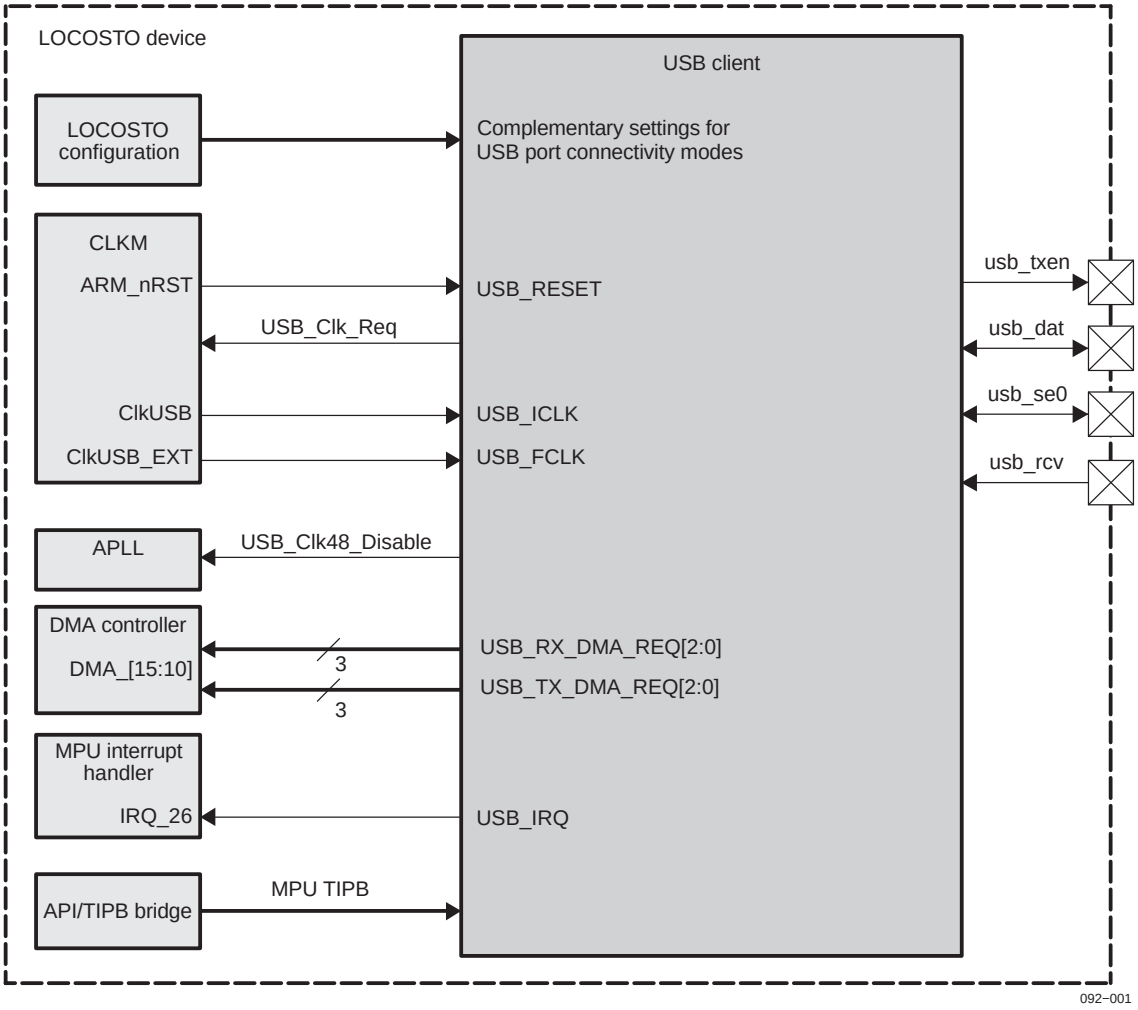
The USB device controller provides an interface between the microprocessor unit (MPU) subsystem and the USB wire and handles USB transactions with minimal MPU software intervention.

The USB device controller module supports one control endpoint (EP0), up to 15 IN endpoints, and up to 15 OUT endpoints. The exact endpoint configuration is software programmable. The specific configuration items are each endpoint, the size in bytes, the direction (IN, OUT), the type (bulk/interrupt or isochronous [ISO]), and the associated number.

The USB device controller module also supports three direct memory access (DMA) channels for IN endpoints and three DMA channels for OUT endpoints for either bulk/interrupt or ISO transactions.

Figure 23-1 shows the USB client and LOCOSTO device interface.

Figure 23-1. USB Highlights



The USB device controller includes the following main features:

- USB 2.0 full-speed (12 Mb/s) and low-speed (1.5 Mb/s) operations
- One control endpoint and up to 15 IN endpoints and 15 OUT software-configurable endpoints

- Each endpoint can be configured for types (interrupt, bulk or isochronous), for size, and for double-buffering operation within a limit of 2K bytes of data.
- Support of suspend/resume and remote wake-up
- Six DMA channels (three IN and three OUT) and three interrupt lines

The LOCOSTO device has no internal USB transceiver and must use external USB transceivers to access an external USB device.

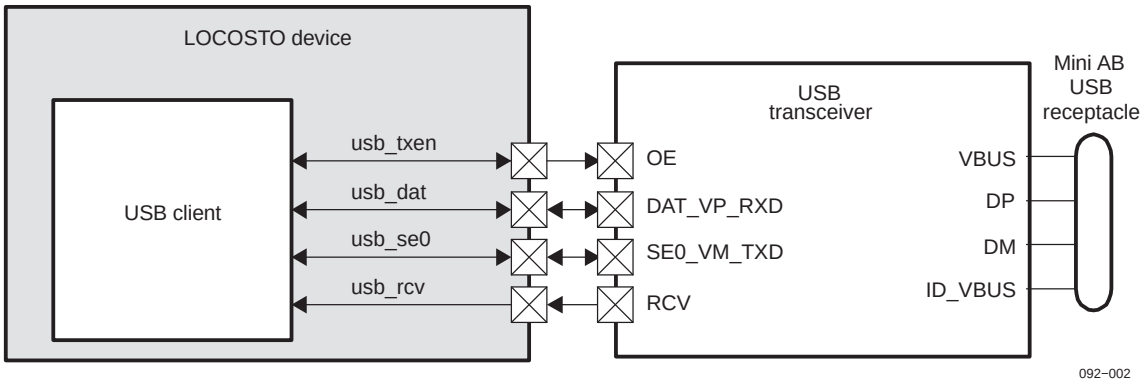
The USB port signal pins interface supports the following external USB transceivers:

- 3-USB device controller pin mode (DAT/SE0 bidirectional mode)
- 4-pin mode (VP/VM bidirectional mode)

23.2 USB Client Environment

Figure 23-2 shows a typical application using the USB device controller.

Figure 23-2. USB Device Controller Typical Application System Overview



The USB device controller physically connects the LOCOSTO device to an external USB transceiver. It converts between signaling appropriate for the LOCOSTO USB device controller and signaling appropriate for the USB wire.

Note: Texas Instruments provides a global solution by connecting the LOCOSTO device to the TWL3031 device. For more information on the TWL3031 device, see your TI representative.

The LOCOSTO device accesses an external USB device using the TWL3031 transceiver. The TWL3031 device includes a single USB on-the-go (OTG) dual-role transceiver compatible with the *Universal Serial Bus Specification Revision 2.0* (high-speed not supported) and the On-The-Go supplement to the USB 2.0 specification, revision 1.0.

The TWL3031 USB interface works in 3-pin bidirectional (DAT/SE0) mode or 4-pin bidirectional (VP/VM) mode. This selection is made through an internal configuration register.

CAUTION

The LOCOSTO-TWL3031 solution does not support the OTG feature because LOCOSTO is slave only.

The LOCOSTO device provides a USB 2.0 full-speed-only controller (no high-speed and no OTG).

23.2.1 Selecting and Configuring USB Connectivity

The process of selecting desired USB connectivity and configuring the LOCOSTO device for that connectivity can be done using the steps provided in this section.

23.2.1.1 Select USB or USB OTG Transceiver Type

USB connectivity on the LOCOSTO USB port requires external components. Several different types of external transceiver signaling are supported. Signaling between the LOCOSTO USB controller and the external USB transceiver for monitoring and controlling the differential USB signal can be performed through a 4-pin or 3-pin signaling interface, with two or more additional control signals provided by additional signals or via an I2C link. OTG transceivers support the 6-pin, 4-pin, and/or 3-pin basic signaling, but generally provide an interrupt output and an I2C link for the additional control and status reporting required by OTG functionality.

Consider which LOCOSTO pins are required in each diagram and determine the best choice (3-pin or 4-pin) in terms of other non-USB functionality that is available for each port. Choose a transceiver type.

Normally, a USB OTG transceiver can be used in place of a USB transceiver; using an OTG transceiver with an I2C interface can reduce the number of required LOCOSTO pins. This can help make non-USB functionality available on the USB ports.

23.2.1.2 Determine USB Mode Control Register Settings

The system control module provides registers that control several aspects of USB pin signaling. These registers must be configured to allow proper USB operation.

- USB_TRX_MODE bits CONFIG.CONF_CORE_REG[9:7]: USB port transceiver mode. These bits select the USB port transceiver interface type. Transceiver signaling type depends on the transceiver signaling mode. The LOCOSTO-supported transceiver signaling types are 3-pin bidirectional (DAT_SE0) mode transceiver signaling (0x0) and 4-pin bidirectional (VP/VM) mode transceiver signaling (0x1).
- REG_USBX_SYNCHRO bit CONFIG.CONF_CORE_REG[10]: This bit avoids glitches on the host and client path, if delay is not balanced, adding a delay of one clock cycle:
  - 0 = No delay is present on the output path.
  - 1 = One 48-MHz clock cycle delay is added to the output signals.
- USB_PRT bits CONFIG.CONF_CORE_REG[6:5]: USB port is available.
  - 0x0 = USB port is not available
  - 0x1 = USB port is available.
  - Others = Reserved
- USB_FILTER_CD bits CONFIG.CONF_CORE_REG[4:2]:
  - 0x0 = 0 ms or no filter
  - 0x1 = 5 ms
  - 0x2 = 10 ms
  - 0x3 = 15 ms
  - 0x4 = 20 ms
  - 0x5 = 25 ms
  - 0x6 = 30 ms
  - 0x7 = 0 ms or no filter
- USB_VBUS_MODE bit CONFIG.CONF_CORE_REG[1]: This bit selects the USB VBUS input source detection mode, which is used to detect USB insertion/disconnection.
  - 0 = Software detection is selected: USB VBUS input uses the USB_VBUS_CTRL bit value.
  - 1 = Hardware detection is selected (not implemented in LOCOSTO).
- USB_VBUS_CTRL bit CONFIG.CONF_CORE_REG[0]: This bit can be programmed to indicate an external USB insertion/disconnection.
  - 0 = Indicates an external USB disconnection
  - 1 = Indicates an external USB insertion

23.2.2 Transceiver Signaling Types

23.2.2.1 USB Transceiver 3-Pin Bidirectional Signaling

Table 23-1 shows the signaling used when a USB or a USB OTG transceiver is connected to the LOCOSTO device and is used in the 3-pin bidirectional DAT/SE0 signaling mode.

Table 23-1. Signaling Between the USB Controller and 3-Pin Bidirectional USB Transceiver Using DAT/SE0 Signaling

Logical Signal Name	LOCOSTO Pin Direction	Transceiver Pin Direction	Description				
TXEN	Output	Input	When low, the USB transceiver drives D+ and D_UnicodeEncodeError_.				
DAT and SE0	Output	Input	When TXEN is low, the LOCOSTO device drives DAT and SE0, and the transceiver drives D+ and D_UnicodeEncodeError_ based on the values of DAT and SE0.				
	Output	Input	TXEN	DAT	SE0	D+	D_UnicodeEncodeError_
			0	0	0	0	0
				1	0	1	0
				X	1	0	0
			TXEN	D+	D_UnicodeEncodeError_	DAT	SE0
			1	0	0	0	1
				0	1	0	0
				1	0	1	0
				1	1	Undefined	Undefined

Note: The LOCOSTO device does not support 3-pin bidirectional signaling using the VP/VM signals.

23.2.2.2 USB Transceiver 4-Pin Bidirectional Signaling

Table 23-2 shows the signaling used when a USB or a USB OTG transceiver is connected to the LOCOSTO device and is used in the 4-pin bidirectional VP/ VM signaling mode.

Table 23-2. Signaling Between the USB Controller and 4-Pin Bidirectional USB Transceiver Using VP/VM Signaling

Logical Signal Name	LOCOSTO Pin Direction	Transceiver Pin Direction	Description		
TXEN	Output	Input	When low, the USB transceiver drives D+ and D_UnicodeEncodeError_.		
VM			Value driven to or received from D_UnicodeEncodeError_		
	Output	Input	TXEN	VM	D_UnicodeEncodeError_
			0	0	0
			0	1	1
	Input	Output	TXEN	D_UnicodeEncodeError_	VM
			1	0	0
			1	1	1
VP			Value driven to or received from D+		
	Output	Input	TXEN	VP	D+

Table 23-2. Signaling Between the USB Controller and 4-Pin Bidirectional USB Transceiver Using VP/VM Signaling (continued)

Logical Signal Name	LOCOSTO Pin Direction	Transceiver Pin Direction	Description		
			0	0	0
			0	1	1
	Input	Output	TXEN	D+	VP
			1	0	0
			1	1	1
RCV	Input	Output	Output from transceiver single-ended D_UnicodeEncodeError_ signal receiver		
			D+	D_UnicodeEncodeError_	RCV
			0	0	X
			0	1	0
			1	0	1
			1	1	X

Note: The LOCOSTO device does not support 4-pin bidirectional signaling using the DAT/SE0 signals.

23.2.3 USB Device Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a non-USB device controller port must implement features, including a USB type-B or mini-B receptacle, VBUS power detection, transient suppression, a controllable pullup resistor to the D+ or D_UnicodeEncodeError_ line, and USB-compatible upstream port transceiver.

As an option, weak pulldown resistors can be used to hold the D+ and D_UnicodeEncodeError_ signals at voltages below the USB transceiver VIL level when no USB host is connected to the USB Type B connector. Keeping D+ and D_UnicodeEncodeError_ voltages below VIL can reduce the USB transceiver IDDQ. The value for the pulldown resistors must be selected carefully so that the circuit meets the requirements of the USB specification.

The LOCOSTO USB device controller supports implementation only as an FS USB device. As such, the pullup resistor must be connected to the D+ signal to indicate implementation of an FS USB device.

23.2.3.1 Device on USB Port Using 3-Pin USB Transceiver

The USB_TRX_MODE bits CONFIG.CONF_CORE_REG[9:7] must be set to 0x0 to allow proper bidirectional operation of the 3-pin USB transceiver.

A USB OTG transceiver can provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity. The software passes the VBUS validity information from the USB OTG transceiver to the device controller by reading the VBUS status through the I^2C link. The OTG transceiver ID pin must be left unconnected, and OTG functionality is not available.

Because the OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D_UnicodeEncodeError_ pulldown controls, those hardware connections can be removed when an OTG transceiver is used. The system software must then control those features through the OTG transceiver I2C register set.

23.2.3.2 Device on USB Port Using 4-Pin USB Transceiver

The USB_TRX_MODE bits CONFIG.CONF_CORE_REG[9:7] must be set to 0x1 to allow proper bidirectional operation of the 4-pin USB transceiver.

A USB OTG transceiver can provide USB device-only functionality instead of a USB transceiver, with appropriate I2C and interrupt connectivity. The software passes the VBUS validity information from the USB OTG transceiver to the device controller by reading the VBUS status through the I^2C link. In this case, the OTG transceiver ID pin is left unconnected and OTG functionality is not available.

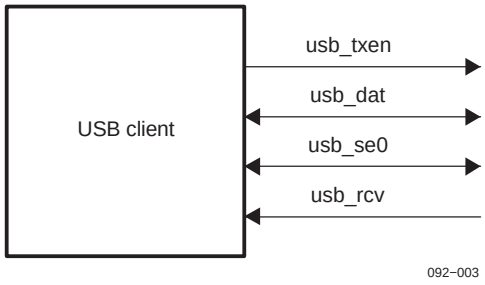
Because the OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D_UnicodeEncodeError_ pulldown controls, these hardware features can be removed when an OTG transceiver is used. The system software must control those features using the OTG transceiver I^2C register set.

23.2.4 USB Client Functional Interfaces

23.2.4.1 Basic USB Device Controller Pins

Figure 23-3 shows the USB device controller functional interface.

Figure 23-3. USB Device Controller Interface Signals



23.2.4.2 USB Device Controller Interface Description

Table 23-3 provides the I/O description.

Table 23-3. I/O Description

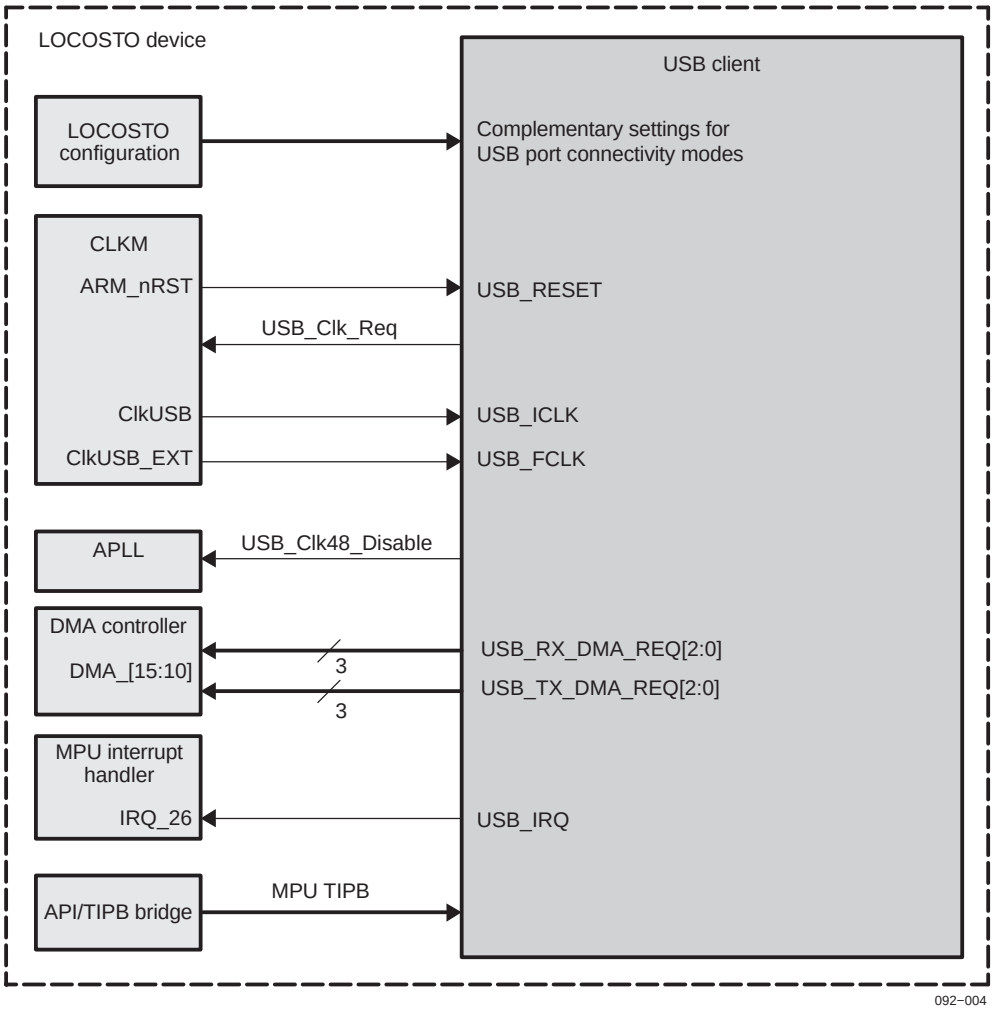
Signal Name	I/O(1)	Description	Reset Value
usb_se0	I/O	SE0 function in 3-pin bidirectional DAT/SE0 mode	Input: Unknown
		VM function in 4-pin bidirectional VP/VM mode	
usb_dat	I/O	DAT function in 3-pin bidirectional DAT/SE0 mode	Input: Unknown
		VP function in 4-pin bidirectional VP/VM mode	
usb_txen	O	Transmit enable	0
usb_rcv	I	Differential receiver signal input (not used in the 3-pin bidirectional DAT/SE0 mode)	Unknown

(1) I = input, O = output

23.3 USB Client Integration

Figure 23-4 shows the USB module integration in the LOCOSTO device.

Figure 23-4. USB Integration



23.3.1 Clocking, Reset and Power-Management Scheme

23.3.1.1 Clocks

23.3.1.1.1 Module Clocks

The USB device controller has a functional clock and an interface clock. Table 23-4 lists the characteristics of each clock.

- The functional clock (48 MHz), USB_FCLK, comes from the analog phase-locked loop (APLL) through the clock manager (CLKM) module. This clock can be shut down by the USB function (USB_Clk48_Disable) when:
  - The USB is suspended (idle state).
  - The USB is disconnected.

  This shutdown must be enabled before it occurs: Set the SOFF_DIS bit USB.SYSCON1[1] TO 0 (default value).
- The interface clock, USB_ICLK, comes from the CLKM module. This is a free-running clock when the system is awake.

Table 23-4. USB Clocks

Type	Name	Source	Frequency	Description
Functional	USB_FCLK	ClkUSB_EXT	48 MHz	Clock provided by the CLKM
Interface	USB_ICLK	ClkUSB	52 or 13 MHz	Clock provided by the CLKM

Note: For more information, see Chapter 6, *Power, Reset, and Clock Management.*

23.3.1.1.2 Clock Requirements

- Input conditions

  The power management circuitry is based on the SOFF_DIS bit USB.SYSCON1[1]: Setting the shutoff disable bit disables the power shutoff circuitry (0: enabled, 1: disabled).
- Output conditions (clock requests)

  To request the external clocks, two signals are generated externally:
  - USB_Clk_Req (1: interface clock can be shut off, 0: interface clock must be activated) is low to request the interface clock; destination is the CLKM.
  - USB_Clk48_Disable (0: functional clock must be free-running, 1: functional clock is shut off) is high to enable the shutoff of the functional (48-MHz) clock; the APLL is the destination.

  Note: The USB_Clk_Req and USB_Clk48_Disable output pins can be used to shut off or wake up the clocks.
- Clock requirement for register accesses

  Any access to the USB device registers (base address 0xFFFF B000) requires the interface clock. In some particular cases, USB_Clk_Req can be inactive 1 (USB suspend or USB cable disconnected), and the system can access the module only by generating the clock of the bus interface.

23.3.1.2 Hardware Reset

The USB device controller reset can be done using a hardware reset.

The hardware reset of the USB device controller can be performed by activating ARM_nRST. The hardware reset signal has a global reset action on the module.

For more information, see Chapter 6, *Power, Reset, and Clock Management*.

23.3.2 Hardware Requests

23.3.2.1 DMA Requests

Six DMA channels (three for IN and three for OUT transactions) support data streaming for USB isochronous or nonisochronous endpoints. Each DMA channel can support one USB endpoint. (Endpoint 0 is not supported.) The DMA channels stream data between the USB endpoint data buffers and other modules through the system DMA.

Table 23-5 lists the DMA channels driven from the USB device controller to the DMA controller to support streaming USB data.

Table 23-5. USB DMA

Name	Mapping	Destination	Description
USB_RX_DMA_REQ0	DMA_10	DMA controller	DMA request for OUT transactions (receive) channel 0
USB_TX_DMA_REQ0	DMA_11	DMA controller	DMA request for IN transactions (transmit) channel 0

Table 23-5. USB DMA (continued)

Name	Mapping	Destination	Description
USB_RX_DMA_REQ1	DMA_12	DMA controller	DMA request for OUT transactions (receive) channel 1
USB_TX_DMA_REQ1	DMA_13	DMA controller	DMA request for IN transactions (transmit) channel 1
USB_RX_DMA_REQ2	DMA_14	DMA controller	DMA request for OUT transactions (receive) channel 2
USB_TX_DMA_REQ2	DMA_15	DMA controller	DMA request for IN transactions (transmit) channel 2

When an RX DMA request is active (0) for a channel (only one active at a given time), data must be read into the DMA FIFO data register (USB.DATA_DMA). When a TX DMA request is active (0) for a channel (only one active at a given time), data must be written into the DMA FIFO data register (USB.DATA_DMA) to be transmitted.

23.3.2.2 Interrupt Requests

One interrupt line is driven out from the USB device controller (USB_IRQ) to the MPU interrupt handler (IRQ_26).

The USB device controller generates the following interrupt types on this interrupt line:

- General USB interrupts (including endpoint 0, DMA, and device states interrupts), USB_IRQ_GEN
- Endpoint-specific interrupts (nonisonchronous interrupt), USB_IRQ_NISO
- Start of frame (SOF) interrupts for isochronous transactions, USB_IRQ_ISO

Table 23-6 describes the USB interrupts generated on the USB_IRQ interrupt line.

Table 23-6. USB Interrupts

Interrupt Name	Mapping	Destination	Description
USB_IRQ	IRQ26	MPU interrupt handler	USB interrupt • DMA transfer completed on transmit channel *n* • Completion of DMA transfer from receive channel *n* • End of transfer (EOT) packet detection on receive channel *n* • SOF detected • OUT transaction detected on endpoint *n* • IN transaction detected on endpoint *n* • Device status change • Setup transaction completed on control endpoint (0) • OUT transaction detected on control endpoint (0) • IN transaction detected on control endpoint (0)

23.3.2.3 USB Detection

When LOCOSTO is in deep sleep mode, the USB device controller can detect a bus connection to an external USB host or hub and generate a deep sleep wake request (USB_Clk_Req) to wake up the system and to get the interface clock (USB_ICLK).

The USB device controller can also generate a USB_Clk_Req request while the USB is connected: If the USB device controller is idle (SUSPEND state) and a resume event occurs, the USB_Clk_Req request is generated. This request does not wake up the system.

Once the interface clock is present, the USB device controller can generate an attached/unattached interrupt when it detects the LOCOSTO device connected to an external USB host or hub or when it becomes disconnected. This attached/unattached interrupt uses the general USB interrupt line (USB_IRQ connected to the MPU interrupt handler, IRQ_26), and the connection state is given by the ATT bit USB.DEVSTAT[0].

The USB device controller can detect whether or not the USB is connected through either software or hardware. This choice (soft/hard) is made by the USB_VBUS_MODE bit CONFIG.CONF_CORE_REG[1] of the LOCOSTO system control module:

- 1 (default value) = Software detection is selected.
- 0 = Hardware detection is selected (not implemented in LOCOSTO).

23.3.2.3.1 Software Detection

Software detection depends on the USB_VBUS_CTRL bit CONFIG. CONF_CORE_REG[19] of the LOCOSTO system control module:

- 0 (default) = The USB is not connected.
- 1 = The USB is connected.

23.3.2.3.2 Hardware Detection

This detection is not implemented in the LOCOSTO device.

23.4 USB Client Functional Description

23.4.1 USB Device Transactions

The MPU subsystem includes an interrupt at the end of a USB transaction if the MPU subsystem has actions to perform. Isochronous transactions are an exception because isochronous interrupt information is available at the SOF interrupts. The MPU interrupt service routine (ISR) code determines which endpoint and direction causes the interrupt and acts appropriately.

The following subsections describe the activities surrounding USB transactions that are not part of a DMA transfer. Cases in which a transaction occurs before the previous transaction is handled by the MPU subsystem are not considered. The information is organized so that each section deals with one type and direction of transaction, such as the following:

- Nonisochronous, nonsetup OUT transactions
- Nonisochronous IN transactions
- Isochronous OUT transactions
- Isochronous IN transactions

Each section thus focuses only on a specific transaction style without adding to the confusion of special cases related to other styles.

23.4.2 Nonisochronous, Nonsetup OUT (USB Host to MPU) Transactions

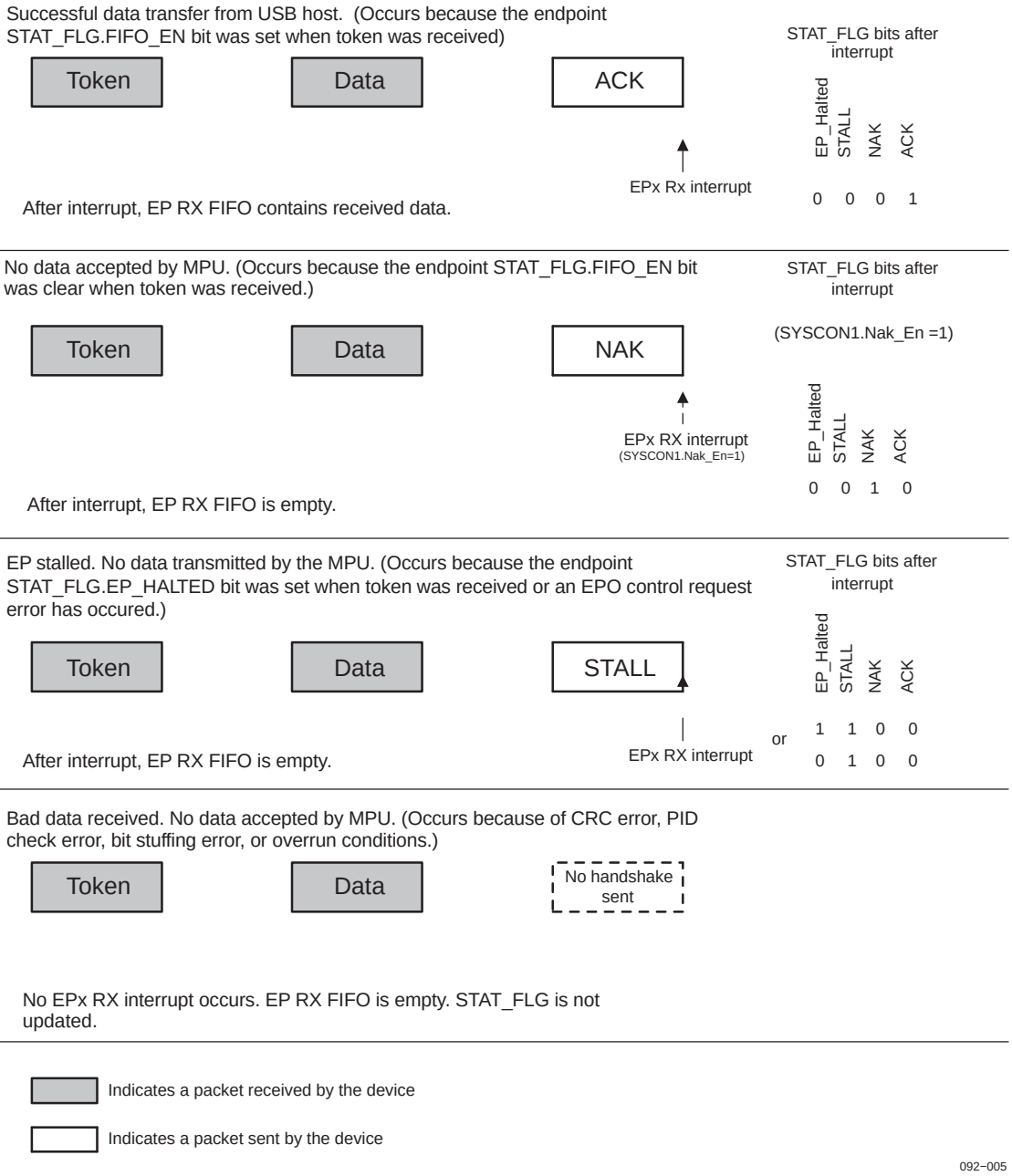
Nonisochronous, nonsetup OUT transactions are USB transactions in which data is moved from the USB host to the MPU subsystem, the USB handshaking protocols are in effect, and data transmission is ensured. These types of transactions apply to all OUT transactions on bulk and interrupt endpoint types, and to nonsetup transactions on control endpoints.

Figure 23-5 shows the various USB protocol conditions that can occur during nonisochronous, nonsetup OUT transactions. Figure 23-5 also shows the three phases that can occur in an OUT transaction, the direction of information flow for each phase, when endpoint interrupts are generated, and the resulting STAT_FLG bits for the endpoint.

The top three cases show the normal USB handshaking modes: acknowledge (ACK, good data received), nonacknowledge (NAK, device not ready to receive data), and STALL (the device is in a condition where the endpoint cannot handle OUT transactions).

The last case is an abnormal instance in which the token packet or the data packet is received with errors. The RX FIFO contains only valid receive data under the first (ACK) case.

Figure 23-5. Nonisochronous, Noncontrol OUT Transaction Phases and Interrupts



23.4.2.1 Nonisochronous, Noncontrol OUT Endpoint Handshaking Conditions

An endpoint SET_FIFO_EN bit CTRL[2] provides the main control for the endpoint capability to allow successful OUT transaction data reception for the endpoint. If at the beginning of an OUT transaction to an endpoint, the endpoint FIFO_EN bit STAT_FLG[2] is 1, the USB module can accept the OUT transaction data to the RX FIFO. When the transaction completes, the USB module can return ACK to the PC to indicate that the data is received correctly (see the top case shown in Figure 23-5.

If, however, the endpoint FIFO_EN bit STAT_FLG[2] is 0 at the beginning of an OUT transaction to the endpoint, the USB module returns NAK during the handshake phase to indicate that the endpoint did not accept the data (see the second case shown in Figure 23-5).

The USB host is not required to send a whole RX FIFO worth of data to the endpoint during an OUT transaction. In this case, the RX FIFO is not full when the endpoint RX interrupt is generated. The MPU code must not read too much data. The MPU code must read the RXF_COUNT value RXFSTAT[9:0] before reading data from the RX FIFO.

At the end of a USB OUT transaction to an endpoint in which the data is acknowledged (ACKed), the hardware clears the endpoint FIFO_EN bit STAT_FLG[2]. After the MPU software deals with the OUT transaction data in the endpoint RX FIFO, it must set the endpoint SET_FIFO_EN bit CTRL[2] to re-enable the endpoint OUT transaction reception. The MPU software can use the endpoint SET_FIFO_EN bit CTRL[2] as a receive flow control mechanism.

23.4.2.1.1 Acknowledged Transactions (ACKs)

At the completion of an OUT transaction to an endpoint, the USB module issues an endpoint-specific interrupt to the MPU subsystem and the STAT_FLG is updated. In response to the endpoint interrupt, the MPU subsystem must read the EPN_STAT register to identify the endpoint causing the interrupt, then write 1 to the interrupt bit to clear it.

The MPU subsystem must then set the EP_NUM bits EP_NUM[3:0] to the endpoint number and the EP_SEL bit EP_NUM[5] to 1, and then read the endpoint status from STAT_FLG. The ACK bit STAT_FLG[3] is set to indicate that the endpoint received a transaction to which the USB module signaled ACK handshaking.

If the NON_ISO_FIFO_EMPTY bit STAT_FLG[1] is cleared, the transaction sent 1 or more bytes of data (but less than or equal to the physical size of the endpoint RX FIFO), and the data is in the endpoint RX FIFO. The MPU subsystem reads the RXF_COUNT value RXFSTAT[9:0] to determine the number of bytes to read from RX FIFO. The MPU subsystem can then read RX data from DATA register.

When the MPU subsystem reads the data from the FIFO, it sets the SET_FIFO_EN bit CTRL[2] to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO, and then clears the EP_SEL bit EP_NUM[5]. This clears the ACK bit STAT_FLG[3] for this endpoint to allow the next transaction status to be written into the STAT_FLG register.

23.4.2.1.2 Nonacknowledged Transactions (NAKs)

The device can be configured by the NAK_EN bit SYSCON1[4] either to inform or not inform the MPU subsystem of a NAKed transaction. If the NAK_EN bit SYSCON1[4] is cleared, no interrupt is asserted to the MPU subsystem when an OUT transaction completes with a NAK handshake and the NAK bit STAT_FLG[4] is not set.

If the NAK_EN bit SYSCON1[4] is set, the USB module issues an endpoint-specific interrupt to the MPU subsystem at the completion of an OUT transaction to an endpoint, and the NAK bit STAT_FLG[4] is set. In response to the endpoint interrupt, the MPU subsystem must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write 1 to the interrupt bit to clear it.

The MPU subsystem must then set the EP_NUM bits EP_NUM[3:0] to the endpoint number and the EP_SEL bit EP_NUM[5] to 1, and then read the endpoint status from the STAT_FLG register. The NAK bit STAT_FLG[4] is set to indicate that the endpoint received a transaction to which the USB module signaled NAK handshaking.

The MPU subsystem must set the SET_FIFO_EN bit CTRL[2] to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO, and then clear the EP_SEL bit EP_NUM[5]. This clears the NAK bit STAT_FLG[4] for this endpoint to allow the next transaction status to be written into the STAT_FLG register.

23.4.2.2 Nonisochronous, Noncontrol OUT Transaction Error Conditions

23.4.2.2.1 STALLed Transactions

The USB module responds to an endpoint OUT transaction with a STALL handshake to indicate an error condition on the endpoint, either when the endpoint EP_HALTED bit STAT_FLG[6] is set or when a request error occurs (control transactions only). When an endpoint OUT transaction is given a STALL handshake, the endpoint STALL bit STAT_FLG[5] is set and an endpoint-specific interrupt is generated for the endpoint.

The FIFO_EN bit STAT_FLG[2] has a lower priority than the EP_HALTED bit STAT_FLG[6] when the EP_HALTED bit STAT_FLG[6] is set and transactions to the RX endpoint are stalled, regardless of the FIFO_EN value STAT_FLG[2]. When the FIFO_EN bit STAT_FLG[2] is set, it is automatically cleared at the end of the STALLed transaction and RX FIFO is cleared.

In response to the endpoint interrupt, the MPU subsystem must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write 1 to the interrupt bit to clear it. The MPU subsystem must then set the EP_NUM bits EP_NUM[3:0] to the endpoint number and the EP_SEL bit EP_NUM[5] to 1, and then read the endpoint status from STAT_FLG. The STALL bit STAT_FLG[5] is set to indicate that the endpoint received a transaction to which the USB module signaled STALL handshaking.

If the EP_HALTED bit STAT_FLG[6] is set by the MPU subsystem and can be removed, the MPU subsystem must set the CLR_HALT bit CTRL[7] to clear the condition and set the SET_FIFO_EN bit CTRL[2] to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO.

If the EP_HALTED bit STAT_FLG[6] is set in response to a SET_FEATURE request sent by the USB host, or if the bit is cleared (control transaction only), the MPU subsystem has no action to perform and must clear the EP_SEL bit EP_NUM[5]. This clears the STALL bit STAT_FLG[5] for this endpoint and allows the next transaction status to be written to the STAT_FLG register.

23.4.2.2.2 Packet Errors

If a receive data error occurs during an endpoint OUT transaction (token or data packet), the USB module does not provide a handshake during the handshake phase of the transaction and no interrupt is asserted to the MPU subsystem (see the fourth case shown in Figure 23-5).

Additionally, the endpoint RX FIFO is not filled, and the FIFO_EN bit STAT_FLG[2] is not cleared. If the MPU subsystem clears the RX FIFO during the data packet of an OUT transaction, no handshake is returned to the USB host to signal an error.

23.4.2.2.3 Sequence Bit Errors

If the core does not receive an expected DATA PID during an OUT transaction, the module automatically returns an ACK handshake to the USB host, regardless of the FIFO_EN bit STAT_FLG[2] (see the USB specification). Data is ignored, and no interrupt is asserted to the MPU subsystem.

This error occurs when the ACK handshake received from a previous OUT transaction is corrupted by the USB host.

23.4.2.3 Nonisochronous, Noncontrol OUT Endpoint FIFO Error Conditions

If the USB host tries to fill more data into an endpoint RX FIFO than the FIFO can hold, a FIFO overrun occurs. The USB module does not provide a handshake during the handshake phase of the transaction and no interrupt is asserted to the MPU subsystem. Additionally, the endpoint RX FIFO is not filled, and the FIFO_EN bit STAT_FLG[2] is not cleared.

The MPU subsystem must not read more data from RX FIFO than the value indicated by the RXF_COUNT bits RXFSTAT[9:0].

23.4.3 Nonisochronous IN (MPU to USB Host) Transactions

Nonisochronous IN transactions refer to USB transactions in which data is moved from the MPU subsystem to the USB host, where the USB handshaking protocols are in effect, and data transmission is ensured. These transactions are the IN transactions that occur on control, bulk, and interrupt endpoints. These transactions do not ensure USB bandwidth.

To provide data for an endpoint IN transaction, the MPU code writes the transmit data into the endpoint transmit FIFO. The MPU code must first wait until the USB finishes with any previous TX data for the endpoint (if data was written previously to the TX FIFO). This must be done by a proper response to the endpoint- specific transmit interrupts.

When an IN transaction to the endpoint occurs, if the endpoint FIFO_EN bit STAT_FLG[2] bit is set, the USB module sends any data that is in the endpoint TX FIFO during the data phase. If the TX FIFO is empty and the FIFO_EN bit STAT_FLG[2] is set when an IN transaction to the endpoint occurs, a 0-byte data packet is sent.

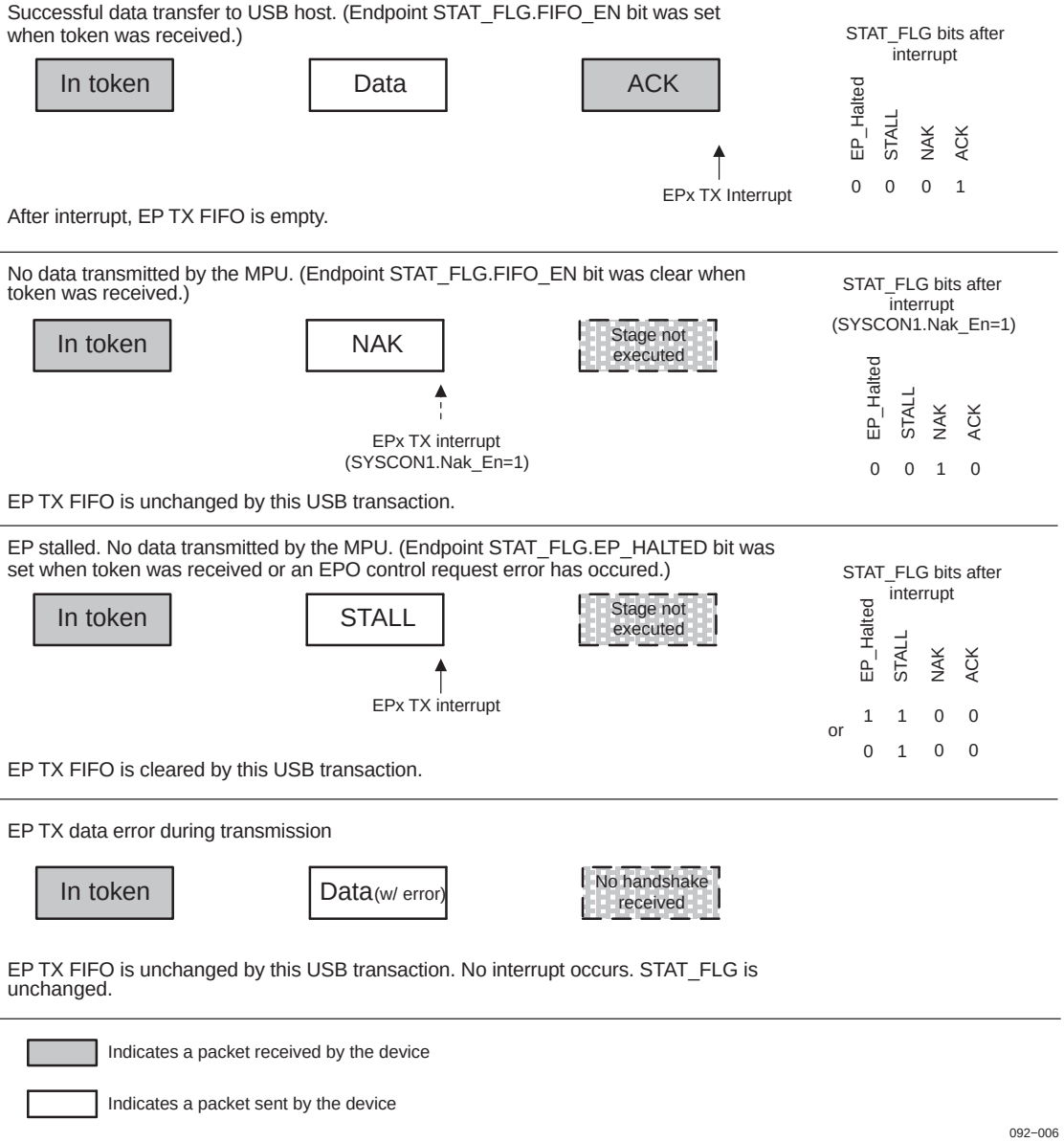
After the endpoint previous transmit activity is handled, the MPU code sets the EP_SEL bit EP_NUM[5] and gains access to the endpoint FIFO and status. The MPU subsystem then can write the new transmit data to the endpoint TX FIFO via the DATA register (being careful not to overflow the FIFO).

Once all of the transmit data is written to the endpoint FIFO, the MPU code sets the SET_FIFO_EN bit CTRL[2] to allow the USB to use the endpoint TX FIFO, and then clears the EP_SEL bit EP_NUM[5]. The data in the endpoint TX FIFO is sent to the USB host the next time an IN transaction to the endpoint occurs.

Figure 23-6 shows the various USB protocol conditions that can occur during nonisochronous IN transactions, diagramming the three phases of the IN transaction, the direction of information flow for each phase, when endpoint-specific interrupts are generated, and the resulting STAT_FLG bits for the endpoint.

The top three cases in Figure 23-6 show the normal USB handshaking: acknowledge ACK (data sent by the USB module and received properly by the USB host), NAK (the device is not ready to send data to the USB host), and STALL (the device is in a condition where the endpoint cannot handle IN transactions). The last case shows an abnormal case in which there is an error, either in the token packet received by the core or in the data packet received by the USB host.

Figure 23-6. Nonisochronous IN Transaction Phases and Interrupts



23.4.3.1 Nonisochronous IN Endpoint Handshaking

Per the USB specifications for IN transactions, the USB host can provide only one of two handshakes to the USB function during the handshake phase: ACK or no handshake at all. The first indicates a successful transfer (see the first case in Figure 23-6). The second indicates that the host received a garbled data packet (see the last case shown in Figure Figure 23-6).

23.4.3.1.1 Acknowledged Transactions (ACKs)

When the endpoint IN transaction completes on the USB with an ACK handshake, the endpoint generates an endpoint-specific interrupt to the MPU subsystem (see the first case in Figure 23-6). In response to the endpoint interrupt, the MPU subsystem must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write 1 to the interrupt bit to clear it.

The MPU subsystem must then set the EP_NUM bits EP_NUM[3:0] to the endpoint number, the EP_DIR bit EP_NUM[4] to 1 (to signal an IN endpoint), and the EP_SEL bit EP_NUM[5] to 1, and then read the endpoint status from the STAT_FLG register. The ACK bit STAT_FLG[3] is set to indicate that the endpoint received an ACK handshake from the USB host, and the TX FIFO is empty (because any data that was in the TX FIFO was transmitted during the IN transaction).

If the MPU subsystem has more data to transmit to the USB host, it must fill the TX FIFO following the process previously described. The MPU subsystem must then clear the EP_SEL bit EP_NUM[5], which clears the ACK bit STAT_FLG[3] for this endpoint to allow the next transaction status to be written to the STAT_FLG register.

23.4.3.1.2 Nonacknowledged Transactions (NAKs)

If the MPU subsystem is not ready to provide transmit data an IN endpoint, the core provides a NAK handshake to the IN transaction to that endpoint. Readiness to transmit data endpoint FIFO_EN bit STAT_FLG[2]; when set to 1, it indicates in the TX FIFO can be sent to the USB host. When the endpoint STAT_FLG[2] bit is set to 0 and an IN transaction to the endpoint NAK handshake is sent, indicating that the MPU subsystem handle the request.

If the NAK_EN bit SYSCON1[4] is cleared, when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, STAT_FLG is not updated and no endpoint-specific interrupt to the MPU subsystem is generated. If the NAK_EN bit SYSCON1[4] is set, when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, the NAK bit STAT_FLG[4] is set and an endpoint-specific interrupt to the MPU subsystem is generated.

In response to the endpoint interrupt, the MPU subsystem must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write 1 to the interrupt bit to clear it. The MPU subsystem must then set the EP_NUM bits EP_NUM[3:0] to the endpoint number, the EP_DIR bit EP_NUM[4] to 1 (to signal an IN endpoint), and the EP_SEL bit EP_NUM[5] to 1, and then read the endpoint status from the STAT_FLG register.

The NAK bit STAT_FLG[4] is set to indicate that the endpoint sent a NAK handshake to the USB host. If the MPU subsystem has data to transmit to the USB host, it must fill the TX FIFO following the process previously described. The MPU subsystem must then clear the EP_SEL bit EP_NUM[5], which clears the NAK bit STAT_FLG[4] for this endpoint to allow the next transaction status to be written to the STAT_FLG register. Signaling NAK does not cause the endpoint TX FIFO to be cleared (because the MPU subsystem still retains control of the FIFO).

Signaling a NAK handshake for several consecutive endpoint transactions can cause the PC host to discard the transaction; therefore, a NAK is not necessarily a good mechanism when the MPU subsystem cannot service a request for long periods of time.

23.4.3.2 Nonisochronous IN Transaction Error Conditions

23.4.3.2.1 STALLed Transactions

The USB module sends a STALL handshake to the USB host during the data phase of the transaction to the IN endpoint under two conditions: If the endpoint EP_HALTED flag STAT_FLG[6] is set or if a request error occurs (control transaction only). A USB STALL handshake indicates that the device endpoint is in a condition in which it cannot transfer data and instructs the USB host not to retry the transaction. The device typically requires intervention using another mechanism to clear the condition, usually a control transfer via endpoint 0.

The MPU can set the endpoint EP_HALTED bit STAT_FLG[6] by writing the appropriate value in the EP_NUM register, and then setting the endpoint SET_HALT bit CTRL[6]. The MPU can clear the endpoint by selecting it, and then setting the endpoint CLR_HALT bit CTRL[7].

When the endpoint EP_HALTED bit STAT_FLG[6] is set, the endpoint signals STALL for its IN transactions until the HALT condition clears. When the STALL handshake is sent in response to a transaction to the endpoint, the STALL bit STAT_FLG[5] is set, and an endpoint-specific interrupt to the MPU is generated.

In response to the endpoint interrupt, the MPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write 1 to the interrupt bit to clear it. The MPU must then set the EP_NUM bits EP_NUM[3:0] to the endpoint number, the EP_DIR bit EP_NUM[4] to 1 (to signal an IN endpoint), the EP_SEL bit EP_NUM[5] to 1, and then read the endpoint status from the STAT_FLG register.

The STALL bit STAT_FLG[5] is set to indicate that the endpoint sent a STALL handshake to the USB host. The MPU must then clear the EP_SEL bit EP_NUM[5], which clears the STALL bit STAT_FLG[5] for this endpoint and allows the next transaction status to be written to the STAT_FLG register.

Except for control endpoint 0, separate endpoint halt bits are defined for each direction; therefore, for a given endpoint number, the TX can be halted when the RX is not.

23.4.3.2.2 Packet Errors

If an error (cyclic redundancy check [CRC], bit stuffing, or a PID check) occurs during the token packet of a USB IN transaction to a nonisochronous endpoint, the USB block ignores the transaction. No endpoint-specific interrupt to the MPU subsystem occurs for transactions with corrupted packets. If the MPU subsystem clears the TX FIFO during the data packet of an IN transaction, a bit-stuffing error is forced.

If the USB host does not return a handshake after an IN transaction (because of an error during transmission), the USB device controller module, after a time-out, detects an error. The data to transmit is still in the TX FIFO to be resent during the next IN transaction, the FIFO_EN bit STAT_FLG[2] is not cleared, and interrupt is not asserted to the MPU.

23.4.3.3 Nonisochronous IN Endpoint FIFO Error Conditions

The MPU cannot write more data to the TX FIFO than the configured FIFO size.

23.4.4 Isochronous OUT (USB Host to MPU) Transactions

Isochronous OUT transactions are USB transactions in which a given amount of data is transferred from the USB host to the USB device controller module device every 1-ms USB frame. No USB handshaking is provided, and no endpoint- specific interrupt to the MPU is generated at completion of an isochronous OUT transaction. The MPU handles isochronous OUT data at each SOF interrupt.

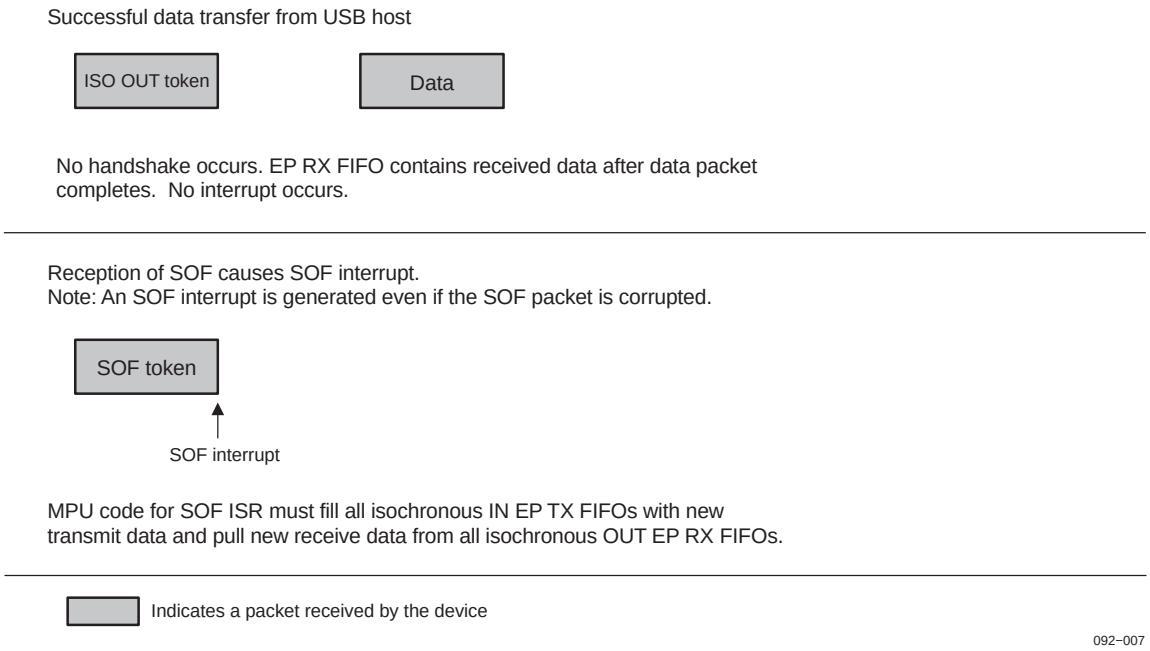
At each SOF interrupt, for each isochronous OUT endpoint, the MPU code must select the endpoint by writing the appropriate value in the EP_NUM register and must check the ISO_FIFO_EMPTY bit STAT_FLG[9]. If the RX FIFO contains data, the code must read the RXF_COUNT value RXFSTAT[9:0] (if the number of bytes to read from RX FIFO is not known), read all the bytes from RX FIFO through the data register, and then clear the EP_SEL bit EP_NUM[5].

Because the USB transaction for the isochronous endpoint can occur at any time during the USB 1-ms frame, the USB interface implements a double-buffering of the endpoint receive data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint. At all times, one of the two FIFOs is foreground and the other is background.

The USB interface side of the USB module is allowed to write to the background RX FIFO, and the MPU is allowed to read to the foreground RX FIFO. The designations foreground and background are swapped at each SOF. Isochronous endpoint FIFOs in the background are always enabled to the USB; the foreground FIFOs are enabled to the MPU.

Figure 23-7 shows the two phases (ISO OUT token and data) of an isochronous OUT data transfer at the top of the figure. No endpoint-specific interrupt to the MPU is generated for the isochronous OUT transaction. The data for isochronous endpoints are instead handled by the MPU at each SOF interrupt, which is shown as the second case in Figure 23-7.

Figure 23-7. Isochronous OUT Transaction Phases and Interrupts



23.4.4.1 Isochronous OUT Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STALL bit STAT_FLG[5], the NAK bit STAT_FLG[4], and the ACK bit STAT_FLG[3] for isochronous endpoints always return 0. Because there is no handshake, the endpoint-specific interrupt for isochronous endpoints is not used.

23.4.4.2 Isochronous OUT Transaction Error Conditions

If the MPU subsystem fails to read all of the data in the ISO OUT endpoint foreground FIFO by the time the foreground and background FIFOs are switched (at the next SOF), the endpoint FIFO being switched to the background is flushed, and the DATA_FLUSH bit STAT_FLG[13] is asserted for the duration of the next frame.

There is no special indication when the USB host does not provide a transaction to an ISO OUT endpoint during a frame, but once the FIFO that was background in that frame is foreground, the FIFO is empty (a zero-length data ISO OUT transaction also results in an empty FIFO and cannot be distinguished from a missed ISO OUT transaction).

If an ISO OUT transaction occurs with data error (CRC, PID check, or bit stuffing), the RX FIFO is empty at the next SOF interrupt, and the ISO_ERR bit STAT_FLG[12] is asserted for the duration of the next frame.

23.4.4.3 Isochronous OUT Endpoint FIFO Error Conditions

The MPU must never read more data than the value given by the RXF_COUNT bits RXFSTAT[9:0].

If the USB host sends more data than the FIFO can contain, the FIFO is cleared and the ISO_ERR bit STAT_FLG[12] is set at the next SOF interrupt. A properly configured USB system does not do this.

Note: Both foreground and background isochronous FIFOs are cleared when the CLR_EP bit CTRL[1] bit is set.

23.4.5 Isochronous IN (MPU to USB Host) Transactions

Isochronous IN transactions are USB transactions in which a given amount of data is transferred from the USB device controller module device to the USB host every 1-ms USB frame. No handshaking is provided.

The USB module provides double-buffering of data for ISO IN endpoints; the background FIFO is the source of data for IN transactions to the isochronous endpoint, and the foreground FIFO can be written to by the MPU. When an IN transaction to an isochronous endpoint occurs, the USB module sends all data found in the endpoint background TX FIFO. The MPU provides new data to the isochronous IN endpoint foreground TX FIFO at each SOF interrupt.

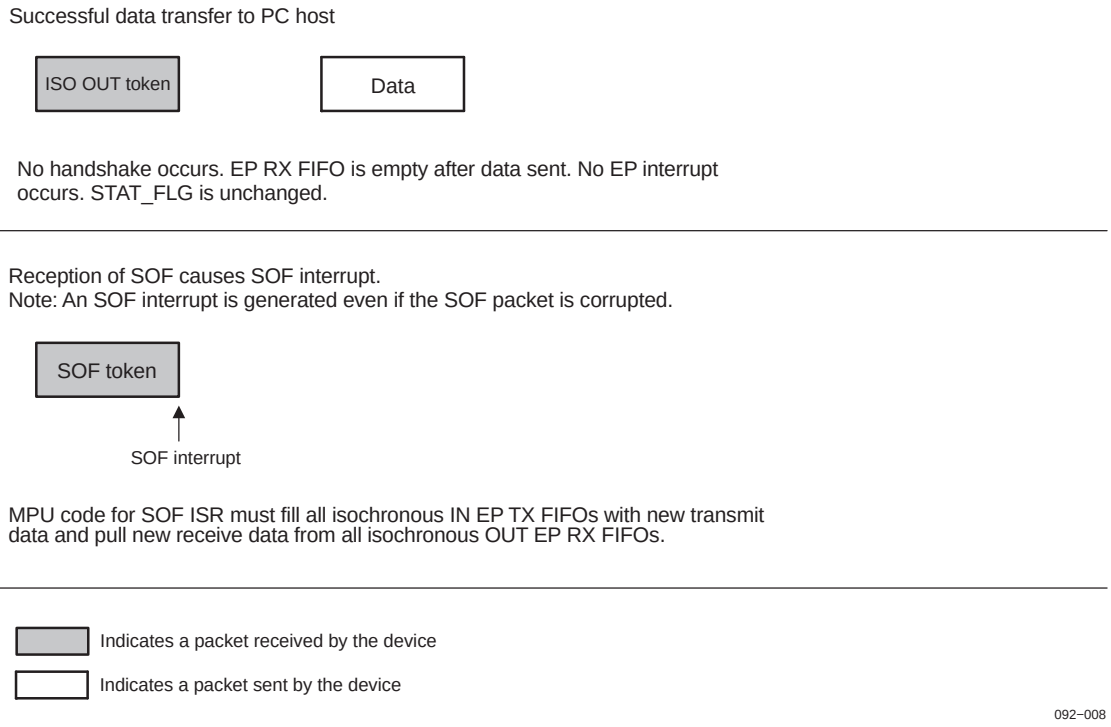
In response to the SOF interrupt, for each isochronous IN endpoint, the MPU code selects the endpoint (via the EP_NUM register), and then fills the endpoint TX FIFO (via the DATA register). When all transmit data are written to the FIFO, the MPU code must clear the EP_SEL bit EP_NUM[5].

Because the USB transaction for the isochronous endpoint can occur at any time during the USB 1-ms frame, the USB interface implements a double-buffering of the endpoint transmit data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint. At all times, one of the two FIFOs is foreground and the other is background.

The USB interface side of the USB module is allowed to read from the background TX FIFO, and the MPU is allowed to write to the foreground TX FIFO. The designations foreground and background are swapped, and the new background TX FIFO is cleared at each SOF. Because isochronous endpoints implement double-buffering, isochronous endpoints do not control access to the FIFOs through the SET_FIFO_EN bit CTRL[2]; the SET_FIFO_EN bit CTRL[2] and the FIFO_EN bit STAT_FLG[2] are not implemented for isochronous IN endpoints.

Figure 23-8 shows the transaction phases associated with isochronous IN transactions and the SOF transaction. No endpoint-specific interrupt to the MPU subsystem is generated as a result of an isochronous IN transaction, and there is no handshake phase. The SOF transaction causes an SOF interrupt to the MPU; it is assumed that the MPU refills the isochronous IN endpoint transmit FIFO at each SOF interrupt.

Figure 23-8. Isochronous IN Transaction Phases and Interrupts



23.4.5.1 Isochronous IN Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STALL bit STAT_FLG[5], the NAK bit STAT_FLG[4], and the ACK bit STAT_FLG[3] for isochronous endpoints always return 0. Because there is no handshake, there is no endpoint-specific interrupt to the MPU subsystem to report handshake results for isochronous endpoints.

23.4.5.2 Isochronous IN Transaction Error Conditions

If the USB host did not successfully complete an isochronous IN transaction in the previous frame, and if data were present in the TX FIFO to be sent at the IN transaction, the MISS_IN bit STAT_FLG[14] is asserted for the duration of the next frame. If the isochronous IN endpoint is cleared in the middle of a USB transaction to the background FIFO, the macro forces a bit-stuffing error for the isochronous transaction.

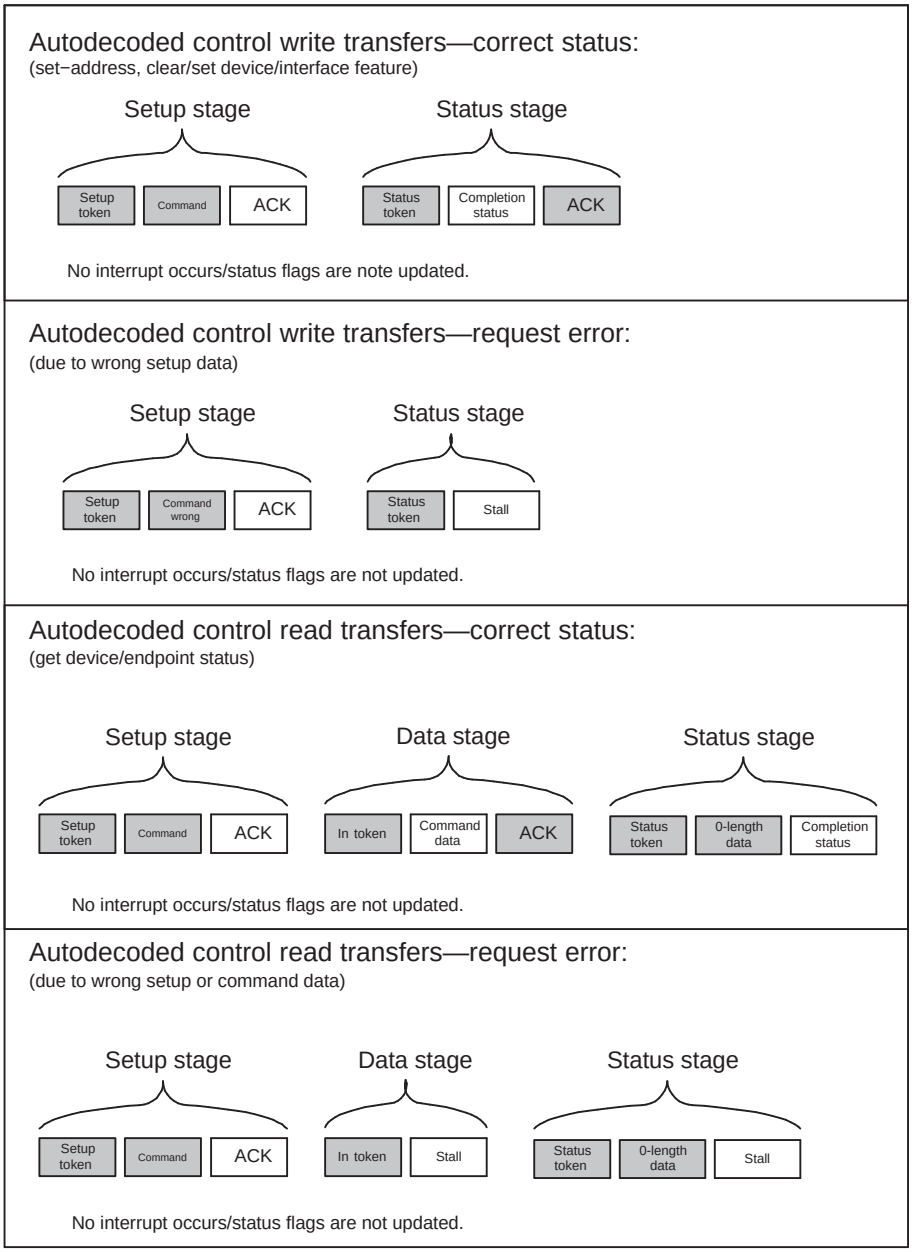
23.4.5.3 Isochronous IN Endpoint FIFO Error Conditions

If the MPU tries to overfill the configured endpoint FIFO, data written to the DATA register after the TX FIFO is full is lost. Any data successfully put in the FIFO is transmitted when the FIFO is the background FIFO and an IN transaction for that endpoint occurs. Because an ISO TX FIFO is cleared automatically on the toggle from background to foreground, there is no reason to clear the FIFO. However, if the MPU does not send the data it wrote, clearing the endpoint is the only mechanism to clear the FIFO.

23.4.6 Control Transfers on Endpoint 0

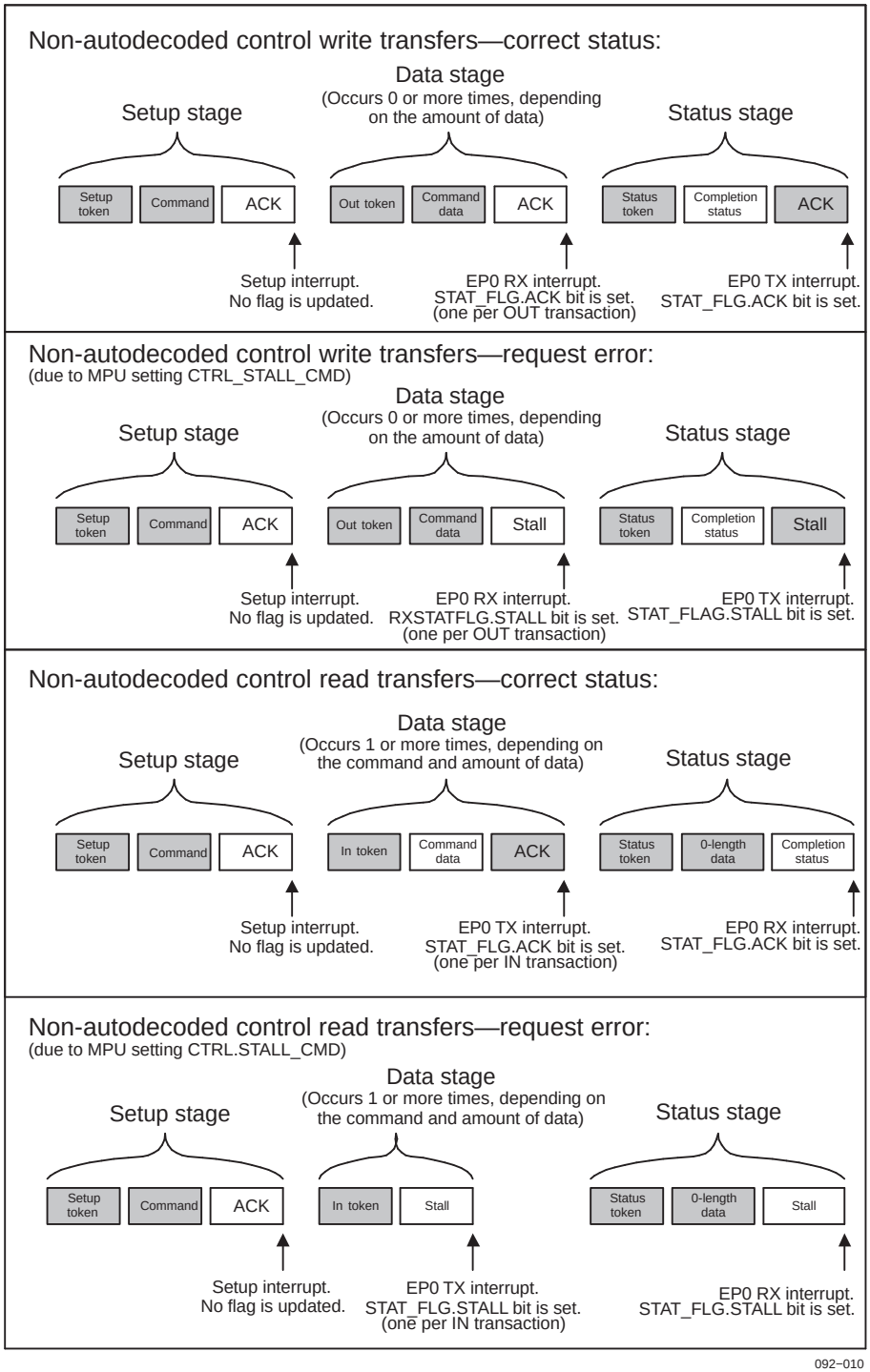
Control transfers on endpoint 0 include control write and control read transfers. Control write and control read transfers are each composed of two or more transactions to endpoint 0. Additionally, the USB device controller module can autodecode some control write and control read transfers. These operations are summarized in Figure 23-9 and Figure 23-10. An IN or an OUT transaction is received from a control request. This transaction is automatically stalled by the core.

Figure 23-9. Stages and Transaction Phases of Autodecoded Control Transfers



092−009

Figure 23-10. Stages and Transaction Phases of Non-Autodecoded Control Transfers



Non-autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0 and have specific USB protocol meanings. These transfers are not handled automatically by the core. The USB device controller block automatically provides an ACK handshake for the setup stage transaction, but the data and status stage transaction handshaking is accomplished using the usual RX and TX control bits that affect transaction handshaking.

A general USB interrupt to the MPU subsystem occurs at the end of each transaction of each stage of a control transfer. The MPU must perform the following actions on non-autodecoded control transfers:

- Process the setup phase setup USB interrupt. The MPU reads the control transfer command from the setup FIFO and decodes the command. For control reads, the MPU subsystem fetches the requested read data and places it (or as much of the read data as fits) into the endpoint 0 FIFO, and then enables the endpoint 0 FIFO. For control writes, the MPU code only enables the endpoint 0 FIFO. The MPU code also sets any flags needed to process endpoint 0 USB interrupts during the control transfer.
- Process the data phase endpoint 0 general USB interrupt(s). For control reads, the data phase general USB endpoint 0 TX interrupt indicates that the previously provided transmit data is sent. Any additional data must be written to the endpoint 0 FIFO. For control writes, the write data must be pulled from the endpoint 0 FIFO, and, when all control write data is available, interpret the write data and act on the write request. After handling the last data phase interrupt, the MPU must set the endpoint 0 control bits to signal the desired status to the host.
- Process the status stage endpoint 0 general USB interrupt. The MPU provides its completion status back to the USB host during this stage, using the status in either the data phase of the transaction (for control write transfers) or the handshake phase of the transaction (for control read transfers).

Autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0 and have specific USB protocol meanings. These transfers are handled automatically by the USB device controller block without intervention by the MPU. The USB device controller block handles all handshaking automatically and without regard to the endpoint 0 control bits that affect normal (noncontrol transfer) transaction handshaking. No interrupt is asserted to the MPU during autodecoded control transfers.

If a request defined by the *Universal Serial Bus Specification Revision 1.1* is not associated with the data of the setup phase, the core stalls the request and the MPU is not informed of its occurrence (autodecoded).

When a setup token is identified, the USB decode module must monitor the setup stage data packet, decode it, and determine if it is an autodecoded or a non-autodecoded transfer and a control read or a control write. If it is a valid non-autodecoded request, the setup FIFO is immediately cleared and control of the FIFO is immediately taken away from the MPU (if the MPU controls the FIFO). New setup data are placed into the setup FIFO, and the setup interrupt flag is set (SETUP bit IRQ_SRC[2]).

In response to the setup interrupt, the MPU must select the setup FIFO by setting the SETUP_SEL bit EP_NUM[6]. This clears the SETUP bit IRQ_SRC[2]. The MPU must then read 8 bytes from the setup FIFO, clear the SETUP_SEL bit EP_NUM[6], and confirm that the SETUP bit IRQ_SRC[2] is not reset by a new setup transaction.

If the SETUP bit IRQ_SRC[2] is asserted, the MPU subsystem must discard the previously read data and handle the new setup packet as previously explained. Thus, the MPU never misses a new setup transaction (per the *Universal Serial Bus Specification Revision 1.1*).

23.4.6.1 Autodecoded Control Write Transfers

For set address control write transfers, the USB address provided in the setup token is captured to the USB module device address register. If the new address is different from 0, the device moves into an addressed state (ADD bit DEVSTAT[2] set) if it is not already addressed.

For set and clear feature control writes, the appropriate feature information bit is set or cleared. When a set or clear feature transfer occurs to set or clear the device remote wake-up feature, the R_WK_OK bit DEVSTAT[6] is set or cleared, as appropriate. If a set or clear interface feature occurs, the core automatically stalls the request because no feature is defined for the interface (see the *Universal Serial Bus Specification Revision 1.1*).

In compliance with the Universal Serial Bus Specification Revision 1.1, a SET_ADDRESS request is effective after the status stage of the request, even if the status stage does not end with an ACK handshake. SET/CLEAR_FEATURE requests are effective after a setup stage, even if a status stage does not occur.

23.4.6.1.1 Autodecoded Control Write Transfer Handshaking

The USB device controller module automatically provides ACK handshaking for all transactions of all stages of autodecoded control write transfers, unless a corrupt packet is received; the USB module ignores corrupt packets. The SET_FIFO_EN bit CTRL[2] and the STALL_CMD bit SYSCON2[5] have no effect on handshaking.

23.4.6.1.2 Autodecoded Control Write Transfer Error Conditions

If the token packet or the data packet of a setup stage transaction has an error (bad CRC, PID check, or bit-stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

23.4.6.2 Autodecoded Control Read Transfers

Autodecoded control reads include the standard device requests GET_ENDPOINT and DEVICE_STATUS. These control read transfers access information kept in registers inside the USB module, so the MPU code is not involved in filling the read data into the TX FIFO.

The USB module returns the currently selected appropriate status information (depending on the wIndex value in the setup stage data packet) during the data phase of the single IN transaction of the data stage, and provides ACK as the handshake for the status stage handshake phase. The MPU receives no interrupt.

23.4.6.2.1 Autodecoded Control Read Transfer Handshaking

The USB device controller module automatically provides ACK handshaking for all transactions of all stages of autodecoded control read transfers, unless a corrupted token packet is received, which the USB module ignores. The SET_FIFO_EN bit CTRL[2] and the STALL_CMD bit SYSCON2[5] have no effect on the handshaking. If the status packet has a DATA0 PID instead of a DATA1 PID, the status is STALLed and an interrupt is not asserted to the MPU. If the setup packet has a DATA1 PID instead of a DATA0 PID, setup transaction is ignored (error).

23.4.6.2.2 Autodecoded Control Read Transfer Error Conditions

If the token phase or the data phase of a setup stage transaction has an error (bad CRC, PID check, or bit-stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

Data errors during the data stage of autodecoded control write transfers are handled in the standard way; any data stage transaction from the host where a data error occurs is ignored.

If the USB host sends a GET_ENDPOINT/DEVICE_STATUS request with a bad parameter, the autodecode mechanism senses the bad parameter in the setup stage data phase and causes a STALL handshake to be signaled during the data phase of the data stage and during the status stage.

23.4.6.3 Non-Autodecoded Control Write Transfers

Non-autodecoded control write transfers include the SET_/CLEAR_ENDPOINT feature, SET_CONFIGURATION, SET_INTERFACE, SET_DESCRIPTOR, and class- or vendor-specific control write transfers. Non-autodecoded control write transfers consist of two or three stages (setup, data [optional], and status).

The setup stage of a valid non-autodecoded control write transfer consists of one SETUP transaction from the USB host to the USB device. At the end of the setup stage handshake, the USB module generates an MPU general USB interrupt with the SETUP bit IRQ_SRC[2] set.

In response to this general USB interrupt, the MPU must set the SETUP_SEL bit EP_NUM[6] to clear the setup interrupt flag. The MPU subsystem must then read 8 bytes from the setup FIFO through the DATA register, clear the EP_SEL bit EP_NUM[5], and check the SETUP bit IRQ_SRC[2].

If the SETUP bit IRQ_SRC[2] is set, the MPU subsystem must discard the previously read setup data and handle the new setup data packet as previously explained. If the SETUP bit IRQ_SRC[2] is cleared, the MPU code interprets this request information and performs any application-specific activity needed because of the setup stage request. If there is one or more data stages for the transfer, the MPU subsystem must set the SET_FIFO_EN bit CTRL[2] for endpoint 0 to allow the core to accept RX data from the coming OUT transaction.

The data stage for non-autodecoded control writes consists of zero or more OUT transactions. Transaction handshaking and interrupt generation applies the same as for nonisochronous, noncontrol OUT endpoints. The MPU can cause NAK, STALL, or ACK signaling for the data stage transactions. If ACK is signaled on a given general USB interrupt, the MPU must respond by reading the data from the endpoint 0 RX FIFO and saving it for processing.

After completion of the data stage, a status stage IN transaction occurs. The USB module provides handshaking to the USB host based on the endpoint 0 handshaking control FIFO_EN bit STAT_FLG[2]. The MPU can delay signaling completion of the control write transfer by either forcing NAK handshaking to the host during the status stage (by holding the FIFO_EN bit STAT_FLG[2] 0) or causing ACK handshaking by setting the SET_FIFO_EN bit CTRL[2] with an empty endpoint 0 FIFO. An endpoint 0 TX general USB interrupt is sent to the MPU subsystem at the completion of the status stage.

After a SET_CONFIGURATION request, the device moves in an addressed or configured state as soon as the MPU subsystem sets the DEV_CFG bit SYSCON2[3] or the CLR_CFG bit SYSCON2[2].

23.4.6.3.1 Specific MPU-Required Actions

If the device receives a valid set endpoint halt feature request, it must set the appropriate SET_HALT control bit CTRL[6].

If the device receives a valid CLEAR_ENDPOINT halt feature request, it must set the appropriate RESET_EP bit CTRL[0] to clear the halt condition, set FIFO flags, and reset data PID to DATA0 for the endpoint. If the specified endpoint number is 0, the MPU must set only the CLR_HALT bit CTRL[7] to clear the halt condition.

If the device receives a valid SET_CONFIGURATION request, it must reset all endpoints by setting the RESET_EP control bit CTRL[0], set the SELF_PWR bit SYSCON1[2] to the appropriate value, and then set halt conditions for endpoints not used by the default interface set for the configuration.

If the device is addressed when the SET_CONFIGURATION is received, the MPU must write 1 to the DEV_CFG bit SYSCON2[3] to allow the device to move into the configured state (CFG bit DEVSTAT[3] set). If the device is configured when the SET_CONFIGURATION is received, and the new configuration value is 0, the MPU must write 1 to the CLR_CFG bit SYSCON2[2] to allow the device to move back into the addressed state (CFG bit DEVSTAT[3] cleared). If the device receives a valid se

If the device receives a valid set interface request, it must reset all endpoints used by the interface set by setting the RESET_EP control bit CTRL[0], and then setting halt conditions for endpoints not used by this interface.

Other MPU-required actions are specific to the request and are not detailed in this document.

23.4.6.3.2 Non-Autodecoded Control Write Transfer Handshaking

Valid setup stage transactions are signaled ACK. Transactions with invalid setup stage token or data packets are ignored and do not receive handshake packets from the USB module. Interrupts are not generated.

Data stage handshaking for non-autodecoded control write transfers is dependent on the endpoint 0 FIFO_EN bit STAT_FLG[2], the EP_HALTED bit STAT_FLG[6], and the STALL_CMD bit SYSCON2[5]. The MPU subsystem can delay completion of any transaction of the data stage by signaling NAK (via SET_FIFO_EN bit CTRL[2] not set). The USB specification requires that once STALL is signaled in a control transfer, it must be signaled on that endpoint until the next setup token is received. Either the STALL_CMD bit SYSCON2[5] or the SET_HALT bit CTRL[6] (reflected in the EP_HALTED register bit STAT_FLG[6]) provides this functionality. The EP_HALTED bit STAT_FLG[6] does not reflect the forced STALL caused by the STALL_CMD bit SYSCON2[5]; it retains its previous value.

Status stage handshaking is controlled by the endpoint 0 FIFO_EN bit STAT_FLG[2] and the STALL_CMD bit SYSCON2[5]. Successful completion of a non-autodecoded control write transfer is indicated by the USB device controller module returning a zero-length data payload for the data phase of the status stage and an ACK handshake from the host for the handshake phase of the status stage. Although NAK handshaking can be used to indicate delays in completion of the requested control write, the USB host can choose to abort the control write after some number of NAKs.

23.4.6.3.3 Non-Autodecoded Control Write Transfer Error Conditions

If an error occurs while dealing with the control write, which the MPU subsystem cannot deal with itself, it must signal STALL to the USB host for all subsequent transactions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting endpoint 0 STALL_CMD bit SYSCON2[5], which causes stalling of all the remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as they are for BULK/INTERRUPT transactions. If a corrupted packet is received, the core ignores the transaction and no interrupt is asserted.

23.4.6.4 Non-Autodecoded Control Read Transfers

Non-autodecoded control read transfers include the GET_INTERFACE_STATUS, GET_CONFIGURATION, GET_INTERFACE, GET_DESCRIPTOR, SYNCH_FRAME, and class- or vendor-specific control read transfers. Non-autodecoded control read transfers consist of three stages (setup, data, and status).

The setup stage of a valid non-autodecoded control read transfer consists of one SETUP transaction from USB host to USB device. At the end of the setup stage handshake, the USB module generates a MPU general USB interrupt with the IRQ_SRC.SETUP flag set.

In response to this general USB interrupt, the MPU subsystem must set the SETUP_SEL bit EP_NUM[6] to clear the setup interrupt flag. The MPU must then read 8 bytes from the setup FIFO through the DATA register, clear the EP_SEL bit EP_NUM[5], and check the SETUP flag IRQ_SRC[2].

If the SETUP bit IRQ_SRC[2] is set, the MPU subsystem must discard the previously read setup data and handle the new setup data packet as previously described. If the SETUP bit IRQ_SRC[2] is cleared, the MPU code interprets this request information and then prepares data for the IN transactions that follow. This includes placing the data requested (or the first few bytes, if more than one FIFO worth of data is returned) into the endpoint 0 FIFO, and setting the SET_FIFO_EN bit CTRL[2].

The data stage of a control read transfer consists of one or more IN transactions. Transaction handshaking and interrupt generation applies the same as for nonisochronous, noncontrol IN endpoints; the MPU subsystem can cause NAK, STALL, or ACK signaling for the data stage transactions. At endpoint 0 TX general USB interrupts, the MPU code must move more data to the endpoint 0 FIFO, until the last bytes of the requested data are provided.

Although SETUP packets have a defined payload length, the USB host can cancel the transaction at any time, without the status stage, and resend another SETUP command. The MPU code must be able to operate correctly in this situation.

After completion of the data stage, a status stage OUT transaction occurs. The USB host sends a zero-length data packet, and the MPU code must return its completion status for the control read standard request via standard handshaking mechanisms.

Note: When returning exactly what the host requests and that request is a multiple of the maximum packet size, no zero-length packet is required. A zero-length packet is required only when the amount of data the device must return is less than the amount requested by the host and the amount returned is a multiple of the maximum packet size (source USB forum).

23.4.6.4.1 Non-Autodecoded Control Read Transfer Handshaking

Handshaking for the setup stage of non-autodecoded control read transfers is forced by the USB module to always be ACK, unless there is a data error in the packet, in which case the USB module ignores the transaction. If the setup packet has a DATA1 PID instead of a DATA0 PID, the setup transaction is ignored (error).

Data stage handshaking for non-autodecoded control read transfers is dependent on the endpoint 0 FIFO_EN bit STAT_FLG[2], the EP_HALTED bit STAT_FLG[6], and the STALL_CMD bit SYSCON2[5].

The handshaking information is used during the data phase of the data stage transaction. The USB specification requires that once STALL is signaled in a control transfer, it must be signaled until the next setup token is received. The STALL_CMD bit SYSCON2[5] and the SET_HALT bit CTRL[6] (reflected through the EP_HALTED register bit STAT_FLG[6]) provide this functionality. The EP_HALTED bit STAT_FLG[6] does not reflect the forced STALL caused by the STALL_CMD bit SYSCON2[5]; it retains its previous value.

The status stage is controlled by the FIFO_EN bit STAT_FLG[2] and the STALL_CMD bit SYSCON2[5].

Successful completion of a non-autodecoded control read transfer is indicated by the host sending an OUT token followed by an empty packet, and the USB device controller responding with an ACK. If the data packet sent by the USB host during the status stage of a control read request is not empty, the OUT transaction is accepted by the core, but OUT data is not put into the endpoint 0 RX FIFO. If the status packet has a DATA0 PID instead of a DATA1 PID, a STALL is returned by the core and an interrupt is asserted.

23.4.6.4.2 Non-Autodecoded Control Read Transfer Error Conditions

If an error occurs while dealing with the control read, which the MPU subsystem cannot deal with itself, it must signal STALL to the USB host for all subsequent transactions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting endpoint 0 STALL_CMD bit SYSCON2[5], which causes the stalling of all remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as they are for BULK/INTERRUPT transactions. The USB device controller module responds to control read status stage transactions that have a bad token or bad data by not sending a handshake packet. In both cases, the transaction is ignored and no general USB interrupt is generated to the MPU subsystem.

23.4.6.5 Autodecoded Versus Non-Autodecoded Control Requests

Table 23-7 lists the autodecoded versus the non-autodecoded control requests.

Table 23-7. Autodecoded Versus Non-Autodecoded Control Requests[1][2][3][4]

Request	Recipient	Status	MPU Required Action	Device Behavior if Device Is Not Configured
GET_ STATUS	Device	Autodecoded (function of AUTODEC_ DIS register bit)	None	Device status is returned (SYSCON1.SELF_PWR and DEVSTAT.R_WK_OK bits).
	Interface	Nonautodecoded	The MPU must stall the command (through the SYSCON2. STALL_CMD bit) if the interface number is not correct. No feature is defined for the interface.	Command is passed to the MPU.
	Endpoint	Autodecoded	None	The core automatically stalls the command if endpoint number is different from 0.

(1) Transactions on endpoints other than 0 are ignored if the device is not configured (addressed state).
(2) If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the MPU must set halt feature for the endpoint. This does not happen if USB host works correctly.
(3) If endpoint 0 is halted, per the *Universal Serial Bus Specification Revision 1.1* (see Section 9.4.5: *Get_Status*), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE.
(4) Requests are handled with respect to the *Universal Serial Bus Specification Revision 1.1* when specified as such, but many device reactions are not specified by the specification.

Table 23-7. Autodecoded Versus Non-Autodecoded Control Requests (continued)

Request	Recipient	Status	MPU Required Action	Device Behavior if Device Is Not Configured
		(function of AUTODEC_ DIS register bit)		
CLEAR/SET FEATURE	Device	Autodecoded	None (DS_CHG IT is asserted to the MPU after any DEVSTAT.R_ WK_OK bit modification.)	The core handles the request.
		(function of AUTODEC_ DIS register bit)		
	Interface	Autodecoded	None (No feature is defined in the *Universal Serial Bus Specification Revision 1.1* for the interface. These requests are stalled.)	Command is stalled in any case.
		(function of AUTODEC_ DIS register bit)		
	Endpoint	Nonautodecoded	The MPU must stall the command (through the SYSCON2. STALL_CMD bit) if the endpoint number/type/direction is not correct. The MPU must reset the EP after handling the pending transactions (if CLEAR) or set halt condition (if SET). For EP 0, the MPU must only clear or set the halt condition; the FIFO and data PID are always correct for the next setup.	Command is passed to the MPU.
SET_ ADDRESS	Device	Autodecoded	None (see (5)) (Whether the device is addressed or not is available in the DEVSTAT register. A valid SET_ADDRESS request with address number from 0 generates a DS_CHG interrupt to the MPU.)	Default: Device moves in the addressed state if address number is different from 0. Addressed: Device takes the new address value or moves in default state if address number is 0. Configured: Request is STALLed.
GET_ DESCRIPTOR	All	Non-autodecoded	The MPU must write descriptor data into endpoint 0 FIFO.	Command is passed to the MPU.
SET_ DESCRIPTOR	All	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_ CMD bit) if it does not support set descriptor requests.	Command is passed to the MPU.
GET/SET CONFIGURATIO	Device	Non-autodecoded	The MPU must stall the command (through the SYSCON2. STALL_CMD bit) if the configuration number is not correct.	Command is passed to the MPU.

(5) During a SET_ADDRESS autodecoded command, only the 7 LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

Table 23-7. Autodecoded Versus Non-Autodecoded Control Requests (continued)

Request	Recipient	Status	MPU Required Action	Device Behavior if Device Is Not Configured
			If the request is SET_CONFIG, the MPU must reset all endpoints, halt endpoints not used by the default interface setting, set the SYSCON1.SELF_PWR value if the device is self-powered for the configuration set, and then set SYSCON2.DEV_ CFG bit (if config nb is not 0), or set the SYSCON2.CLR_CFG bit (if config nb is 0) before allowing the status stage to complete. The device moves to configured state (if DEV_CFG set), or moves to addressed state (if CLR_CFG set) and a DS_CHG interrupt is asserted to the MPU.	
GET/SET INTERFACE	Interface	Non-autodecoded	The MPU must stall the command (through the SYSCON2. STALL_CMD bit) if the interface/setting number is not correct. If the request is SET_INTERFACE, the MPU must reset endpoints used by the interface and halt endpoints not used by the interface setting before allowing status stage to complete.	Command is passed to the MPU.
SYNCH_ FRAME	Endpoint	Non-autodecoded	The MPU must stall the command if it does not support the SYNCH_FRAME request or write requested data in the endpoint 0 FIFO.	Command is passed to the MPU.

23.4.6.6 Note on Control Transfers Data Stage Length

The control transfer data stage length is indicated in the setup data packet.

During control reads, if the USB host requests more data than indicated in the setup packet, an unexpected IN transaction is STALLed, causing STALL handshake for all remaining transactions of the transfer until next SETUP. If the USB host requires less data than indicated in the setup packet, the transfer is not STALLed. However, if the host moves to status stage earlier than expected for a non-autodecoded request, the OUT status stage is NAKed because the MPU subsystem has not enabled the RX FIFO.

During control writes, if the USB host sends more bytes than indicated in the setup packet, the transfer is STALLed. If the USB host sends fewer bytes than were expected, the request is accepted. But if the USB host moves to status stage earlier than expected for a non-autodecoded request, the IN status stage is NAKed because the MPU subsystem has not enabled the TX FIFO.

23.4.7 USB Device Initialization

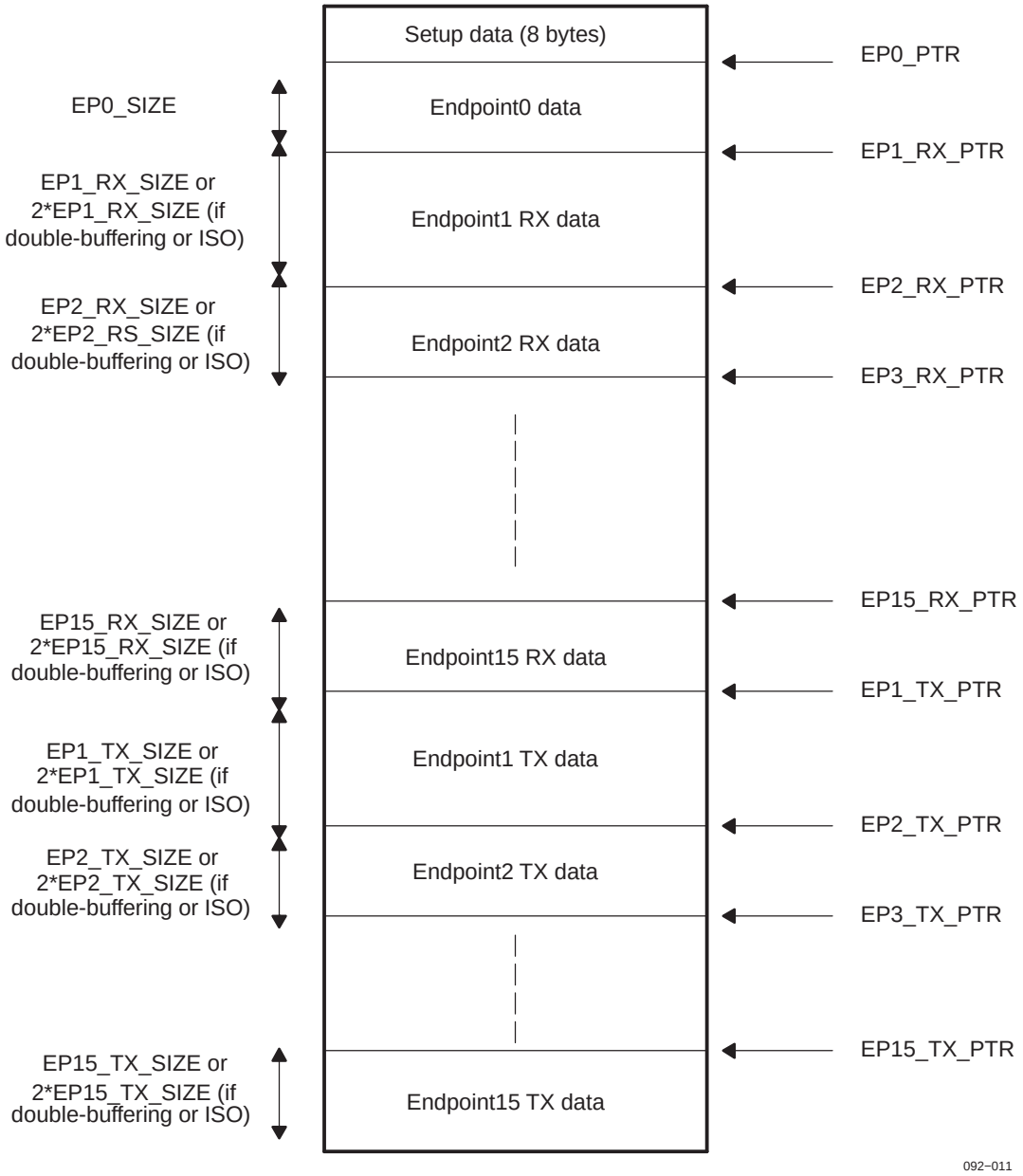
To allow communication between the device and a USB host, the MPU subsystem configures the device by filling the configuration registers.

For each endpoint, the MPU subsystem must write on the dedicated register:

- Endpoint size
- Whether double-buffering is allowed for endpoint or not
- Endpoint type (isochronous or nonisochronous)
- Address of the pointer

The system software must choose how to allocate the 2040 available bytes of the USB device controller RAM to the USB endpoints. The receive endpoint size and type are configured using the EP1_RX through the EP15_RX registers. The transmit endpoint size and type are configured using the EP1_TX through EP15_TX registers. Figure 23-11 shows an example of the RAM organization, obtained using the flowchart shown in Figure 23-12.

Figure 23-11. Example of RAM Organization



Once the endpoints are configured, the MPU subsystem must set the CFG_LOCK bit SYSCON1[8]. If this bit is not set, all transactions are ignored by the core. When the MPU subsystem is ready to communicate with the USB host, it must set the PULLUP_EN bit SYSCON1[0].

The MPU subsystem can wait until the DS_CHG attach interrupt is detected and handled before setting the PULLUP_EN bit SYSCON1[0]. The USB host cannot detect the device until this bit is set.

Figure 23-12 and Figure 23-13 show flowcharts for the configuration phase.

Figure 23-12. Device Configuration Routine

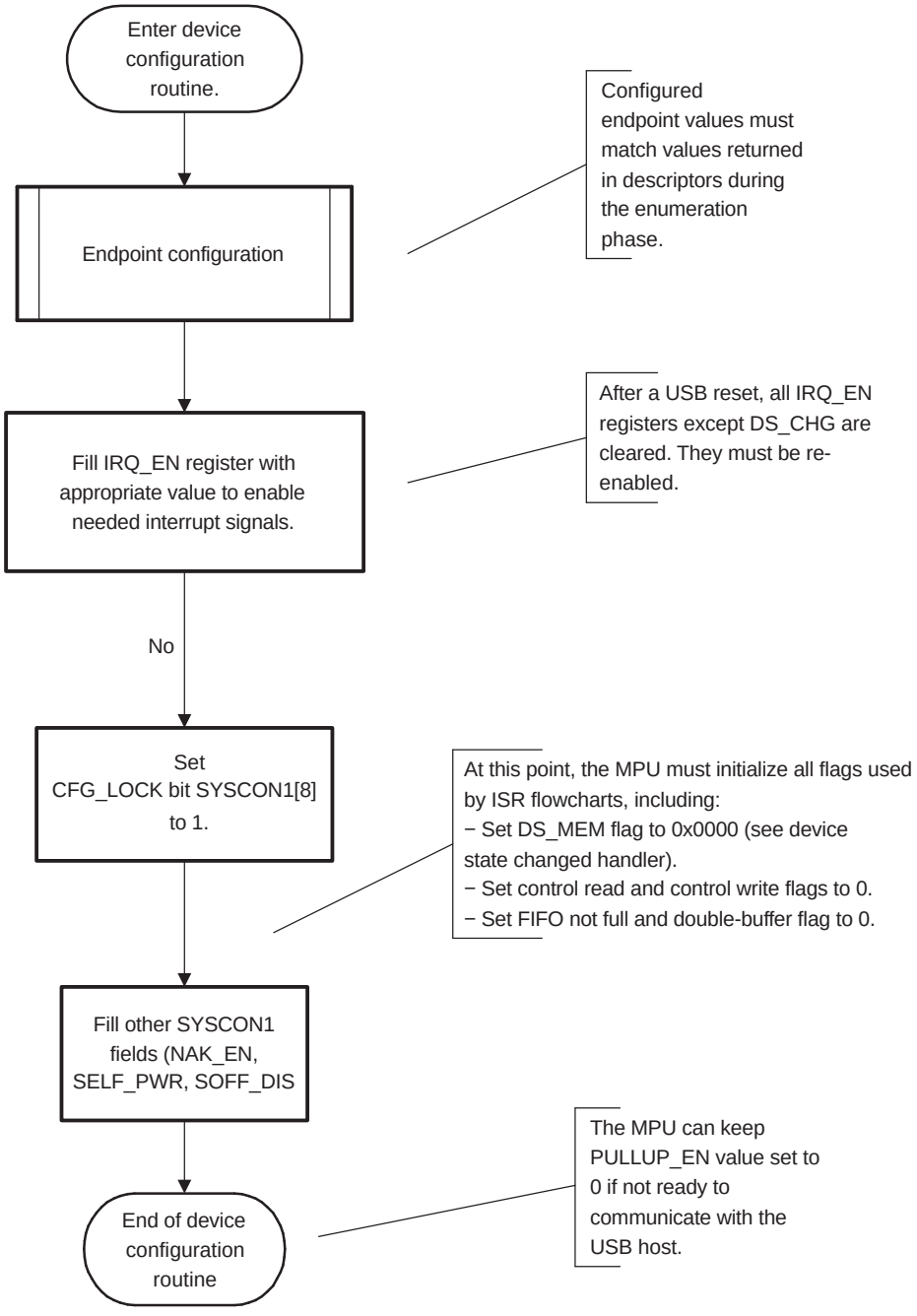
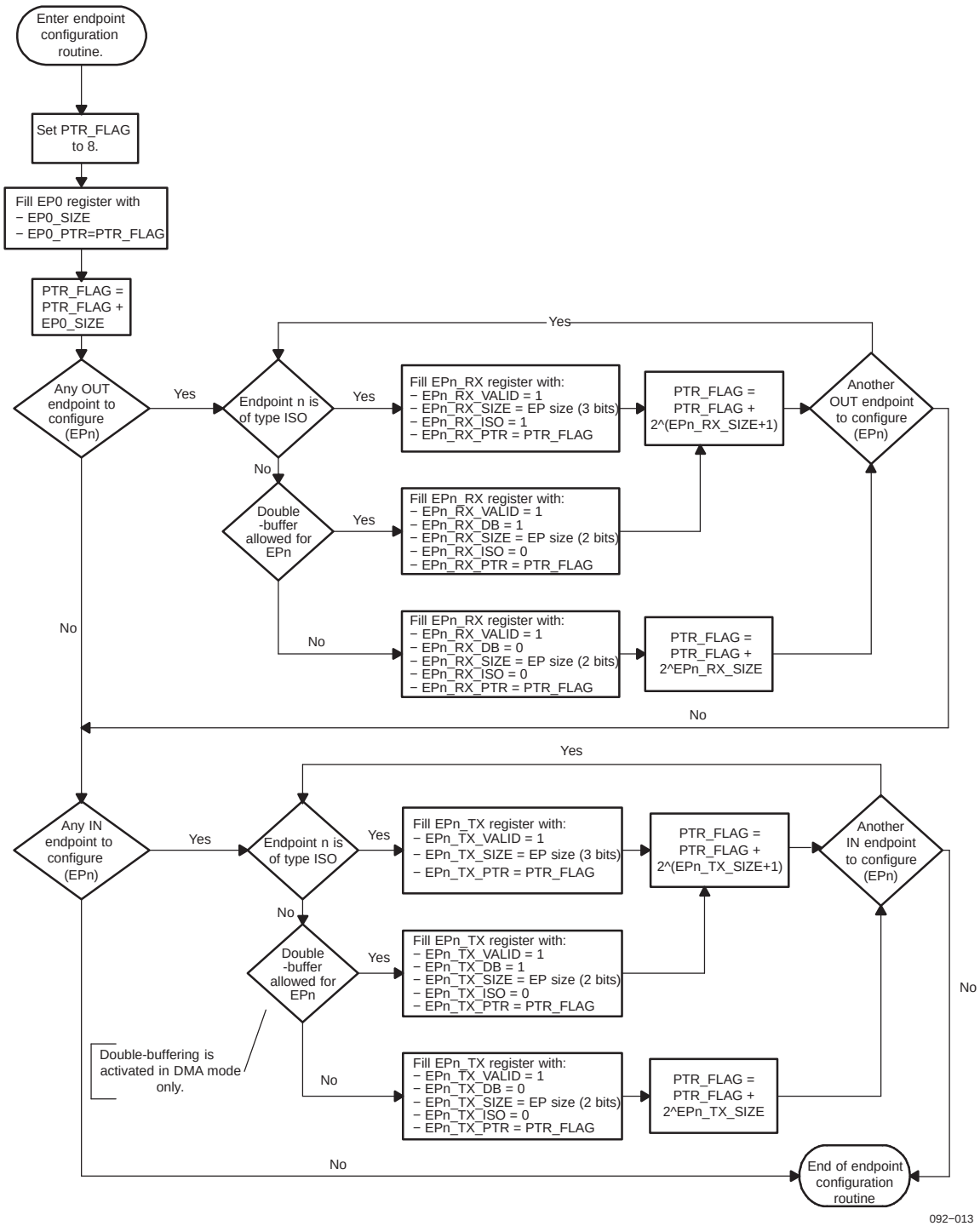


Figure 23-13. Endpoint Configuration Routine



NOTE: The local host must fill these fields during an initial endpoint configuration before setting the CFG_LOCK bit SYSCON1[8] and must not change the values once the CFG_LOCK bit SYSCON1[8] bit is set. If an endpoint is no longer used and is cleared by software, you can proceed with a new configuration of the endpoint.

23.4.8 Preparing for Transfers

To avoid NAK handshakes for the first transaction on an endpoint, the MPU subsystem must prepare the endpoint FIFO to receive or transfer data. After the first transaction, the FIFO is enabled during the interrupt handling (ISR flowchart).

For receive endpoints, this phase consists of enabling the FIFO to receive data from the USB host. If double-buffering is allowed for the endpoint, setting the SET_FIFO_EN bit CTRL[2] enables both FIFOs. Therefore, it is not possible to allow a single transaction when double-buffering is used.

The MPU subsystem enters the prepare-for-USB-RX-transfers routine, presented once after the enumeration phase, and then properly reacts to EP interrupts. Whether double-buffering is allowed or not is transparent to the MPU subsystem, unless both FIFOs are cleared through the CLR_EP bit CTRL[1] or the RESET_EP bit CTRL[0]. In this case, and in one where the MPU subsystem finishes to handle an interrupt without setting the SET_FIFO_EN bit CTRL[2], the MPU subsystem must re-enter the prepare-for- USB-RX-transfers routine.

For transmit endpoints, the MPU subsystem enters the prepare-for-endpointn- TX-transfer routine, presented each time a new file must be transmitted from endpoint n to the USB host. The MPU subsystem must not enter this routine until data written into TX FIFO from the previous transfer have all been received successfully by the USB host (ACK interrupt received), unless TX FIFO is cleared through the CLR_EP bit CTRL[1] or the RESET_EP bit CTRL[0] (see Figure 23-14 and Figure 23-15).

Note: This does not apply to endpoint 0, which is not used before a setup interrupt occurs. At setup interrupt, the MPU subsystem reacts appropriately, and enables EP0 FIFO only if necessary.

To ensure proper use of the module, you cannot prepare data on different endpoints at the same time. You can enter a routine when the routine is not used by another endpoint (no parallelism); the EP_NUM register can be accessed through different routines.

Figure 23-14. Prepare-for-USB-RX-Transfers Routine

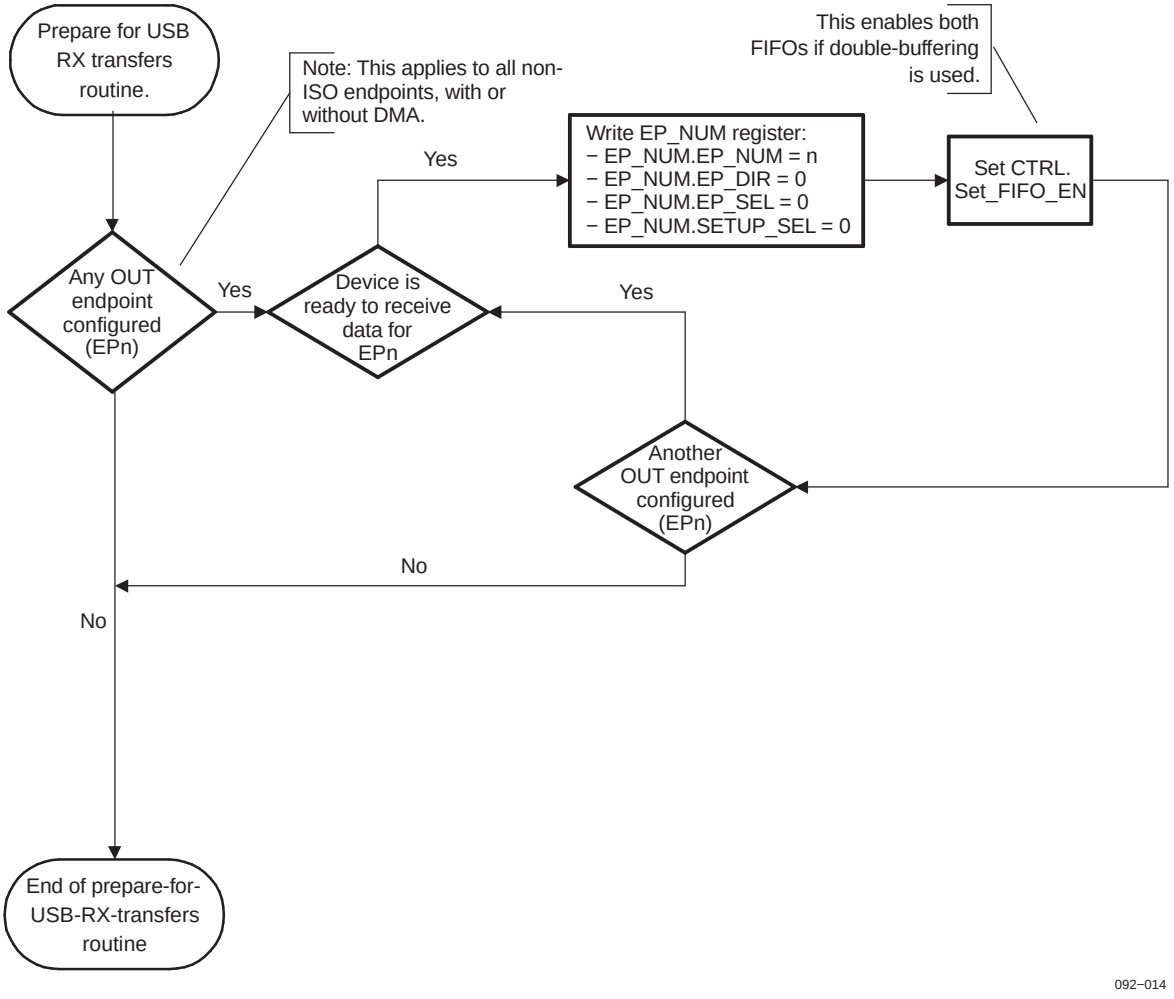
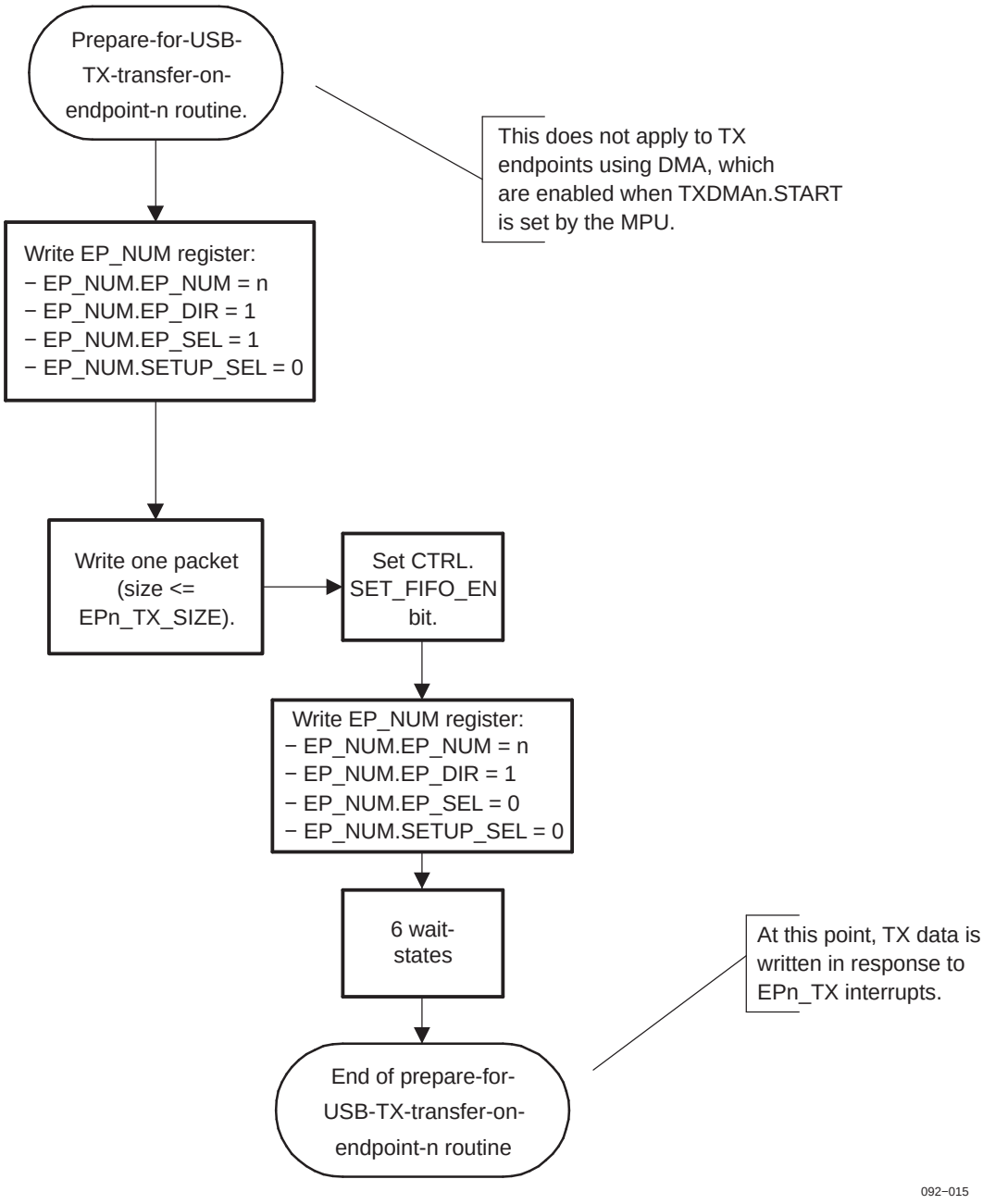


Figure 23-15. Prepare-for-USB-TX-Transfer-on-Endpoint-n Routine



23.4.9 USB Device ISR Flowcharts

The flowcharts in this section give general operational guidelines for USB device ISR processing. System-architecture-specific details are left to the engineers who write the MPU and USB host code. One USB-specific interrupt register is provided (IRQ_SRC), including:

- General USB interrupts (USB_IRQ_GEN), including endpoint 0, DMA, and device states interrupts
- Nonisochronous endpoint-specific interrupt (USB_IRQ_NISO)
- SOF interrupt for isochronous transactions (USB_IRQ_ISO)

The general USB interrupt ISR must handle non-autodecoded control transfers on endpoint 0 and some specialty interrupts generated because of USB device state modifications or DMA transfers. The ISR for

the endpoint-specific interrupt must handle interrupts from the USB module that are generated because of USB activity for nonisochronous endpoints. The SOF ISR handles isochronous endpoints and, if needed by the application, tracks the USB frame number. Many flowcharts show guidelines for how to handle the interrupts related to the USB device controller module. The flowcharts in this part assume that the NAK_EN bit SYSCON1[4] is cleared.

Note: A key assumption behind the flowcharts is that the application provides separate buffers for each direction of endpoint, except endpoint 0. The flowcharts read from these application buffers for IN transactions on TX endpoints and write to these application buffers for OUT transactions on RX endpoints.

The USB device controller does not support re-entrant interrupts. Each USB device controller must be handled completely before it can handle another USB device controller interrupt. This restriction occurs because there is only one EP_NUM register, so endpoint control operations must be completed before working with another endpoint or endpoint direction.

23.4.9.1 Important Note on USB Device Interrupts

When an endpoint interrupt is asserted, the MPU subsystem writes the EP_NUM register with the EP_SEL bit EP_NUM[5] set to 1. The MPU subsystem must finish the interrupt handling before clearing the EP_SEL bit EP_NUM[5], because clearing this bit clears the corresponding status bit in the STAT_FLG register (ACK, NAK, STALL). When an interrupt is pending on an endpoint, the MPU subsystem must not select and then deselect the endpoint without handling the interrupt, because this clears the pending transaction status flags. The MPU subsystem does not need to set the EP_SEL bit EP_NUM[5] to 1 when setting the SET_FIFO_EN bit CTRL[2], the SET_HALT bit CTRL[6], and the CLR_HALT bit CTRL[7].

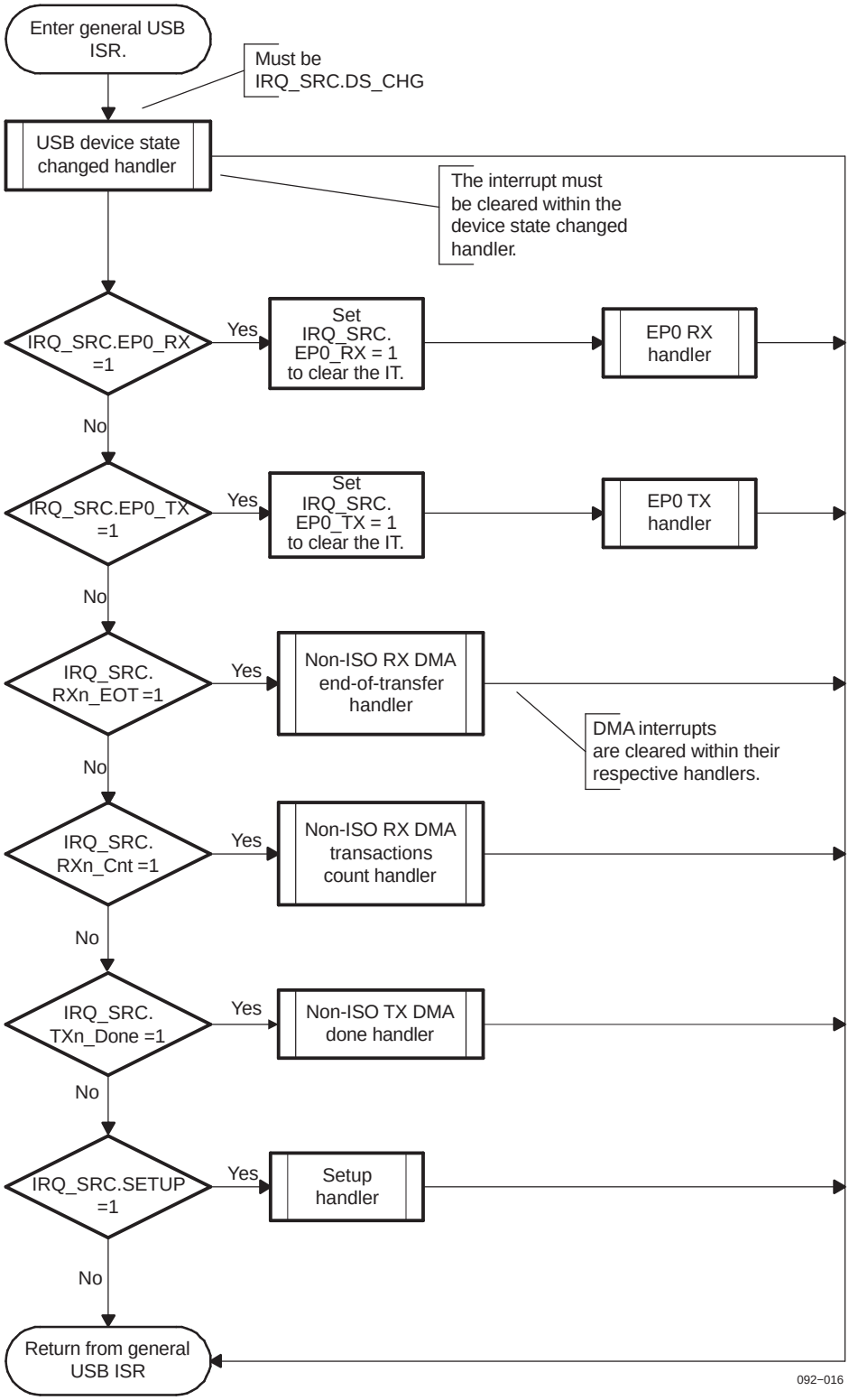
The endpoint status (STAT_FLG register) is updated at the end of each USB transaction if the previous transaction is handled. If a pending interrupt has not been handled when a new nontransparent transaction occurs, status flags are not updated (and NAK is returned, even if FIFO was enabled, or STALLed if the EP halt feature was set), so that the MPU subsystem never misses an ACKed transaction. If double-buffering is used for an endpoint, the STAT_FLG register is updated if there is zero or one interrupt pending for the endpoint; STAT_FLG is not updated if there are already two interrupts pending on the endpoint.

The MPU subsystem does not need to set the NAK_EN bit SYSCON1[4] during normal operation. However, for the debugging process, this bit can be set when the MPU subsystem finishes handling an EP interrupt without having set the corresponding SET_FIFO_EN bit CTRL[2]. During TX transaction, if the NAK_EN bit SYSCON1[4] is set, the MPU subsystem must wait for a NAK interrupt to write the TX data, to avoid a possible conflict caused by the NAK interrupt received while the MPU subsystem was writing the TX data.

23.4.9.2 Parsing General USB Device Interrupt

The USB_IRQ_GEN general USB interrupt ISR must parse the interrupt identifier register IRQ_SRC to determine the types of general USB interrupts that are active. These include interrupts relating to USB device state modifications (USB reset, suspend/resume, during enumeration phase) and control transfers on endpoint 0 or nonisochronous DMA transfers in either receive or transmit mode. Multiple interrupts can be active at any time, and all must be dealt with by the ISR before returning from the ISR.Figure 23-16 is a flowchart for parsing the general USB interrupts.

Figure 23-16. General USB Interrupt ISR Source Parsing Flowchart



23.4.9.3 Setup Interrupt Handler

A separate interrupt flag exists for setup transactions so that the MPU subsystem cannot miss a setup transaction, even if it occurs during the data or status phase of another transfer (such as an aborted transfer). The setup parsing function captures the control transfer request information used to determine which USB bus activity is needed and to control how the MPU subsystem must generate or respond to the control transfer. This information includes the following:

- bmRequestType
- bmRequest
- wValue
- wIndex
- wLength

The setup interrupt handler shown in Figure 23-17 processes setup transactions occurring on endpoint 0. It calls the routine that parses the control transfer request information, shown in Figure 23-18, to set flags that the rest of the ISR code can use to control proper response to control transfers. Two flags used during endpoint 0 interrupt handlers are set by the setup interrupt handler:

- Control read flag
- Control write flag

Figure 23-17. Setup Interrupt Handler

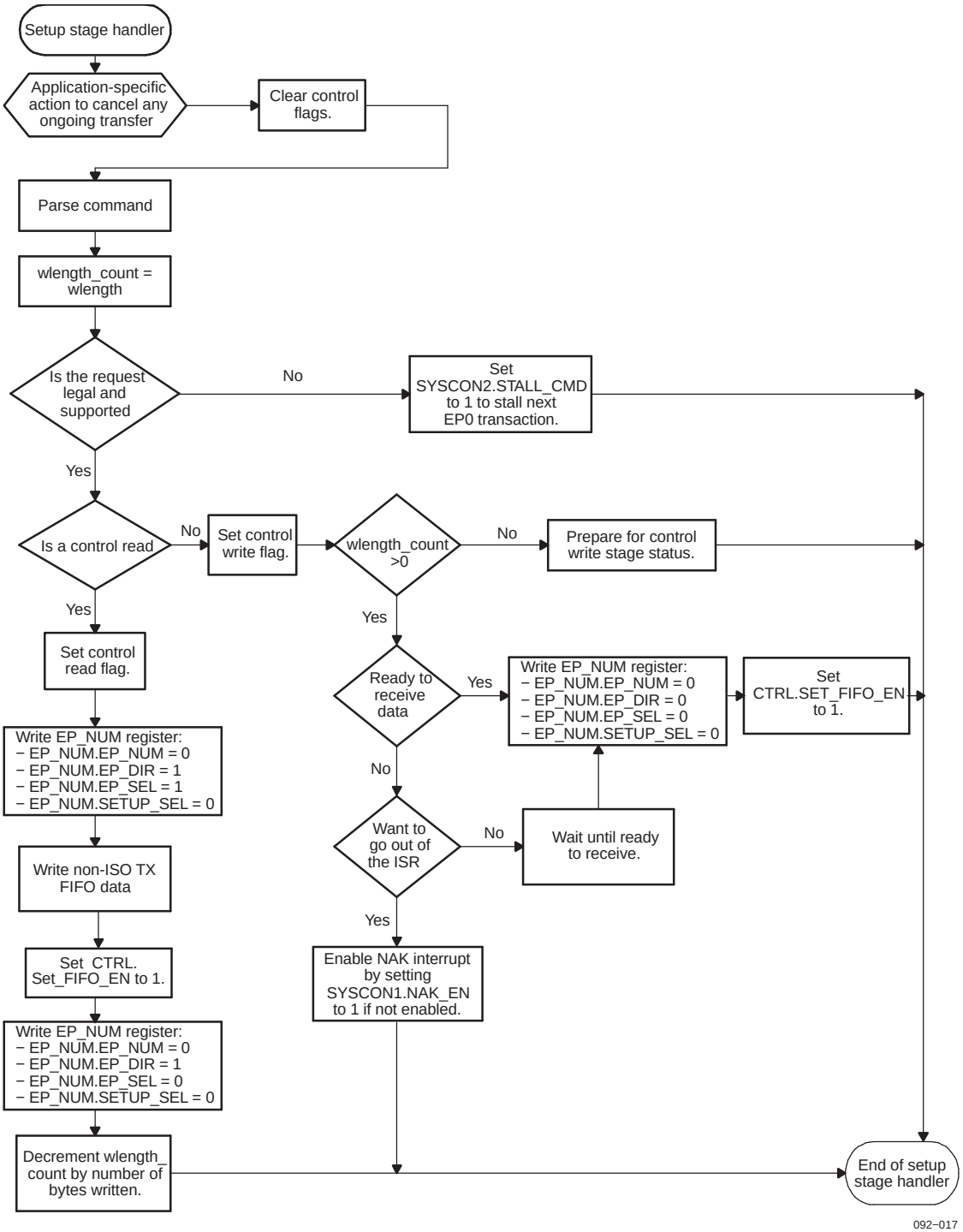
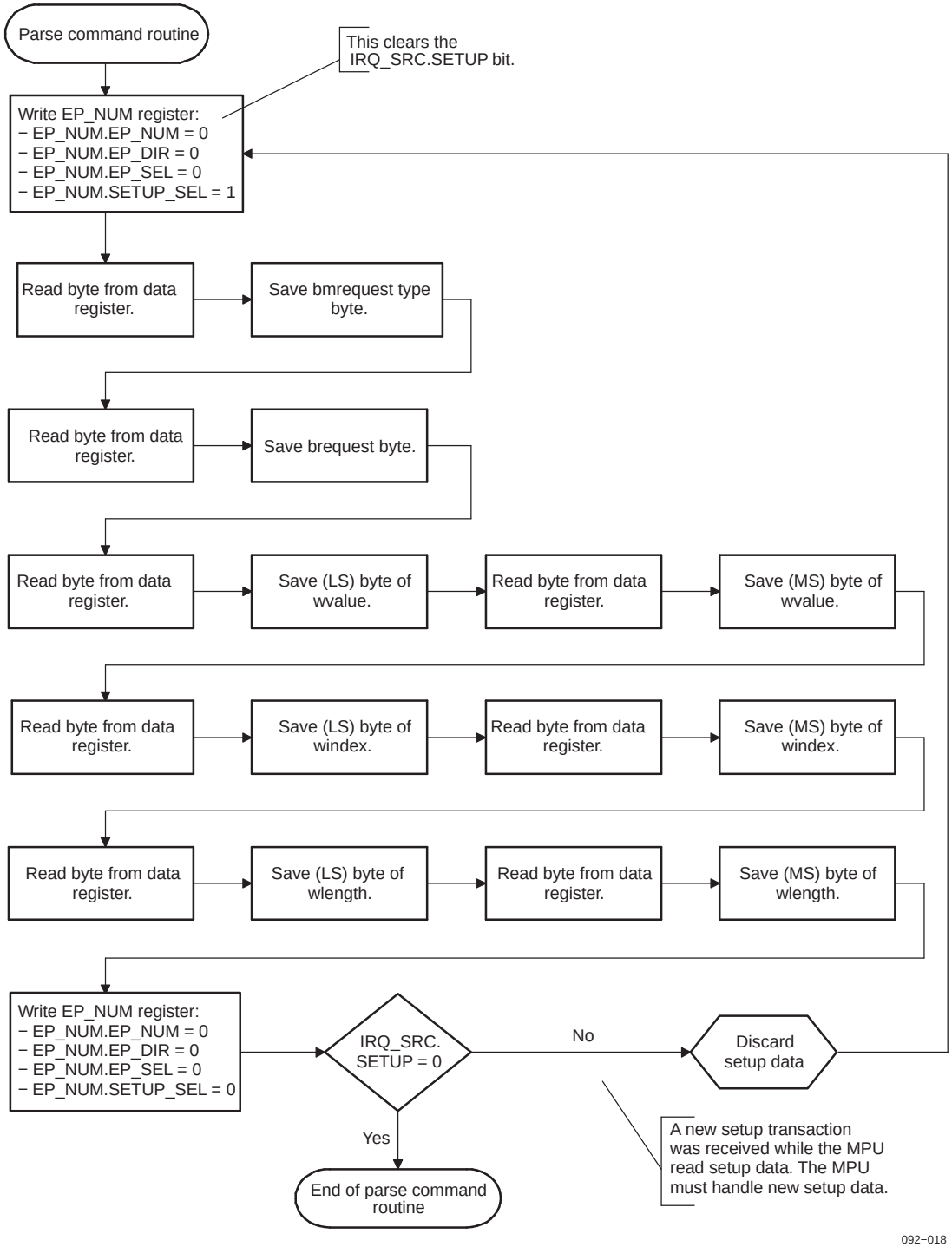


Figure 23-18. Parse Command Routine (Setup Stage Control Transfer Request)



23.4.9.4 Endpoint 0 RX Interrupt Handler

Figure 23-19 shows the endpoint 0 RX portion of the general USB interrupt handler, which must handle general USB interrupts related to control OUT transactions on endpoint 0. No EP0 interrupt is generated for autodecoded control transfers.

Figure 23-19. Endpoint 0 RX Interrupt Handler

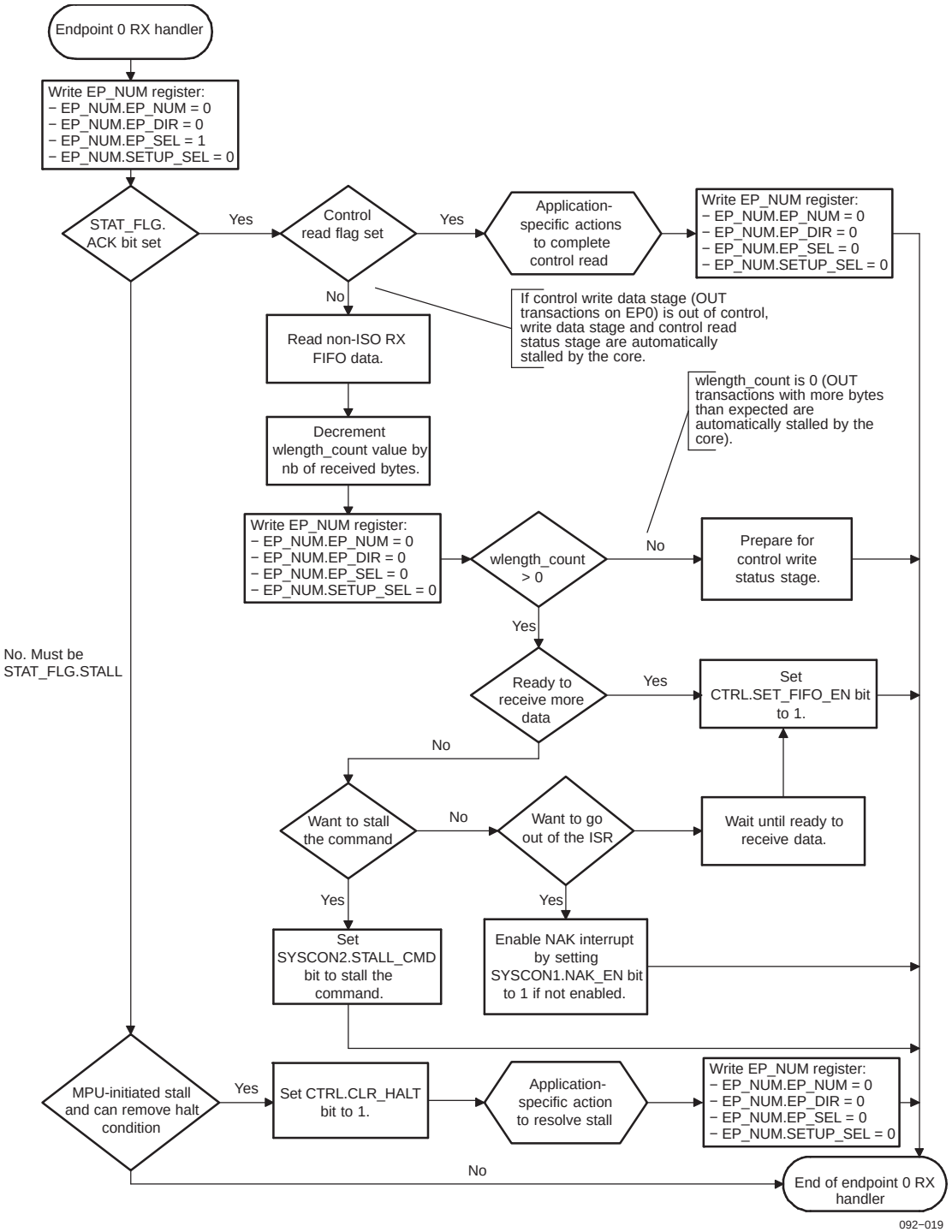
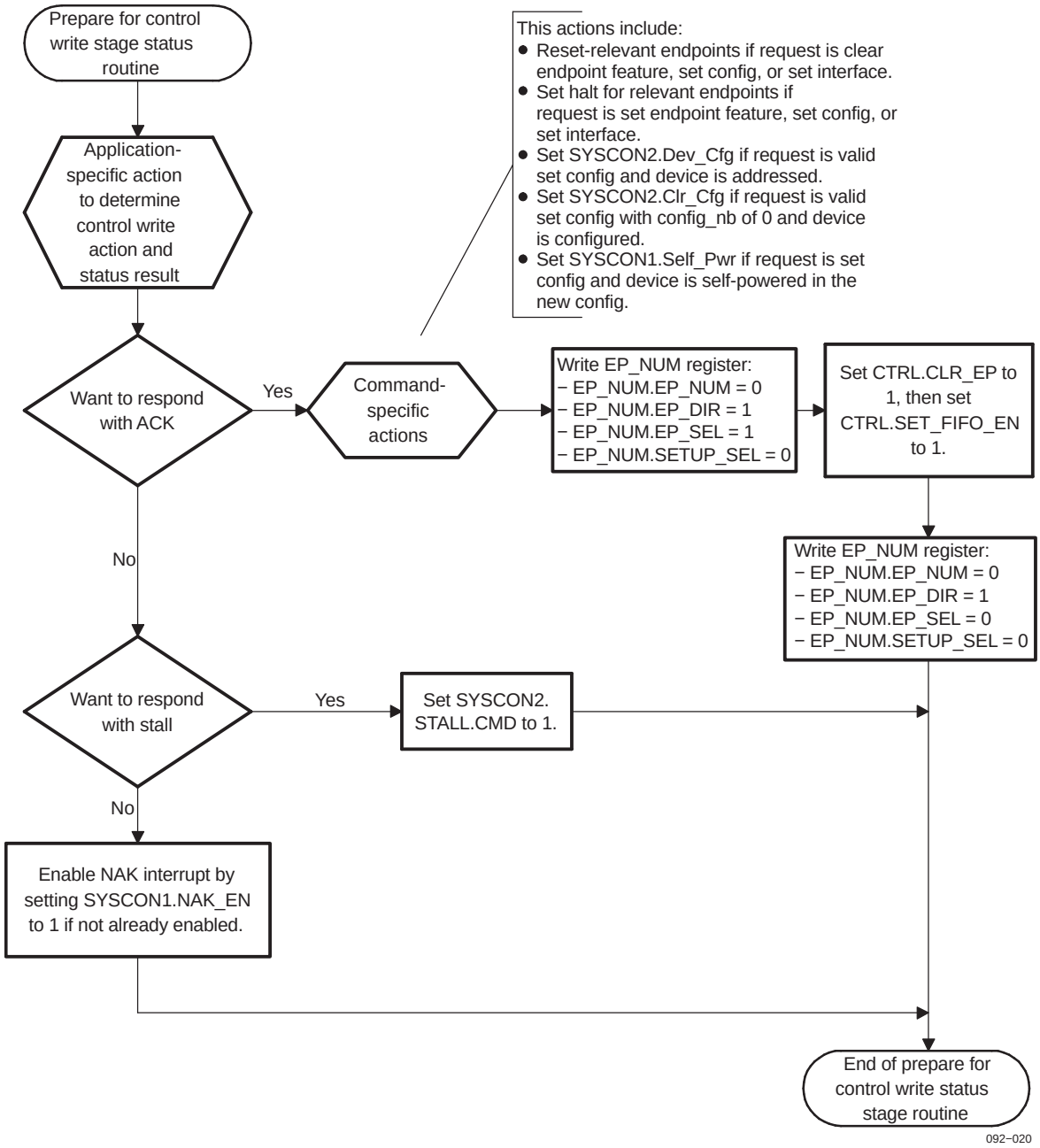


Figure 23-20. Prepare for Control Write Status Stage Routine



To support the SET_DESCRIPTOR command, when in communication-specific actions, you must first reset all endpoints by selecting the endpoint and then clear it, unlock the configuration, charge the new one, lock the configuration, go in prepare-for-transfers, and enable all interrupts (see Figure 23-20).

23.4.9.5 Endpoint 0 TX Interrupt Handler

Figure 23-21 shows the endpoint 0 TX portion of the general USB interrupt handler, which must handle general USB interrupts related to control IN transactions on endpoint 0.

The endpoint 0 TX interrupt handler must be able to move data into the endpoint 0 TX FIFO when the application buffer for endpoint 0 TX data is not empty and an endpoint 0 TX interrupt occurs, thus signaling an ACKed non-autodecoded endpoint 0 IN transaction. This data can be control read data stage information or control write status stage handshaking information (see Figure 23-22).

Figure 23-21. Endpoint 0 TX Interrupt Handler

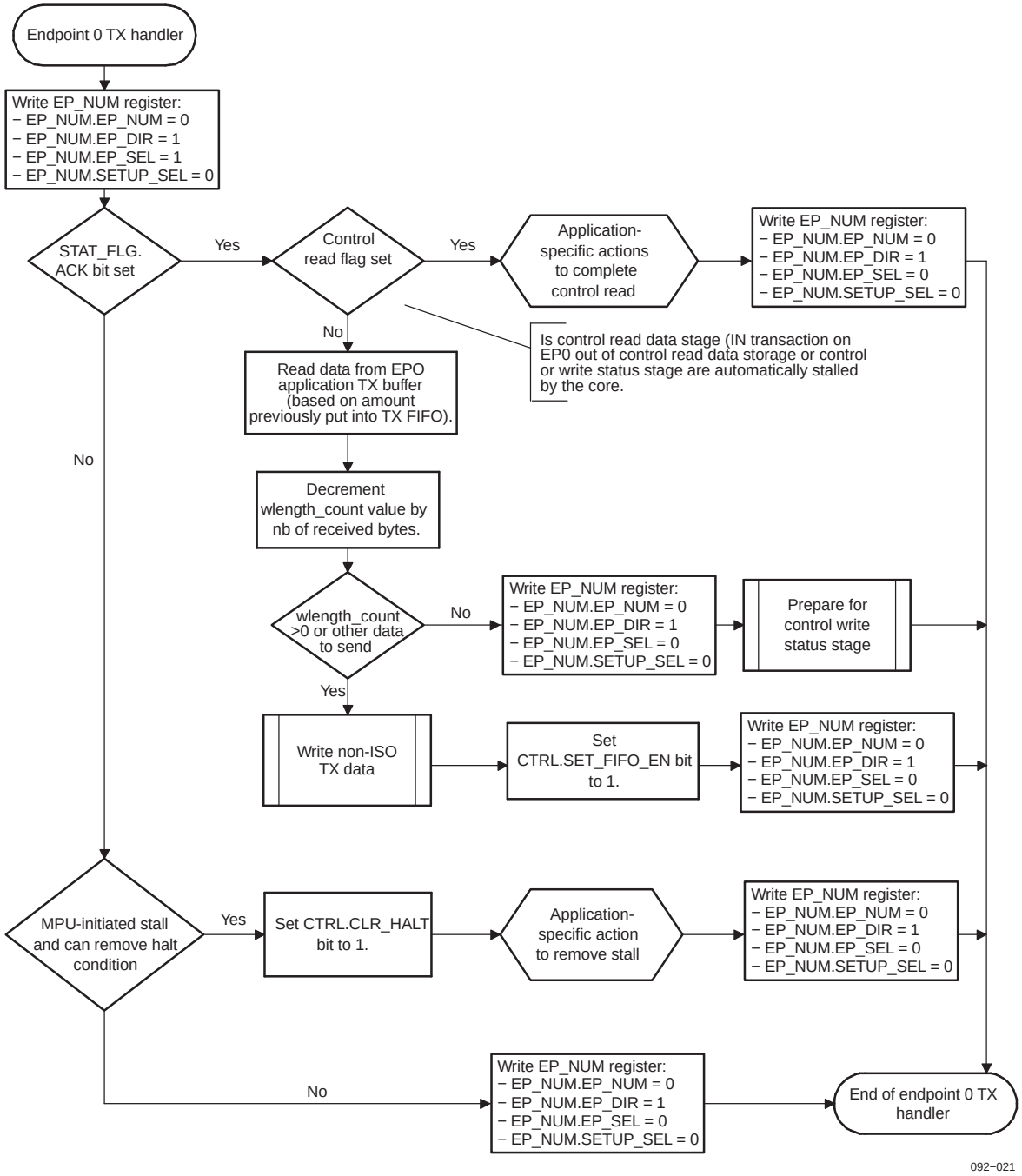
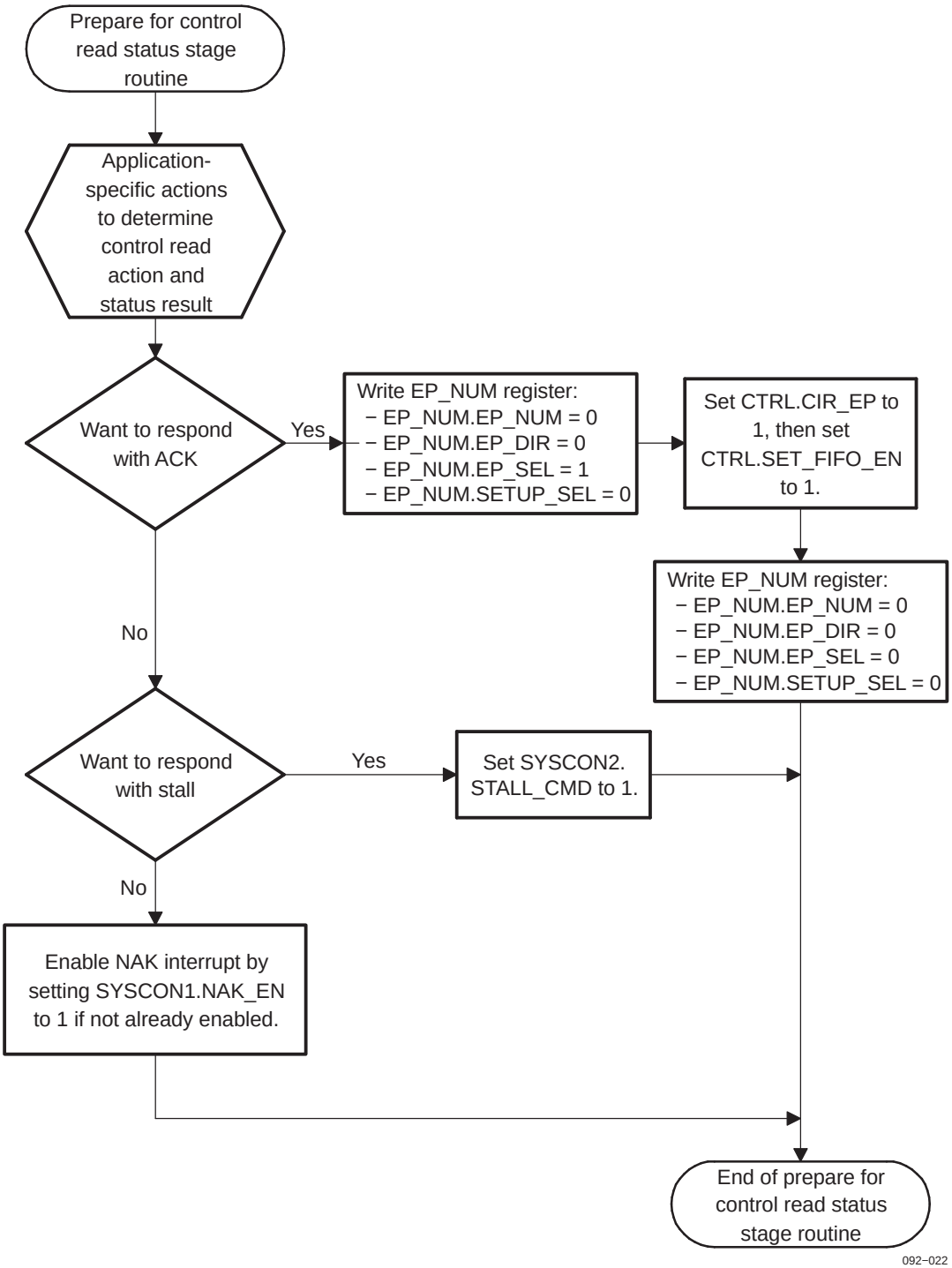


Figure 23-22. Prepare for Control Read Status Stage Routine



23.4.9.6 Device States Changed Handler

This section describes how USB device states and transition states are decoded by the USB device controller and how they can be handled.

The state-machine (see Figure 23-23) shows how the USB device controller device moves from one state to another state with respect to the *Universal Serial Bus Specification Revision 1.1*. The attach/unattach transition is not shown in the transition flow.

Because SET_CONFIGURATION is not decoded by the core, the MPU subsystem must distinguish a SET_CONFIGURATION with a nonvalid configuration value from other SET_CONFIGURATION requests; the MPU also must set the DEV_CFG bit SYSCON2[3] only if the configuration value is valid (value 0 is nonvalid), when the device is in addressed state. When the device is in configured state, the MPU subsystem must set the CLR_CFG bit SYSCON2[2] if the configuration number is 0 so that the device moves to the addressed state.

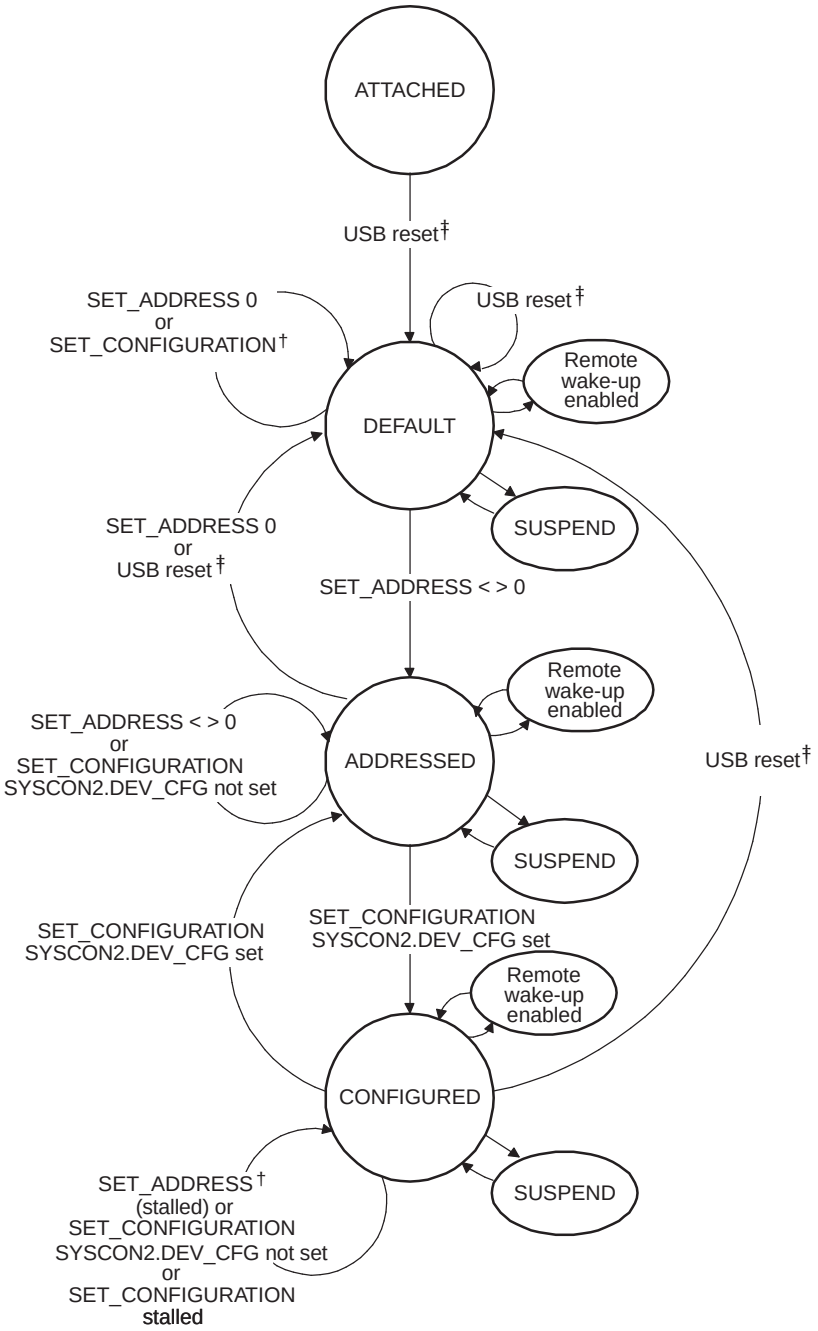
Device states are visible in the DEVSTAT register and are decoded as follows:

- Attached: The device is attached to the USB and is powered.
- Default: The device is attached to the USB and is powered and reset.
- Addressed: The device is attached to the USB and is powered and reset with an address assigned. The device moves into the addressed state after a SET_ADDRESS request with an address number other than 0.
- Configured: The device is attached to the USB, is powered and reset with an address other than 0, and is configured. The device moves into the configured state after a valid SET_CONFIGURATION request, only if the MPU subsystem has set the DEV_CFG bit SYSCON2[3] (meaning the configuration is valid).
- Suspended: The device is at minimum default and has not had bus activity for 5 ms.
- Reset: When set, the device is receiving a valid USB host reset.
- R_WK_OK: This bit is set (cleared) automatically after a valid SET_ DEVICE_FEATURE (CLEAR_DEVICE_FEATURE) request from the USB host.

Any change in the DEVSTAT register bits triggers a device change interrupt (DS_CHG bit IRQ_SRC[3]), if enabled.

The device moves to the addressed state after the status stage of a valid SET_ADDRESS, even if the status stage ACK handshake is corrupted when received or is not sent by the USB host. A SET_DEVICE_FEATURE or a CLEAR_DEVICE_FEATURE is effective after setup transaction, even if no status stage occurs. A SET_CONFIGURATION request is effective before status stage, when the MPU subsystem sets the CLR_CFG bit SYSCON2[2] or the DEV_CFG bit SYSCON2[3] (see Figure 23-24).

Figure 23-23. USB Device Controller Device State Transitions

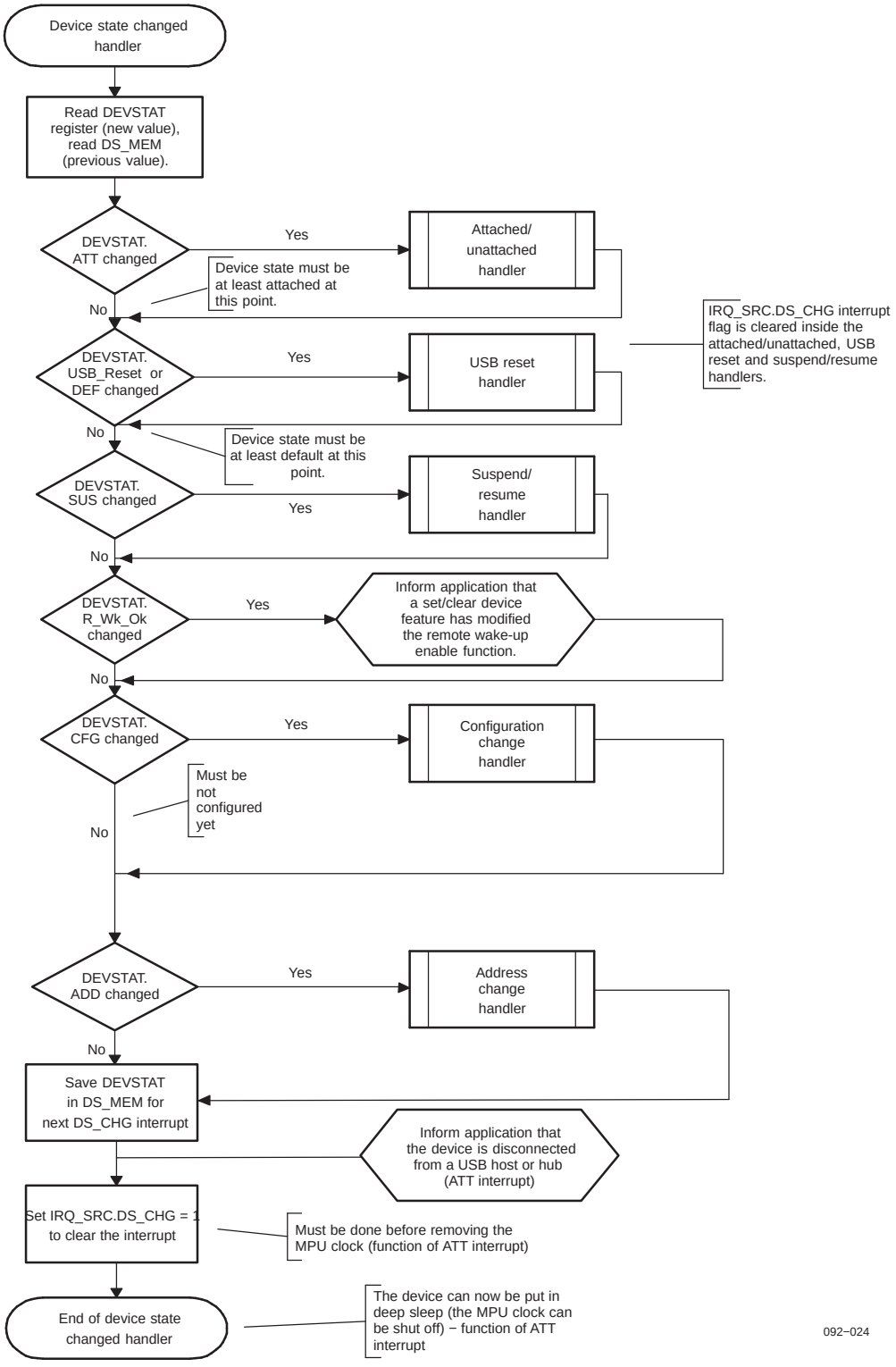


†Behavior is not specified by the *Universal Serial Bus Specification Release 1.1.*

‡USB reset generates two interrupts (when USB reset is asserted and then when USB reset completes). No interrupt is asserted by the core for transitions shown with dashed lines.

092−023

Figure 23-24. Typical Operation for USB Device State Changed Interrupt Handler

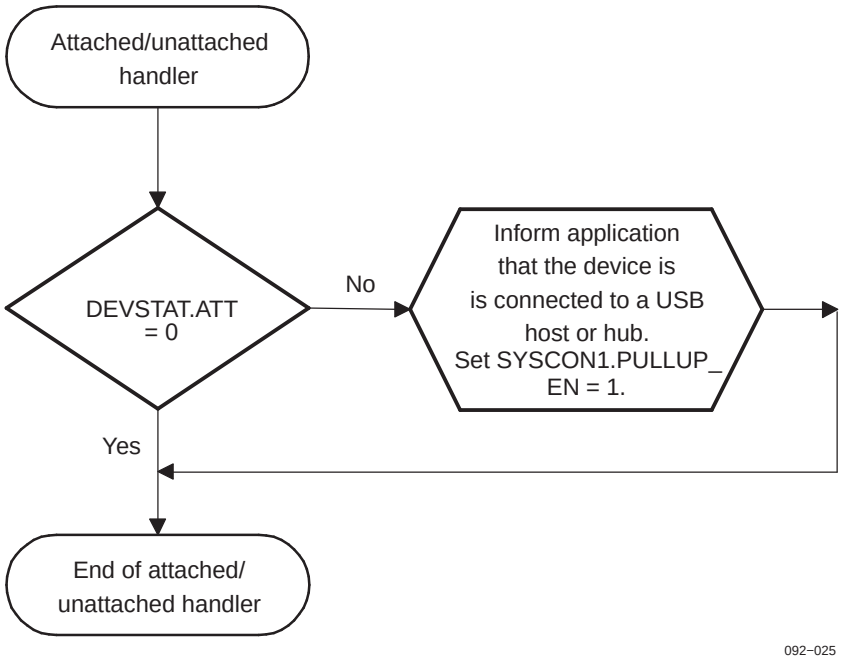


23.4.9.7 Device States Attached/Unattached Handler

The device attached/unattached interrupt occurs either when the device detects that it is connected to the USB host or hub (VBUS is on) or when it is disconnected (VBUS is off). The MPU can use this interrupt to put the device into deep-sleep or to initialize any application-specific information relating to the USB device controller.

Figure 23-25 shows attached/unattached handler operation.

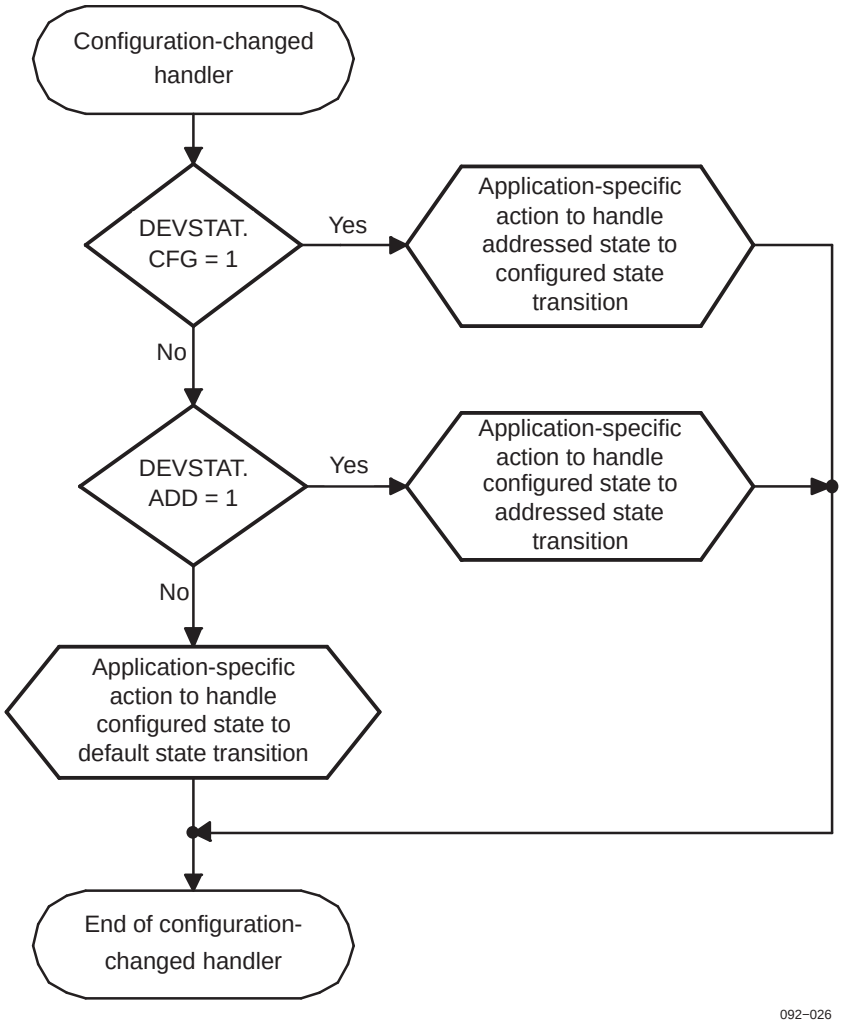
Figure 23-25. Attached/Unattached Handler



23.4.9.8 Device State Configuration-Changed Handler

When a configuration-changed interrupt occurs, the USB device has received a set configuration operation. When this occurs, the configuration-changed handler performs the operations shown in Figure 23-26.

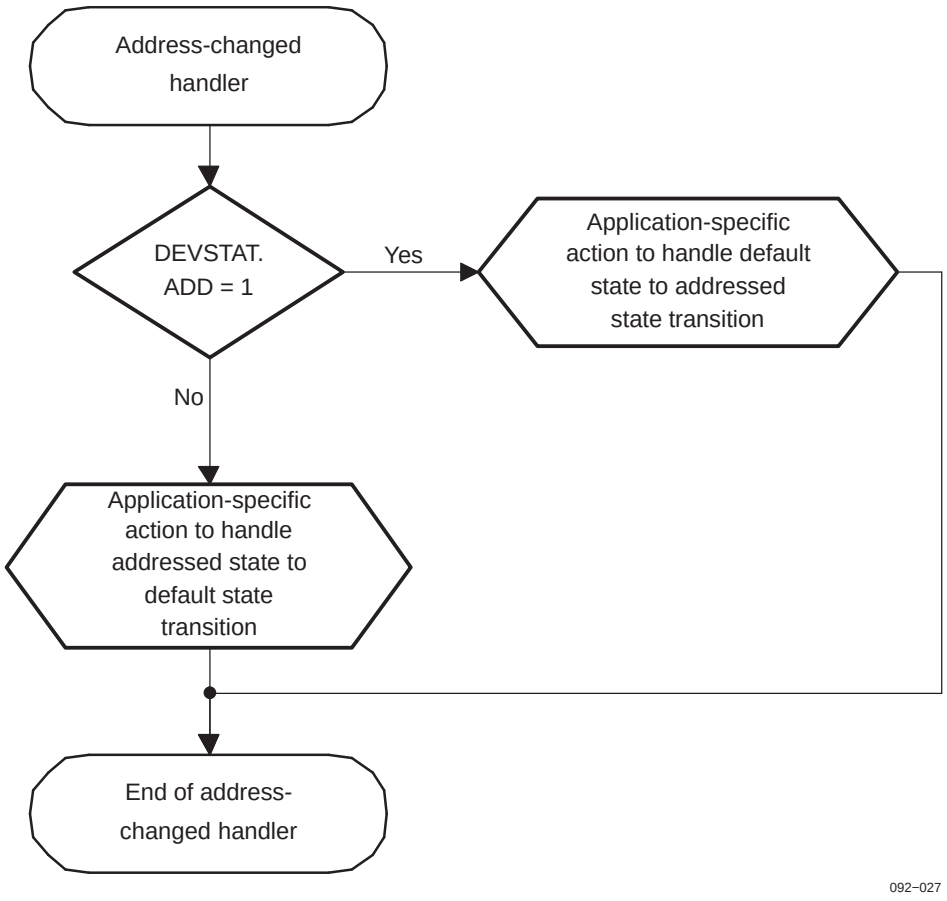
Figure 23-26. Configuration-Changed Handler



23.4.9.9 Device State Address-Changed Handler

When an address-changed interrupt occurs, the USB device has received a set address operation. When this occurs, the address-changed handler performs the operations shown in Figure 23-27.

Figure 23-27. Address-Changed Handler

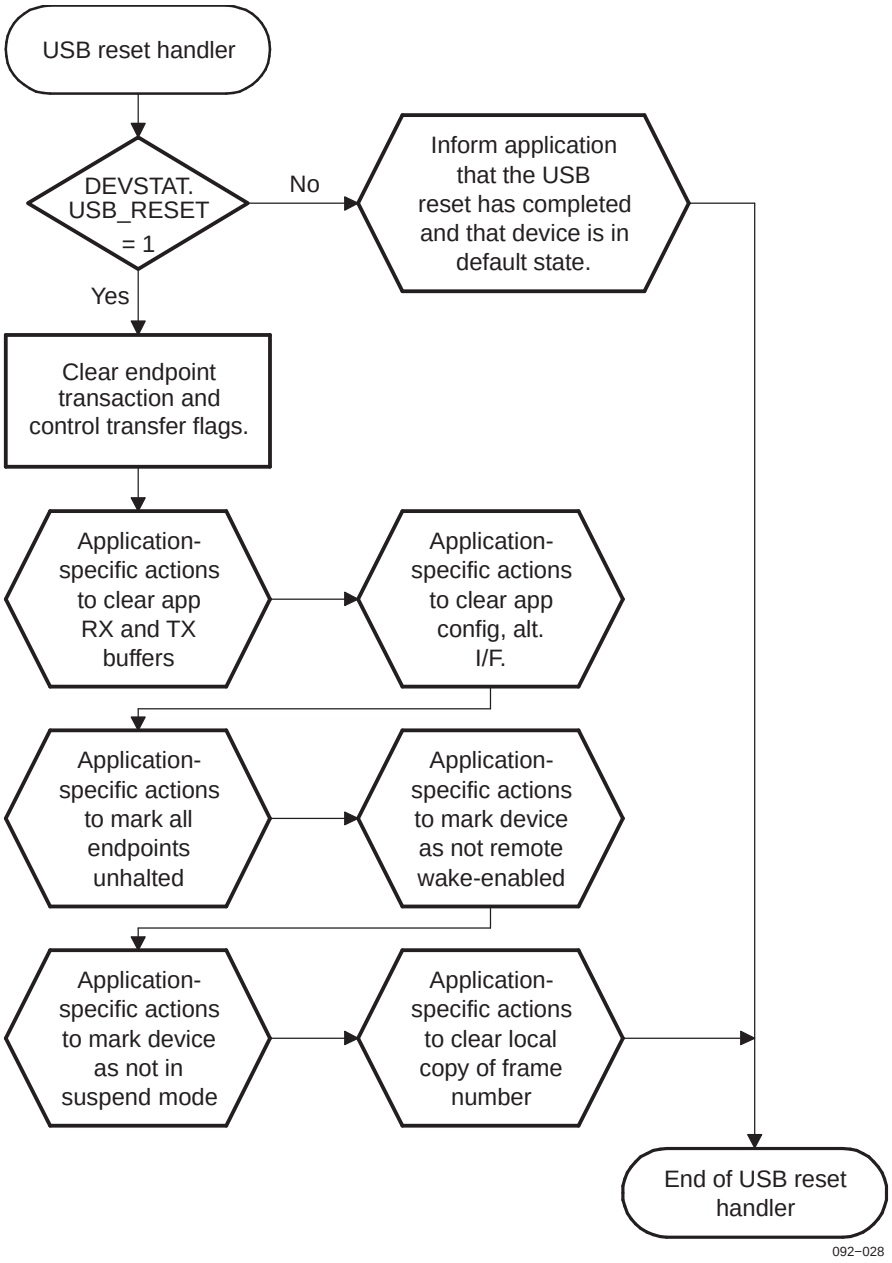


23.4.9.10 USB Device Reset Interrupt Handlert

When a USB reset occurs, the USB module generates a general USB interrupt to the MPU subsystem (see Figure 23-28). The MPU subsystem responds to this interrupt by performing the following operations:

- Cancels any ongoing USB transaction and/or control transfer handling
- Clears any copies that the application has of configuration number or alternate interface numbers
- Clears any application-specific information relating to halted endpoints
- Clears any application-specific information relating to the remote wakeenable flag
- Clears any application-specific information relating to the suspend mode flag_UnicodeEncodeError_
- Clears any application-specific copy of the frame number

Figure 23-28. USB Device Reset Handler Flowchart



23.4.9.11 Suspend/Resume Interrupt Handler

When a USB device suspend/resume general USB interrupt occurs, the USB module has either entered or left suspend mode. The MPU code must determine which and react appropriately (see Figure 23-29).

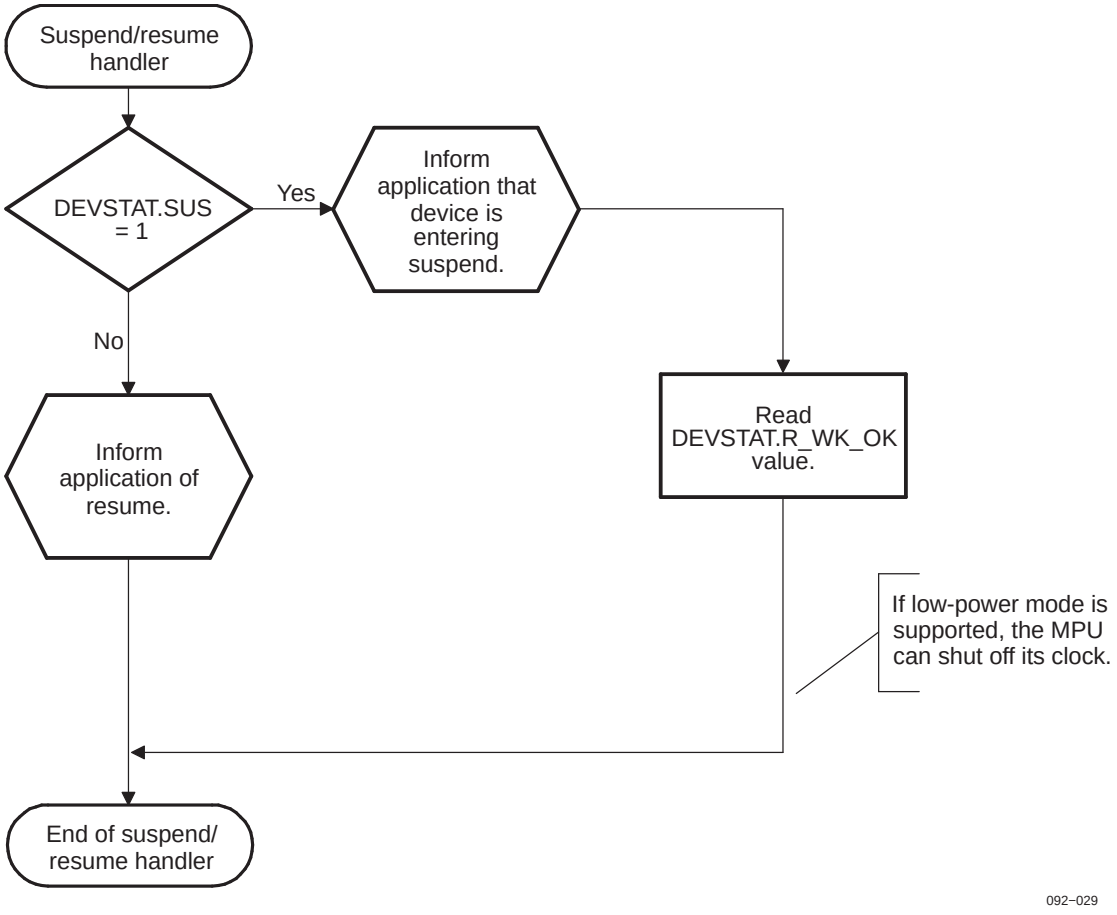
The suspend sense hardware is implemented to trigger only after 5 ms of bus idle. This forces compliance with the TWTRSM timing parameter (3 ms of IDLE to identify suspend, 2 ms before the remote device can signal resume) of the*Universal Serial Bus Specification Revision 1.1*.

If the MPU subsystem wants to wake the device from suspend mode and remote wake-up enable is set (DEVSTAT.R_WK_OK = 1), it must first turn its clock on (if stopped), and then set the RMT_WKP bit SYSCON2[6]. The device then drives resume.

If shutoff is enabled (SYSCON1.SOFF_DIS = 0), the 48-MHz clock is automatically shut off at suspend and turned on at resume (USB host or MPU subsystem driven). Setting or not setting the SOFF_DIS bit SYSCON1[1] is part of the device configuration. However, the MPU subsystem can modify its value at suspend interrupt time if necessary.

A USB reset is also a valid way to exit suspend mode. But the suspend/resume handler and the USB reset handler do not have to consider this because three interrupts are generated in that case (one for resume, one for reset, and one for end of reset).

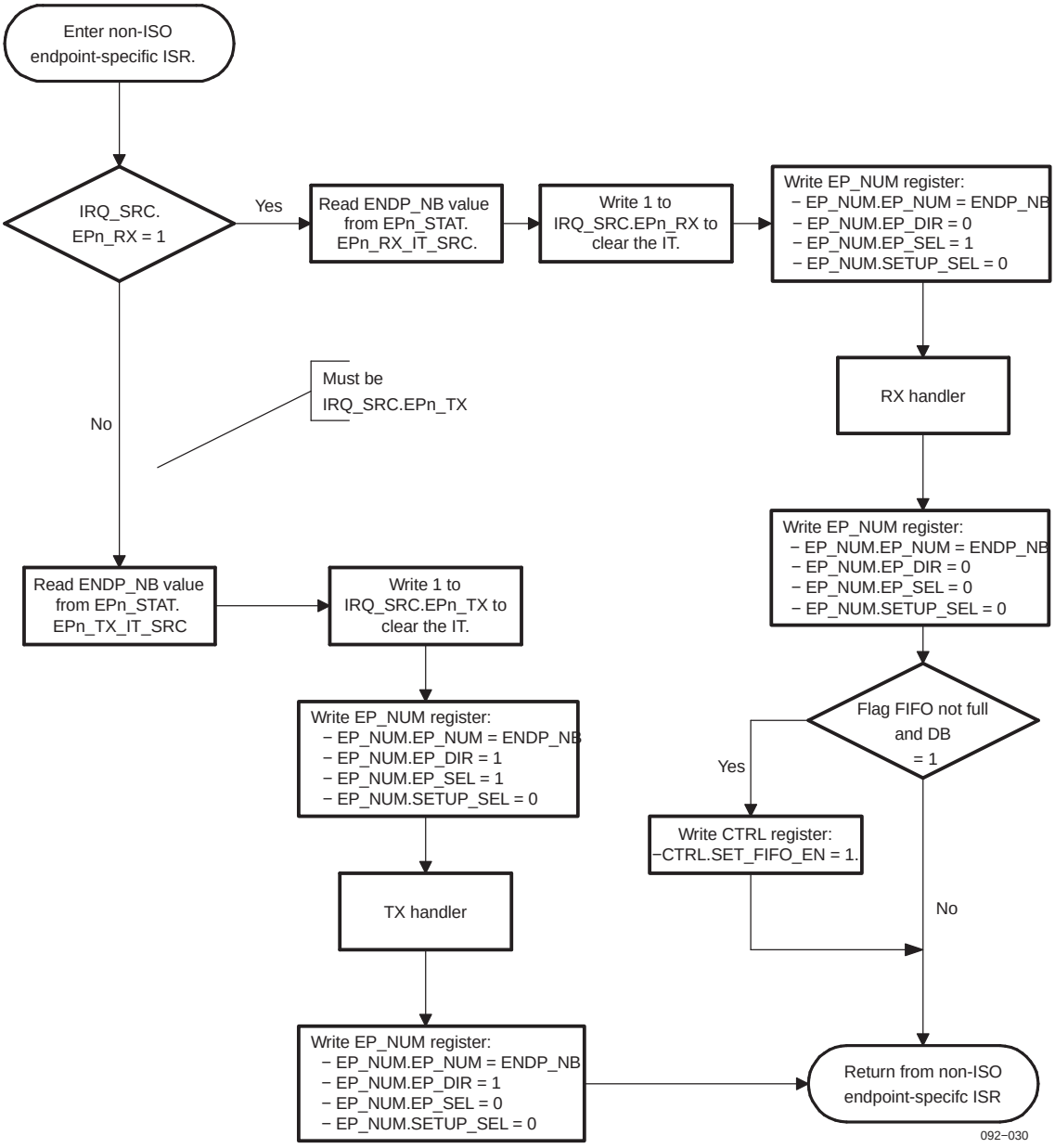
Figure 23-29. Typical Operation for USB Suspend/Resume General USB Interrupt Handler



23.4.9.12 Parsing Nonisochronous Endpoint-Specific Interrupt

The USB_IRQ_NISO endpoint-specific interrupt ISR (also known as the nonisochronous interrupt) must parse the interrupt identifier register IRQ_SRC to determine which interrupts are active (EPN_RX bit IRQ_SRC[5], EPN_TX bit IRQ_SRC[4], or both). The two interrupts can be active at any time. The ISR must then read the EPN_STAT register to determine the endpoint causing the interrupt. For each direction, only one endpoint interrupt can be active at a time (see Figure 23-30).

Figure 23-30. Nonisochronous Endpoint-Specific (Except EP0) ISR Flowchart



23.4.9.13 Nonisochronous, Noncontrol OUT Endpoint Receive Interrupt Handler

Figure 23-31 shows the operations required to handle nonisochronous, noncontrol OUT endpoint-specific receive interrupts. This flowchart shows two RX transaction handshaking interrupts. A third interrupt handshaking possibility, not shown in Figure 23-31, occurs when NAK interrupts are enabled.

Depending on the application-specific actions required for various endpoints in the real system, it is possible to use one routine that is common to all of the nonisochronous, noncontrol receive endpoints, where the only differences are the EP_NUM register value set and selecting the application RX data buffer in the read nonisochronous RX FIFO data routine (see Figure 23-32).

Figure 23-31 does not document control endpoint 0 receive interrupts, which are discussed separately because of the more complex 3-stage transfer mechanism used for control writes.

Figure 23-31. Nonisochronous, Noncontrol Endpoint Receive Interrupt Handler

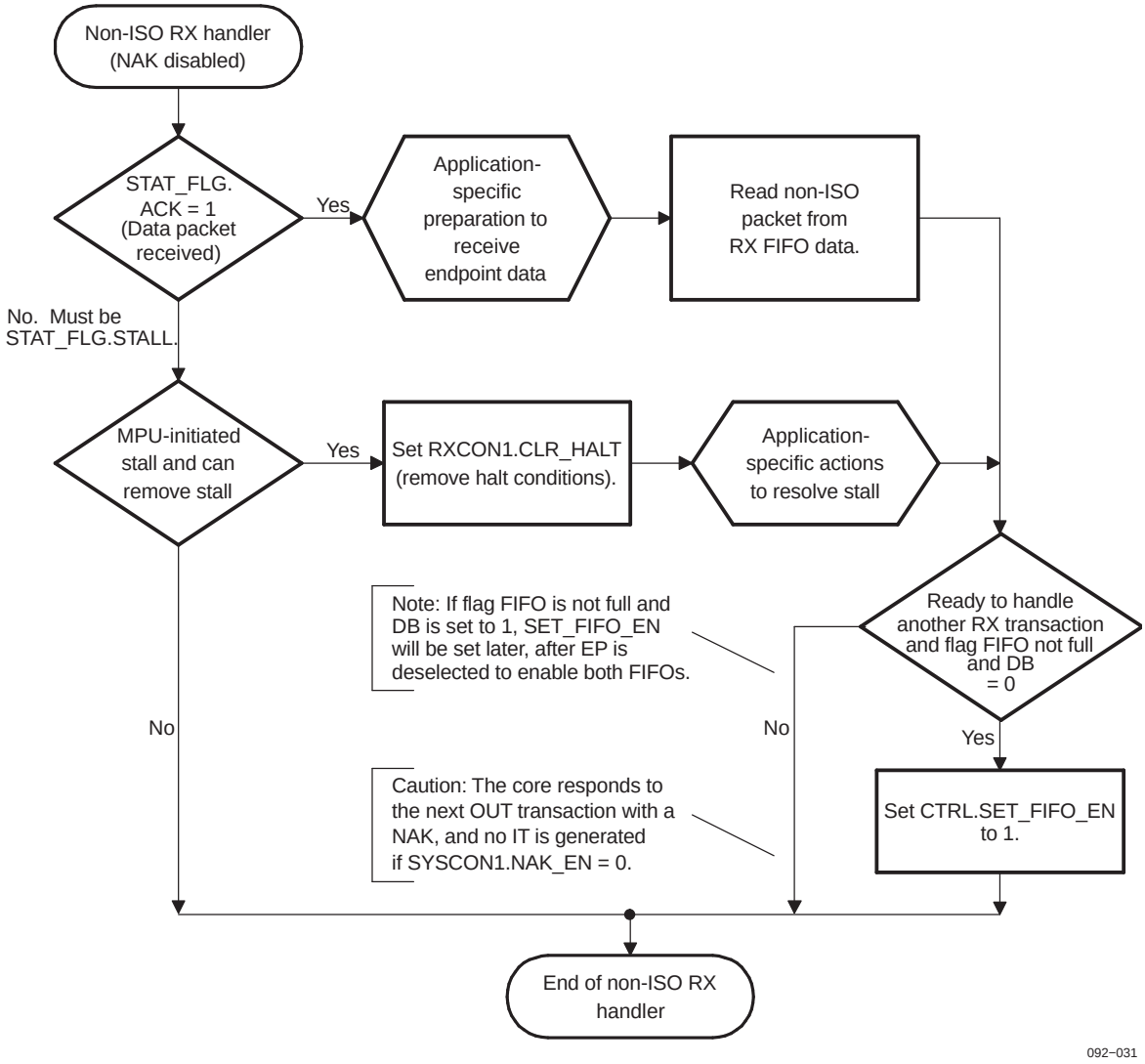
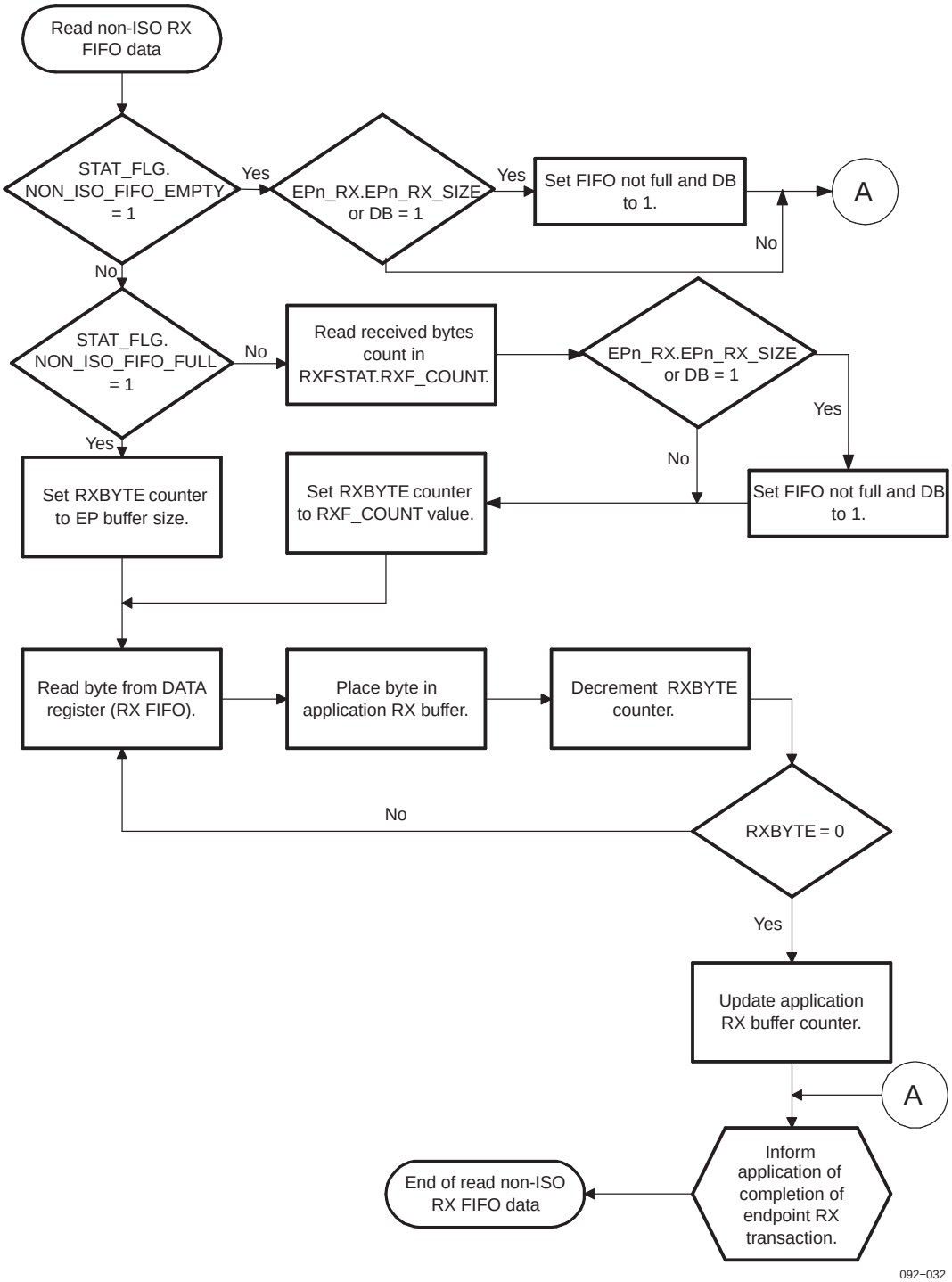


Figure 23-32. Read Nonisochronous RX FIFO Data Flowchart



23.4.9.14 Nonisochronous, Noncontrol IN Endpoint Transmit Interrupt Handler

Figure 23-33 shows the operations required to handle nonisochronous, noncontrol IN endpoint-specific transmit interrupts. This flowchart shows two TX transaction handshaking interrupts. A third interrupt handshaking possibility, not shown in Figure 23-33, occurs when NAK interrupts are enabled.

Depending on the application-specific actions required for various endpoints in the real system, it is possible to use one routine common to all nonisochronous, noncontrol transmit endpoints, where the only differences are the EP_NUM register value set and in the routine selection of the application TX buffer (see Figure 23-34).

Figure 23-33 does not document control endpoint 0 transmit interrupts, which are discussed separately because of the more complex 3-stage transfer mechanism used for control reads.

Figure 23-33. Nonisochronous, Noncontrol Endpoint Transmit Interrupt Handler

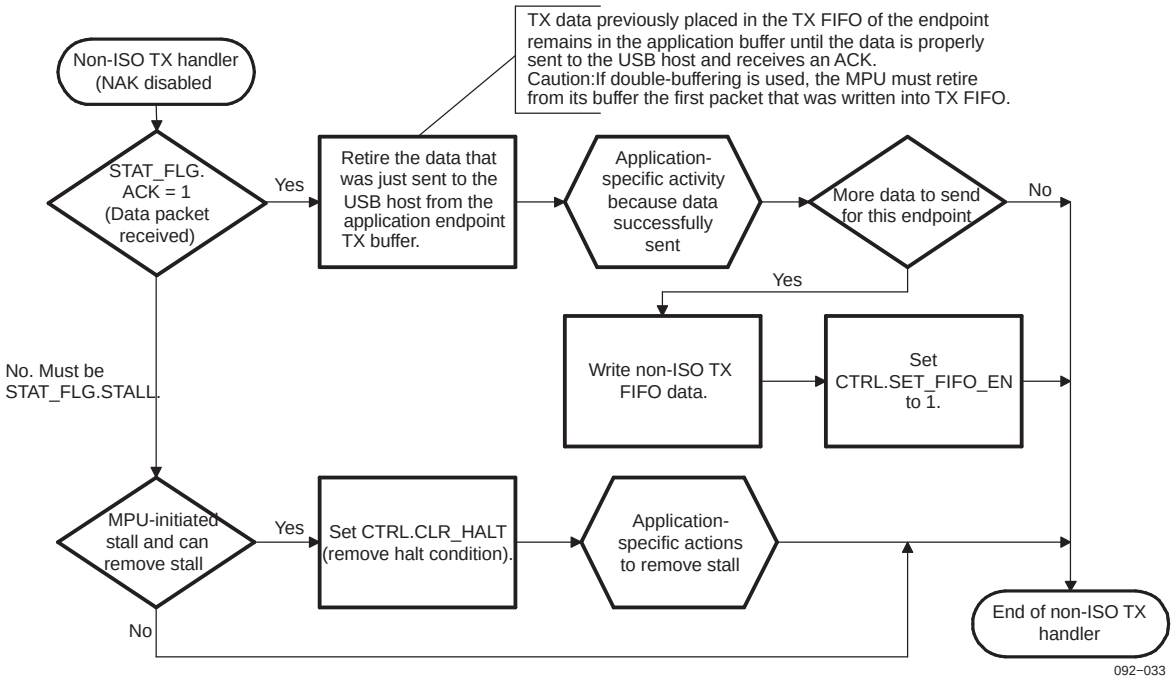
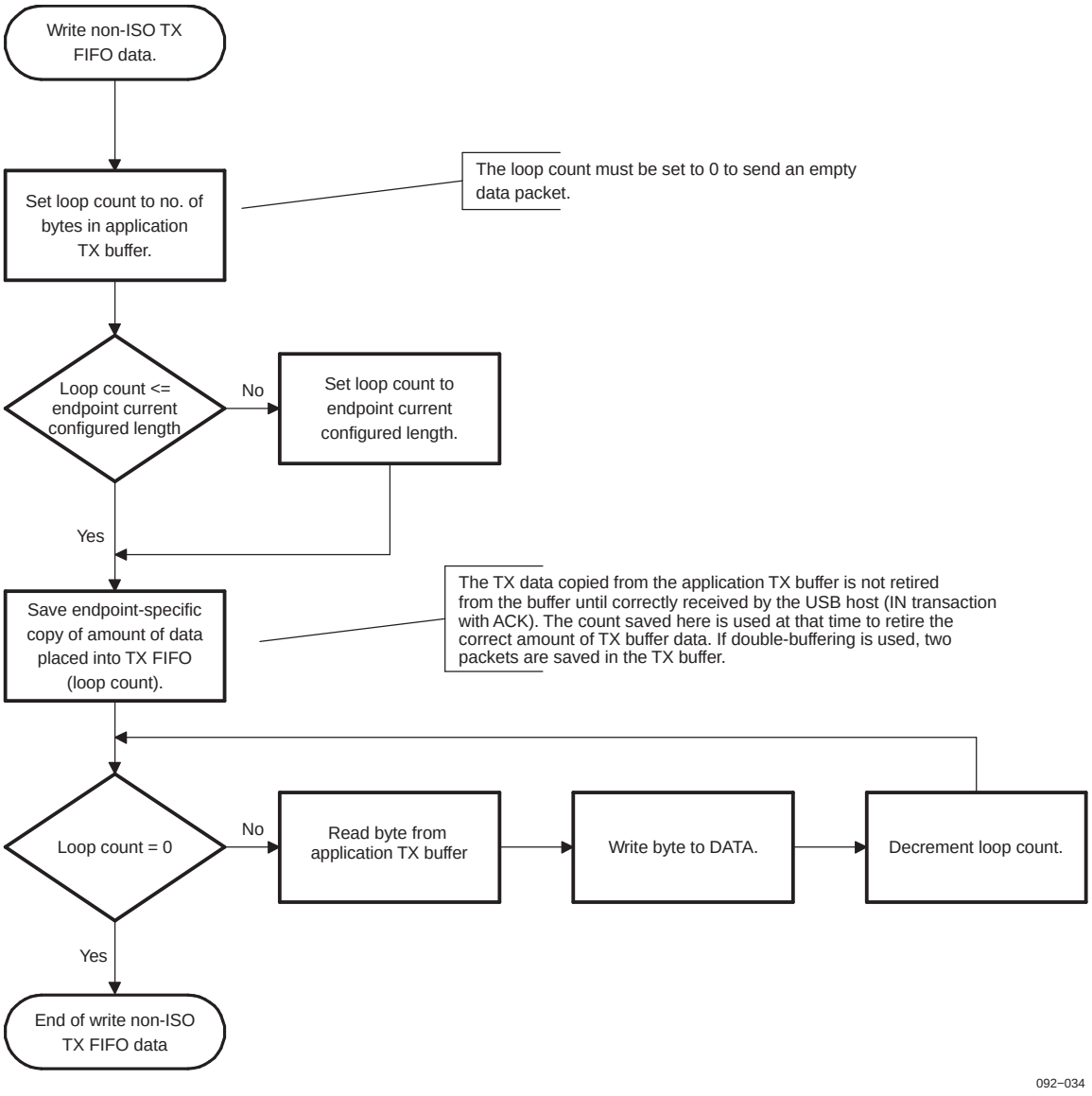


Figure 23-34. Write Nonisochronous TX FIFO Data Flowchart



23.4.9.15 SOF Interrupt Handler

The USB_IRQ_ISO SOF interrupts to the MPU subsystem (also known as isochronous interrupts) occur once per USB frame. The MPU subsystem must handle data traffic for the isochronous endpoints at each SOF interrupt. In addition, the SOF ISR can handle any application-specific activity related to the implicit timing of the SOF interrupt. Figure 23-35 shows the SOF ISR flowchart. The read ISO RX FIFO data and write ISO TX FIFO data procedures are shown in Figure 23-36 and Figure 23-37, respectively.

Figure 23-35. SOF Interrupt Handler Flowchart

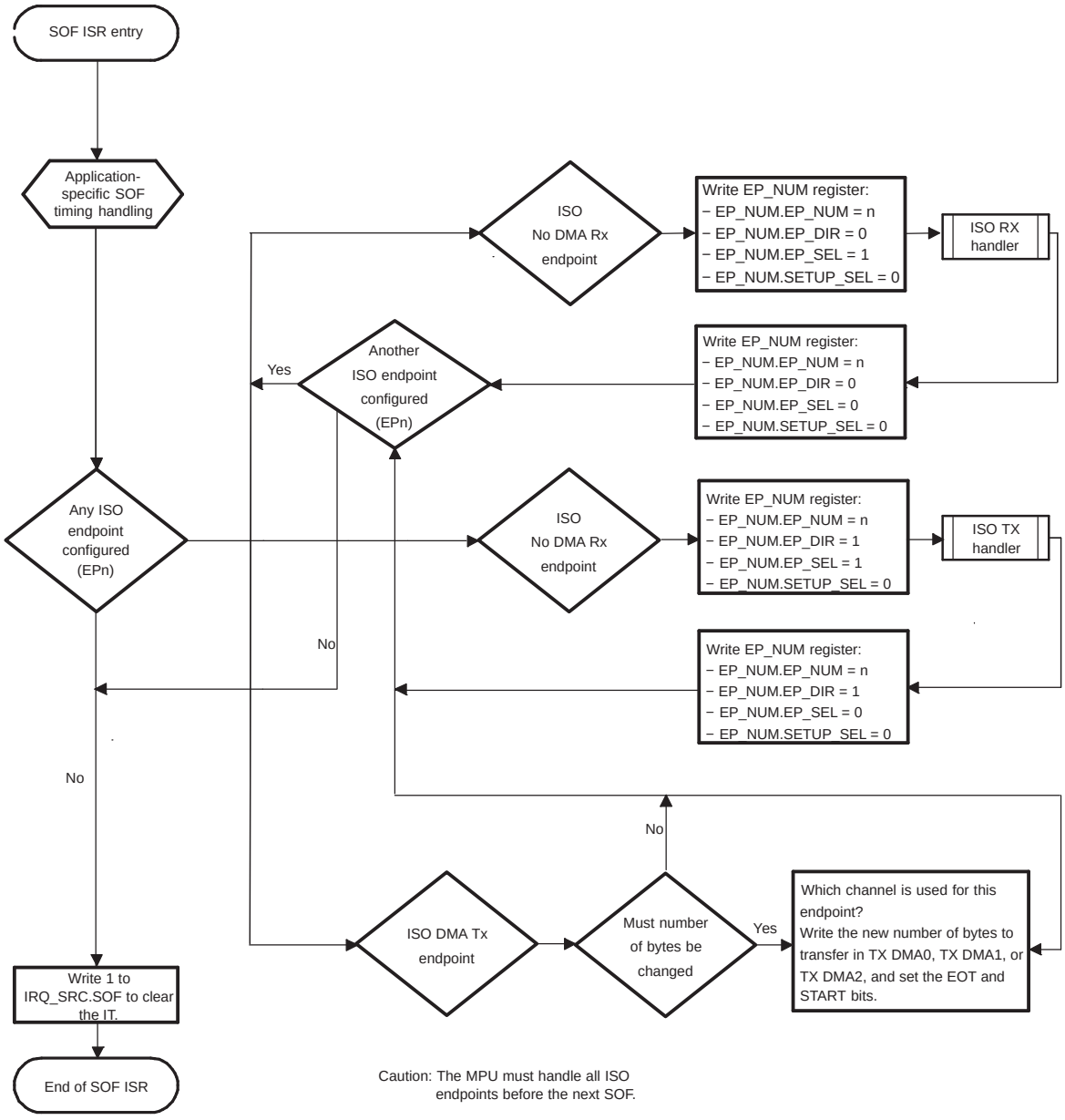


Figure 23-36. Read Isochronous RX FIFO Data Flowchart

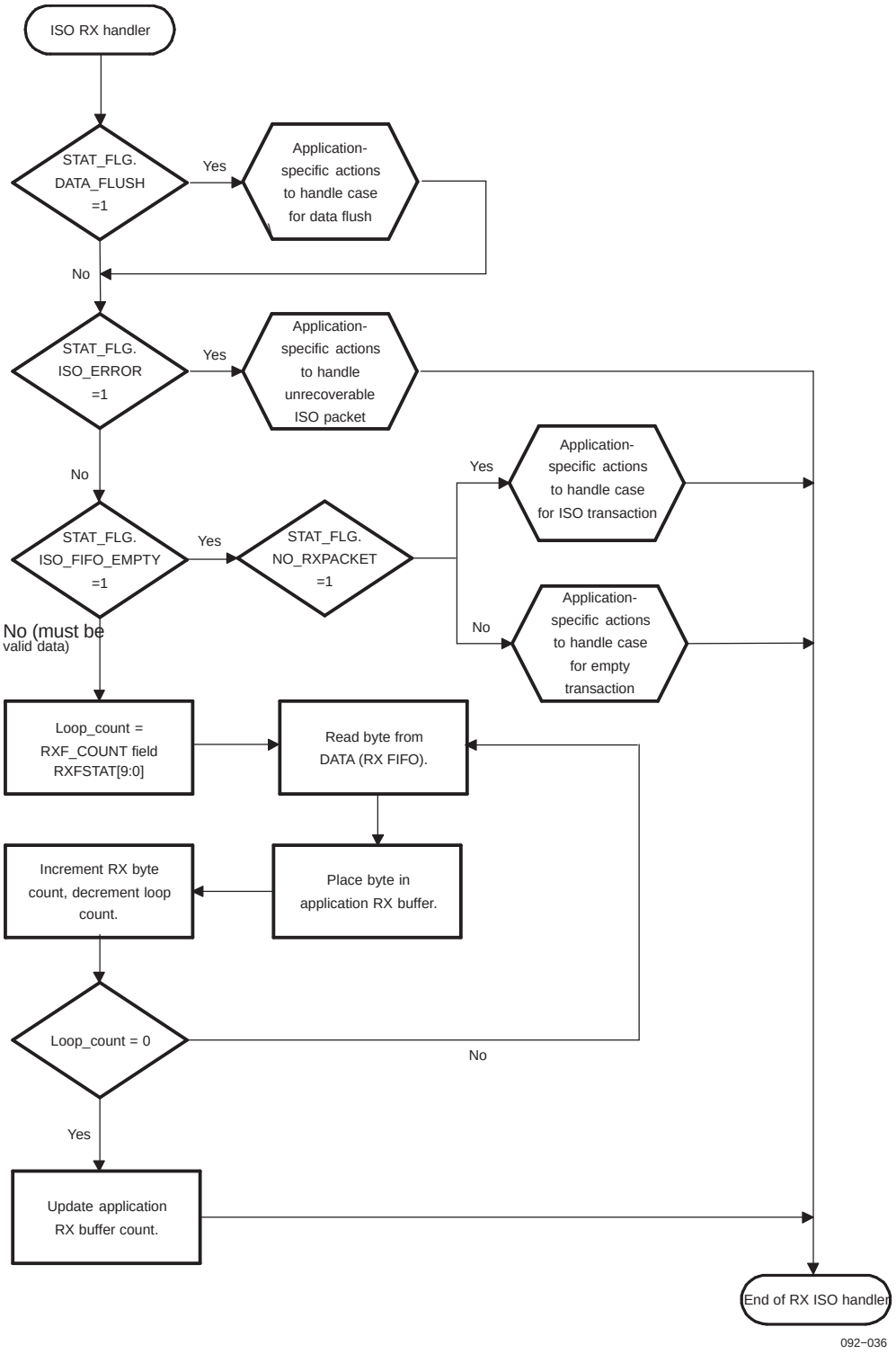
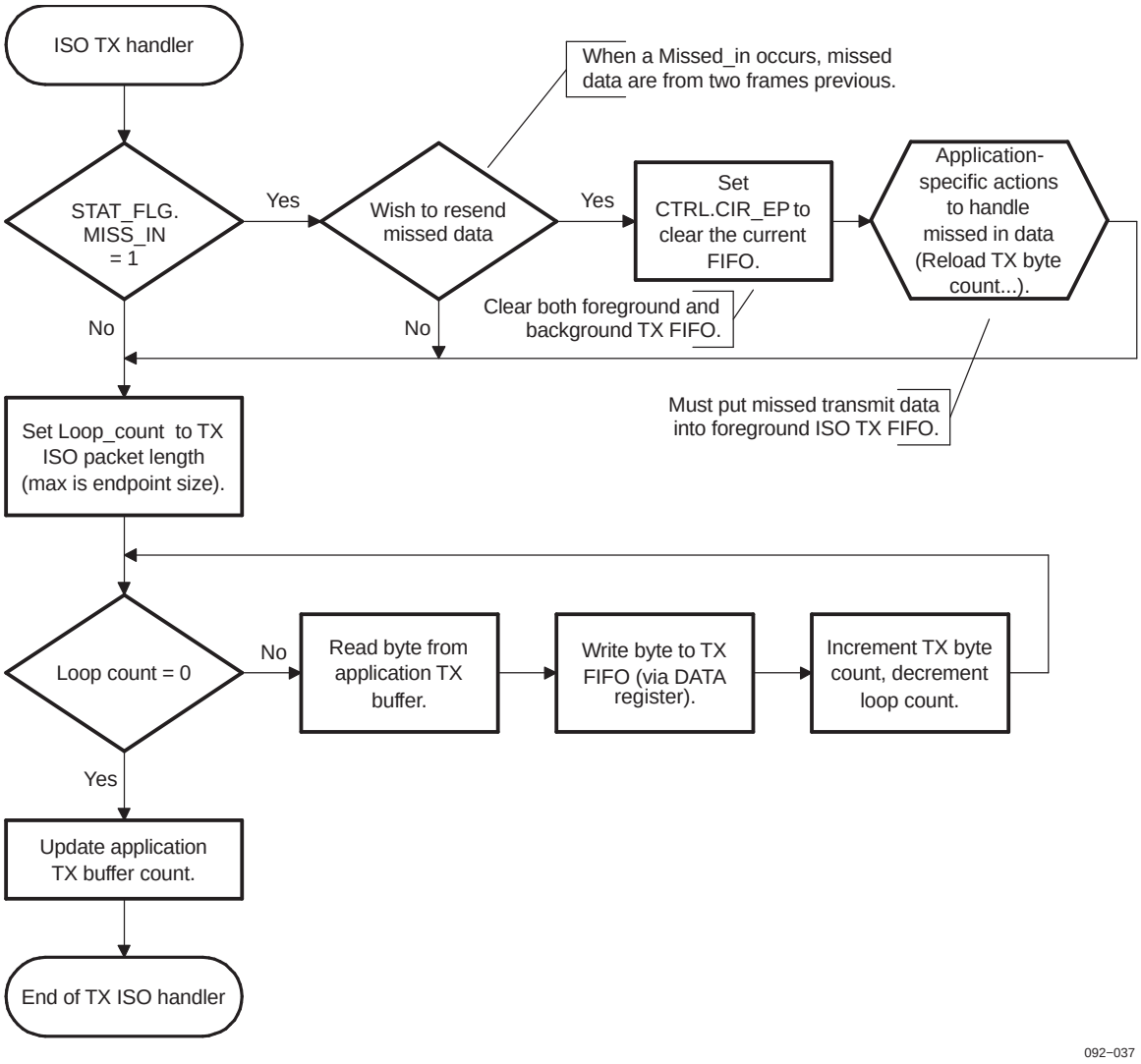


Figure 23-37. Write Isochronous TX FIFO Data Flowchart



23.4.9.16 Summary of USB Device Controller Interrupts

Table 23-8 lists the interrupt types by endpoint types.

Table 23-8. USB Device Controller Interrupt Type by Endpoint Type

Interrupt Type	USB_IRQ_GEN General USB IRQs				USB_IRQ_NISO EP-Specific IRQs (Non-ISO Interrupt On)		USB_IRQ_ ISO SOF ISO IRQs
	Setup (EP0)	Control (EP0) Out	Control (EP0) In	Other	Bulk or Interrupt Out	Bulk or Interrupt In	(ISO) SOF
Transaction ACKed		X	X		X	X	
Transaction NAKed (if enabled)		X	X		X	X	
Transaction STALLed		X	X		X	X	
Setup	X						
SOF							X

Table 23-8. USB Device Controller Interrupt Type by Endpoint Type (continued)

Interrupt Type	USB_IRQ_GEN General USB IRQs				USB_IRQ_NISO EP-Specific IRQs (Non-ISO Interrupt On)		USB_IRQ_ ISO SOF ISO IRQs
Device state changed				X			
RX DMA EOT (non-ISO)				X			
RX DMA trans count (non-ISO)				X			
TX DMA done (non-ISO)				X			

23.4.10 DMA Operation

The USB device controller provides support for six DMA channels. Three receive DMA channels are reserved to OUT transfers (isochronous or nonisochronous) and three transmit DMA channels are reserved to IN transfers (isochronous or nonisochronous). It is not possible to operate DMA transactions on control EP0.

CAUTION

The MPU subsystem must not access an endpoint used in a DMA transfer through the EP_NUM, CTRL, and STAT_FLG registers (in DMA, this remark applies after the MPU subsystem has set the SET_FIFO_EN bit CTRL[2] to enable the RX DMA transfer). In particular, the MPU subsystem must not set the halt feature while the endpoint is selected in the RXDMA_CFG register.

Note: To use the DMA channels properly, you must set the DMA configuration during the address state interrupt (DS_CHANGE).

The parameters used for DMA transactions (FIFO size, isochronous endpoint, double-buffering, and pointers) are those defined for the associated endpoint.

23.4.10.1 Receive DMA Channels Overview

Receive DMA channels are programmed through the three RXDMA control registers. Each channel can be assigned to a given endpoint number using a non-zero value in the RXDMAn_EP fields RXDMA_CFG (a value of 0 means the DMA channel is deselected).

Received OUT data must be read when an RX DMA request is active through the DATA_DMA register. The RX FIFO accessed is of the endpoint for which the DMA request is active (only one RX DMA request at a time is active).

The USB device controller receive DMA channels 0 through 2 are connected to LOCOSTO DMA controller requests DMA_10, DMA_12, and DMA_14, respectively. These DMA requests are made to the system DMA (sDMA).

23.4.10.2 Nonisochronous OUT (USB Host to MPU) DMA Transactions

During nonisochronous transfers to a DMA-operated OUT endpoint, a request to the MPU DMA controller is generated when data are placed into the endpoint FIFO and must be read. ACK and NAK interrupts are always disabled automatically by the core for DMA operated endpoints.

There are two dedicated maskable interrupts per DMA channel to control nonisochronous OUT transfers.

23.4.10.2.1 End-of-Transfer Interrupt (RXn_EOT Bit IRQ_SRC[8])

The end-of-transfer interrupt signals that the core detects an EOT, which occurs in the following two cases:

- When the last valid transaction to the endpoint is either an empty packet (ACK and buffer empty) or a packet whose size is less than the physical endpoint buffer size (ACK and buffer not full)
- When the number of received transactions has reached the programmed value in the RXn_TC field RXDMAn[7:0], if the RXn_STOP bit RXDMAn[15] is set by the MPU subsystem

After an end-of-transfer interrupt, the MPU subsystem must set the SET_FIFO_EN bit CTRL[2] for the endpoint to re-enable the channel.

The MPU subsystem must not initiate a new RX DMA transfer until it receives an end-of-transfer interrupt.

23.4.10.2.2 Transaction Count Interrupt (RXn_CNT Bit IRQ_SRC[9])

The intent of this interrupt is watermark control. The MPU subsystem can use it to monitor the file size of incoming transfers and take appropriate actions if the file received exceeds an expected size.

A transaction count interrupt does not disable the ongoing DMA transfer.

A transaction count interrupt occurs each time the number of received transactions (and not bytes) reaches the programmed value in the receive transaction counter for the DMA channel. One transaction has a size equal to the buffer size of the selected nonisochronous endpoint. RXn_COUNT interrupt is asserted even if the RXn_STOP bit RXDMAn[15] is set; in that case, both RXn_COUNT and RXn_EOT interrupts are asserted (see Figure 23-38 through Figure 23-41).

The transaction count watermark is programmed in the RXDMAn register.

Figure 23-38. Nonisochronous RX DMA Transaction Example (RX_TC=2)

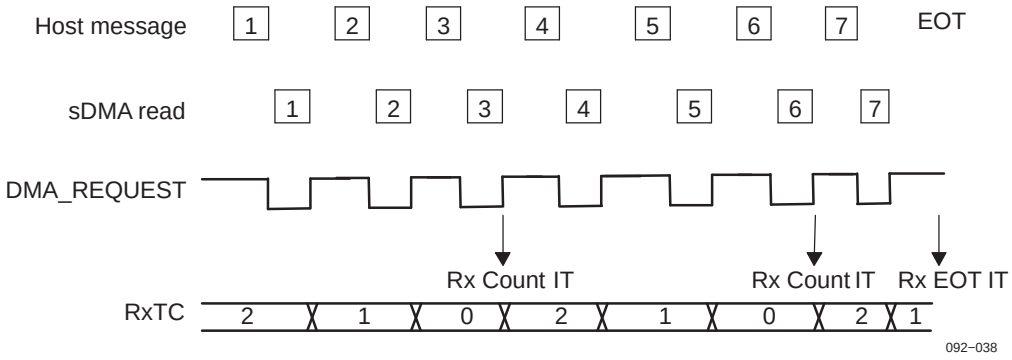


Figure 23-39. Nonisochronous RX DMA Start Routine

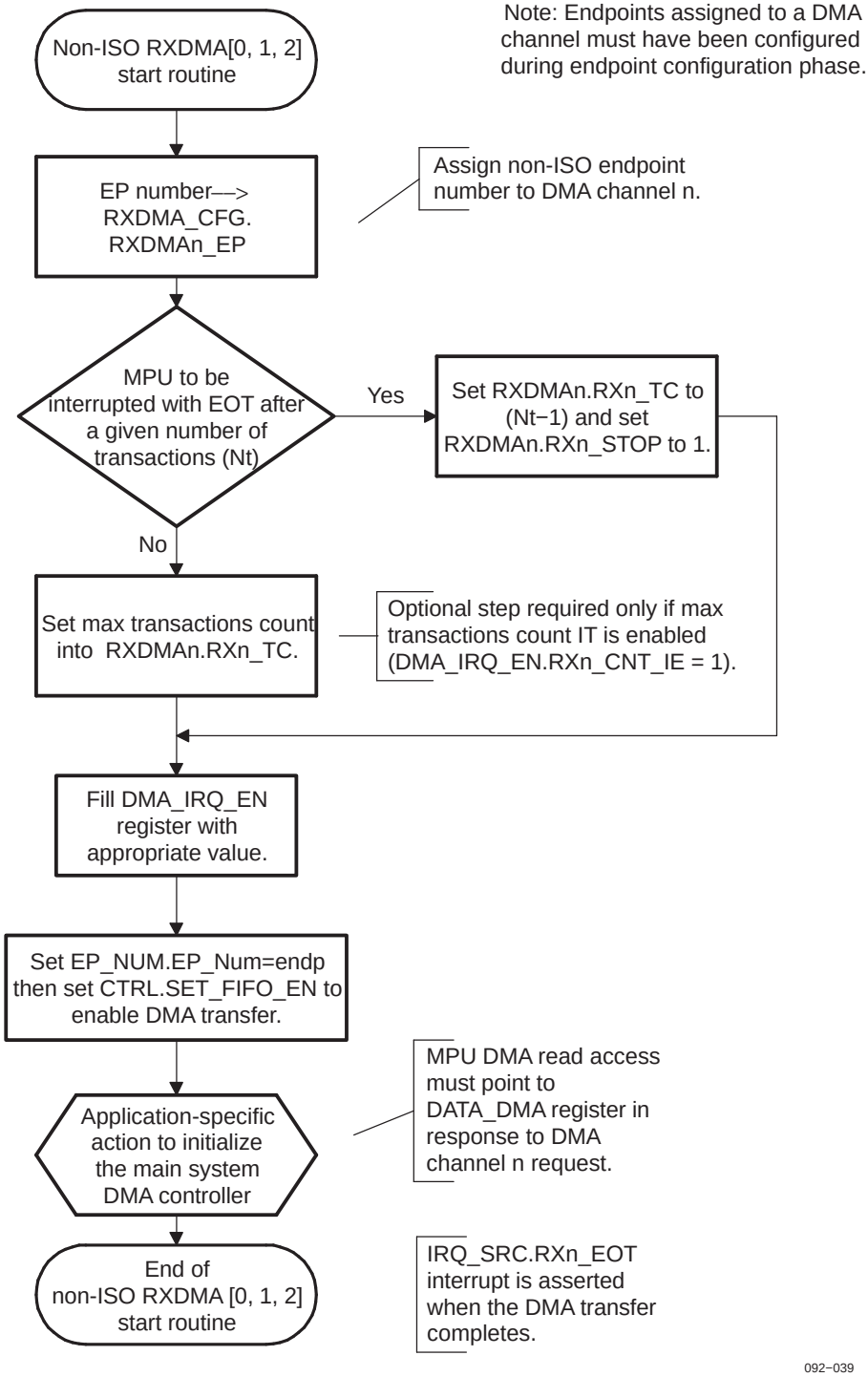


Figure 23-40. Nonisochronous RX DMA EOT Interrupt Handler

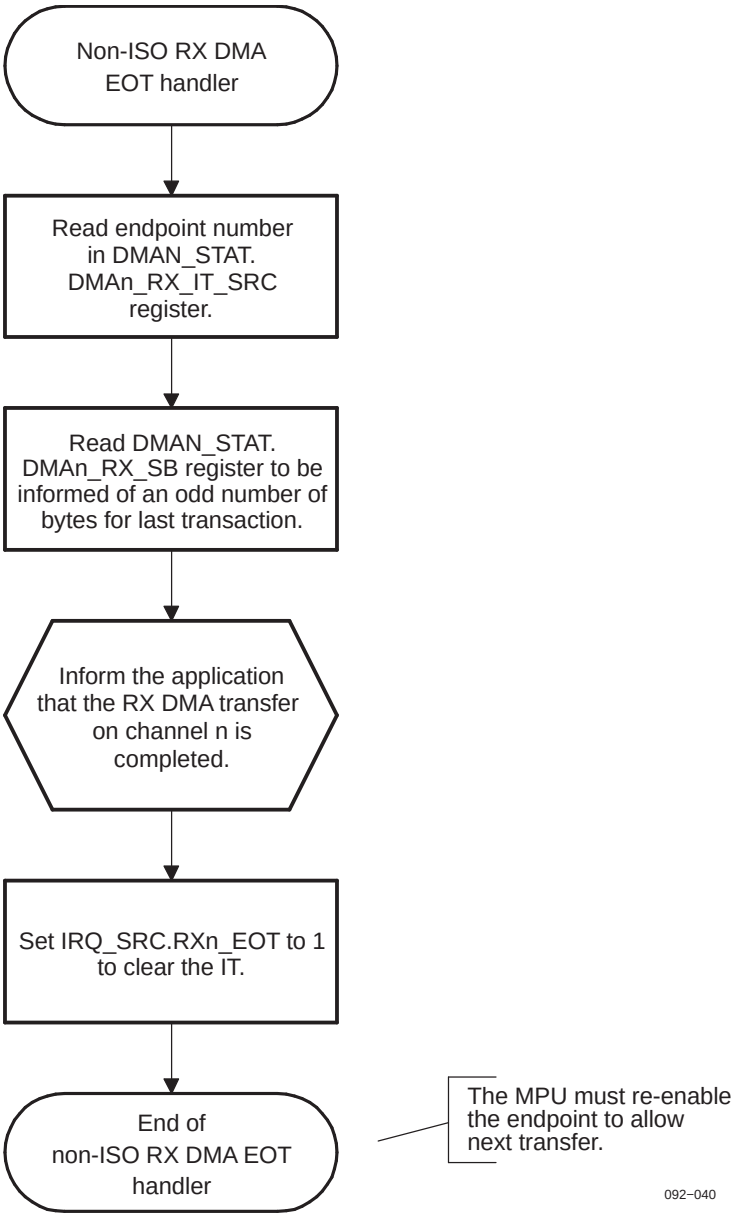
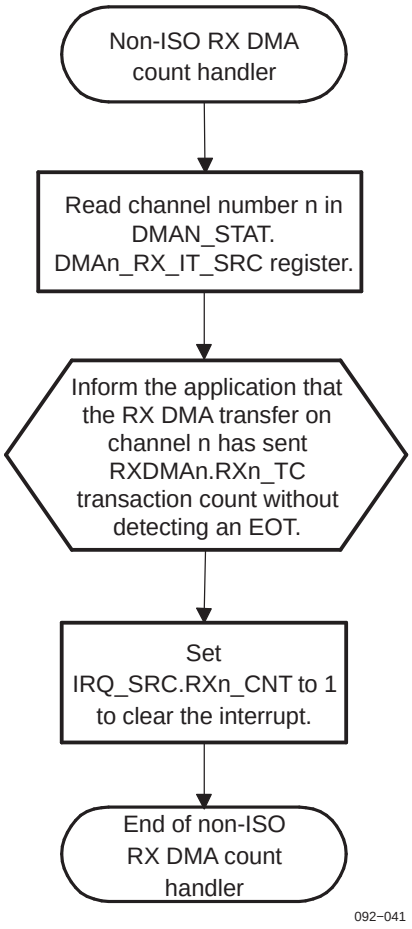


Figure 23-41. Nonisochronous RX DMA Transactions Count Interrupt Handler



23.4.10.3 Isochronous OUT (USB Host to MPU) DMA Transactions

During isochronous transfers to a DMA-operated OUT endpoint, a request to the MPU DMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. There is no interrupt associated with DMA transfer to ISO OUT endpoints (see Figure 23-42 and Figure 23-43).

Figure 23-42. ISO RX DMA Transaction

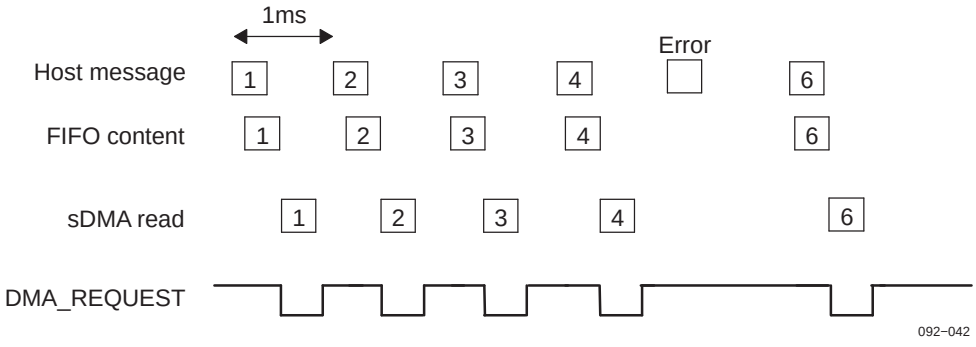
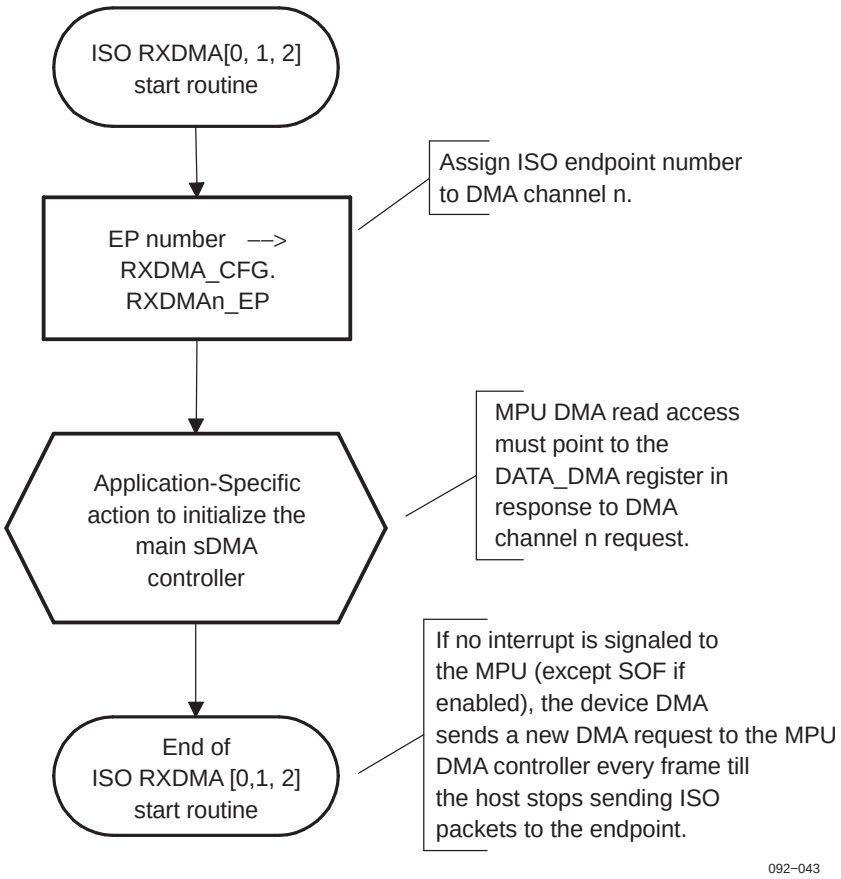


Figure 23-43. ISO RX DMA Start Routine



23.4.10.4 Transmit DMA Channels Overview

Transmit DMA channels are programmed through the three TXDMA control registers. Each channel can be assigned to a given endpoint number using a non-zero value in the TXDMAn_EP fields TXDMA_CFG (a value of 0 means the DMA channel is deselected).

The other three control registers (TXDMA0, TXDMA1, and TXDMA2) operate differently for isochronous and nonisochronous endpoints. Transmitted data must be written to the DATA_DMA when a TX DMA request is active. They are written to the TX FIFO of the endpoint associated with an active request (only one TX DMA request at a time is active).

The USB device controller transmit DMA channels 0 through 2 are connected to LOCOSTO DMA controller requests DMA_11, DMA_13, and DMA_15, respectively. These DMA requests are made to the sDMA.

23.4.10.5 Nonisochronous IN (MPU to USB Host) DMA Transactions

Nonisochronous (bulk) TX DMA file transfers are virtually unlimited in size. The flowcharts in Figure 23-45 and Figure 23-46 show how to handle small, medium, and large file transfers.

Registers TXDMA0, TXDMA1, and TXDMA2 operate for nonisochronous endpoints in the following manner: The transfer size counter (TXN_TSC bits TXDMAn[9:0]) corresponds to either the number of bytes to transmit (EOT bit set) or the number of buffers to transmit (EOT bit cleared). The buffer size corresponds to the programmed size of the TX endpoint.

A request is generated to the MPU main DMA controller when the endpoint buffer is empty, initially after the START bit is set, and then each time there is space free in the TX FIFO for other TX packets to be written, until the TXn_TSC bits TXDMAn[9:0] count down to 0. The request is removed when the buffer is full or when there are no more bytes of data to be sent.

A DMA transmit transfer done interrupt is signaled to the MPU subsystem after the last IN transaction completes successfully. This occurs after the START bit is set and after the TXn_TSC bits TXDMAn[9:0] equal 0 for the selected DMA channel.

The MPU subsystem must not initiate a new TX DMA transfer until it receives a TX_DONE interrupt.

A small file transfer of less than 1024 bytes can be achieved in a single-pass DMA signaled by a single interrupt completion. A file size equal to or greater than 1024 bytes requires two or more DMA passes, signaled by an interrupt completion after each pass.

The file to transfer size (FTZ) can be viewed as a concatenation of three arguments, as shown in Figure 23-44 .

Figure 23-44. File Transfer Size

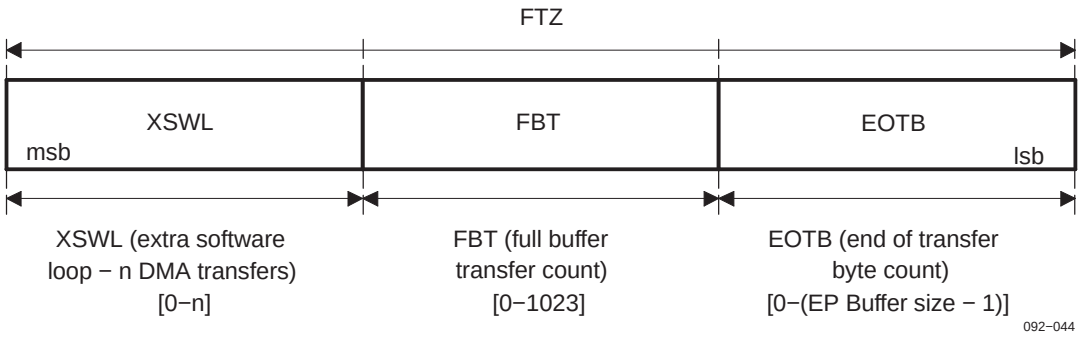


Figure 23-45 shows the steps required to prepare and permit a TX DMA transfer of any size. It also effectively starts the initial DMA transfer. The completion of this DMA task is signaled to the MPU subsystem through a DONE interrupt, the handler of which is shown in Figure 23-46. The start routine and the associated interrupt handler are tightly coupled. Example 1 shows a TX DMA transfer.

Figure 23-45. Nonisochronous TX DMA Start Routine

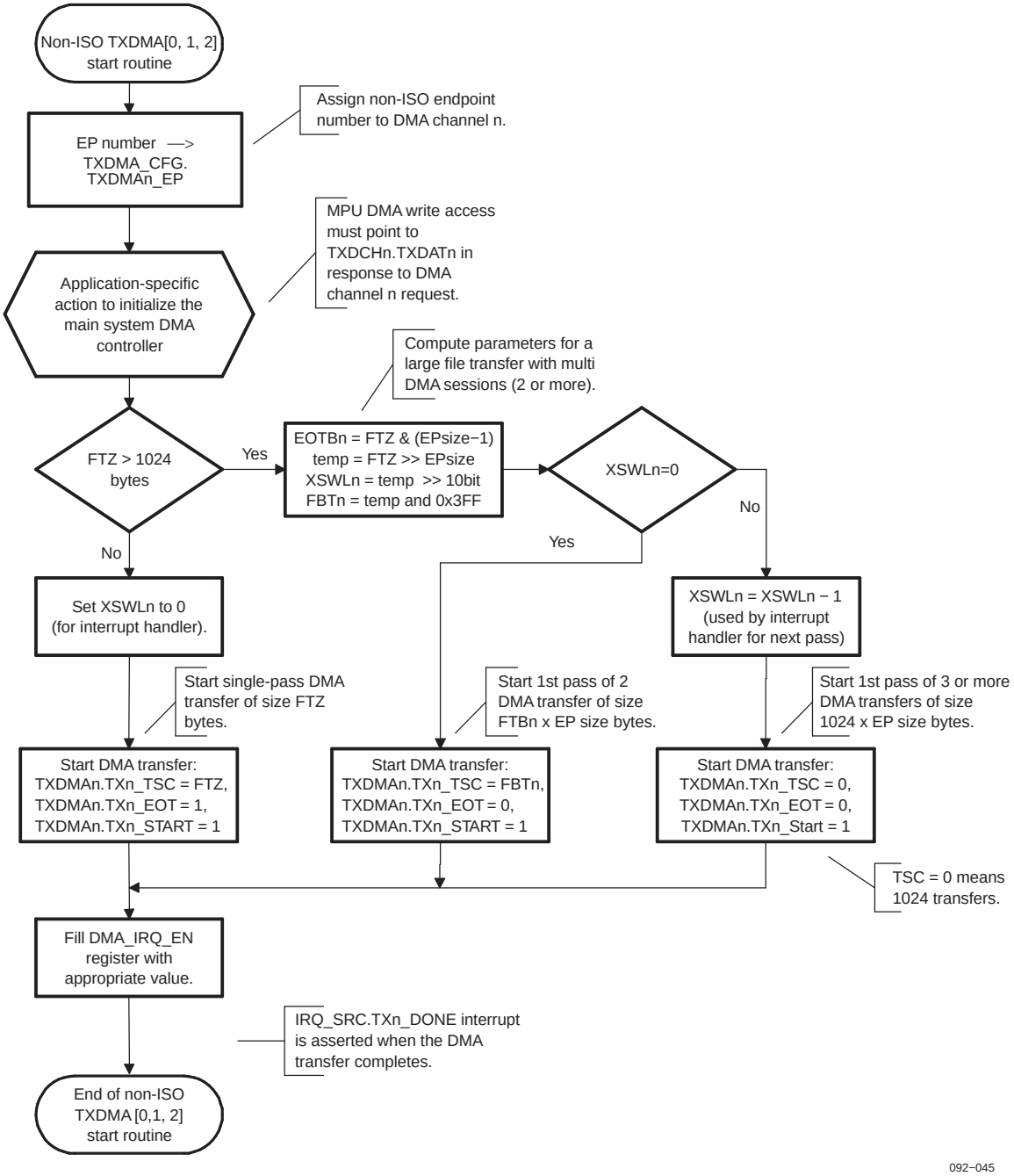
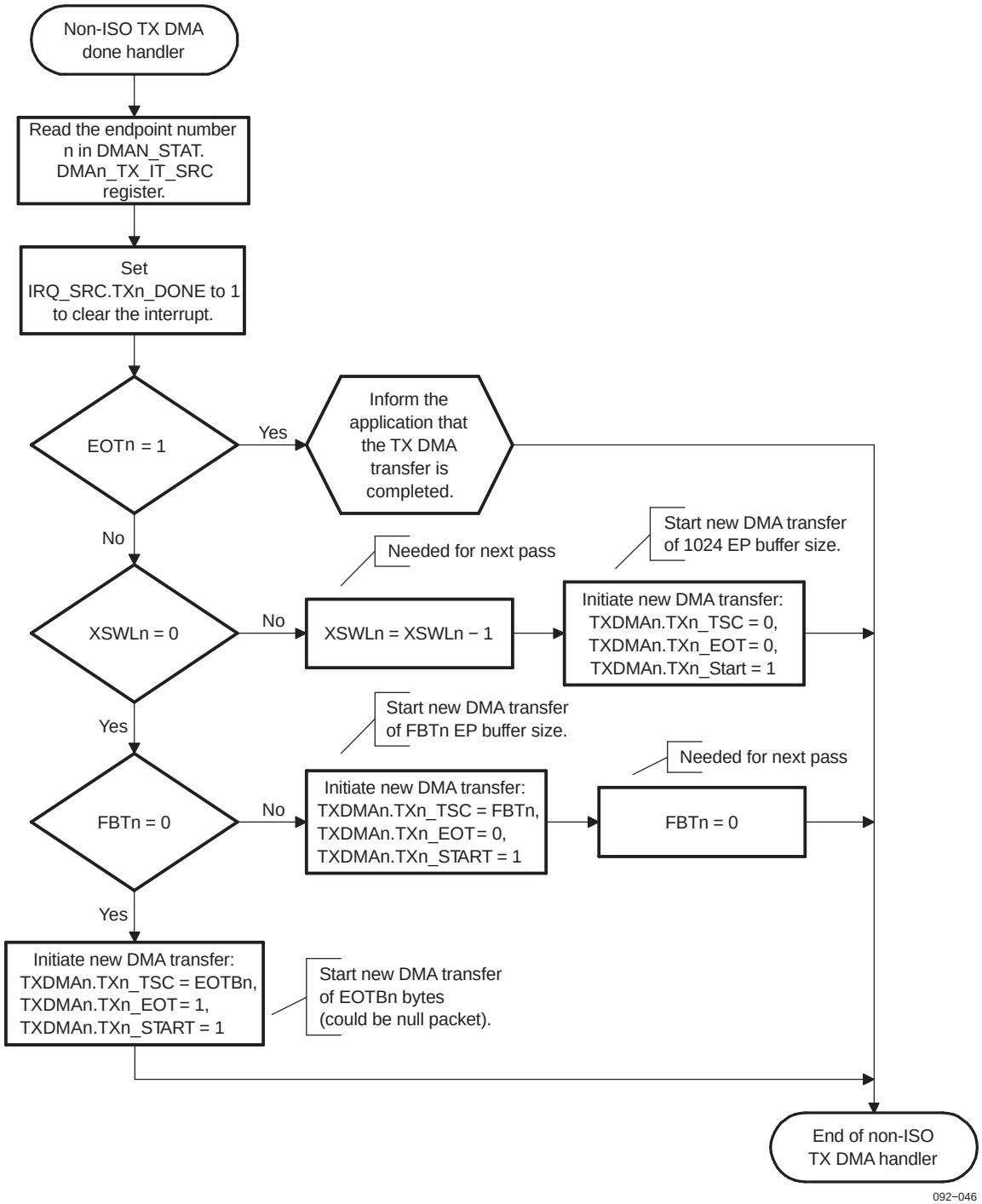


Figure 23-46. Nonisochronous TX DMA Done Interrupt Handler



Example 1. 100,603 Bytes to Transfer Through 32 Bytes IN Bulk Endpoint

This gives XSWL = 0x3, FBT = 0x47, EOTB = 0x1b,

which means five passes of DMA transfer, signaled by five TXn_DONE interrupts, are required:

1) EOT= 0, FBT= 0, loop=3 32,768 bytes transferred (1024 x 32 bytes)

2) EOT=0, FBT=0, loop=2	32,768 bytes
3) EOT=0, FBT=0, loop=1	32,768 bytes
4) EOT=0, FBT=0x47, loop=0	2272 bytes (71 x 32 bytes)
5) EOT=1, FBT=0x1B, loop=0	27 bytes

23.4.10.6 Isochronous IN (USB Host to MPU) DMA Transactions

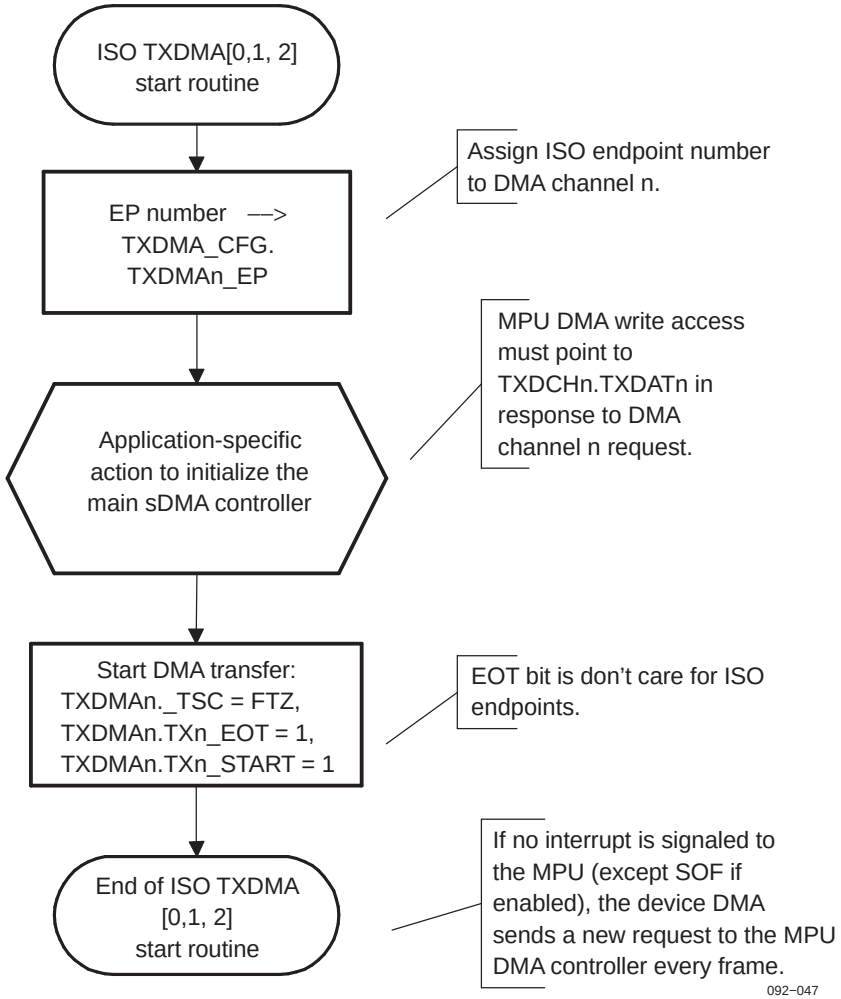
For isochronous endpoints, the transfer size counter (TXn_TSC bits TXDMAn[9:0]) corresponds to the number of bytes to transmit. The programmed size must not exceed the programmed buffer size of the endpoint. Otherwise, the result is unpredictable (see Figure 23-47).

A request is generated to the MPU main DMA controller when the endpoint buffer is empty, initially after the START bit is set, and then after each SOF (every 1 ms). The request is removed when the number of bytes written in the buffer matches the TXn_TSC bits TXDMAn[9:0] value.

During isochronous transfers to a DMA-operated IN endpoint, a request to the MPU sDMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. There is no special interrupt associated with the DMA transfer.

No interrupt is signaled to the MPU subsystem during DMA operation to ISO IN endpoints.

Figure 23-47. ISO TX DMA Start Routine



23.4.10.7 Important Note on DMA Requests

For each direction, only one DMA request can be active at any time. A request must then be serviced to allow the next pending request on the same direction to be asserted. In particular, a TX DMA request is asserted at each SOF if a TX DMA channel is configured for an isochronous endpoint; this request must be serviced imperatively.

23.4.10.8 Note on DMA Channels Deconfiguration

It is recommended that the MPU subsystem wait for an EOT (RX) or a DONE (TX) interrupt before disabling the channel by writing 0 in the TX/ RXDMA_CFG register. However, if needed by the application, the MPU subsystem can deselect the endpoint number in the TX/RXDMA_CFG register during a DMA transfer. The resulting behavior is:

- For RX transfer:
  - If the RX DMA request is active when the endpoint is unselected, deconfiguration is effective only at the end of the RX DMA request (that is, after all the DMA data have been read). When double-buffering is used, the deconfiguration is effective after both buffers are read (if both buffers are not empty at deselection). An EOT interrupt is asserted if an EOT is detected.
  - If the RX DMA request is not active when deselection occurs, the effect is immediate.
- For TX transfer:
  - If the TX DMA request is active when the endpoint is unselected, deconfiguration is effective after the TX DMA request is handled and the TX data is sent through an IN transaction (both buffers if double-buffering with both buffers full). No TX_DONE interrupt is asserted, even if the TXn_TSC bits TXDMAn[9:0] value is 0 after the transaction.
  - If the TX DMA request is inactive when the endpoint is unselected, deconfiguration is effective when all data available in TX buffer(s) are sent through IN transaction(s). If the TXn_TSC bits TXDMAn[9:0] value is 0 at this point, no TX_DONE interrupt is asserted. If the TX_DONE interrupt has already been asserted when the endpoint is deselected, configuration is effective only after the TX_DONE interrupt handling.

The TX/RXDMAn_EP bits TX/RXDMA_CFG reflects the endpoint value until deconfiguration is effective. The MPU subsystem must read this register to know if the channel is disabled yet or not. The MPU must wait until the read value is 0 before performing other actions to the endpoint. After effective deconfiguration, all transactions to the endpoint generate an endpoint-specific interrupt (if nontransparent).

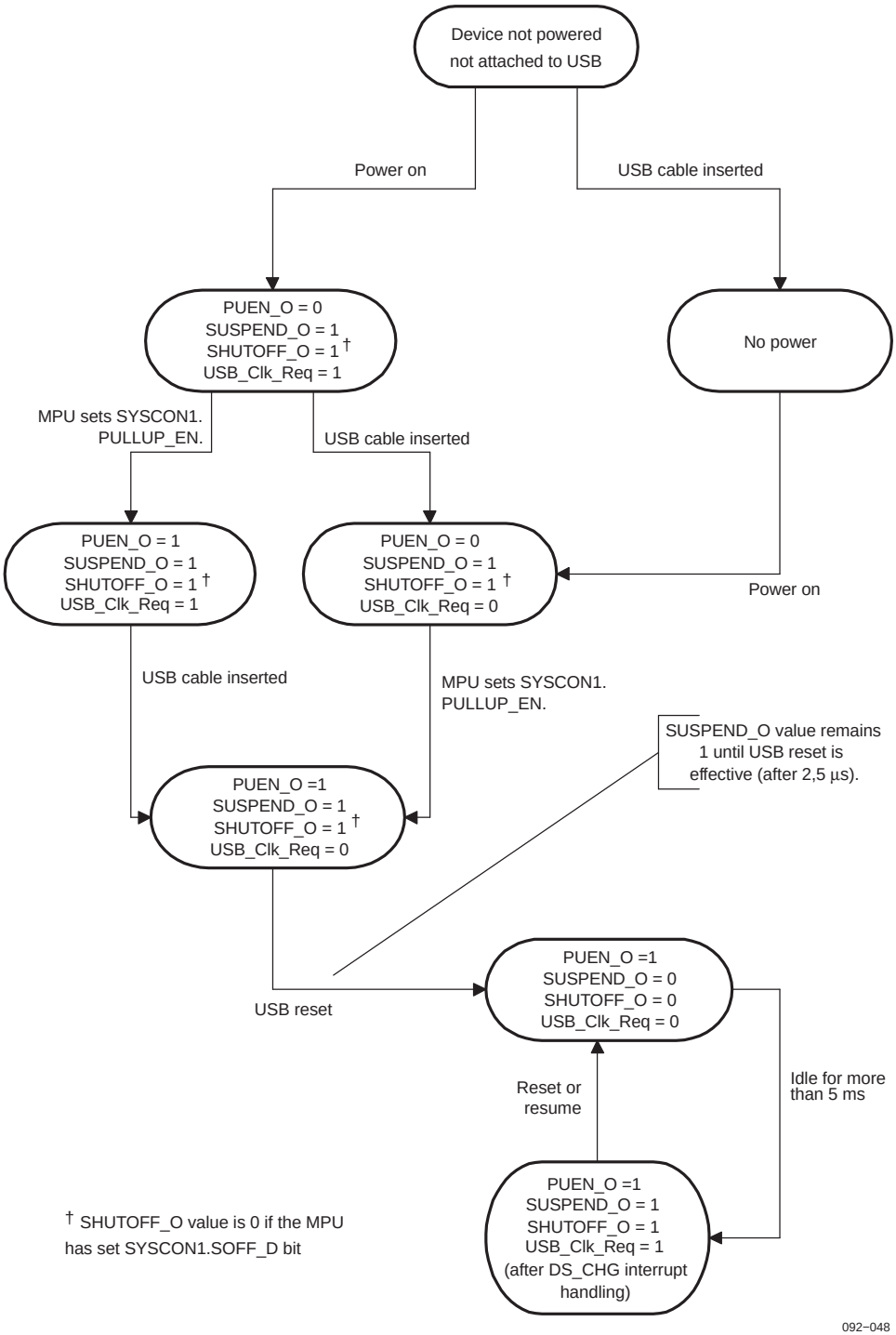
If the selected endpoint is isochronous, deconfiguration is effective after the TX/RX request is serviced, and the subsequent isochronous transactions are handled at SOF interrupt through the endpoint registers (EP_NUM and STAT_FLG).

23.4.11 Power Management

Figure 23-48 shows the values assigned to the USB device controller signals concerned with power management, in the function of the device state. These signals are:

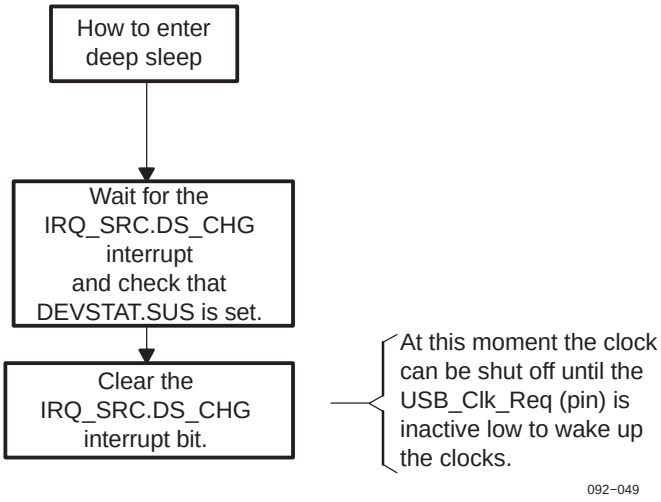
- PUEN_O: Pullup enable signal, always reflecting the SYSCON1.PULLUP_ EN register bit
- SHUTOFF_O: Power circuitry shutoff signal, controlled by the core and the SOFF_DIS bit SYSCON1[1]
- USB_Clk_Req: Asserted low to request the interface clock
- SUSPEND_O: Suspend signal, asserted high when the device is in suspend mode

Figure 23-48. Power Management Signal Values



From a software point of view, Figure 23-49 shows the reaction. This flowchart does not need to be implemented; it only reflects the way the module can enter deep sleep.

Figure 23-49. Power Management Flowchart



23.5 USB Client Register Manual

Table 23-9 lists the base address and address space for the USB module.

Table 23-9. Instance Summary

Module Name	Base Address	Size
USB	0xFFFF B000	2K bytes

CAUTION

The MPU processor can read from or write to registers using one of the following accesses:

- 16-bit access: All bits of the register are accessed.
- 8-LSB access: The 8 LSBs are accessed.
- 8-MSB access: The 8 MSBs are accessed.

MPU processor actions are required depending on the access mode, in some particular cases when reading or writing data.

23.5.1 USB Register Mapping Summary

Table 23-10 lists the USB module registers.

Table 23-10. USB Module Registers

Register Name	Type	Register Width (Bits)	Offset
REVDEV	R	16	0x00
EP_NUM	RW	16	0x02
DATA	RW	16	0x04
CTRL	RW	16	0x06
STAT_FLG	R	16	0x08
RXFSTAT	R	16	0x0A
SYSCON1	RW	16	0x0C
SYSCON2	RW	16	0x0E
DEVSTAT	R	16	0x10
SOFREG	R	16	0x12
IRQ_EN	RW	16	0x14
DMA_IRQ_EN	RW	16	0x16
IRQ_SRC	RW	16	0x18
EPN_STAT	R	16	0x1A
DMAN_STAT	R	16	0x1C
RXDMA_CFG	RW	16	0x20
TXDMA_CFG	RW	16	0x22
DATA_DMA	RW	16	0x24
TXDMA0 _UnicodeEncodeError_ TXDMA2	RW	16	0x28_UnicodeEncodeError_0x 2C
RXDMA0 _UnicodeEncodeError_ RXDMA2	RW	16	0x30_UnicodeEncodeError_0x 34
EP0	RW	16	0x40
EP_RX1 _UnicodeEncodeError_ EP_RX15	RW	16	0x42_UnicodeEncodeError_0x 5E

Table 23-10. USB Module Registers (continued)

Register Name	Type	Register Width (Bits)	Offset
EP_TX1 _UnicodeEncodeError_ EP_TX15	RW	16	0x62_UnicodeEncodeError_0x 7E

23.5.2 Register Description

Table 23-11 through Table 23-33 describe the register bits.

Table 23-11. REVDEV

Address Offset	0x00		
Physical Address	0xFFFF B200	**Instance**	USBOT1
Description	Revision Register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REV_NB							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	0x_UnicodeEncodeError__UnicodeEncodeError_
7:0	REV_NB	Revision number of current USB device controller. Examples: 0x01 for 0.1, 0x21 for 2.1	R	See (1)

(1) TI internal data

Table 23-12. EP_NUM

Address Offset	0x02		
Physical Address	0xFFFF B02	**Instance**	USBOT1
Description	Endpoint Selection Register		
Type	RW	**Associated Register**	DATA, STAT_FLG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SETUP_SEL	EP_SEL	EP_DIR	EP_NUM				

Bits	Field Name	Description	Type	Reset
15:7	Reserved	Reserved	RW	0x_UnicodeEncodeError__UnicodeEncodeError__UnicodeEncodeError_
6	SETUP_SEL	The setup FIFO select bit. Set by the local host in response to a Setup general USB Interrupt, to access the EP0 read-only Setup FIFO when reading DATA register. Setting this bit has for effect to clear IRQ_SRC. SETUP interrupt bit. When this bit is set, other EP_NUM register bits value must be 0. 0x0: No access 0x1: Access permitted	RW	0

Bits	Field Name	Description	Type	Reset
5	EP_SEL	Set by the local host to access the status (STAT_FLG, RXFSTAT) and data (DATA) registers for the endpoint selected. After each access to an endpoint during IT handling, the local host must ABSOLUTELY clear this bit. **Caution:** When the local host sets this bit, it must set SETUP_SEL bit to 0. After having accessed the endpoint FIFO, either for read or write access, the local host must clear this bit (write 0x0). 0x0: No access 0x1: Access permitted	RW	0
4	EP_DIR	Endpoint direction bit. Gives the direction associated with the endpoint number selected in EP_NUM.EP_NUM. 0x0: OUT endpoint 0x1: IN endpoint	RW	0
3:0	EP_NUM	Endpoint number binary. Encoded in these four bits, associated to the direction given by EP_NUM.EP_DIR bit, is the current endpoint selected. All reads and writes to the endpoint status, control and data locations are for this endpoint. 0x0:: EP0 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6 0x7: EP7 0x8: EP8 0x9: EP9 0xA: EP10 0xB: EP11 0xC: EP12 0xD: EP13 0xE: EP14 0xF: EP15	RW	0x0

Table 23-13. DATA

Address Offset	0x04		
Physical Address	0xFFFF B004	**Instance**	USBOT1
Description	Data Register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Bits	Field Name	Description	Type	Reset
15:0	DATA	EP_NUM.EP_DIR = 1: this register contains the data written by the local host to be sent to the USB host during the next IN transaction. Data can only be written successfully if the EP_NUM.EP_SEL bit is asserted. EP_NUM.EP_DIR = 0: this register contains the data received by the USB core from USB host OUT or SETUP transactions. Data can be read successfully only if the EP_NUM.EP_SEL bit is asserted, or if the EP_NUM.SETUP_SEL bit is asserted (for setup data).	RW	0x_UnicodeEncodeError__UnicodeEncodeError__UnicodeEncodeError__UnicodeEncodeError_

Bits	Field Name	Description	Type	Reset
		Note: A write into the DATA register when EP_NUM.EP_DIR=0 and a read from the DATA register when EP_NUM.EP_DIR=1 are denied.		

Table 23-14. CTRL

Address Offset	0x06		
Physical Address	0xFFFF B006	**Instance**	USBOT1
Description	Control Register		
Type	RW 1toClr		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CLR_HALT	SET_HALT	Reserved			SET_FIFO_EN	CLR_EP	RESET_EP

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	RW	0x_UnicodeEncodeError__UnicodeEncodeError_
7	CLR_HALT	Clear halt endpoint (nonisochronous) bit. EP_NUM.EP_DIR = 1: this register contains the data written by the local host to be sent to the USB host during the next IN transaction. Data can be written successfully only if the EP_NUM.EP_SEL bit is asserted. EP_NUM.EP_DIR = 0: this register contains the data received by the USB core from USB host OUT or SETUP transactions. Data can be read successfully only if the EP_NUM.EP_SEL bit is asserted, or if the EP_NUM.SETUP_SEL bit is asserted (for Setup data). Note: A write into the DATA register when EP_NUM.EP_DIR=0 and a read from the DATA register when EP_NUM.EP_DIR=1 are denied. 0x0: No action 0x1: Clear halt condition	RW 1toClr	0
6	SET_HALT	Set halt endpoint (nonisochronous). Only concerns nonisochronous endpoints. Used by the local host to halt the selected endpoint. **Caution:** If the endpoint to halt is used by a DMA channel, the local host must disable the DMA channel before setting halt condition for this endpoint. Always reads 0x0. 0x0 No action 0x1: Clear halt condition	RW 1toClr	0
5:3	Reserved	Reserved	RW	0x-
2	SET_FIFO_EN	Set FIFO enable (nonisochronous) bit. Only concerns nonisochronous endpoints. If the selected endpoint direction is IN, this bit is used by the local host to enable the USB device to transmit data from the FIFO at the next valid IN token. If the selected endpoint direction is OUT, this bit is used by the local host to enable the USB device to receive data from the USB host at the next valid OUT transaction. If not set the device will return a NAK handshake.	RW 1toClr	0

Bits	Field Name	Description	Type	Reset
		Caution: the local host must never enable endpoint 0 FIFO out of control transfers. For Bulk and Interrupt endpoints, FIFO must never be enabled when Halt feature is set, or when RX FIFO is not empty. Furthermore, during EP Interrupts Handling, the local host must have cleared the interrupt bit before setting CTRL.SET_FIFO_EN bit (to avoid masked ACK interrupts). Always reads 0x0. 0x0: No action 0x1: FIFO enabled		
1	CLR_EP	Clear endpoint. Set by the local host to clear the selected endpoint FIFO pointers and flags. It will also clear previous transaction handshake status. For isochronous endpoints or nonisochronous double-buffered endpoints, both foreground and background FIFO are cleared. Always reads 0x0. 0x0: No action 0x1 Clear endpoint	RW 1toClr	0
0	RESET_EP	Endpoint reset (noncontrol) bit. Only concerns non-Ctrl Endpoints. Set by the local host to reset the selected endpoint. It forces for an Interrupt or a Bulk Endpoint Data PID to DATA0, clears halt condition, clears FIFO and previous transactions handshake status. For an isochronous endpoint, it only clears the FIFO (both foreground and background). Always reads 0x0. 0x0: No action 0x1 Reset endpoint	RW 1toClr	0

Table 23-15. STAT_FLAG

Address Offset	0x08		
Physical Address	0xFFFF B008	**Instance**	USBOT1
Description	Status Register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NO_RXPACKET	MISS_IN	DATA_FLUSH	ISO_ERR	Reserved		ISO_FIFO_EMPTY	ISO_FIFO_FULL	Reserved	EP_HALTED	STALL	NAK	ACK	FIFO_EN	NON_ISO_FIFO_ EMPTY	NON_ISO_FIFO_ FULL

Bits	Field Name	Description	Type	Reset
15	NO_RXPACKET	Isochronous no packet received (ISO OUT) bit. Only concerns ISO OUT endpoints. Notify the local host that the core has not received any isochronous packet on the endpoint. Updated on an SOF. Read 0x0: Endpoint received a packet on the previous frame Read 0x1: Endpoint did not receive a packet on the previous frame	R	1
14	MISS_IN	Isochronous missed IN token for previous frame (ISO IN) bit. Only concerns ISO IN endpoints.	R	0

Bits	Field Name	Description	Type	Reset
		Notify the local host that the core missed a valid ISO IN token during previous frame, and that TX Data were flushed from the FIFO instead of being transmitted to the USB host. Updated on an SOF. Read 0x0: Endpoint received an IN token the previous frame Read 0x1: Endpoint did not receive an IN token the previous frame and TX data were flushed.		
13	DATA_FLUSH	Isochronous receive data flush (ISO OUT) bit. Only concerns ISO OUT endpoints. When set, indicates that data was flushed from the isochronous FIFO that was moved from the foreground to the background (the local host does not read all of the data from the foreground FIFO in a frame). Updated every frame. Read 0x0: Not significant Read 0x1: Data was flushed	R	0
12	ISO_ERR	Isochronous receive data error (ISO OUT) bit. Only concerns ISO OUT endpoints. When set, indicates that the ISO data packet was received incorrectly (the core detects an error in the data packet (CRC, Bit Stuffing, PID check) or an overrun condition occurs in the FIFO). When this bit is set the FIFO contents are automatically flushed by the core and the FIFO status is empty. Updated every frame. Read 0x0: Not significant Read 0x1: ISO packet received with errors	R	0
11:10	Reserved	Reserved	R	0x-
9	ISO_FIFO_EMPTY	ISO FIFO empty bit concerns ISO endpoints only. Only concerns ISO endpoints. Set when the FIFO for the selected ISO endpoint is empty, either through an appropriate CTRL.CLR_EP bit or CTRL.RESET_EP bit or after successful reads from the selected FIFO. Read 0x0: ISO FIFO is not empty. Read 0x1: ISO FIFO is empty.	R	1
8	ISO_FIFO_FULL	ISO FIFO full bit only concerns ISO endpoints. Only concerns ISO endpoints. Set when the FIFO for the selected ISO endpoint is full. This condition is cleared by setting the CTRL.CLR_EP bit or the CTRL.RESET_EP bit, or after one successful read (by the local host or the USB host). Read 0x0: ISO FIFO is not full. Read 0x1: ISO FIFO is full.	R	0
7	Reserved	Reserved	R	-
6	EP_HALTED	Endpoint halted flag (nonisochronous) bit. Only concerns nonisochronous endpoints. When set, indicates the selected endpoint is halted. The endpoint can be put into the halt state only by the local host writing the endpoint halt control bit (in response to a SET_FEATURE request for instance). 0x0 Selected endpoint is not halted. 0x1 Selected endpoint is halted. Read 0x0: The selected endpoint is not halted. Read 0x1: The selected endpoint is halted.	R	0
5	STALL	Transaction stall (nonisochronous) bit. Only concerns nonisochronous endpoints.	R	0

Bits	Field Name	Description	Type	Reset
		Set at the end of a transaction if a STALL handshake packet is returned to the USB host, and if no nonhandled interrupt is pending on the current buffer. The core automatically returns a STALL packet if a valid IN token is received by a halted TX endpoint, or if a valid OUT transaction is received by an halted RX endpoint, or if there is a request error (endpoint 0). Cleared when the local host finishes handling the corresponding interrupt (at EP_NUM.EP_Sel bit deselection). Read 0x0: No STALL handshake is returned. Read 0x1: STALL handshake is returned.		
4	NAK	Transaction nonacknowledge (nonisochronous) bit. Set at the end of a transaction if a NAK handshake is returned to the USB host, and if no nonhandled interrupt is pending on current buffer. The USB core automatically returns a NAK handshake to the USB host if a valid IN token is received by a TX endpoint or if a valid OUT transaction is received by an RX endpoint, and the STAT_FLG.FIFO_En bit is not set for the endpoint. Cleared when the local host finishes handling the corresponding interrupt (at EP_NUM.EP_Sel bit deselection). Read 0x0: No NAK handshake is returned. Read 0x1: NAK handshake is returned.	R	0
3	ACK	Transaction acknowledge (nonisochronous) bit. Set at the end of a nontransparent valid IN transaction if the data packet is sent successfully to the USB host, and the ACK handshake is received, or at the end of a nontransparent valid OUT transaction, if the data packet is received successfully by the USB device, and the ACK handshake is returned. Cleared when the local host finishes handling the corresponding interrupt (at EP_NUM.EP_Sel bit deselection). Read 0x0: No ACK handshake is returned. Read 0x1: ACK handshake is returned.	R	0
2	FIFO_EN	FIFO enable status (nonisochronous) bit. This bit is asserted when CTRL.SET_FIFO_EN is set to 1 and cleared automatically after a transaction completes with an ACK, STALL. Read 0x0: Nonisochronous endpoint FIFO is disabled. Read 0x1: Nonisochronous endpoint FIFO is enabled.	R	0
1	NON_ISO_FIFO_EMPTY	Nonisochronous FIFO empty bit. Set when the FIFO for the selected nonisochronous endpoint is empty, either through an appropriate CTRL.CLR_EP bit or CTRL.RESET_EP bit or after successful reads from the selected FIFO. Read 0x0: Nonisochronous endpoint FIFO is not empty. Read 0x1: Nonisochronous endpoint FIFO is empty.	R	1
0	NON_ISO_FIFO_FULL	Nonisochronous FIFO full bit. Concerns nonisochronous endpoints only. Set when the FIFO for the selected nonisochronous endpoint is full. This condition is cleared by setting the CTRL.CLR_EP bit or the CTRL.RESET_EP bit, or after one successful read (by the local host or the USB host). Read 0x0: Nonisochronous endpoint FIFO is not full. Read 0x1: Nonisochronous endpoint FIFO is full.	R	0

Table 23-16. RXFSTAT

Address Offset	0x0A		
Physical Address	0xFFFF B00A	**Instance**	USBOT1
Description	Receive FIFO Status Register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RXF_COUNT							

Bits	Field Name	Description	Type	Reset
15:10	Reserved	Reserved	R	0x–
9:0	RXF_COUNT	Receive FIFO byte count 10_UnicodeEncodeError_bit field. Indicates the number of bytes currently in the receive FIFO.	R	0x000

Table 23-17. SYSCON1

Address Offset	0x0C		
Physical Address	0xFFFF B00C	**Instance**	USBOT1
Description	System Configuration Register 1		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CFG_LOCK	DATA_ENDIAN	DMA_ENDIAN	Reserved	NAK_EN	AUTODEC_DIS	SELF_PWR	SOFF_DIS	PULLUP_EN

Bits	Field Name	Description	Type	Reset
15:9	Reserved	Reserved	RW	0x00
8	CFG_LOCK	Device configuration locked bit. After the local host enters the device configuration (registers 0x20 to 0x3F), it must set this bit to use the device. When the device configuration is not locked, the device is not ready to used. 0x0 Device configuration is not locked. Device is not ready. 0x1 Device configuration is locked.	RW	0
7	DATA_ENDIAN	Data endian bit. Can be set by local host to select little- or big-endian format on data access (read or write). 0x0: Little-endian 0x1 Big-endian	RW	0
6	DMA_ENDIAN	DMA data endian bit. Can be set by local host to select little- or big-endian format on data access (read or write). 0x0 Little-endian 0x1: Big-endian	RW	0
5	Reserved	Reserved	RW	-
4	NAK_EN	NAK enable bit. Can be set by the local host to be signaled for a NAK transaction handshake response. Note: If the local host sets this bit, it must wait for a NAK interrupt before selecting TX endpoint to write TX data. 0x0: NAK is disabled. 0x1: NAK is enabled.	RW	0
3	AUTODEC_DIS	Autodecode process disabled. This functionality can be set to force all EP0 transactions (except the SET_ADDRESS transaction) to be handled by software 0x0: Autodecode process is deactivated 0x1: Autodecode process is activated	RW	0

Bits	Field Name	Description	Type	Reset
2	SELF_PWR	Self-powered bit. Indicates to the USB host whether the device is bus-powered or self-powered. This is needed for a GET_DEVICE_STATUS auto-decoded request. 0x0: Bus powered 0x1: Self powered	RW	0
1	SOFF_DIS	Shutoff disable bit 0x0: Power shutoff circuitry is enabled. 0x1: Power shutoff circuitry is disabled.	RW	0
0	PULLUP_EN	External pullup enable bit. Allows the device to disconnect itself from the USB bus, forcing the host to reset and reconfigure the device. This bit can be used to prevent from USB traffic when the device is not ready. 0x0: Pull-up is disabled. USB host cannot detect the device. In this mode, the 48-MHz USB clock is forced off. 0x1: Pull-up is enabled	RW	0

Table 23-18. SYSCON2

Address Offset	0x0E		
Physical Address	0xFFFF B00E	**Instance**	USBOT1
Description	System Configuration Register 2		
Type	RW 1toClr	**Associated Register**	DEVSTAT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									RMT_WKP	STALL_CMD	Reserved	DEV_CFG	CLR_CFG	Reserved	

Bits	Field Name	Description	Type	Reset
15:7	Reserved	Reserved	RW	0x_UnicodeEncodeError__UnicodeEncodeError__UnicodeEncodeError_
6	RMT_WKP	Set-only remote wake-up bit. This set-only bit when written with a 1 initiates the remote wakeup sequence even if DEVSTAT.R_WK_OK bit is not previously set to 1 by the USB host. Then, to generate a resume, the software must check itself the remote wakeup enable value before initiating any wakeup sequence. Always reads 0x0. 0x0: No action 0x1: Initiates remote wake-up sequence	RW 1toClr	0
5	STALL_CMD	Set-only stall command bit. Only concerns nonautodecoded requests on control endpoint (EP0). This is asserted in response to a USB command where either the command itself or its data is invalid. Asserting this bit forces the non-autodecoded command to complete with a STALL handshake. It has no effect for autodecoded requests. Always reads 0x0. 0x0: No action 0x1: Stall current USB command	RW 1toClr	0
4	Reserved	Reserved	R	-
3	DEV_CFG	USB device controller. Only concerns nonautodecoded requests on control endpoint (EP0).	RW 1toClr	0

Bits	Field Name	Description	Type	Reset
		This is asserted in response to a USB command where either the command itself or its data is invalid. Asserting this bit forces the non-autodecoded command to complete with a STALL handshake. It has no effect for autodecoded requests. Always reads 0x0. 0x0: No action 0x1: Allows DEVSTAT.CFG to be set		
2	CLR_CFG	Clear configured. If the local host receives a SET_CONFIGURATION with a configuration value of 0, and if device is configured it must write a 1 to this bit to inform the command decode that the device becomes deconfigured (moves to addressed state). Always reads 0x0. 0x0: No action 0x1: Allows DEVSTAT.CFG to be cleared	RW 1toClr	0
1:0	Reserved	Reserved	RW	0x-

Table 23-19. DEVSTAT

Address Offset	0x10		
Physical Address	0xFFFF B010	**Instance**	USBOT1
Description	Device Status Register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved		R_WK_OK	USB_RESET	SUS	CFG	ADD	DEF	ATT

Bits	Field Name	Description	Type	Reset
15:10	Reserved	Reserved	R	0x–
9:7	Reserved	Reserved	R	0x0
6	R_WK_OK	Remote wake-up granted bit. Automatically set and cleared when the core receives a valid set/clear device feature request from the USB host. It returns a 1 when the USB host grants the function the ability to assert remote wakeup. Read 0x0: Not significant Read 0x1: Remote wakeup granted	R	0
5	USB_RESET	USB reset signaling is active. A valid USB reset has for effects to reset all the endpoint FIFOS, all the other control registers bits except SYSCON1.CFG_LOCK and associated configuration (registers 0x20 to 0x3F), IRQ_EN.DS_CHG_IE and IRQ_SRC.DS_Chg and for DEVSTAT forces the device to the default state. This bit is cleared at end of reset. Read 0x0: Device is not being reset by USB host. Read 0x1: Device is being reset by USB host.	R	0
4	SUS	Suspended state bit. Device is, at a minimum, attached to the USB and is powered, is reset by the USB host, and does not see bus activity for 5 ms. It can also have a unique address and be configured for use. However, because the device is suspended, the host might not use the device function. Read 0x0: Not suspended Read 0x1: Suspended	R	0

Bits	Field Name	Description	Type	Reset
3	CFG	Configured state bit. Device is attached to the USB, is powered, is reset, has a unique address, and is configured. The host can now use the function provided by the device. Returns 1 when the USB device is configured after a set SYSCON2.DEV_CFG=1. Remains set to 1 until the device becomes deconfigured. Cleared when the core receives a valid SET_CONFIGURATION request and the local host sets SYSCON2.CLR_CFG bit. While not set to 1 any transaction, which are not for Control EP0, are ignored. A GET_ENDPOINT_STATUS to a noncontrol endpoint is stalled. Read 0x0: Not configured Read 0x1: Configured	R	0
2	ADD	Addressed state bit. Device is attached to the USB, powered, is reset, and a unique device address is assigned. Returns 1 after a SET_ADDRESS standard request. Remains set to 1 until the device becomes de-addressed. Read 0x0: Not addressed Read 0x1: Addressed	R	0
1	DEF	Default state bit. Returns 1 when the USB device is attached to the USB and powered and is reset. Remains set to 1 until the device becomes depowered. Device moves to default state as soon as the USB reset is effective. Read 0x0: Not in default Read 0x1: Default	R	0
0	ATT	Attached state bit. Returns 1 when the device is attached to the USB and powered. Remains set to 1 until the device becomes depowered. Read 0x0: Not attached Read 0x1: Attached	R	0

Table 23-20. SOFREG

Address Offset	0x12		
Physical Address	0xFFFF B012	**Instance**	USBOT1
Description	Start of Frame Register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			FT_LOCK	TS_0K	TS										

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	R	0x0
12	FT_LOCK	Frame timer locked bit. The USB host sends out an SOF packet every millisecond. When an SOF packet is not received by the device due to a bus error, a local SOF is generated, used for an ISO FIFO switch. When the core receives two valid SOFs separated by a TF (time frame), it sets SOF.FT_LOCK to 1, only if TF is in a frame interval IF allowed by the Universal Serial Bus Specification Revision 1.1 (IF = [11964:12036] USB bit time). If the TF is out of this interval, SOF.FT_LOCK value remains 0, and a local SOF is generated by the core. When SOF.FT_LOCK is set, and the frame timer is locked to the timing TF, a local SOF is generated, if no valid SOF is received in an interval of TF since the last valid SOF. The SOF.FT_LOCK bit is cleared if a valid SOF is received out of the interval IF. If the core receives a valid SOF in this interval, the frame timer locks to the new frame time. If the core does not receive a valid SOF, the frame timer remains locked to TF. When SOF.FT_LOCK clears, a local SOF is generated after 12036 USB bit time, if no valid SOF is received, and SOF.FT_LOCK remains 0. Note : Each time a valid SOF is received by the core out of an allowed interval IF, a local SOF is generated and ISO FIFO switch. Read 0x0: Frame timer is not locked.	R	0

Bits	Field Name	Description	Type	Reset
		Read 0x1: Frame timer is locked.		
11	TS_0K	Timestamp OK bit. This bit indicates that the timestamp in SOF.TS is valid for the current frame. Read 0x0: Timestamp is invalid. Read 0x1: Timestamp is valid. A valid SOF packet is received from the USB host.	R	0
10:0	TS	Timestamp number field .This field returns the timestamp from the last USB host valid SOF packet. The frame number is valid, if SOF.TS_OK is 1. For an SOF miss, this value is not updated and TS_OK is cleared.	R	0x000

Table 23-21. IRQ_EN

Address Offset	0x14		
Physical Address	0xFFFF B014	**Instance**	USBOT1
Description	Interrupt Enable Register (Enables all non-DMA interrupts)		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SOF_IE	Reserved	EPN_RX_IE	EPN_TX_IE	DS_CHG_IE	Reserved		EP0_IE

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	RW	0x–
7	SOF_IE	Start of frame interrupt enable. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
6	Reserved	Reserved	RW	-
5	EPN_RX_IE	Receive endpoint n interrupt enable (non-ISO)9 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
4	EPN_TX_IE	Transmit endpoint n interrupt enable (non-ISO) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
3	DS_CHG_IE	Device state changed interrupt enable 0x0: Interrupt disabled int 0x1: Interrupt enabled	RW	0
2:1	Reserved	Reserved	RW	0x-
0	EP0_IE	EP0 transactions interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0

Table 23-22. DMA_IRQ_EN

Address Offset	0x16		
Physical Address	0xFFFF B016	**Instance**	USBOT1
Description	DMA Interrupt Enable Register (enables all DMA interrupts)		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					TX2_DONE_IE	RX2_CNTT_IE	RX2_EOT_IE	Reserved	TX1_DONE_IE	RX1_CNT_IE	RX1_EOT_IE	Reserved	TX0_DONE_IE	RX0_CNT_IE	RX0_EOT_IE

Bits	Field Name	Description	Type	Reset
15:11	Reserved	Reserved	RW	0x–
10	TX2_DONE_IE	Transmit DMA channel 2 done interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
9	RX2_CNTT_IE	Receive DMA channel 2 transactions count interrupt enable. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
8	RX2_EOT_IE	Receive DMA channel 2 end of transfer interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
7	Reserved	Reserved	RW	-
6	TX1_DONE_IE	Transmit DMA channel 1 done interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
5	RX1_CNT_IE	Receive DMA channel 1 transactions count interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
4	RX1_EOT_IE	Receive DMA channel 1 end of transfer interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
3	Reserved	Reserved	RW	-
2	TX0_DONE_IE	Transmit DMA channel 0 done interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
1	RX0_CNT_IE	Receive DMA channel 0 transactions count interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0
0	RX0_EOT_IE	Receive DMA channel 0 end of transfer interrupt enable 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0

Table 23-23. IRQ_SRC

Address Offset	0x18		
Physical Address	0xFFFF B018	**Instance**	USBOT1
Description	Interrupt Source Register		
Type	RW 1toClr		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					TXN_DONE	RXN_CNT	RXN_EOT	SOF	Reserved	EPN_RX	EPN_TX	DS_CHG	SETUP	EP0_RX	EP0_TX

Bits	Field Name	Description	Type	Reset
15:11	Reserved	Reserved	RW	0x–
10	TXN_DONE	Transmit DMA channel n done interrupt flag (nonisochronous) bit. Only for nonisochronous DMA transfer. Never set for isochronous DMA transfer. Set automatically by the core when a transmit DMA channel completes the programmed transfer by servicing the last IN transaction from the USB host (TXDMAn.TXn_TSC (transfer size counter) equals 0 and the last IN transaction completes with an ACK). When asserted, the local host must read the DMAN_STAT register to identify the endpoint number for which the transfer completed. Note: the endpoint interrupt IRQ_SRC.EPn_TX is never set for the assigned endpoint to TX DMA channel *n.* Write 0x0: No action Write 0x1: Nonisochronous transmit DMA transfer for a channel is ended.	W 1toClr	0
9	RXN_CNT	Receive DMA channel n transactions count interrupt flag (nonisochronous) bit. Only for nonisochronous DMA transfer. Never set for isochronous DMA transfer and never set if the interrupt enable bit associated is not set. It means that no poll operation is possible on the DMA. Set automatically by the core during an active receive DMA transfer on each time RXDMAn.RXn_TC equals 0 after an OUT transaction with ACK status. This bit is set after the RX DMA data is read (end of DMA request). When this bit is asserted, the local host must read the DMAN_STAT register to identify the endpoint number for which the transfer completed. Note: An RXn_CNT IT is asserted also if RXDMAn.RXn_STOP is set. In this case, both IRQ_SRC.RXn_EOT and IRQ_SRC.RXn_CNT are asserted. Write 0x0: No action Write 0x1: Nonisochronous receive DMA transfer for a channel reached the transactions count level.	W 1toClr	0
8	RXN_EOT	Receive DMA channel n end of transfer interrupt flag (nonisochronous) bit. Only for nonisochronous DMA transfer. Never set for isochronous DMA transfer and never set if the interrupt enable bit associated is not set, which means that no poll operation is possible on the DMA. Set automatically by the core when a receive DMA channel detected an EOT packet during the last OUT transaction from the USB host. This bit is set after RX DMA data is read (end of DMA request). When this happens, the DMA assigned endpoint FIFO is kept disabled (STAT_FLG.FIFO_En=0) to avoid receiving a new packet data from the USB host. The local host can grant again a DMA transfer to the same endpoint by enabling the FIFO again (STAT_FLG.FIFO_EN=1). An EOT is detected when the core receives a data packet the size of which is less than the configured endpoint FIFO Size (or empty), or when RXDMAn.RXn_TC equals 0 after an OUT transaction with ACK status, and RXDMAn.RXn_STOP bit is set. When this bit is asserted, the local host must read DMAN_STAT.DMAn_RX_IT_SRC to identify the endpoint number for which the transfer completes, and DMAN_STAT.DMAn_RX_SB to be informed of an odd number of bytes received during the last transaction (useful for 16-bit read access from the DATA_DMA register). Note: The endpoint interrupt IRQ_SRC.EPn_RX is never set for the assigned endpoint to the RX DMA channel. Write 0x0: No action Write 0x1: Nonisochronous receive DMA transfer for a channel is ended.	W 1toClr	0
7	SOF	Start of frame interrupt flag bit. Every 1 ms, the USB host outputs a nSOF packet to the functions. The SOF bit reflects when a new SOF is received. Write 0x0: No action Write 0x1: SOF packet received (or internal SOF)	W 1toClr	0
6	Reserved	Reserved	RW	-

Bits	Field Name	Description	Type	Reset
5	EPN_RX	EPn OUT transactions interrupt flag bit. Only concerns nonisochronous endpoints. Automatically set by the core when a handshake sequence occurs for an OUT transaction to an interrupt of bulk endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). The local host must read EPN_STAT register to identify the endpoint causing the interrupt. Write 0x0: No action Write 0x1: OUT transaction detected on an endpoint	W 1toClr	0
4	EPN_TX	EPn IN transactions interrupt flag bit. Only concerns nonisochronous endpoints. Automatically set by the core when a handshake sequence occurs for an IN transaction to an interrupt of bulk endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). The local host must read EPN_STAT register to identify the endpoint causing the interrupt. Write 0x0: No action Write 0x1: IN transaction detected on an endpoint value after MPU or USB reset is low.	W 1toClr	0
3	DS_CHG	Device state changed interrupt flag bit. Automatically set by the core when the state of the device changes (when the core modifies any of the bits present in the DEVSTAT register). When cleared, the background DEVSTAT register moves into the foreground position. Write 0x0: No action Write 0x1: Device state change detected	W 1toClr	0
2	SETUP	Setup transaction interrupt flag bit. Automatically set by the core when a valid setup transaction completes on the control endpoint for a nonautodecoded control request. Cleared automatically by the core when the local host sets EP_NUM.SETUP_SEL bit when reading setup data. Write 0x0: No action Write 0x1: Valid setup transaction occurred on endpoint 0.	W 1toClr	0
1	EP0_RX	EP0 OUT transactions interrupt flag bit. Set automatically by the core when a handshake sequence occurs for a non-autodecoded OUT transaction to the control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). Write 0x0: No action Write 0x1: OUT transaction on EP0	W 1toClr	0
0	EP0_TX	EP0 IN transactions interrupt flag bit. Set automatically by the core when a handshake sequence occurs for a non-autodecoded IN transaction to the control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). Write 0x0: No action Write 0x1: IN transaction on EP0	W 1toClr	0

Table 23-24. EPN_STAT

Address Offset	0x1A		
Physical Address	0xFFFF B01A	**Instance**	USBOT1
Description	Nonisochronous Endpoint Interrupt Status Register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				EPN_RX_IT_SRC				Reserved				EPN_TX_IT_SRC			

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Reserved	R	0x-
11:8	EPN_RX_IT_SRC	Receive endpoint interrupt source (nonisochronous) bit. Only concerns nonisochronous endpoints. When IRQ_SRC.EPn_RX flag is set, the endpoint causing the interrupt condition is encoded in these four register bits. When IRQ_SRC.EPn_RX flag is cleared, the four bits read as 0. Read 0x0: No receive endpoint interrupt is pending. Read 0x1: EP1 Read 0x2: EP2	R	0x0

Bits	Field Name	Description	Type	Reset
		Read 0x3: EP3		
		Read 0x4: EP4		
		Read 0x5: EP5		
		Read 0x6: EP6		
		Read 0x7: EP7		
		Read 0x8: EP8		
		Read 0x9: EP9		
		Read 0xA: EP10		
		Read 0xB: EP11		
		Read 0xC: EP12		
		Read 0xD: EP13		
		Read 0xE: EP14		
		Read 0xF: EP15		
7:4	Reserved	Reserved	R	0x-
3:0	EPN_TX_IT_SRC	Transmit endpoint interrupt source (nonisochronous) bit. Only concerns nonisochronous endpoints. When the IRQ_SRC.EPn_TX flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_TX flag is cleared, the four bits read 0. Read 0x0: No transmit endpoint interrupt is pending. Read 0x1: EP1 Read 0x2: EP2 Read 0x3: EP3 Read 0x4: EP4 Read 0x5: EP5 Read 0x6: EP6 Read 0x7: EP7 Read 0x8: EP8 Read 0x9: EP9 Read 0xA: EP10 Read 0xB: EP11 Read 0xC: EP12 Read 0xD: EP13 Read 0xE: EP14 Read 0xF: EP15	R	0x0

Table 23-25. DMAN_STAT

Address Offset	0x1C		
Physical Address	0xFFFF B01C	**Instance**	USBOT1
Description	Nonisochronous DMA Interrupt Status Register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DMAN_RX_SB	DMAN_RX_IT_SRC				Reserved				DMAN_TX_IT_SRC			

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	R	0x-
12	DMAN_RX_SB	DMA Receive Single Byte. Concerns nonisochronous endpoints (isochronous endpoints receive a constant number of bytes). Set when an IRQ_SRC.RXn_EOT interrupt is asserted and the core receives an odd number of bytes during the last transaction. This is used to know the exact number of bytes received in case of a 16-bit read access from the DATA_DMA register. When the IRQ_SRC.RXn_EOT flag is cleared, the bit reads 0. Read 0x0: No EOT DMA interrupt is pending or the core received an even number of bytes during the last transaction. Read 0x1: EOT DMA interrupt is pending and the core received an even number of bytes during the last transaction.	R	0
11:8	DMAN_RX_IT_SRC	DMA Receive Interrupt Source. Only concerns nonisochronous endpoints. When the IRQ_SRC.EPn_RX flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When the IRQ_ SRC.EPn_RX flag is cleared, the four bits read as 0. Read 0x0: No receive DMA interrupt is pending. Read 0x1: EP1 Read 0x2: EP2 Read 0x3: EP3 Read 0x4: EP4 Read 0x5: EP5 Read 0x6: EP6 Read 0x7: EP7 Read 0x8: EP8 Read 0x9: EP9 Read 0xA: EP10 Read 0xB: EP11 Read 0xC: EP12 Read 0xD: EP13 Read 0xE: EP14 Read 0xF: EP15	R	0x0
7:4	Reserved	Reserved	R	0x-
3:0	DMAN_TX_IT_SRC	DMA Transmit Interrupt Source. Only concerns nonisochronous endpoints. When the IRQ_SRC.EPn_TX flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When the IRQ_ SRC.EPn_TX flag is cleared, the four bits read 0. Read 0x0: No transmit DMA interrupt is pending Read 0x1: EP1 Read 0x2: EP2 Read 0x3: EP3 Read 0x4: EP4 Read 0x5: EP5 Read 0x6: EP6 Read 0x7: EP7 Read 0x8: EP8 Read 0x9: EP9 Read 0xA: EP10 Read 0xB: EP11 Read 0xC: EP12 Read 0xD: EP13 Read 0xE: EP14 Read 0xF: EP15	R	0x0

Table 23-26. RXDMA_CFG

Address Offset	0x20		
Physical Address	0xFFFF B020	**Instance**	USBOT1
Description	DMA Receive Channels Configuration Register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			RX_REQ	RXDMA2_EP				RXDMA1_EP				RXDMA0_EP			

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	RW	0x_UnicodeEncodeError_
12	RX_REQ	The RX DMA request active. Allows the RXDMA request to be configurable level or pulse sensitive. When pulse sensitive, the request is active during 2 cycles. 0x0: RX DMA request active level 0x1: RX DMA request active pulse	RW	0
11:8	RXDMA2_EP	Receive endpoint number for DMA channel 2. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint. 0x0: Receive DMA channel 2 is deactivated. 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6 0x7: EP7 0x8: EP8 0x9: EP9 0xA: EP10 0xB: EP11 0xC: EP12 0xD: EP13 0xE: EP14 0xF: EP15	RW	0x0
7:4	RXDMA1_EP	Receive endpoint number for DMA channel 1. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint. 0x0: Receive DMA channel 1 is deactivated. 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6 0x7: EP7	RW	0x0

Bits	Field Name	Description	Type	Reset
		0x8: EP8 0x9: EP9 0xA: EP10 0xB: EP11 0xC: EP12 0xD: EP13 0xE: EP14 0xF: EP15		
3:0	RXDMA0_EP	Receive endpoint number for DMA channel 0.The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint. 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6 0x7: EP7 0x8: EP8 0x9: EP9 0xA: EP10 0xB: EP11 0xC: EP12 0xD: EP13 0xE: EP14 0xF: EP15	RW	0x0

Table 23-27. TXDMA_CFG

Address Offset	0x22		
Physical Address	0xFFFF B022	**Instance**	USBOT1
Description	DMA Transmit Channels Configuration Register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			TX_REQ	TXDMA2_EP				TXDMA1_EP				TXDMA0_EP			

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	R	0x-
12	TX_REQ	The TX DMA request active. Allows the TXDMA request to be configurable level or pulse sensitive. When pulse sensitive, the request is active during 2 cycles. 0x0: TX DMA request active level 0x1: TX DMA request active pulse	RW	0
11:8	TXDMA2_EP	Transmit endpoint number for DMA channel 2. The endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is de_UnicodeEncodeError_activated. Any other value automatically enables transmit DMA transfer for the selected endpoint.	RW	0x0

Bits	Field Name	Description	Type	Reset
		0x0: Transmit DMA channel 2 is deactivated. 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6 0x7: EP7 0x8: EP8 0x9: EP9 0xA: EP10 0xB: EP11 0xC: EP12 0xD: EP13 0xE: EP14 0xF: EP15 0xF: EP15		
7:4	TXDMA1_EP	Transmit endpoint number for DMA channel 1. The endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0x0: Transmit DMA channel 1 is deactivated. 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6 0x7: EP7 0x8: EP8 0x9: EP9 0xA: EP10 0xB: EP11 0xC: EP12 0xD: EP13 0xE: EP14 0xF: EP15 0xF: EP15	RW	0x0
3:0	TXDMA0_EP	Transmit endpoint number for DMA channel 0. The endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0x0: Transmit DMA channel 0 is deactivated. 0x1: EP1 0x2: EP2 0x3: EP3 0x4: EP4 0x5: EP5 0x6: EP6 0x7: EP7	RW	0x0

Bits	Field Name	Description	Type	Reset
		0x8: EP8		
		0x9: EP9		
		0xA: EP10		
		0xB: EP11		
		0xC: EP12		
		0xD: EP13		
		0xE: EP14		
		0xF: EP15		
		0xF: EP15		

Table 23-28. DATA_DMA

Address Offset	0x24		
Physical Address	0xFFFF B024	**Instance**	USBOT1
Description	DMA FIFO Data Register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_DMA															

Bits	Field Name	Description	Type	Reset
15:0	DATA_DMA	DMA FIFO data. **Warning:** It is possible for both an RX DMA request and a TX DMA request to be active at the same time. In this case, the main DMA controller engine can access both transmit endpoint and receive endpoint FIFO. A read access to DATA_DMA register affects the endpoint that causes the RX DMA request to be active, and a write access affects the endpoint that causes the TX DMA request to be active. **Caution:** The local host must not access this register directly; however, there is no hardware mechanism that protects from this access. No access into this register must be done out of DMA requests handling.	RW	0x_UnicodeEncodeError__UnicodeEncodeError__UnicodeEncodeError__UnicodeEncodeError_

Table 23-29. TXDMA0 _UnicodeEncodeError_ TXDMA2

Address Offset	0x28_UnicodeEncodeError_0x2C in 0x2 byte increments		
Physical Address	0xFFFF B028_UnicodeEncodeError_0 xFFFF B02C	**Instance**	USBOT1
Description	Transmit DMA Control Register n		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXN_EOT	TXN_START	Reserved						TXN_TSC							

Bits	Field Name	Description	Type	Reset
15	TXN_EOT	Transmit DMA channel n end of transfer. Can be either 0 or 1 for BULK DMA transfer. When set by the local host, it signals to the core that the transfer size set in TXDMAn.TXn_TSC is in bytes. A TX Done Interrupt (IRQ_SRC.TXn_Done) will be asserted with the last IN transaction. Note that if the number of bytes set in TXDMAn.TXn_TSC is a multiple of the endpoint s buffer Size, the TX Done interrupt will be asserted only after an IN transaction with an empty data packet. When cleared, the transfer size set in TXn_TSC is in full buffer size for the endpoint selected (BULK only). A TX Done interrupt will be asserted when the last buffer is sent with the last IN transaction. This mode is to be used for a partial bulk transfer of a large file exceeding 1023 bytes. 0x0: DMA transfer size is in buffers. 0x1: DMA transfer size is in bytes.	RW	0
14	TXN_START	Transmit DMA channel n start. Set by the local host to tell the device that the main DMA system is ready to transmit the number of bytes or buffers. Once set, the DMA transfer cannot be interrupted, except if the local host clears endpoint in TXDMA_CFG register. The IRQ_SRC.TXn_DONE interrupt bit will be asserted when the DMA transfer ends. Always reads 0x0. 0x0: No action 0x1: DMA transfer start	RW	0
13:10	Reserved	Reserved	RW	0x-
9:0	TXN_TSC	Transmit DMA channel n transfer size counter. The binary encoded value from 0 to 1023, written by the local host into this register, corresponds to the number of bytes or number of buffer transfers (function of TXDMAn.TXn_EOT) which will be transmitted by the Transmit DMA channel n. When Read, the register reflects the number of bytes/buffers the USB device has still to transmit. This Read mode is only provided for software debug purpose. Caution: For an ISO transfer, you must verify that the set value does not exceed the ISO FIFO size for the endpoint. There is no hardware mechanism to protect from this situation. If it happens, results are unpredictable. For bulk transfer, when TXDMAn.TXn_EOT = 0, a set value of TXDMAn.TXn_TSC = 0 means 1024 buffers and not 0. The counter then operates as follows: 000, 3FF, 3FE, &001, 000, stop. When TXDMAn.TXn_EOT = 1, a set value of TXDMAn.TXn_TSC = 0 a NULL packet is sent in response to the next IN token.	RW	0x000

Table 23-30. RXDMA0 _UnicodeEncodeError_ RXDMA2

Address Offset	0x30_UnicodeEncodeError_0x34 in 0x2 byte increments		
Physical Address	0xFFFF B030_UnicodeEncodeError_0 xFFFF B034	**Instance**	USBOT1
Description	Receive DMA Control Register n		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXN_STOP	Reserved							RXN_TC							

Bits	Field Name	Description	Type	Reset
15	RXN_STOP	Receive DMA channel n transfer stop. When set, an RXn_EOT interrupt is asserted to the local host after n OUT transactions, where n is the encoded binary value + 1 programmed into RXDMAn.RXn_TC field. This register is used when no smaller than a buffer size packet is received at an EOT, and the local host expects a given amount of data for the transfer. **Caution:** At end of transfer, the DMA channel is disabled and all OUT transactions received to the assigned endpoint are sent NAK by the core. The local host must set CTRL.SET_FIFO_En for the endpoint to re-enable the channel.	RW	0
14:8	Reserved	Reserved	RW	0x–
7:0	RXN_TC	Receive DMA channel n transactions count. The local host can ask for an interrupt each n OUT transaction, where n is the encoded binary value + 1 programmed into RXDMAn.RXn_TC field. This register must be programmed to the desired transactions watermark limit before enabling the DMA transfer for the receive DMA channel n. **Caution**: A reached watermark does not disable an active DMA transfer if RXDMAn.RXn_Stop is not set. If RXDMAn.RXn_Stop is set for the transfer, both RXn_CNT and RXn_EOT interrupts are asserted. A read to that register returns the number of transactions remaining before the IRQ_SRC.RXn_CNT interrupt flag is asserted. This read mode is provided only for software debug purposes.	RW	0x00

Table 23-31. EP0

Address Offset	0x40		
Physical Address	0xFFFF B040	**Instance**	USBOT1
Description	Endpoint 0 Configuration Register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				Reserved				EP0_PTR							
		EP0_SIZE													

Bits	Field Name	Description	Type	Reset
15:14	Reserved		RW	0x_UnicodeEncodeError_
13:12	EP0_SIZE	Endpoint 0 FIFO size. Contains the endpoint 0 FIFO Size value, and must match the value sent by the local host to the USB host during the GET_DEVICE_DESCRIPTOR request preceding configuration phase. Status Flags (STAT_FLG.NON_ISO_FIFO_EMPTY, STAT_FLG.NON_ISO_FIFO_FULL), overrun, and underrun conditions are based on this value for all IN and OUT transactions to endpoint 0. Note: The local host must fill this field before setting SYSCON1.CFG_LOCK. 0x0: 8 bytes 0x1: 16 bytes 0x2: 32 bytes 0x3: 64 bytes	RW	0x_UnicodeEncodeError_
11	Reserved	Reserved	RW	-
10:0	EP0_PTR	Endpoint 0 pointer. Contains the address of the endpoint 0 pointer. Value 0x000 is forbidden (reserved for Setup FIFO).	RW	0x—

Bits	Field Name	Description	Type	Reset
		Note: The pointer value should be set to a value < 0xFF because the memory size is 2K bytes and a pointer coded value = 0xFF corresponds to 2040 bytes. Addressing upper bytes issues in memory overlap.		

Table 23-32. EP_RX1 _UnicodeEncodeError_ EP_RX15

Address Offset	0x42_UnicodeEncodeError_0x5E in 0x2 byte increments		
Physical Address	0xFFFF B042_UnicodeEncodeError_0 xFFFF B05E	**Instance**	USBOT1
Description	Receive Endpoint n Configuration Register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPN_RX_VALID	EPN_RX_SIZE_DB	EPN_RX_SIZE		EPN_RX_ISO	EPN_RX_PTR										

Bits	Field Name	Description	Type	Reset
15	EPN_RX_VALID	Receive endpoint *n* valid bit. Must be set by the local host to allow receive endpoint *n* to be used for USB transfers as part of the device configuration. If not set, all transactions to the endpoint are ignored by the core. 0x0: Receive endpoint *n* does not exist for this configuration. 0x1: Receive endpoint *n* is part of the device configuration.	RW	-
14	EPN_RX_SIZE_DB	Receive nonisochronous (or isochronous) endpoint *n* double buffer (DB). Must be set by the local host to allow double-buffering for receive nonisochronous endpoint n. This reduces the number of transactions resulting in a NAK handshake. Note: This bit is for nonisochronous endpoints only. For isochronous endpoints, which are always double buffered, this bit is endpoint s Size MSB. 0x0: No double buffer for nonisochronous receive endpoint *n.* 0x1: Double buffer used for nonisochronous receive endpoint *n.*	RW	-
13:12	EPN_RX_SIZE	Receive endpoint n size field. Contains the endpoint *n* FIFO Size value. Status Flags (STAT_FLG.NON_ISO_FIFO_EMPTY, STAT_FLG.NON_ISO_FIFO_FULL, STAT_FLG. overrun, and underrun conditions are based on this value for all OUT transactions to endpoint *n.* Note: This paragraph includes a description of EPn_RX.[14] bit for isochronous endpoints. **Warning:** For isochronous endpoints, a size of 1023 bytes takes the whole memory and then cannot be programmed with a 2K bytes USB W2FC. 0x0: 8 bytes in nonisochronous mode or isochronous mode and EPn_RX_SIZE_DB = 0 else if isochronous mode and EPn_RX_SIZE_DB=1 128 bytes 0x1: 16 bytes in nonisochronous mode or isochronous mode and EPn_RX_SIZE_DB = 0 else if isochronous mode and EPn_RX_SIZE_DB=1 256 bytes 0x2: 32 bytes in nonisochronous mode or isochronous mode and EPn_RX_SIZE_DB = 0 else if isochronous mode and EPn_RX_SIZE_DB=1 512 bytes	RW	0x-

Bits	Field Name	Description	Type	Reset
		0x3: 64 bytes in nonisochronous mode or isochronous mode and EPn_RX_SIZE_DB = 0; else if isochronous mode and EPn_RX_SIZE_DB=1 1023 bytes		
11	EPN_RX_ISO	Receive isochronous endpoint n field. Must be set if the Receive endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from interrupt). 0x0: Receive endpoint n type is bulk or interrupt. 0x1: Receive endpoint n type is isochronous.	RW	-
10:0	EPN_RX_PTR	Receive endpoint n pointer field. Contains the address of the receive endpoint *n* pointer. Value 0x000 is forbidden (reserved for setup FIFO). Caution: For isochronous endpoints or nonisochronous endpoints that allow double-buffering, 2*RX Buffer Size must be reserved for Ping-Pong. Note: pointer value should be set to a value < 0xFF because the memory size is 2K bytes and a pointer coded value = 0xFF correspond to 2040 bytes. Addressing upper bytes issues in memory overlap.	RW	0x—

Table 23-33. EP_TX1 _UnicodeEncodeError_ EP_TX15

Address Offset	0x62_UnicodeEncodeError_0x7E in 0x2 byte increments		
Physical Address	0xFFFF B062_UnicodeEncodeError_0 xFFFF B07E	**Instance**	Transmit Endpoint n Configuration
Description	Transmit Endpoint n Configuration Register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPN_TX_VALID	EPN_TX_SIZE_DB	EPN_TX_SIZE		EPN_TX_ISO	EPN_TX_PTR										

Bits	Field Name	Description	Type	Reset
15	EPN_TX_VALID	The transmit endpoint *n* valid bit. Must be set by the local host to allow transmit endpoint *n* to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core. 0x0: Transmit endpoint *n* does not exist for this configuration. 0x1: Transmit endpoint *n* is part of the device configuration.	RW	-
14	EPN_TX_SIZE_DB	Transmit nonisochronous (or isochronous) endpoint n double buffer. Must be set by the local host to allow double-buffering for transmit nonisochronous endpoint n, when used in a DMA transfer. This reduces the number of transactions resulting in a NAK handshake. Note: This bit is for only nonisochronous endpoints used in DMA mode. For isochronous endpoints, which are always double-buffered, this bit is endpoint s Size MSB. For nonisochronous endpoints not used in a DMA transfer, double-buffering is not provided. 0x0: No double buffer for nonisochronous transmit endpoint *n.* 0x1: Double buffer used for nonisochronous transmit endpoint *n.*	RW	-
13:12	EPN_TX_SIZE	Transmit endpoint n size. Contains the endpoint n FIFO Size value. Status flags (FIFO_Empty, FIFO_Full) and underrun condition are based on this value for all IN transactions to endpoint n.	RW	0x-

Bits	Field Name	Description	Type	Reset
		Note: EPn_TX.[14] bit description applies only to isochronous endpoints. 0x0: bytes in nonisochronous mode or isochronous mode and EPn_TX_SIZE_DB = 0; else, if isochronous mode and ETXn_RX_SIZE_ DB=1 128 bytes 0x1: 16 bytes in nonisochronous mode or isochronous mode and EPn_TX_SIZE_DB = 0; else, if isochronous mode and ETXn_RX_SIZE_ DB=1 256 bytes 0x2: 32 bytes in nonisochronous mode or isochronous mode and EPn_TX_SIZE_DB = 0; else, if isochronous mode and ETXn_RX_SIZE_ DB=1 512 bytes 0x3: 64 bytes in nonisochronous mode or isochronous mode and EPn_TX_SIZE_DB = 0; else, if isochronous mode and ETXn_RX_SIZE_ DB=1 1023 bytes		
11	EPN_TX_ISO	Transmit isochronous endpoint n field. Must be set if the transmit endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from Interrupt). 0x0: Transmit endpoint n type is bulk or interrupt. 0x1: Transmit endpoint n type is isochronous.	RW	-
10:0	EPN_TX_PTR	Transmit endpoint n pointer. Contains the address of the transmit endpoint n pointer. **Caution**: For isochronous endpoints or for nonisochronous endpoints that allow double-buffering, 2*TX Buffer Size must be reserved for Ping-Pong. Note: Pointer value should be set to a value < 0xFF because the memory size is 2K bytes and a pointer coded value = 0xFF corresponds to 2040 bytes. Addressing upper bytes issues in memory overlap. 0x0: address = BASE (forbidden). 0x1: address = BASE + 8 bytes 0x2: address = BASE + 16 bytes 0x3: address = BASE + 24 bytes. 0xFF: address = BASE + 2040 bytes	RW	0x—

Universal Subscriber Identity Module

This chapter discusses the universal subscriber identity module (USIM) for the LOCOSTO device.

Topic **Page**

24.1 USIM Controller Overview

The new possibilities offered by the 3rd generation mobile communication systems are leading to a system migration of the existing single application processor subscriber identity module (SIM) to a multi-application platform for the future universal subscriber identity module (USIM)

To support numerous applications for value-added services (VAS), the Smart Card™ integrates a true card operating system (COS) with increased memory capacity and computing capability. Thus, the card copes with the requirements of remote application management in terms of dynamic allocation and adaptation of data. Moreover, the Smart Card should implement security features (crypto-processors, key generation, access control) as an element of the security frameworks for mobile commerce.

The USIM interface supports many simultaneous Smart Card applications such as SIM (GSM), USIM (3GPP), banking (EMV), and loyalty application. This module implements the hardware interfaces to the Smart Card and a sequencer that manages the transmission protocols T0 and T1, defined in the ISO7816.3 standard, thus freeing the M-controller of real-time constraints.

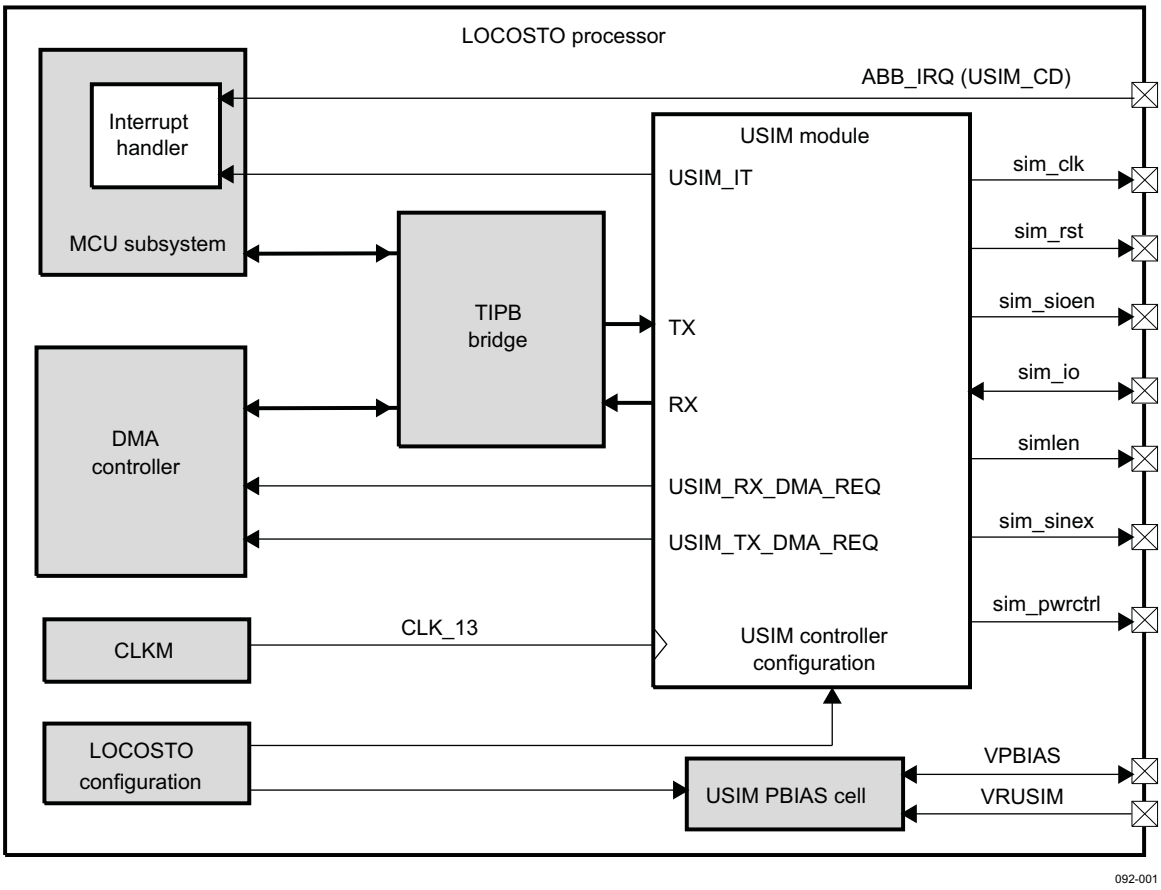
In the LOCOSTO chip-set environment, the USIM interface is restricted for controlling the GSM SIM-card. Thus, in this document only this feature is described.

The USIM interface handles the following:

- The activation sequence (including hardware interpretation of coding convention and timing management for the answer-to-reset [ATR] sequence)
- T = 0 transmit and receive protocol with parity error and timer management
- T = 1 transmit and receive protocol with error (parity/EDC) and timer management
- The warm reset sequence
- The clock-stop sequence
- The deactivation sequence

Figure 24-1 is a general overview of the USIM controller.

Figure 24-1. USIM Controller Overview



The USIM controller is clocked from CLK_13. It interfaces with the following:

- The Smart Card through the ISO Smart Card interface
- The microprocessor unit (MPU) through a shared TI peripheral bus (TIPB)-compliant parallel data interface

The operation of the Smart Card interface is under the control of the MPU.

Data between the USIM and the MPU are 16 bits wide, and data transfer can be executed by two methods:

- Direct memory access (DMA): DMA controller requests
- TIPB: MPU interrupts

The USIM controller supports two DMA request lines for RX and TX, includes two FIFOs (RX_FIFO and TX_FIFO), which are both 16 bytes deep, and generates interrupts (USIM_IT) to the MPU interrupt handler for time-out, error, and status management.

Note: The USIM controller hardware has no input for card detection/extraction (USIM_CD signal). Card plug/unplug information must be provided by an external companion chip. Thus, the LOCOSTO device external pad ABB_IRQ is connected to the MPU interrupt handler through EXTERNAL_IRQ for USIM card presence detection (USIM_CD). Texas Instruments provides such a global solution with the TWL3031 power IC. For information about the TWL3031 power IC, contact your TI representative.

In addition, the LOCOSTO processor external pad ABB_IRQ is connected to the MPU interrupt handler for USIM card presence detection (USIM_CD).

USIM module external pads are controlled by the LOCOSTO configuration to provide the following:

- USIM controller muxed pin configuration
- USIM controller I/O power-down mode
- USIM PBIAS cell configuration

An embedded PBIAS cell provides the bias voltage to the USIM controller extended drain I/Os.

24.1.1 USIM Controller Environment

The exchange of data between the USIM card and the USIM controller is based on a half-duplex asynchronous serial line.

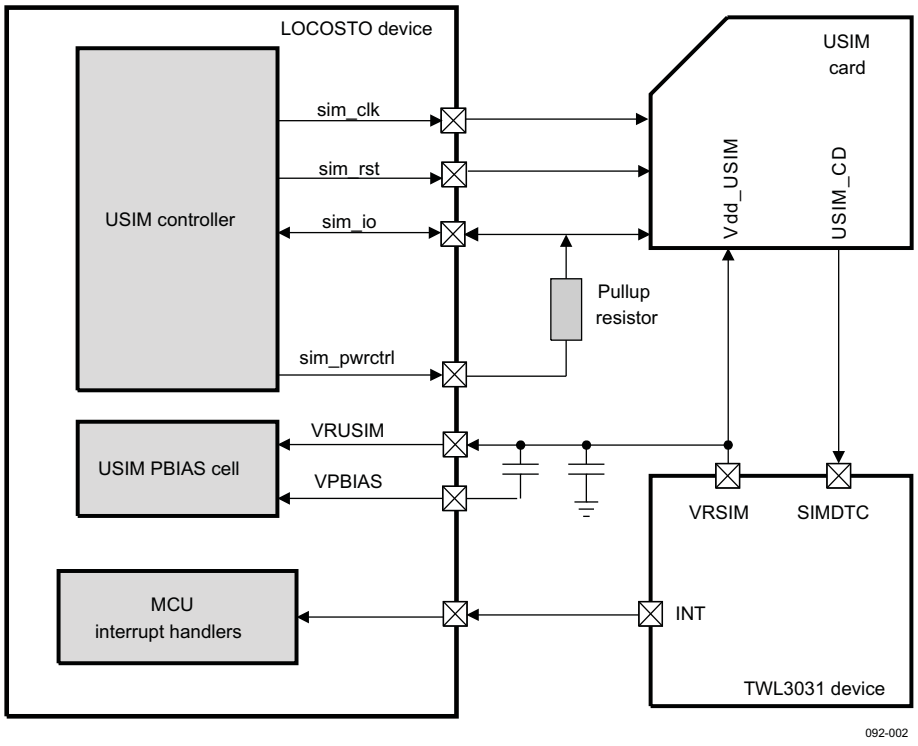
An external power device TWL3031 IC must be provided to:

- Ensure THE power supply of the USIM positive-bias (PBIAS) cell dedicated to LOCOSTO processor USIM applicative pins
- Indicate to the USIM controller any insertion or extraction of the card by using an external interrupt request to MPU interrupt handler

An external pullup resistor is also needed to accurately time-control the voltage-level transition of the SIM_I/O line.

Figure 24-2 shows the external connections between USIM card, LOCOSTO processor, and TWL3031 power IC.

Figure 24-2. USIM Controller External Connections



24.1.2 Main Features

24.1.2.1 Smart Card Features

The Smart Card features are as follows:

- Power supply Vcc (VRUSIM) equal to either:
  - 3 V ± 10% (class B)
  - 1.8 V ± 10% (class C)

- Icc current supply:
  - Max 6 mA at Vcc = 3.0 V (class B and C)
  - Max 4 mA at Vcc = 1.8 V (class C)
- Programming voltage VPP connected to Vcc (power supply)
- External clock only
- USIM card frequency in the range [1MHz–4MHz] for class B/C
- Clock duty-cycle of 45%_UnicodeEncodeError_55%
- Extra guard time between characters equal to 0 (for GSM card only)
- Transmit protocol T = 0 and T = 1 must be supported:
  Transmit T = 0 protocol refers to a half-duplex asynchronous transfer of characters; T = 1 refers to a half-duplex asynchronous transfer of blocks.

24.1.2.2 Smart Card Interface Features

24.1.2.2.1 General

- Vcc power supply:
  - 3 V (class B)
  - 1.8 V (class C)
- External clock frequency equals 13/4 MHz or 13/8 MHz during ATR sequence.
- Mandatory transmission factor values: Di/Fi = 1/372, 8/512, and 16/512
- Start-bit detection logic activated at 11 ETU (elementary time unit) (T = 0) or at 10 ETU (T = 1) (feature strengthened by comparison to ISO: minimum 12 ETU for T = 0)
- Automated or manual card activation and deactivation sequencing management
- Oversampling frequency in reception equal to F_{ETU}*8
- Card presence detection signal with 32-kHz based filtering logic
- Warm reset generation mandatory (with timers)
- External clock-stop mode:
  - External clock autodisabling at least 1860 + 2 ETU clock cycles after last RX character
  - TX character at least 744 clock cycles after re-enabling external clock

24.1.2.2.2 ATR Procedure

- Hardware error handling of character parity check
- Hardware error handling of ATR time-out
- Hardware identification of character coding convention (direct/inverse)

24.1.2.2.3 T = 0 Protocol

- Hardware error handling of the work waiting time (WWT) timer time-out
- Delayed release of I/O data line after successful completion of last byte transmission

24.1.2.2.4 T = 1 Protocol

- Transmit protocol (T) of type 1 (TD1 character equal to xH01)
- Delayed release of I/O data line after successful completion of last byte transmission (asynchronous half-duplex block transmission protocol)
- Character waiting time (CWT) management (hardware timer based)
- Block waiting time (BWT) management (hardware timer based)
- Block guard time (BGT) management (hardware timer based)
- Need to keep parity checking at character level without acknowledgement (that is, no repeat)

24.1.3 Functional Features for Improved Performance

The increase of the card role in the 3rd generation VAS is leading to a related increase of the data flow between the card and the ME. A hardware management of the transmission protocol is the answer to the reduction of the consequent impact on the host CPU loading.

General

External clock frequency (F_{SCK}) equals 13/N MHz (with N = 4 or 8)

ATR Procedure

- Hardware identification of character coding convention (direct/inverse) from TS character with update for processing of subsequent characters (T0, Tai...etc). See *Coding Convention Protocol* in Section 24.1.5.3.2.4, *ATR Sequence Reception*.
- Update for processing of subsequent characters (T0, etc.)
- Software processing of PPS/PTS procedure
- Software identification of procedure bytes of PTS0 and PTS1 with update of protocol type T and transmission factor value Fi/Di

T = 1 Protocol

- Hardware management of block length (interpretation of LEN character)
- Hardware checking of EDC error code, if LRC is used
- Hardware error handling of the following:
  - LEN block length error (in RX mode)
  - Character parity check in complement to EDC error
  - CWT, BWT, and BGT timers time-out (with IT generation)

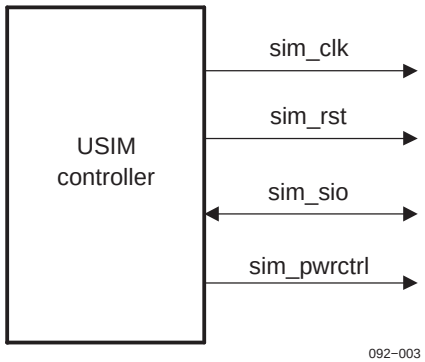
Note: If cyclic redundancy check (CRC) is used, error code checking must be done by software. Handling of printed circuit board (PCB) erroneous encoding must be done by software.

24.1.3.1 Signals and I/O Description

Basic USIM Controller Pins for Smart Card Driving

Figure 24-3 shows the signals interfacing to the Smart Card.

Figure 24-3. USIM Controller Interface Signals



24.1.3.2 USIM Controller Description

The USIM controller and the Smart Card are linked by a bidirectional asynchronous half-duplex interface.

The USIM controller provides the card the reference clock sim_clk from which the baud rate for character transmission is derived. One bidirectional serial line (sim_io) supports the exchange of data.

The activity of the Smart Card is under control of the interface that can switch on and off the clock sim_clk and the power supply of the card through the control signal sim_pwctrl. The interface drives the reset of the card through the dedicated pin sim_rst.

The MPU monitors the USIM through a parallel memory interface supporting a TIPB protocol.

Table 24-1 describes the USIM controller I/O.

Table 24-1. USIM Controller I/O Description

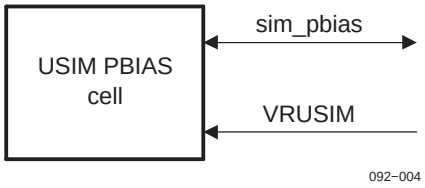
Signal Name	Definition	Type	Active Level	Reset Level
sim_clk	Smart Card clock	O	/ \	0
sim_sio	Smart Card interface data in out line	I/O	0 – 1	HZ
sim_rst	Smart Card reset	O	0	0
sim_pwrctrl	Smart Card power supply control	O	1	0

24.1.4 USIM PBIAS Cell Functional Interfaces

24.1.4.1 USIM PBIAS Cell Pins for Smart Card Power Supply

Figure 24-4 shows the signals interfacing to the Smart Card.

Figure 24-4. USIM PBIAS Cell Interface Signals



24.1.4.2 USIM PBIAS Cell Interface Description

The LOCOSTO USIM interface can be 1.8 V or 3 V. USIM_PBIAS must be connected to Vdd_USIM through a decoupling capacitor. The VSS_Pbias must be connected to the main VSS.

The SIM card power comes from the TWL3031 VRSIM LDO, and the SIM card detection pin is connected to the TWL3031 SIMDTC signal.

The sim_pbias pin can be used in output mode to monitor the PBIAS voltage generated by the USIM PBIAS cell. In input mode, the sim_pbias pin can be used to provide (externally to the LOCOSTO device) the PBIAS reference voltage to the extended drain I/Os when PBIAS is set in high-impedance mode.

Table 24-2. USIM PBIAS Cell I/O Description

Signal Name	I/O	Description	Reset Value
sim_pbias	I/O	Backup solution: PBIAS reference voltage provider	0
VRUSIM (VDDS3)	I	USIM extended drain I/O power supply	0

For more details on PBIAS cell and extended drain I/O, see Chapter 18, *Configuration*.

24.1.5 USIM Card Protocol and Data Format

24.1.5.1 Protocol Overview

The operating procedure detailed in this section applies to every Smart Card with contacts.

The dialogue between the USIM controller and the card includes the following consecutive operations:

1. The USIM controller connects and activates the contacts.
2. Data is transmitted and received between the USIM controller and the card.
3. The USIM controller deactivates the contacts.

A warm reset procedure and sleep mode can be performed by the USIM controller.

The MPU controls the operation of the Smart Card interface.

During activation and deactivation sequences, two operating modes are provided:

- Automatic mode: Card activation and deactivation sequences (respectively, start and stop commands from the MPU) are monitored by the USIM controller.
  The MPU is in charge of the following:
  - Initiation of a transmission sequence
  - Writing and reading characters in the USIM controller address space
  - Deactivation of contacts in case of a card-reject decision
  - Power management in idle mode (sim_clk activation/deactivation)
  - Resetting the card (deactivation and activation) in case of consecutive error detection
- Manual mode: The MPU directly controls the interface signals (sim_clk, sim_rst, sim_pwrctrl, sim_io) during the card activation and deactivation sequences.

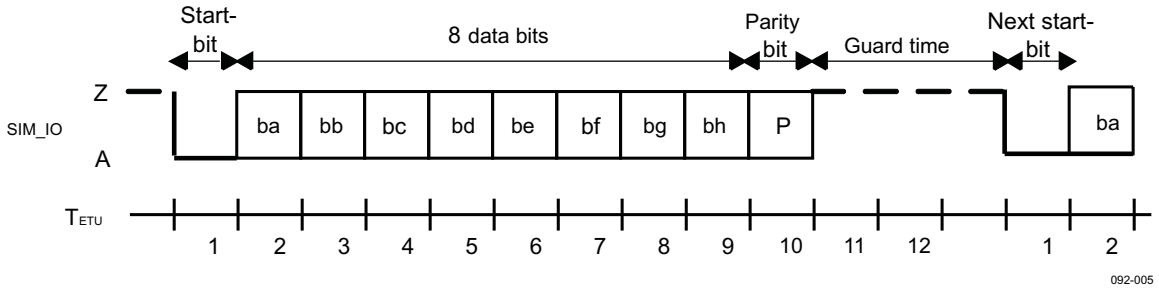
24.1.5.2 Data Format

Figure 24-5 shows the character frame sent or received by the USIM controller.

Two possible states exist for I/O:

- Mark or high state (state Z), if the card and the interface device are in reception mode or if the state is imposed by the transmitter
- Space or low state (state A), if this state is imposed by the transmitter

Figure 24-5. Character Frame



A character consists of 10 consecutive bits:

- A start-bit in state A
- Eight bits of information, designated ba to bh and conveying a data byte
- A tenth bit P used for even parity checking

A data byte consists of 8 bits designated ba to bh:

- Direct convention: ba is b0 (least-significant bit [LSB] first) and bh is b7.
- Inverse convention: ba is b7 (mos-significant bit [MSB] first) and bh is b0.

On hardware or software reset, the default coding convention is set as direct convention mode. For details about the coding convention, see *Coding Convention Protocol* in Section 24.1.5.3.2.4, *ATR Sequence Reception*.

Note: Conventions (level coding, connecting levels Z/A to digits 1 or 0 and a bit significance, connecting ba...bh to b1...b8) are specified in the initial character, called TS, which is transmitted by the card in response to reset (ATR). An ATR sequence is automatically detected by hardware.

Parity is correct when the number of ones is even in the sequence from ba to P.

The delay between consecutive characters (between start leading edges) is at least 12 ETU, including a character duration (10_UnicodeEncodeError_0.2) ETU plus a guard time, and the USIM controller and the Smart Card both remain in reception, so that the sim_io line is in state Z.

24.1.5.3 Operating Modes

24.1.5.3.1 Idle State

The idle state characterizes the state of the USIM controller when the contacts of the card are deactivated. In terms of the USIM controller signal, this means that:

- sim_clk = 0
- sim_rst = 0
- sim_pwrctrl = 0
- sim_io = 0 with the line in transmission mode

24.1.5.3.2 Activation Sequence

To initiate an interaction with a mechanically connected card, the USIM controller must activate the electrical circuits first (in response to a START command from the MPU).

The activation of the card must also be consecutive:

1. Reset the USIM controller hardware to initialize the logic at power up, or reset the software of the Smart Card interface module.
2. Enable internal clocks for the USIM controller.
3. Detect the card insertion event.

The activation process has the following steps:

1. Connect the contacts.
2. Activate the contacts (card power up).
3. Reset the card (cold reset).
4. Detect ATR.

The entire sequence can be executed either automatically or manually (bypass mode). For more information, see Section 24.3.2.1, *Automatic and Bypass Modes*.

24.1.5.3.2.1 Connection of IC Card Contacts

The contacts of the Smart Card are established when as the card is inserted in the driver. This operation is controlled by the external Smart Card driver, which sets the signal USIM_CD when card insertion is detected. The TWL3031 power IC connected to this Smart Card pin generates an external interrupt to the LOCOSTO processor to inform the MPU of the USIM card detection.

24.1.5.3.2.2 Activation of the Contact (Card Powerup)

After card insertion is detected, the contacts must be activated in the following sequence:

1. Card reset is maintained in LOW state (that is, sim_rsT = 0) during contact activation.
2. Card power-supply is switched on (that is, sim_pwrctrl = 1).
3. Bidirectional data line is set in reception mode and is high-impedance-driven (that is, sim_io = HZ).
4. The card external clock is activated (that is, sim_clk toggling).

Each of these steps is executed with timing constraints based on a predefined and software-configurable time period T_{ACTV} (TDUSIM field in the USIM_CONF3 register).

24.1.5.3.2.3 Card Reset (Cold Reset)

After the contacts are activated, the LOCOSTO processor must initiate a cold reset sequence to receive expected ATR from the card.

Two types of GSM SIM cards exist:

- Cards active on an internal reset (that is, synchronous cards)
- Cards active on an external low-level reset (that is, asynchronous cards)

The following protocol according to a specific schedule must be adopted:

1. At time *T0*:
   a. Maintain reset signal in LOW state (sim_rsT = 0) and Smart Card clock ON (sim_clk toggling).
   b. During *tc*, with T0 + 400Tclk < *tc* < T0 + 40,000Tclk
   With a synchronous card, THE ATR sequence must begin. For details, see Section 24.1.5.3.2.4, *ATR Sequence Reception*.

   Note: sim_rst signal must be maintained in LOW state.
   If ATR starts during *tc*, sim_rst must be maintained low, and Steps 2 and 3 are ignored.
   If no ATR is received by the USIM controller, go to Step 2.

2. At time *T1* = T0 + 40,000Tclk:
   a. Switch reset to HIGH state (sim_rsT = 1).
   b. During *tc* with T1 + 400Tclk < *tc* < T1 + 40,000Tclk
   With an asynchronous card, the ATR sequence begins. For details, see Section 24.1.5.3.2.4, *ATR Sequence Reception*.

   Note: If no ATR is received by the USIM controller, go to Step 3.

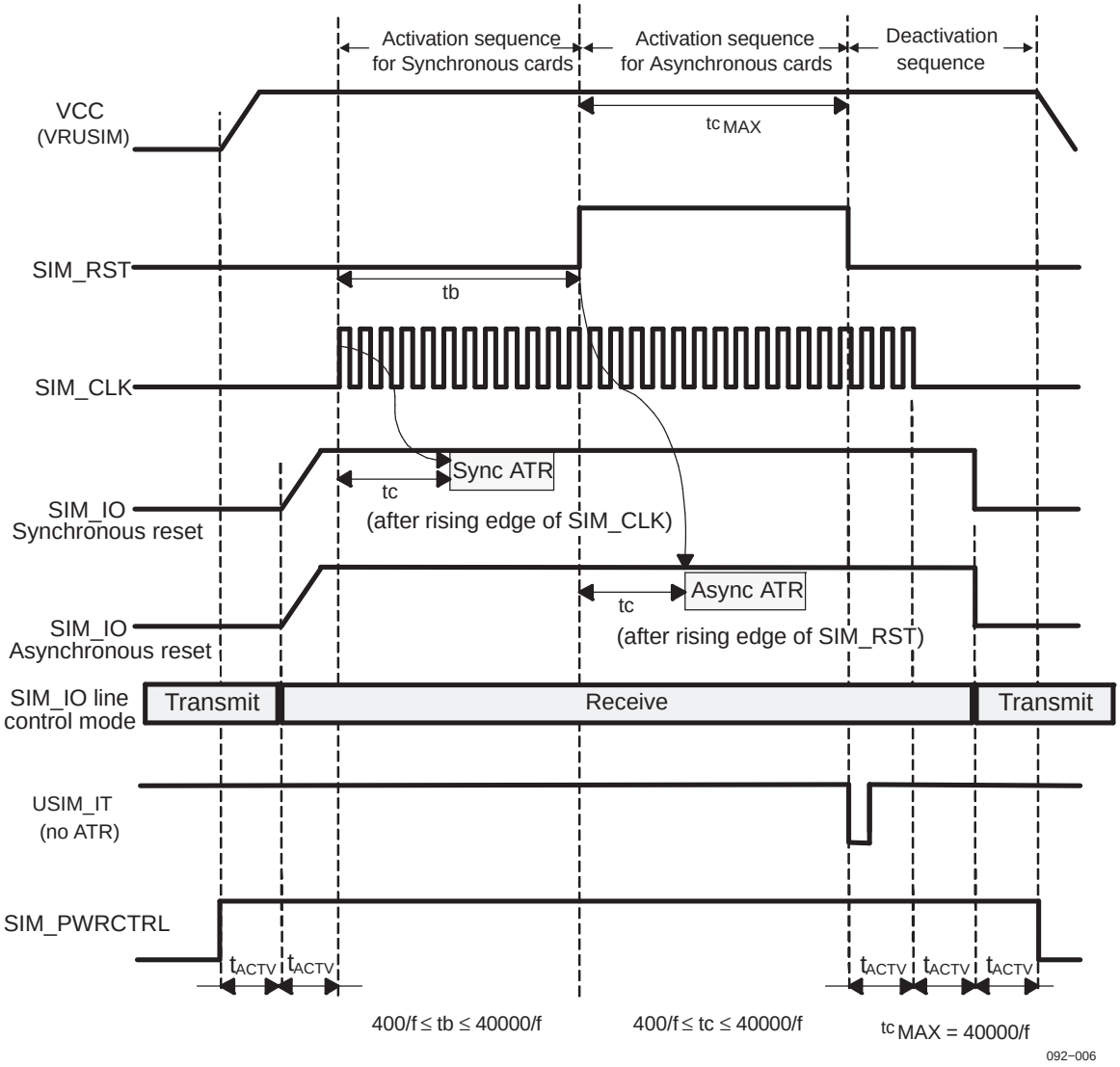
3. At time *T2* = T1 + 40,000Tclk:
   a. Switch reset to LOW state (sim_rsT = 0).
   b. Initiate the deactivation of contacts. For details, see Section 24.1.5.8, *Disconnection of Contacts*.

If no ATR is detected, the USIM controller automatically runs the last phase (contact deactivation sequence) after the contacts are activated.

GSM standard requires that an activation sequence for an asynchronous card always follow the activation sequence for synchronous cards. However, because synchronous cards are not often used, the Smart Card interface allows a bypass to the activation sequence for a synchronous card. This can be done by either hardware implementation or software programming. This can save 40,000 clock cycles in the case of the more frequently used asynchronous cards.

The timing sequence for a GSM card is shown in Figure 24-6.

Figure 24-6. Contact Activation and Cold Reset Sequence



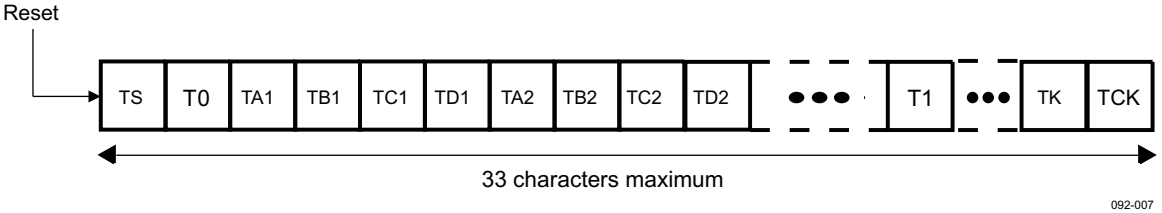
24.1.5.3.2.4 ATR Sequence Reception

Structure and Contents

Once the reset is done, the Smart Card sends an ATR to the USIM controller. An ATR sequence is automatically detected by hardware without any software configuration.

This data string can have a maximum length of 33 bytes, as shown in Figure 24-7.

Figure 24-7. ATR Data String



At the end of a reset operation, the answer from the card consists of the initial character TS followed by at most 32 characters in the following order:

1. T0: format character (mandatory)
2. TA1, TB1, TC1, TD1, …: interface characters (optional)
3. T1, T2, …, Tk: historical characters (15 characters maximum, optional)
4. TCK: check character (conditional)

ATR Sequence

The ATR reception sequence protocol has the following steps:

1. After reception of TS, the coding convention (direct or inverse) is performed and the corresponding configuration register is set accordingly (the CONFCODCONV bit of the USIMSTAT register).
2. The USIM controller adjusts some settings by hardware according to the coding convention for the automatic interpretation of the characters that follow.
3. Automatic reception of the other ATR characters that follow TS is enabled.

Coding Convention Protocol

The coding convention is based on the value of the first byte (TS) transmitted in the ATR of the Smart Card. Two conventions are proposed:

- Direct:
  - LSB first
  - Logic 1 coded on state Z (high level) and logic 0 coded on state A (low level)
- Inverse:
  - MSB first
  - Logic 1 coded on state A (low level) and logic 0 coded on state Z (high level)

Hardware identification of convention mode selection from the TS character is adopted. Upon reset, the default coding convention is set as direct convention mode; hardware then sets up based on the decoding of character TS.

When the TS character is interpreted (based on the direct convention set by default), there should be only two correct possibilities:

- TS = 0x3B (that is, convention mode is direct)
- TS = 0x3F (that is, convention mode is inverse)

If TS does not equal 3B H or 3F H, a coding convention error occurs, and an interrupt should be generated to the MPUr (with the TS_ERROR bit in the USIM_IT register). However, direct convention (that is, default setup) is maintained.

Parity Check Protocol

The parity bit is calculated on the basis of the even parity of logic one rule as defined by the coding convention.

- For the direct convention, the parity is based on an even number of state Z, that is, P = EXOR(b_0,b_1,b_2,b_3,b_4,b_5,b_6,b_7).
- For the inverse convention, the parity is based on an even number of state A, that is, P = EXNOR(b_0,b_1,b_2,b_3,b_4,b_5,b_6,b_7).

24.1.5.4 Data Receive and Transmit Procedure

All data exchanges between the USIM controller and the Smart Card are initiated by the MPU.

Several transmission protocols are part of the standards:

- ATR (answer to reset) ⇒ Receive procedure only (T = 0 only)
- PPS (protocol and parameter selection) ⇒ Receive and transmit procedure
- T (transmit protocol/T = 0 or T = 1) ⇒ Receive and transmit procedure

24.1.5.4.1 Data Transmit Procedure

In transmit mode, characters are sent from the USIM controller (transmitter) to the Smart Card (receiver). A different data transmit procedure is applied for the T= 0 and T= 1 protocols.

24.1.5.4.1.1 T = 0 Protocol

The transmit procedure is based on a parity test by the receiver, with a confirmation of the check sent to the transmitter. The parity of the information byte is computed by the USIM controller and is concatenated to the byte for the transmission.

The transmitter sends characters sequentially to meet minimum character guard time (CGT). A timer is reactivated at each new transmitted character to control this minimum delay. See Section 24.3.2.6.2, *Character Guard Time (T = 0 Protocol Only)*, for more details.

The WWT timer also flags the potential overflow of the waiting time between a transmitted character and the next character (sent by the card to the USIM controller). The timer is reactivated at each new transmitted character, and an interrupt is generated in case of overflow (wait time-out). This is the maximum amount of time to wait for the card to answer after the switch from transmit to receive mode. For details, see Figure 24-15 and Section 24.3.2.3.2, *Switching From Transmit to Receive Mode*.

The required steps of the transmit procedure are as follows:

1. Set bidirectional data sim_io line in transmit mode.
2. Transmit the start-bit active at low level (0). All consecutive time delays are referenced to the falling edge of this start-bit.

   At 1 ETU, the information byte is serialized and transferred followed by the parity bit (at 9 ETU).
3. Release the sim_io line and behave as a receiver at 10.5 ETU to allow the receiver to force on the line the state value corresponding to the result of the checking of the parity bit:
   - State value = 0 = > parity false
   - State value = 1 = > parity good
   - The receiver maintains the value 1 ETU minimum (whatever the parity check result) or 2 ETU maximum (if wrong parity is detected) before releasing the line.

Note: In hardware implementation of the USIM controller, the duration of the reverse of the I/O line is fixed to 2 ETU (whatever the parity check result) so that the next character in TX does not occur before 13 ETU (which complies with ISO7816-3), and after the minimum required guard time (CGT value).

The transmitter checks the parity confirm bit on-the-fly at 11 ETU.

The value of this parity confirm bit determines whether the transmitter repeats the transmission of the current character (this should be handled as a software decision). The next character can be transmitted only after guard time *t*.

The sequences for transmit mode without parity error and with parity error are shown in Figure 24-8 and Figure 24-9, respectively.

Figure 24-8. Transmit Mode Without Parity Error (T = 0) (t> = (CGT–10)ETU)

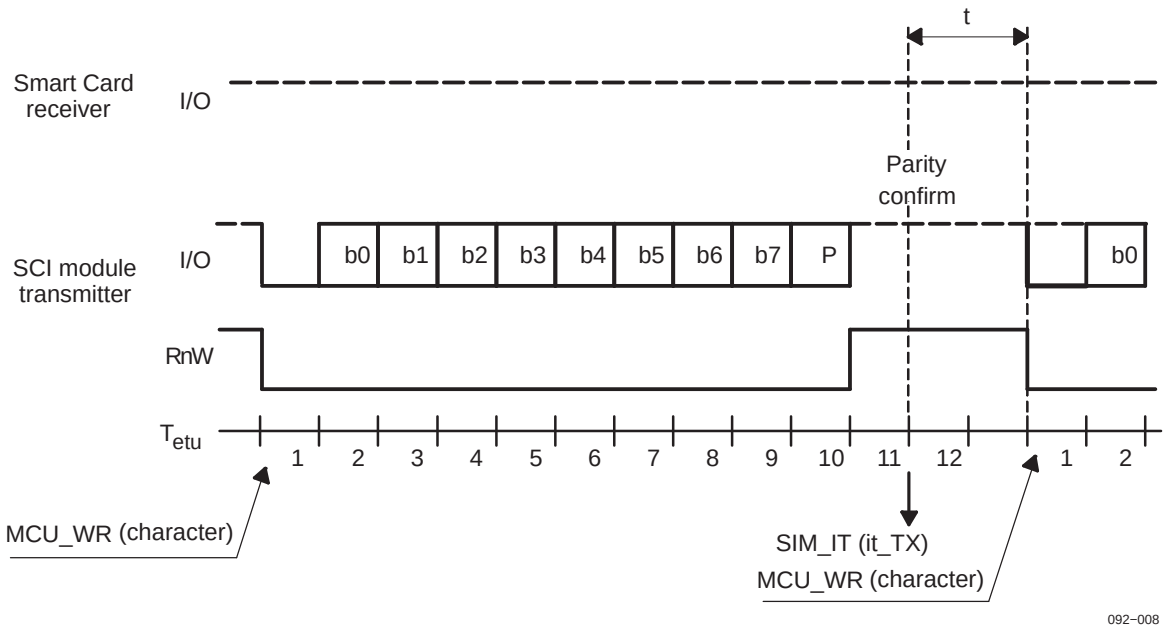
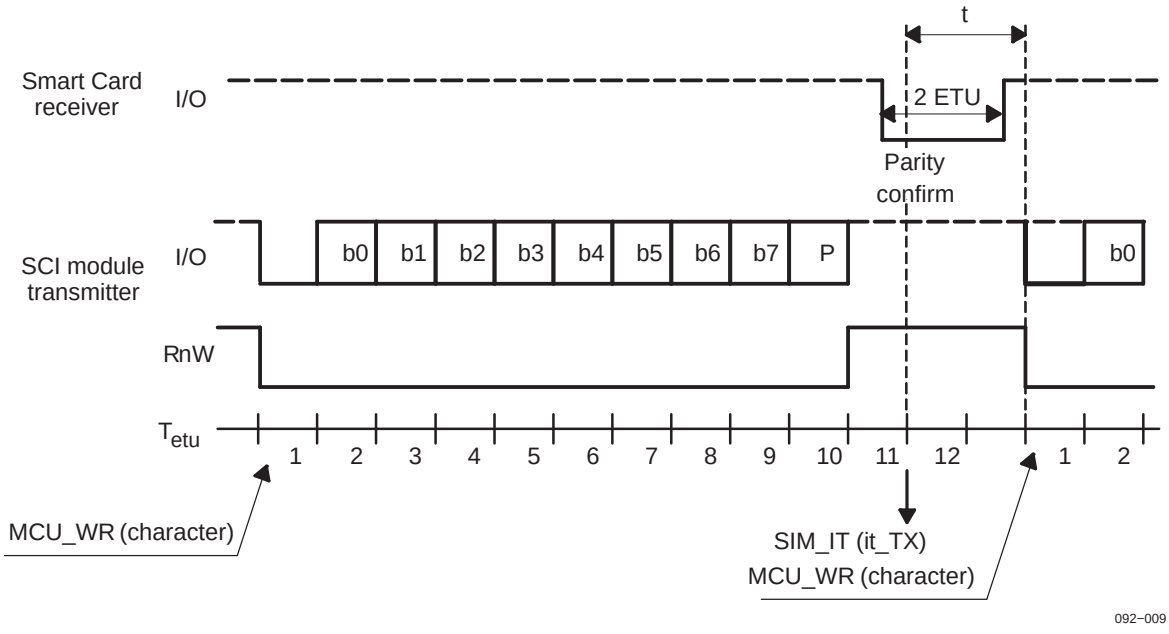


Figure 24-9. Transmit Mode With Parity Error (T = 0) (t> = (CGT-10)ETU)



24.1.5.4.1.2 T = 1 Protocol

The transmit procedure of the T = 1 protocol is more complex than that of the T = 0 protocol and is based on block transmission. A block is the smallest data unit that can be transferred. The transmitter sends characters of a block sequentially to meet minimum CGT and maximum CWT; the CGT timer is reactivated at each new transmitted character to control this minimum guard time between transmitted characters. For details, see Section 24.3.2.6.2, *Character Guard Time (T = 0 Protocol Only)*. The receiver (Smart Card) stays in reception mode during the entire block transmission.

The BWT also flags the potential overflow of the waiting time between the last transmitted character of a block and the first character of next block (sent by the card to the USIM controller). The timer is reactivated at each new transmitted character, and an interrupt is generated in case of overflow (wait time-out). This is the maximum amount of time to wait for the card to answer after the switch from transmit to receive mode.

For the T = 1 protocol, the parity error management as used in the T = 0 protocol does not apply. The error handling includes block error detection code (EDC), and the parity checking is a complement to the EDC. The EDC uses either CRC or a longitudinal redundancy check (LRC) for parity checking.

The required steps of the T = 1 transmit procedure are as follows:

1. Set bidirectional data sim_io line in transmit mode.
2. Transmit the start-bit active at low level (0). All consecutive time delays are referenced to the falling edge of this start-bit.

   At 1 ETU, the first block is serialized and transferred, followed by the parity bit (at 9 ETU).
3. Release the sim_io line and behave as a receiver at 10.5 ETU to allow the receiver to force on the line the state value corresponding to the result of the parity bit checking.
   - State value = 0 = > parity false
   - State value = 1 = > parity good
   - The receiver maintains the value 1 ETU minimum (whatever the parity check result) or 2 ETU maximum (if wrong parity is detected) before releasing the line.

Note: In hardware implementation of the USIM controller, the duration of the reverse of the I/O line is fixed to 2 ETU (whatever the parity check result) so that the next character in TX does not occur before 13 ETU (which complies with ISO7816-3), and after the minimum required guard time (CGT value).

- The transmitter checks the parity confirm bit on-the-fly at 11 ETU.

The value of this parity confirm bit determines whether the transmitter repeats the transmission of the current character (this should be handled as a software decision). The next character can be transmitted only after guard time *t*.

The sequences for the transmit mode without parity check and with parity check are shown in Figure 24-10 and Figure 24-11, respectively.

Figure 24-10. Transmit Mode Without Parity Check (T = 1) (t ≥ (CGT–10)ETU)

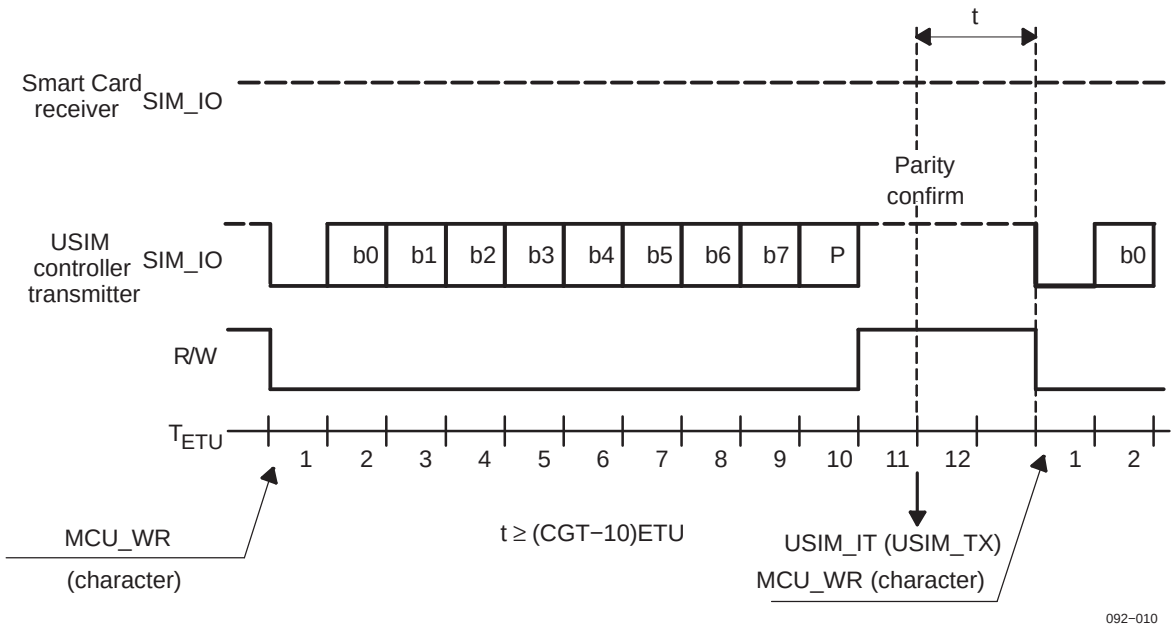
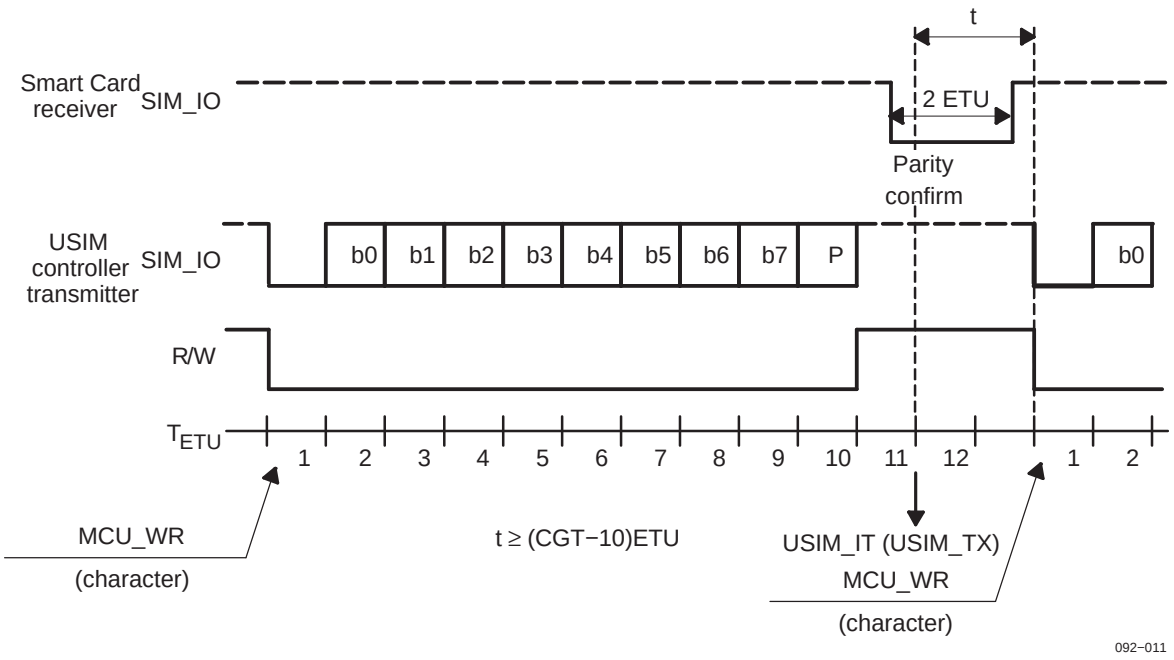


Figure 24-11. Transmit Mode With Parity Check (T = 1) (t ≥ (CGT−10)ETU)



24.1.5.4.2 Data Receive Procedure

In receive mode, characters are sent from the Smart Card (transmitter) to the USIM controller (receiver). In addition, a different process is applied according to the corresponding T = 0 or T = 1 protocol.

24.1.5.4.2.1 T = 0 Protocol

As with the transmit procedure, the receive procedure is based on a parity test by the receiver with a confirmation check to the transmitter. The parity of the information byte is computed by the USIM controller to the received byte, and is compared to the parity sent by the card.

The WWT timer flags the potential overflow of the waiting time between consecutive received characters. The timer is reactivated at each new character and an interrupt is generated in case of overflow (wait time-out). For details, see Section 24.3.2.6.1, *Work Waiting Time*.

The required steps of the receive procedure are as follows:

1. Set the serial line sim_io in reception mode.
2. Detect the start-bit with the oversampling clock at the rate of $8xF_{ETU}$ (*Fsam1*). On the point where the start-bit is confirmed, the mid-start-bit sampling clock (*Fsam2*) is activated and set from LOW to HIGH state. All time delays in the subsequent text are referenced to the time origin, which is the falling edge of the start-bit.
3. On the start-bit occurrence, the USIM controller initializes its receiver submodule and reinitializes the WWT timer. For details, see Section 24.2, *USIM Controller Functional Description*.
4. Using the mid-start-bit sampling clock, the bits of the received character (including the parity bit) can be sampled at each ETU with a reference time based on the middle of the start-bit.
5. At 9 ETU, the information byte is paralleled and its parity is computed. Parity checking can be either enabled or disabled by appropriate setting of one of the configuration registers. For details, see Section 24.3, *USIM Controller Programming Guide*.
6. If parity check is activated, the calculated parity is then compared to the transmitted parity sampled at 9.5 ETU.
7. If parity check is inactivated, the parity is assumed to be correct, whatever its value.
8. At 10.5 ETU the interface forces on the data line, released by the Smart Card, the confirm bit, the value of which is the result of the parity comparison.

- sim_io value = 0 stands for parity false.
- sim_io value = 1 stands for parity true.

The USIM controller maintains this value for 1 ETU only (regardless of good parity or wrong parity detected) before releasing the line at 11.5 ETU. The USIM controller can detect the start-bit of a new character either at 11 ETU or at 12 ETU (after the start-bit of the previous character), depending on the parity checking (either OK or FALSE). The transmitter (Smart Card) checks the parity confirm bit on the fly at 11 ETU. The value of this parity confirm bit determines whether the transmitter repeats the transmission of the current character.

The sequences for receive mode without or with parity error are shown in Figure 24-12 and Figure 24-13, respectively.

Figure 24-12. Receive Mode Without Parity Check

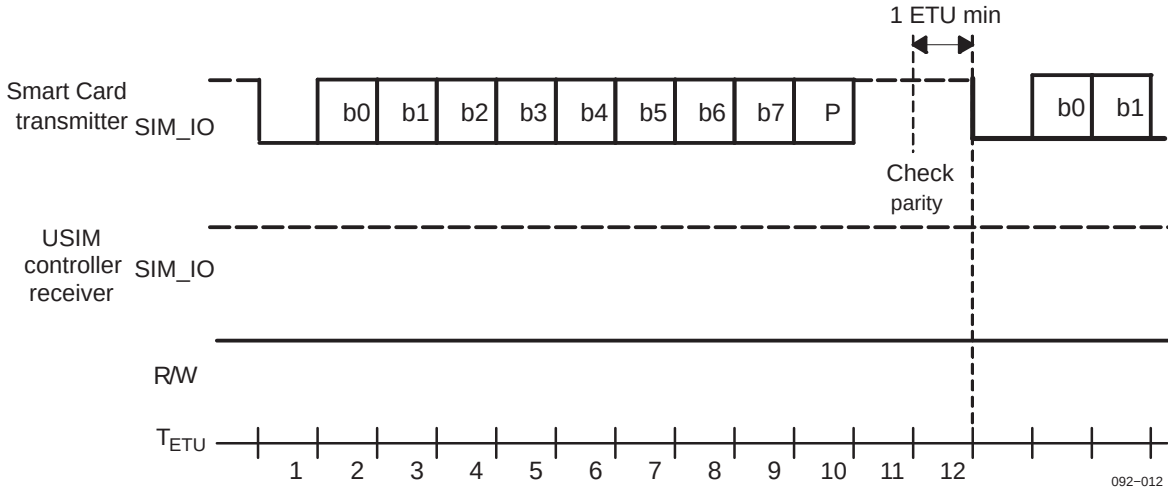
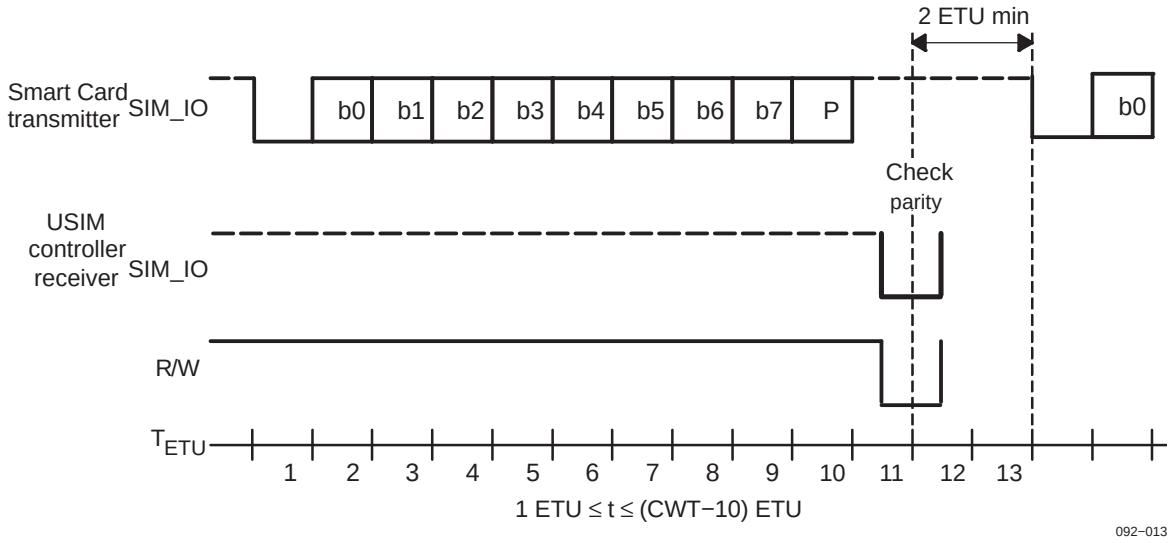


Figure 24-13. Receive Mode With Parity Check



24.1.5.4.2.2 T = 1 Protocol

For the T = 1 protocol, the receiver receives the characters of a block sequentially, without any confirmation to the transmitter of the parity checking result.

As a receiver, the USIM controller processes blocks to flag potential overflow of the waiting time between consecutive received characters (CWT), and counts the number of incoming characters. The CWT timer is reactivated each time a new character is received and an interrupt is generated if an overflow occurs (wait time-out). In the T = 1 protocol, block EDC code uses either CRC or LRC.

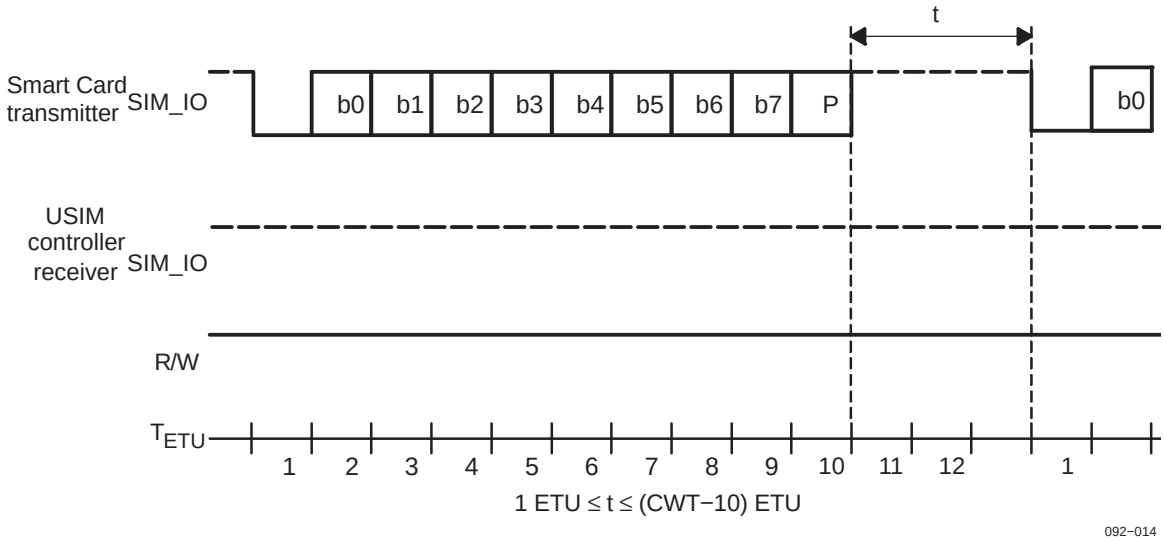
A block is the smallest data unit that can be transferred.

The required steps of the receive procedure are as follows:

1. Set the serial line sim_io in reception mode.
2. Detect of the start-bit with the oversampling clock at the rate of $8xF_{ETU}$(*Fsam1*). On the point where the start-bit is confirmed, the mid-start-bit sampling clock (*Fsam2*) is activated and set from LOW to HIGH state. All time delays in the subsequent text are referenced to the time origin, which is the falling edge of the start-bit.
3. On the start-bit occurrence, the USIM controller initializes its receiver submodule and reinitializes the CWT timer. For details, see Section 24.3.2.6.3, *Character Waiting Time (T = 1 Protocol Only)*.
4. Using the mid-start-bit sampling clock, the bits of the received character (including the parity bit) can be sampled at each ETU with a reference time based on the middle of the start-bit.
5. At 9 ETU, the information byte is paralleled and its parity is computed. The counter used to log the number of incoming characters is incremented.
6. After the reception of a character, the USIM controller checks the parity but does not issue an error signal on the sim_io line or ask for character repeat (even if a parity error occurs). The parity checking is used as a complement to EDC (block Error Detection Code).
7. When the counter of incoming characters equals 3, the received character is interpreted to extract the value for the length of the character (LEN). When the total number of received characters (including prologue and epilogue) equals either [LEN+4] if EDC is based on LRC, or [LEN+5] if EDC is based on CRC, the end of the block is reached. The CWT timer is deactivated and the counter of incoming characters is reset. Thus:
   - If the block is not received completely, the USIM controller can detect the start-bit of a new character after the guard time *t*, and then the receiver can process the next character of the block.
   - If the block reception is finished, the USIM controller checks the EDC and releases the sim_io line and can send an acknowledgement to the other side (that is, the card).

Figure 24-14 shows receive mode without parity check error.

Figure 24-14. Receive Mode Without Parity Check Error



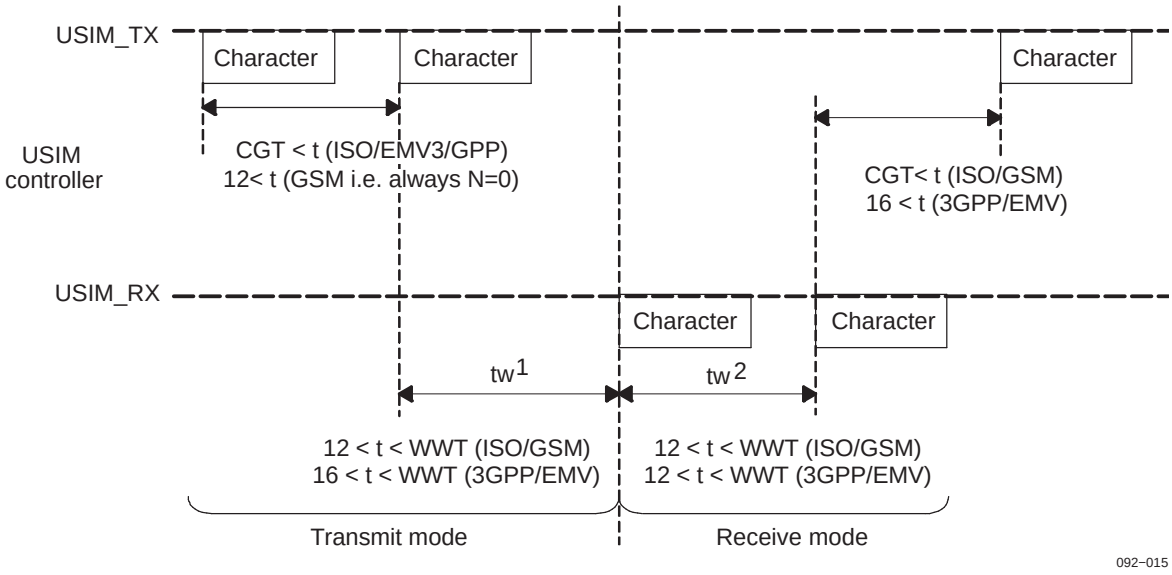
24.1.5.4.2.3 Switching Sequence

Switching from transmit mode to receive mode is software-programmed with 1 bit in the appropriate configuration register (bit TXNRX in the USIMCONF2 register), which controls the data line direction (1→TX / 0→RX). For details, see Table 24-27.

For details about the switching sequence, see ISO 7816-3 and ETSI GDM 11.11 standards.

For T = 0, the WWT timers flag the potential time-out (t_{w1}) between the last character transmitted by the USIM controller and the next character sent by the card to the USIM controller with (t_{w1} = WWT). This is the maximum time to wait for the card to answer after the switch from transmit to receive mode. For details about this timer, see Section 24.2.3.1.1, *Work Waiting Time*.

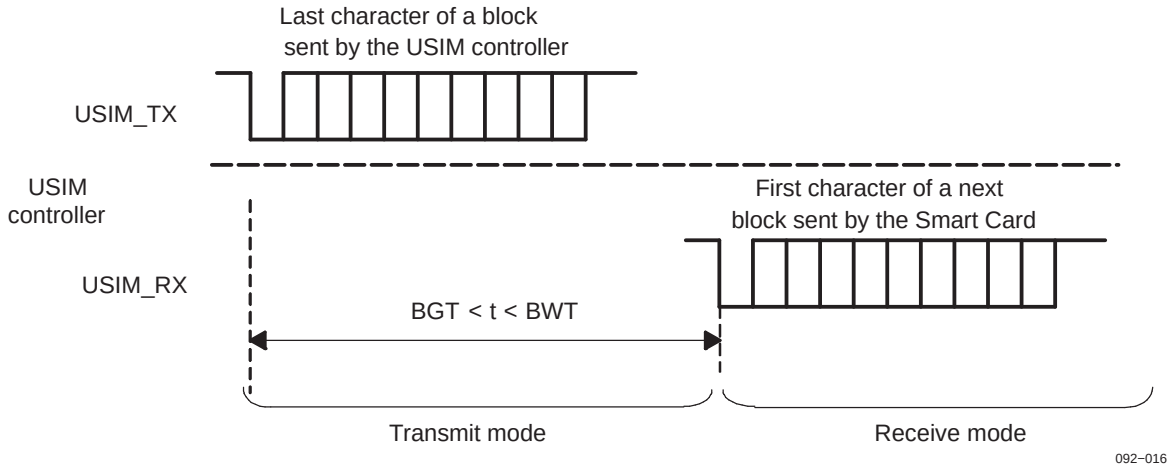
Figure 24-15. Switching from Transmit Mode to Receive Mode (T = 0)



For T = 1, the BGT timers and the BWT timers flag the potential time-out between the last character of a block sent by the USIM controller and the first character of the next block sent from the card. The time interval *t* should be such that BGT<t<BWT. For details, see Section 24.2.3.1.5, *Block Guard Time*, and Section 24.2.3.1.4, *Block Waiting Time*.

Figure 24-16 shows the switching procedure from transmit mode to receive mode (T = 1).

Figure 24-16. Switching from Transmit Mode to Receive Mode (T = 1)



24.1.5.5 Warm Reset Procedure

The USIM controller can decide to reset the Smart Card when no data exchanges are ongoing. For details about warm reset procedures, see the ISO 7816-3, ETSI GSM 11.11, and 3GPP TS 21.111 standards.

The following sequence should be applied:

1. At a notional time T0:

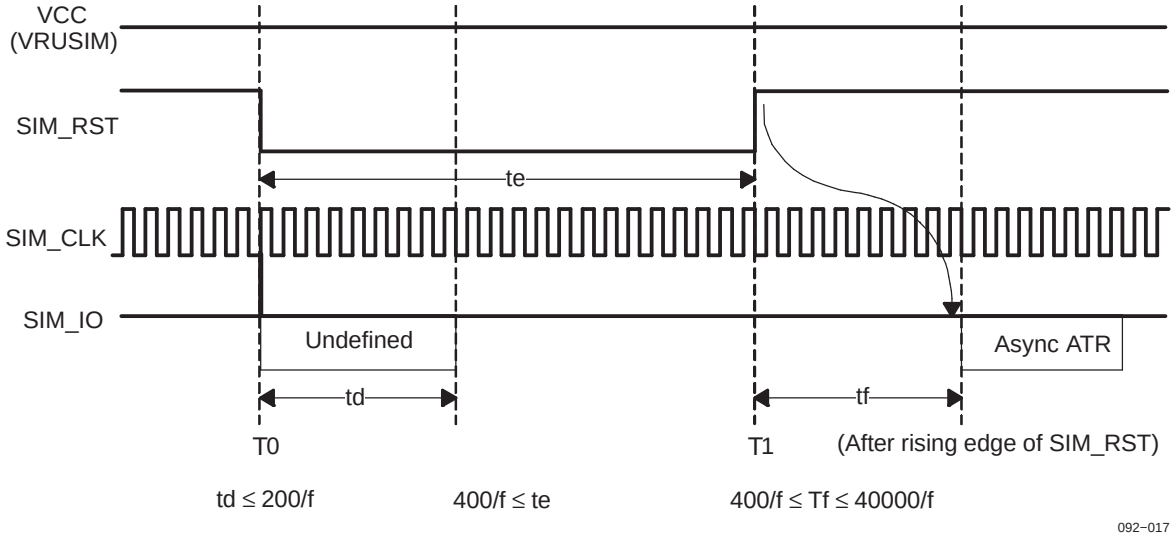
- Card reset is asserted low (sim_rsT = 0).
- Maintain Vcc and CLK stable (signal sim_clk toggling).
- Keep sim_io line in reception mode and driven High-Z (sim_io = HZ).

2. At time T1:
   - Card reset is released high (sim_rsT = 1).

Note: For GSM and 3GPP cards, the interval between time T0 and T1 must be *te* (that is, *te* = T1–T0): with *400Tclk* = *te*.

The warm reset protocol for GSM/3GPP cards is detailed in Figure 24-17 and in the ISO 7816-3 document.

Figure 24-17. ISO 7816-3 Compliant Warm Reset Sequence



24.1.5.6 Sleep Mode (Clock-Stop Mode)

In transmit mode, the MPU can decide to stop the Smart Card clock to save power if the card can support clock-stop mode and if there is no data transmission processing.

The Smart Card should support the clock-stop procedure as defined in this clause: The USIM controller should wait at least 1860 clock cycles after receiving the last character, including the guard time (GUARD_TIME), of the response before it switches off the clock (if it is allowed to do so). Thus, the GUARD_TIME equals at least 744 clock cycles and the USIM controller waits at least 744 clock cycles before it sends the first command after having started the clock:

744 clock cycles = GUARD_TIME = 4096 clock cycles

GUARD_TIME can be either defined by hardware (GUARD_TIME = 2×ETU) or programmed by software (GUARD_TIME = SOFT_TC_GUARD_ TIME_ADD). SOFT_TC_GUARD_TIME_ADD is a field of the TC_GUARD_ TIME_ADD register (see Table 24-44).

The clock-stop sequence is as follows:

1. Init

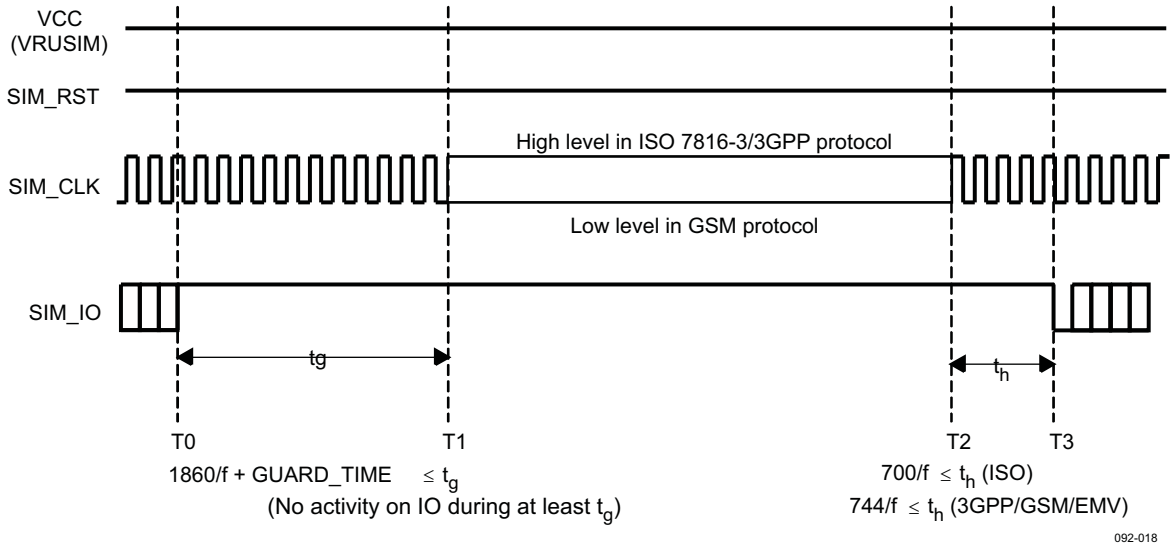
   Configure the idle level of the external card clock (sim_clk) in the corresponding configuration register (SCLKLEV bit of the USIMCONF1 register).
2. Time T0:

   No more transmission (sim_io = Z, sim_clk active), timer t_g = (1860 + GUARD_TIME) × Tclk started (timer t_g is reactivated after each new character)

3. Time T1 = T0 + (1860 + GUARD_TIME) × Tclk
   Smart Card clock stops.
4. Time T2 = T1
   On a new data transmission request, restart the Smart Card clock (sim_clk active) and start timer t_h = 744×Tclk.
5. Time T3 = T2 + 744×Tclk
   Restart transmission.

Figure 24-18 shows the clock-stop sequence.

Figure 24-18. Clock-Stop Sequence



During the clock-stop sequence, some delays differ according to card protocol. For details about the clock-stop sequence, see the ISO 7816-3, ETSI GSM 11.11, and 3GPP TS 21.111 standards.

24.1.5.7 Deactivation Sequence

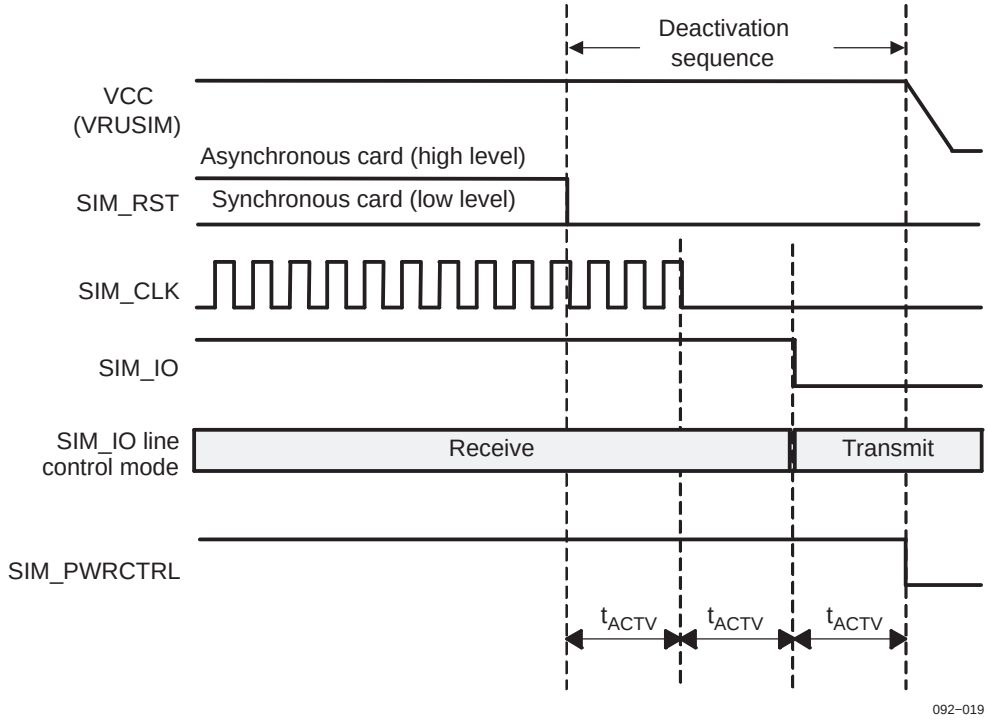
When information exchange is concluded or aborted (for example, unresponsive card or detection of card extraction), the USIM controller must deactivate the electrical circuits (in response to a STOP command from the MPU).

The deactivation process is as follows:

1. Card reset is asserted low (sim_rsT = 0).
2. Card clock is disabled (signal sim_clk inactive low).
3. Bidirectional data line is set in transmit mode and driven low.
4. Card power supply is switched off (signal sim_pwrctrl = 0).

Each step is executed with a schedule based on the configurable time period (t_{ACTV}), as used for the scheduling of the contacts activation phase (see Figure 24-19).

Figure 24-19. Deactivation Sequence



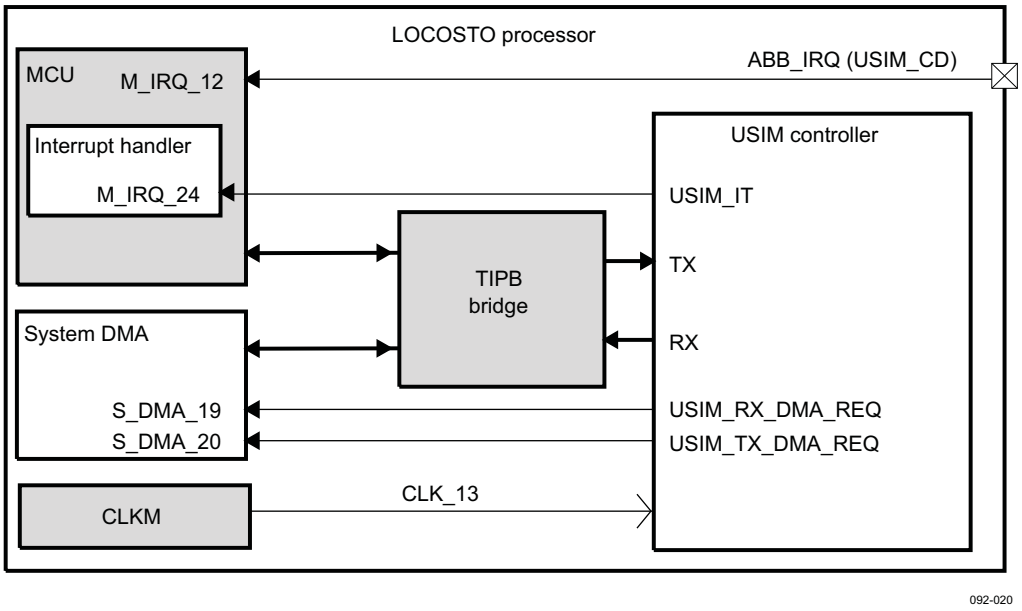
24.1.5.8 Disconnection of Contacts

The card can be extracted from the Smart Card driver only if the deactivation sequence is finished. Because the Smart Card can be extracted without any warning from the user, the duration of the deactivation sequence must be shorter than the time of disconnection. That means that the parameter t_{ACTV} must be calculated from the parameters of speed of extraction and length of the contacts.

24.1.6 USIM Controller Integration

This section describes the USIM controller integration inside the LOCOSTO device. Figure 24-20 shows the USIM controller integration.

Figure 24-20. USIM Controller Integration



24.1.7 Clocking, Reset, and Power-Management Scheme

24.1.7.1 Clocks

The USIM controller operates from a fixed functional clock of 13-MHz clock (CLK_13) generated by the CLKM module (see Table 24-3).

CLK_13 is the common clock of all GSM modules and must be activated with the CLOCK_GSM_REG register from the CLKM module.

Software users enable the GSM 13-MHz clock by programming the CLOCK_GSM_EN field (bit 0 of the CLOCK_GSM_REG register) in the CLKM module. The reset value of this field is 0 (that is, the GSM clock is deactivated by default).

Table 24-3. USIM Controller Clocking

Type	Name	Source	Frequency	Description
Functional clock	CLK_13	CLKM	13 MHz	Used as GSM 13-MHz reference clock
Interface clock	TIPB_CLK	TIPB	52 MHz	Used for data exchange between TIPB and USIM interface

24.1.7.2 Power Management

See Chapter 6, *Power, Reset, and Clock Management*.

24.1.7.3 Power Domain

Because of the voltage regulation and power control state-machine in the TWL3031 power IC, the LOCOSTO device has all its power domains (core voltage VRDBB, I/O voltage VRI/O, APLL and ADPLL voltage VRPLL, and memory interface voltage VRMEM) on when the On_nOff signal is rising. USIM LDO is kept off: VRSIM that supplies the USIM controller with extended-drain I/Os. This LDO is enabled after a software-controlled power-up sequence. For details about the PBIAS cell and extended-drain I/O, see Chapter 18, *Configuration*.

The USIM PBIAS cell handles three power domains:

- Power domain at 1.05 V /1.3 V /1.4 V to interface with the control logic (VDD core/VRDBB)
- Power supply at either 1.8 V or 3.0 V for generating the PBIAS reference voltage domain (VDDS3/VRUSIM)
- PBIAS reference voltage domain

The sim_pbias pin can be used in output mode to monitor the PBIAS voltage generated by the USIM PBIAS cell. In input mode, it can be used to provide (externally to the LOCOSTO device) the PBIAS reference voltage to the extended-drain I/Os when PBIAS is set in high-impedance mode.

24.1.7.4 Resets

24.1.7.4.1 Hardware Reset

Hardware reset is performed by the ARM_nRST signal. It completely resets the module. The registers and the state-machine are reset.

24.1.7.4.2 Software Reset

Software reset is performed with the CMDIFRST field of the USIMCMD[0] register. By setting the CMDIFRST field to 1, the USIM controller is reset. The default value of this field is 0. This reset does not reset the registers, so the previous configuration is kept. For details about USIMCMD register description, see Section 24.4, *USIM Controller Registers*.

Table 24-4 lists the USIM hardware and software reset information.

Table 24-4. USIM Hardware and Software Reset

Type	Name	Source	Activation	Domain
Hardware	ARM_nRST	CLKM	0	Entire module
Software	CMDIFRST bit of the USIMCMD register	Register programming	1	Does not reset the registers

24.1.8 Hardware Requests

24.1.8.1 DMA Requests

The USIM controller DMA request mapping is shown in Table 24-5.

Table 24-5. USIM DMA Requests

Type	Source Interrupt Request Name	Destination Interrupt Request Name	Source	Destination	Description
DMA request	USIM_TX_DMA_REQ	DMA_20	USIM interface	DMA controller	DMA request from USIM interface used in transmit mode
DMA request	USIM_RX_DMA_REQ	DMA_19	USIM interface	DMA controller	DMA request from USIM interface used in receive mode

24.1.8.2 Interrupt Requests

Interrupt mapping from the USIM controller and from an external device to the MPU is shown in Table 24-6.

Table 24-6. USIM Interrupts Names and MPU IRQ Mapping

Type	Source Interrupt Request Name	Destination Interrupt Request Name	Source	Destination	Description
Internal interrupt	USIM_IT(1)	IRQ_24	USIM interface	MPU	Interrupt from USIM controller
External interrupt	USIM_CD(2)	IRQ_12	USIM interface	MPU	External interrupt for USIM card detection

(1) The USIM_IT interrupt request is generated by the USIM controller. The source of the interrupt can be determined with USIM_IT register fields. This interrupt can be masked by using USIM_MASK_IT register fields (see Table 24-32).

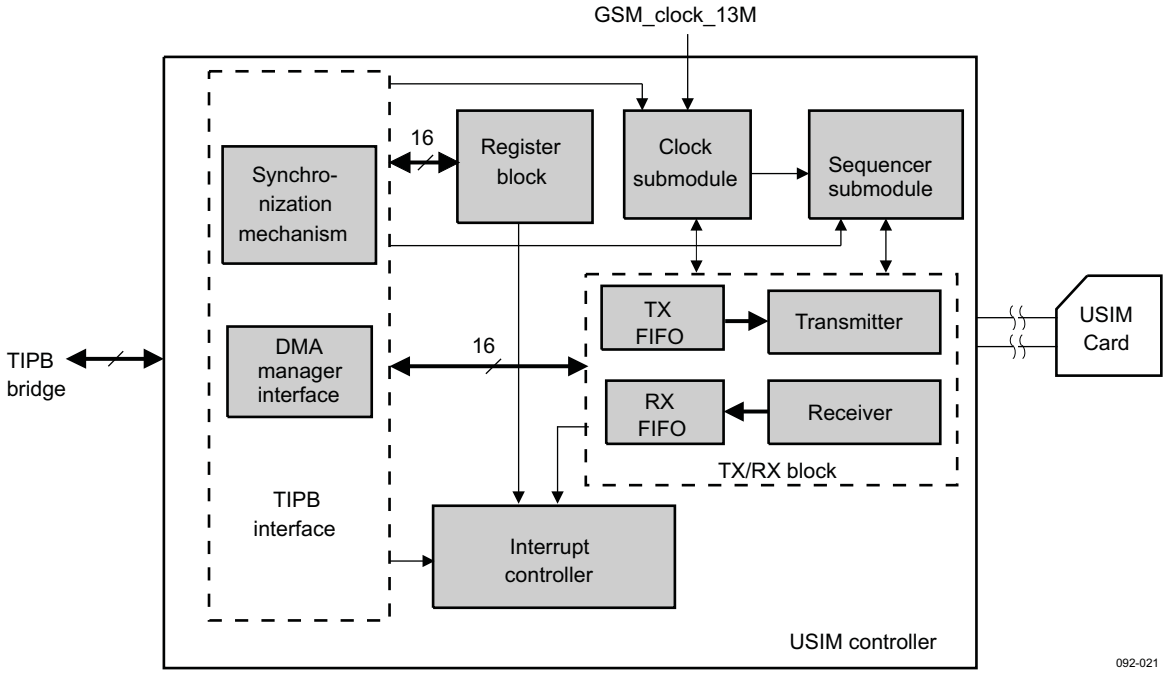
(2) The USIM_CD interrupt request is provided by an external power device (TWL3031 IC) that generates an external interrupt to the MPU interrupt handler to inform of any extraction or insertion of the USIM card. This interrupt is provided by an external pin (ABB_IRQ).

Note: When an interrupt is detected on the ABB_IRQ pad, software users must first read the TWL3031 IC registers to determine the cause of the interrupt. The MPU then initializes (if insertion) or deactivates (if extraction) the USIM controller. For more information about ABB_IRQ, see Chapter 12, *Interrupt Handlers*. For information about the TWL3031 IC, contact your TI representative.

24.2 USIM Controller Functional Description

Figure 24-21 is the functional block diagram of the USIM controller hardware blocks.

Figure 24-21. USIM Controller Functional Blocks



The USIM controller has the following submodules:

- TIPB interface composed of a synchronization mechanism and a DMA manager interface
- Clock receiving the 13-MHz GSM clock
- Sequencer submodule
- Transmitter/receiver (TX/RX) block composed of a transmitter, a receiver, and the two associated FIFOs
- Interrupt controller
- Register block
- Error reporting

24.2.1 TIPB Interface

This submodule includes the synchronization mechanism, DMA management interface, register address decoder, and other related circuits.

24.2.1.1 Synchronization Mechanism

Because the TIPB and USIM module clock are asynchronous, the synchronization mechanism ensures that the MPU can correctly exchange data with the USIM.

24.2.1.2 DMA Manager Interface

Data transfer can be executed in two ways, DMA or interrupts. In both methods, FIFOs are used to transmit and receive data. The corresponding control register bit is DMA_MODE in the USIM_FIFOS register (see Table 24-33). With DMA_MODE set to 1, DMA transfer mode is adopted.

24.2.2 Clock Submodule

The clock submodule generates reference clocks for the main sequencer, the TX/RX blocks and associated timers, and other submodules as follows:

- Provides SIM_CLK to Smart Card clock contact (that is, 13/N MHz with N = 1, 2, 4, or 8)
- Provides RX and TX ETU clocks (FETU) to both receiver and transmitter blocks
- Provides sampling clocks: $F_{sam1} = 8 \times F_{ETU}$ and $F_{sam2} = F_{ETU}$ to the receiver block

The Smart Card clock (external clock F_{SCK}) and all submodules clocks (internal clocks) are derived from the 13-MHz reference clock signal CLK_13.

The Smart Card external clock frequency (F_{SCK}) can equal $1/N x F_{CLK}$ with N = 1, 2, 4, or 8, and F_{CLK} is the free-running clock of TIPB (13 MHz). However, the maximum frequency allowed on Smart Card clock contact (F_{SCK}) depends on the selected conversion factor for the clock (F). For details, see Section 24.3.2.3.6.1, *Transmission Factors*. Also, the default clock frequency applied to the Smart Card during the ATR sequence (after a cold and warm reset) must be 13/4 MHz or 13/8 MHz to fulfill ISO 7816-3 requirements.

The following clocks are generated:

- The ETU clock (F_{ETU}), which schedules bit/character transmission in the sim_io line
- The oversampling clocks (F_{SAM1}/F_{SAM2}), which schedule start-bit detection and bit/character reception on the sim_io line

These two clocks are used to interpret characters or blocks sent by the card.

24.2.2.1 ETU Clock or Bit Duration

The Smart Card external clock frequency (F_{SCK}) can equal $1/N x F_{CLK}$ with N = 4, or 8.

The nominal duration of a transmitted/received bit on the sim_io line is the ETU. This duration depends on the two parameters F (conversion factor of the SIM clock rate) and D (adjusting factor of the binary baud rate), which are defined in character TA1 of the ATR sequence:

$$T_{ETU} = (F/D) x T_{SCK}$$

The supported transmission factors (Fi/Di) are described in Section 24.3.2.3.6.1, *Transmission Factors*.

Accuracy for bit positioning in receive mode must be better than 0.2 ETU. Therefore, an oversampling frequency of $8 \times F_{ETU}$ is adopted for the detection of the start-bit. This enables sampling of the bits with an accuracy of 0.125 ETU (better than 0.2 ETU).

Also, the switching of the data line from receive to transmit and vice-versa for the confirmation of the parity is realized in the time range of [_UnicodeEncodeError_0.2 ETU, +0.2 ETU], as requested by ISO.

24.2.2.2 Over-Sampling Clocks

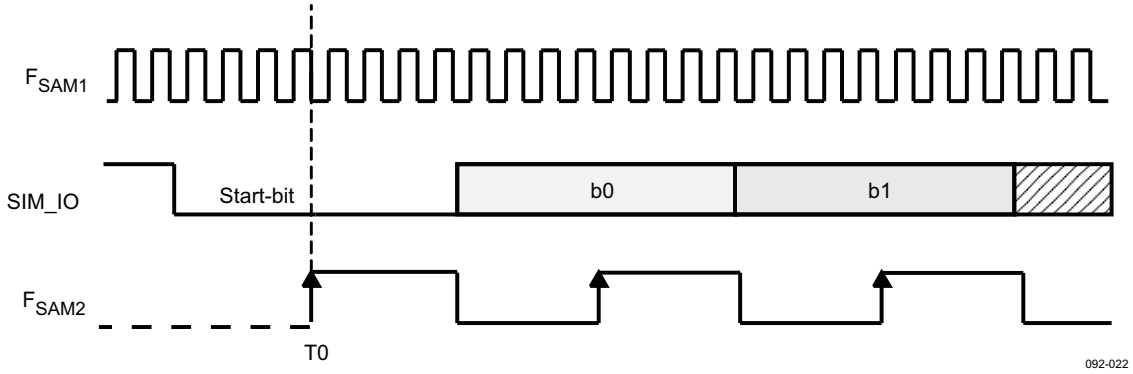
Accuracy for bit positioning in receive mode must be better than 0.2 ETU. Therefore, an oversampling frequency of $8 \times F_{ETU}$ is adopted for the detection of the start-bit. This enables sampling of the bits with an accuracy of 0.125 ETU (better than 0.2 ETU):

$$T_{SAM} = 1/8 \times T_{ETU}, \text{ which equals to } T_{SAM} = (F/D) \times (N/8) \times T_{13MHz}$$

Two sampling clocks are used to ensure correct reception of characters.

Sampling clock 1 ($F_{Sam1} = 8 \times F_{ETU}$) is used to over-sample the sim_io line to detect the start-bit of a new character or block. When the start-bit is detected, sampling clock 2 is activated ($F_{Sam2} = F_{ETU}$) and each bit of a character is sampled in the middle of its period within ±0.2 ETU. Figure 24-22 shows the over-sampling clock generation.

Figure 24-22. Over-Sampling Clocks



At time T0, after start-bit detection (that is, in the middle of start-bit period), Fsam2 is activated.

Subsequent bits of a character are sampled at successive times T0 + NxTetu, with N in the range [1,9].

24.2.2.3 Transmission Factors (F/D)

The ETU period depends on two parameters: F (conversion factor of the USIM clock rate) and D (adjusting factor of the binary baud rate); that is, $T_{ETU} = (F/D) \times T_{SCK}$.

During the ATR sequence, the initial ETU period corresponds to the F and D default values (Fd and Dd). Fd and Dd equal 372 and 1, respectively. The MPU can either keep this period or select other (F, D) parameters as defined during the ATR and/or the PPS procedures.

Two clocks ensure transmission and/or reception of characters: the ETU clock and the over-sampling clock. Both clocks must be generated based on F and D factors.

To support all (F, D) pairs as mentioned in ISO/IEC 7816-3, the clock module must allow generation of all corresponding ETU and sampling clocks. It is adopted to implement clock generation using a hardware prescalar divider instead of using PLL. Instead of synchronously generating F_{ETU} from F_{SAM} by a 1:8 clock divider, both clocks are generated independently from the 13-MHz GSM reference clock (F_{CLK}) using the following formulas:

$$T_{ETU\ approximated} = \text{Round}\ [(F/D) \times N] \times T_{13MHz}$$

$$T_{SAM\ approximated} = \text{Round}\ [(F/D) \times (N/8)] \times T_{13MHz}$$

The corresponding values of the ETU clock divider (that is, round [(F/D) x N]) and sampling clock divider (that is, round [(F/D) x (N/8)]) for a certain (F, D) pair is decided by hardware based on the combinatorial decoding of the tables. See Section 24.3.2.3.6.3, *(F, D) Pairs Supported by the USIM Controller in Hardware Mode*.

$$F_{SAM} = F_{CLK} / \text{CONF\_SAM1\_DIV}$$

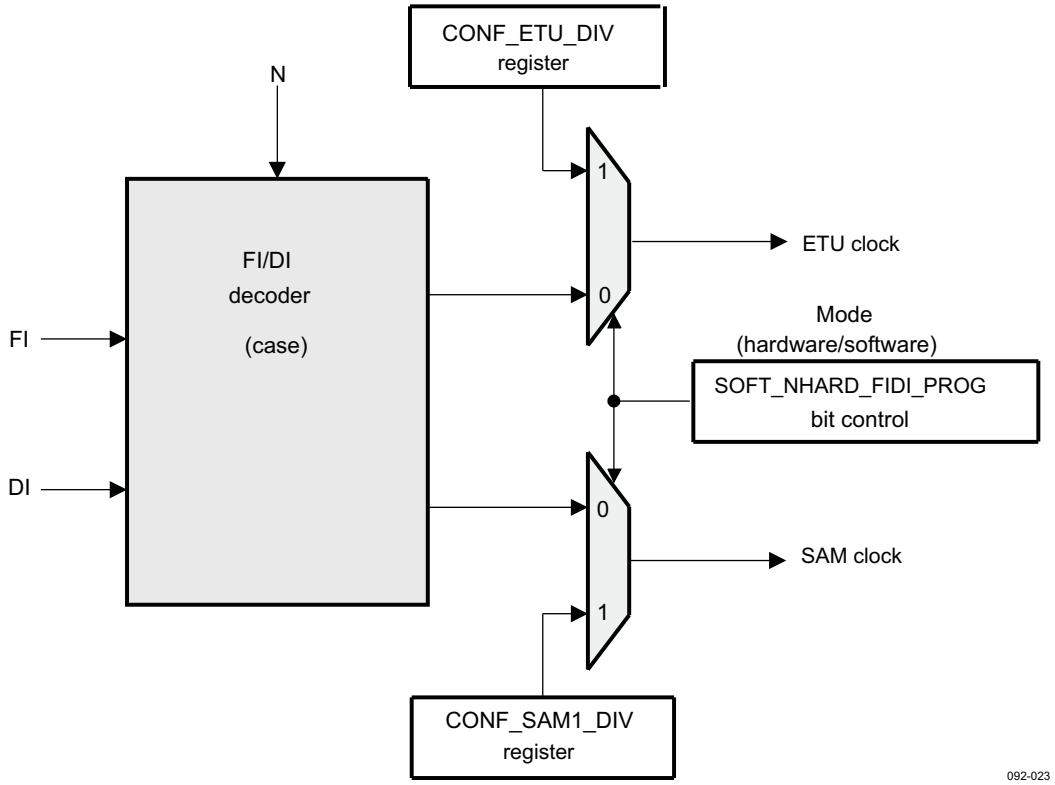
$$F_{ETU} = F_{CLK} / \text{CONF\_ETU\_DIV}$$

CONF_ETU_DIV and CONF_SAM1_DIV factors are generated from the binary values FI and DI given in the TA1 character of ATR, and the binary coding of the clock divider for Smart Card clock (factor N) given in the TC1 character of ATR. To increase debugging and/software programming flexibility, for instance, CONF_ETU_DIV and CONF_SAM1_DIV factors can also be software-configurable. These values are stored in separate registers:

- The CONF_ETU_DIV value is stored in THE CONF_ETU_DIV register (see Table 24-42).
- The CONF_SAM1_DIV value is stored in the CONF_SAM1_DIV register (see Table 24-39).

Figure 24-23 shows the clock generation prescalar divider.

Figure 24-23. Clock Generation Prescalar Divider



24.2.3 Sequencer Submodule

The sequencer submodule manages all functional modes of the USIM interface with a possibility of software-bypassing some automatic sequences if manual mode is selected. The sequencer handles all waiting times as defined by the ISO with the possibility of removing the corresponding hardware timers to give full control to the MPU.

24.2.3.1 Timer Descriptions

24.2.3.1.1 Work Waiting Time

WWT is the maximum delay of the interval between the leading edge of any character sent by the card and the leading edge of the previous character (sent by either the card or the card interface module). WWT flags the potential overflow of the waiting time between a received character (sent by the card to the USIM controller) and the previous character sent by the card. However, the WWT timer is also flags the potential overflow of the waiting time between a transmitted character and the next character (sent by the card to the USIM controller). This is the maximum time to wait for the card to answer after switching from transmit to receive mode. For WWT use, see Figure 24-15.

The WWT value is defined by the ATR sequence; WWT is based on the WI parameter from the TC2 character of the ATR:

For SIM and USIM cards, the WWT is given by the following formula:

WWT = 960 x D x WI x ETU

In other words,

WWT = CONFWAITI x 480 x D x ETU (with CONFWAITI = 2 × WI)

The CONFWAITI field is defined in the CONF4_REG register, and this value can be software-programmed. For details, see Section 24.3, *USIM Controller Programming Guide*.

This timer is used for the T = 0 protocol only, and the timer is reactivated each time a new character is received (start-bit detection).

24.2.3.1.2 Character Guard Time

CGT is the minimum guard time between a character sent from the USIM controller to the card, and the previous character (sent by either the card or the USIM controller).

CGT = (12 + N) × ETU for 0 = N = 254

CGT = 12 for N = 255 and T = 0 protocol

CGT = 11 for N = 255 and T = 1 protocol

With:

- Parameter N (additional guard time) defined by the TC1 character of the ATR. The default value is N = 0. There is no additional guard time to send characters from the card to the USIM controller.
- ETU based on the following:
  - F/D; that is, the values used to compute the ETU if T = 15 is absent from the ATR
    or
  - Fi/Di: If T = 15 is present in the ATR, Fi and Di are the values indicated by the card in TA1. If TA1 is absent, Fi and Di are set at default values.

Parameter N of the CGT timer can be software-programmed. It corresponds to the CGT field defined in the USIM_CGT(7:0) register. For details, see Section 24.3, *USIM Controller Programming Guide*.

The CGT timer is applicable to both T = 0 and T = 1 protocols, and is activated each time a new character is transmitted. It is also activated for switching from receive mode to transmit mode (that is, a minimum delay is required before the card is ready to accept a new input character).

24.2.3.1.3 Character Waiting Time

CWT is the maximum delay between the leading edges of consecutive characters in a block.

The CWT value is defined by thr ATR sequence; CWT is based on the CWI parameter from the first Tbi character (the 4 LSBs) of the ATR, after the first occurrence of T = 1 in TD_{i-1} for i > 2.

CWT is given by the following formula:

$CWT = (11 + 2^{CWI})$ ETU

For 3GPP, CWI should be in the range 00 to 05.

CWT can be defined as follows:

CWT = CWT_value x Tetu

with:

$$CWT\_value = 11 + 2^{CWI} \text{ (for USIM card)} \Leftrightarrow CWT\_value \text{ in the range } [12,43]$$

CWT_value is software-configurable and defined in the USIM_CWT(15:0) register. For details, see Section 24.3, *USIM Controller Programming Guide*.

Note: The CWT timer is used for the T = 1 protocol only. When there is a potential error in the length, the CWT timer can detect the end of a block. This timer is reactivated each time a new character is received (start_bit detection).

24.2.3.1.4 Block Waiting Time

BWT is the maximum delay between the last character of the block received by the card and the first character of the next block sent by the card to the USIM controller.

The BWT value is defined by the ATR sequence; BWT is based on the BWI parameter from the first Tbi character (the 4 MSBs) of the ATR after the first occurrence of T = 1 in TD_{i-1} for i > 2.

The BWT is given by following formula:

$$BWT = (11 + 2^{CONFBWI} \times 960 \times 372 \times (D/F)) \times ETU$$

CONFBWI value is software-programmed and defined in the USIM_BWT registers. For details, see the programming guide.

For 3GPP, BWI should be in the range 00 to 09, and the default value is 4.

BWT is used for the T = 1 protocol only. The timer is reactivated each time a block is transmitted to the card (start-bit of the last character of the block), and it is used to detect an unresponsive card.

24.2.3.1.5 Block Guard Time

BGT is the minimum guard time between consecutive blocks sent in opposite directions.

The BGT has a fixed hardware value as follows:

$$BGT = 22 \text{ ETU}$$

Because BGT is constant, a fixed timer is adopted.

BGT is used for the T = 1 protocol only. This timer is reactivated each time a new block is received or transmitted.

24.2.3.2 State-Machines

The main sequencer is based on a finite state-machine (FSM) scheduled on the ETU clock. The reception and transmission state-machines are initiated by the main sequencer (when in data communication mode) and scheduled on the ETU clock.

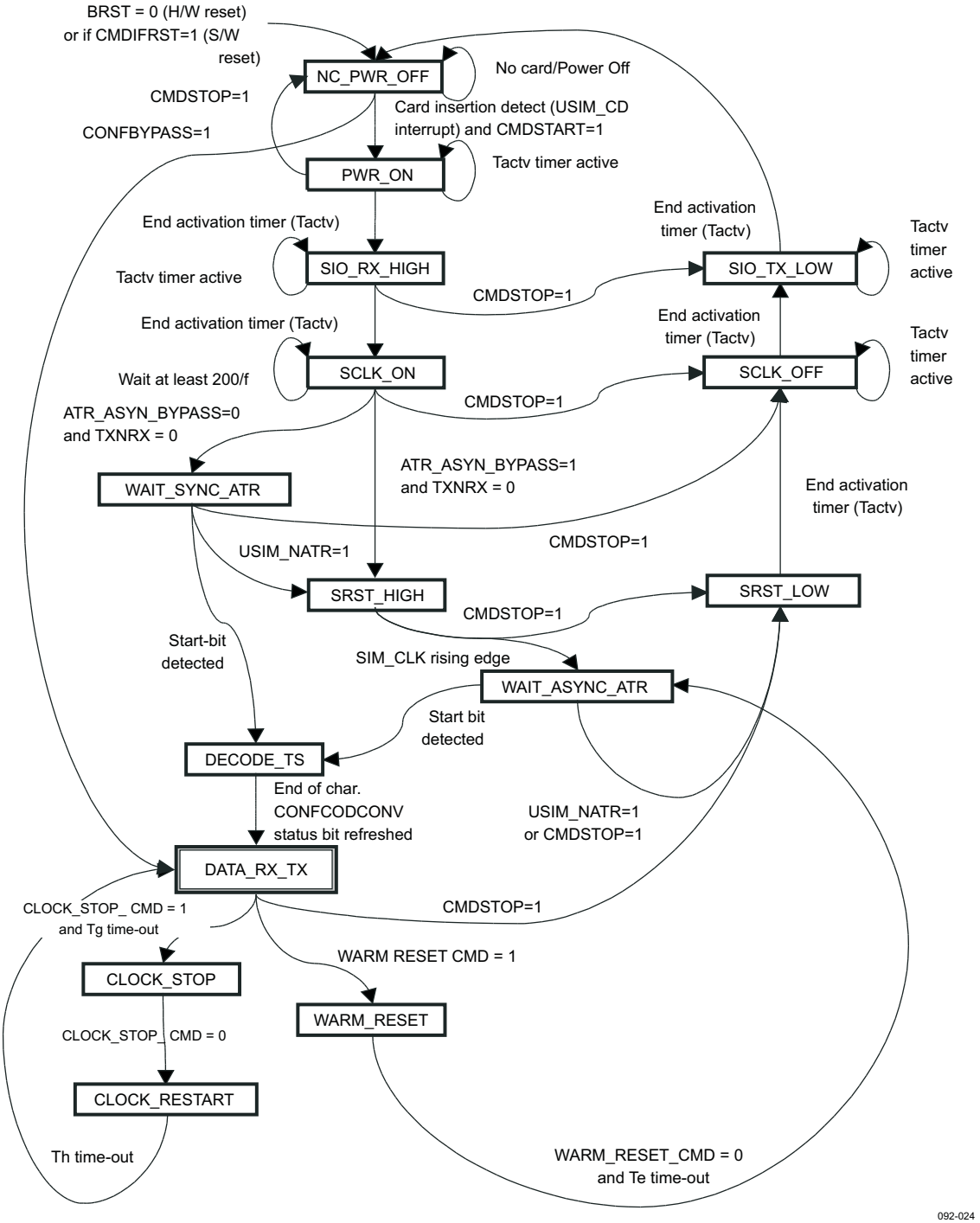
24.2.3.2.1 Main State-Machine

The states of the main FSM are listed in Table 24-7. Figure 24-24 shows the main FSM.

Table 24-7. Main FSM Sequencer States

State	Description
NC_PWR_OFF	No card inserted and no card power supply
PWR_ON	Card power supply activated (signal sim_pwrctrl high)
SI/O_RX_HIGH	sim_io line set in receive (RX) mode and driven high-impedance
SCLK_ON	Card clock activated (signal sim_clk toggling)
WAIT_SYNC_ATR	Wait for ATR sequence of synchronous card
SRST_HIGH	Card reset released (signal sim_rst inactive high)
WAIT_ASYNC_ATR	Wait for ATR sequence of asynchronous card
SRST_LOW	Card reset asserted (signal sim_rst active low)
SCLK_OFF	Card clock disabled (signal sim_clk inactive low)
SI/O_TX_LOW	sim_io line set in transmit (TX) mode and driven low
DECODE_TS	Coding convention interpreted (direct/inverse)
DATA_RX_TX	Interface set in data receive (RX) or transmit (TX) mode
WARM_RESET	Card reset asserted for time te (signal sim_rst active low)
CLOCK_STOP	Card clock stopped (signal sim_clk inactive low or high)
CLOCK_RESTART	Card clock restarted (signal sim_clk toggling)

Figure 24-24. Main Finite State-Machine



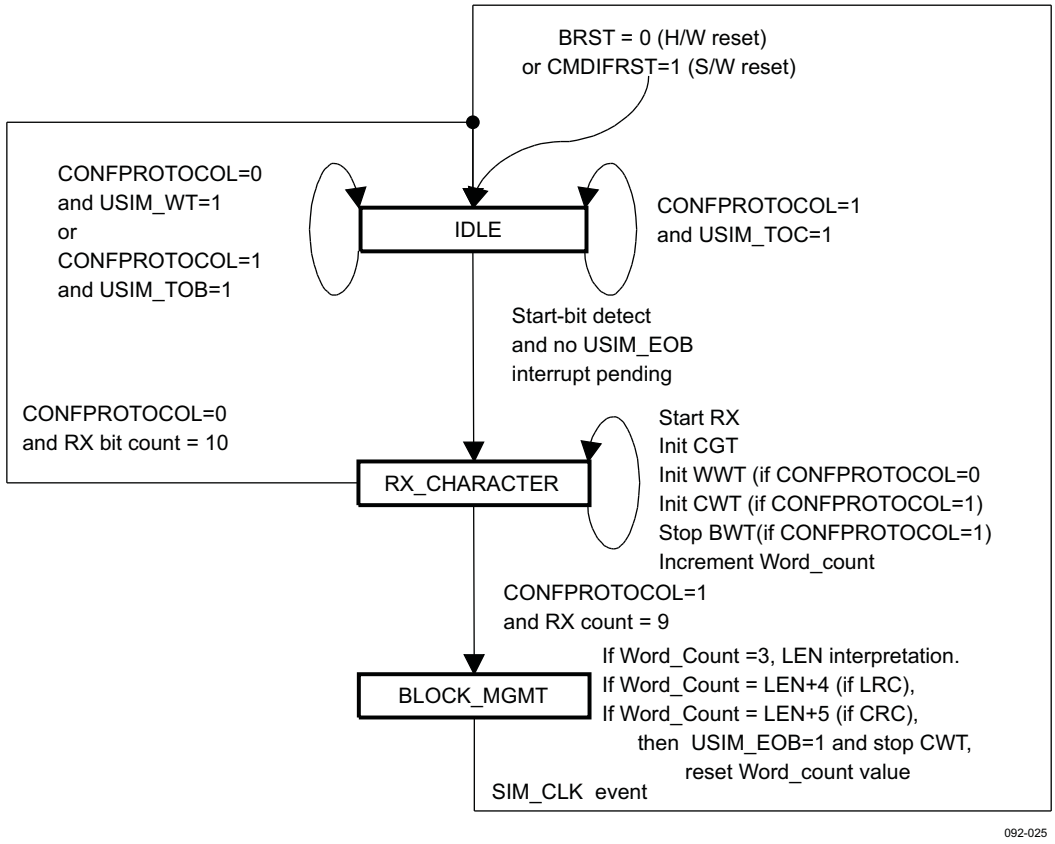
24.2.3.2.2 Reception Finite State-Machine

The states of the reception finite state-machine (RX-FSM) are listed in Table 24-8. Figure 24-25 shows the RX-FSM.

Table 24-8. RX-FSM Sequencer States

State	Description
IDLE	Idle mode (no transmission with card clock active)
RX_CHARACTER	Data reception (RX)
BLOCK_MGMT	Block counter management

Figure 24-25. RX-FSM



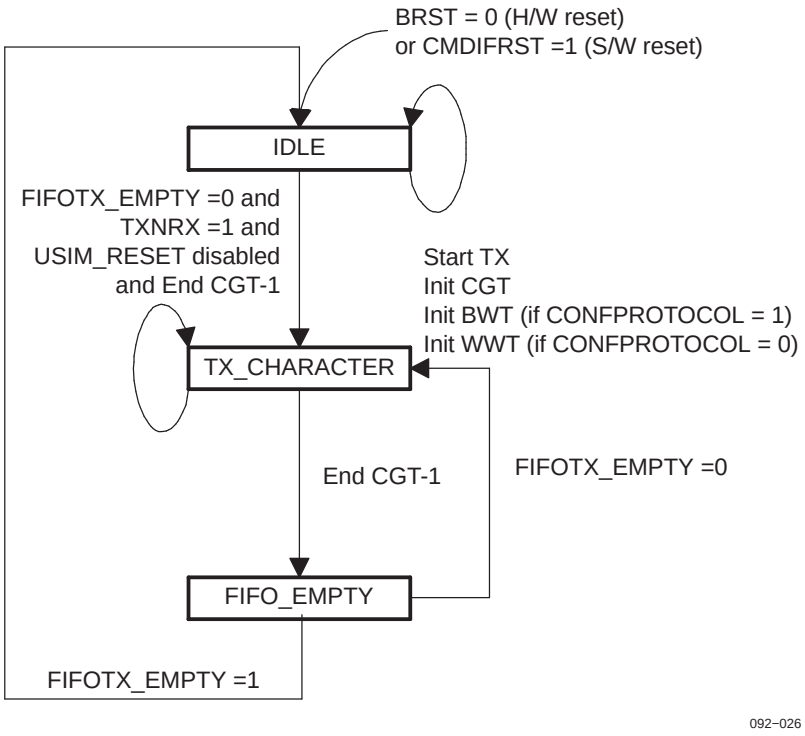
24.2.3.2.3 Transmission Finite State-Machine

The states of the transmission finite state-machine (TX-FSM) are listed in Table 24-9. Figure 24-26 shows the TX-FSM.

Table 24-9. TX-FSM Sequencer States

State	Description
IDLE	Idle mode (no transmission with card clock active)
TX_CHARACTER	Data transmission (TX)
FIFO_EMPTY	FIFO management

Figure 24-26. TX-FSM



24.2.4 Transmitter/Receiver Module

The transmitter/receiver submodule consists of the receiver block (RX), the transmitter block (TX), and the FIFO block.

24.2.4.1 Receiver Block

The receiver block implements all the features defined in ISO/GSM/3GPP standards for T = 0 and T = 1 protocols. It is scheduled on the receive ETU and sampling clocks with baud rates as defined by the F and D parameters.

Two clocks are used to interpret characters or blocks sent by the card. For details, see Section 24.2.2, *Clock Submodule*.

The main features of the receiver block are as follows:

- Decoding of convention mode (direct or inverse)
- Detection of start-bit (with Fsam1 = 8x F_{ETU})
- Sampling of character byte plus parity bit (with Fsam2 = F_{ETU})
- Check between computed and receive parity, and set parity flag STATRXPAR (USIM_DRX register)
- For T = 0 only, automatic request for character repetition if error detected
- For T = 1 only, checking the EDC and setting the parity flag with the CONFLRCCHECK bit (see the following note)
- Ready-to-read interrupt generated to the MPU
- Generating write signal to RX FIFO
- Hardware management of block length and automatic reception of a block

Note: EDC is checked by hardware if based on LRC. However, if CRC is used, EDC is checked by software. For details, see Section 24.3, *USIM Controller Programming Guide*.

24.2.4.2 Transmitter Block

The transmitter block implements all the features defined in ISO/GSM/3GPP/EMV standards for T = 0 and T = 1 protocols. It is scheduled on the transmit ETU clock with baud rates as defined by F and D parameters. See Section 24.2.2.3, *Transmission Factors (F/D)*.

The main features of the transmitter block are as follows:

- Parity computation based on inverse
- EDC computation (T = 1 only)
- Ready to transmit interrupt generated to the MPU
- Read signal generated to TXFIFO
- Hardware management of block length and automatic transmission of a block

When the USIM controller is configured in transmission mode, the transfer of a character starts when a byte is available in the FIFO (FIFO is not empty).

24.2.4.3 FIFO Block

Although transmission between the USIM controller and the Smart Card is asynchronous and half-duplex, both receive path and transmit path integrate a FIFO (two pages, FIFO_RX and FIFO_TX) because the MPU does not need to reload the last block in the FIFO when an error occurs; the USIM controller must retransmit the block.

The main features of the FIFO are as follows:

- Each page of FIFO has a 16-byte storage capacity.
- Dynamically programmable trigger level: 1 to 16 (for both FIFO_RX and FIFO_TX)
- Automatic flow control
- Flag FIFO full, FIFO empty, and FIFO trigger reached

The generation of the receive interrupt USIM_RX is related to the trigger level of the FIFO. In case of a FIFO RX full detection, a dedicated interrupt USIM_OV is generated. The four status bits FIFOTX_FULL, FIFOTX_EMPTY, FIFORX_FULL, and FIFORX_EMPTY of the USIM_FIFOS register represent the state of FIFO_TX and FIFO_RX.

The FIFO trigger level can be modified on the flow during an ongoing reception phase. This allows the FIFO size to fit the expected number of characters to be received, removing the need to wait for the WWT interrupt when the transmission is complete.

For details, see Section 24.3, *USIM Controller Programming Guide*.

24.2.5 Interrupt Controller

The interrupts for Smart Card protocol management are mapped on the ARM IRQ line, and the Smart Card detect (insertion/extraction) is mapped on the ARM FIQ. The interrupt controller is associated with IT, MASK_IT, FIFO, and STATUS registers. For register details, see Section 24.4, *USIM Controller Registers*.

The main features of the interrupt controller include the following:

- Sets IT bits in the interrupt register: USIM_IT
- Sets flag bits in the status register: USIM_STAT
- Sets flag bits in the FIFO register: USIM_FIFO
- Generates two interrupt signals (nirq and nirq_cd) to the MPU controller

24.2.6 Register Block

The register block gathers the command and status registers accessible (read/write) by the MPU:

- Command registers (CMD_START, CMD_STOP, etc.)
- Status registers (FIFO states, Parity in TX/RX, etc.)
- Status of card registers (SIM, USIM, etc.)
- Interrupt status registers (RX, TX, FIFO full, etc.)
- Interrupt mask register
- Configuration registers (coding convention, parity check, timer ranges, FIFO trigger level, control on external signals, etc.)

For register details, see Section 24.4, *USIM Controller Registers*.

24.2.7 Error Reporting

Error reporting can be divided into protocol error or time-out. The USIM_IT register contains several fields dedicated to error and time-out management.

24.2.7.1 Protocol Errors

24.2.7.1.1 TS Error

Protocol error on the first character sent on the ATR sequence is managed by the TS_ERROR bit of the USIM_IT register.

The TS character can have only two correct possibilities:

- TS = 0x3B (direct convention)
- TS = 0x3F (inverse convention)

If TS does not equal 0 x 3B or 0 x 3F, a coding convention error occurs, and an interrupt is generated to the MPU (the TS_ERROR bit in the USIM_IT register).

Note: In case of a TS protocol error, the direct convention (that is, default setup) is maintained.

24.2.7.1.2 No ATR Received

If no ATR is received during card activation, an interrupt is generated to the MPU (the USIM_NATR bit of the USIM_IT register).

24.2.7.1.3 Character Underflow

If the card sends fewer characters than the USIM controller expects, an interrupt is generated to the MPU (the USIM_WT bit of the USIM_IT register).

24.2.7.1.4 FIFO Overflow

When receiving an overflow on the FIFO (FIFO_RX is full), an interrupt flag USIM_OV is set in the USIM_IT register.

24.2.7.2 Time-Out

24.2.7.2.1 Block Time-Out

If the BWT timer reaches its time-out value, an interrupt flag is generated in the USIM_TOB bit of the USIM_IT register. The BWT timer is used only in the T = 1 protocol.

24.2.7.2.2 Character Time-Out

If the CWT timer reaches its time-out value, an interrupt flag is generated in the USIM_TOC bit of the USIM_IT register. The CWT timer is used only in the T = 1 protocol.

24.3 USIM Controller Programming Guide

24.3.1 Setup for Typical Configuration

24.3.1.1 USIM Controller Software Reset

The USIM controller can accept a general software reset that is propagated through all submodules. This reset is useful to initialize the module or after a specific issue.

A software reset is performed with the CMDIFRST bit of the USIMCMD register. The USIM controller is reset by setting the CMDIFRST bit to 1. The default value of this field is 0. Hardware automatically sets this bit to 0 when the software reset is over.

CAUTION

Do not confuse the USIM controller software reset (the CMDIFRST bit of the USIMCMD register) with the warm reset command (the WARM_RESET_CMD bit of the USIMCMD register). The warm reset sequence is not dedicated to module reset; is part of the Smart Card protocol.

24.3.1.2 Initialization of the Card and USIM Controller Registers

Before trying to communicate between the Smart Card and the USIM controller with the associated protocol, the module must be initialized in the following order:

1. USIMCMD register: software reset of the USIM controller
2. USIMCONF1 register: reset of the register
3. USIM_FIFOS register:
   a. Enable FIFO R/W access by setting the FIFO_ENABLE bit to 1 (USIM_ FIFOS[1]).
   b. RX/TX trigger values set: Enter values from 0 x 0 to 0 x F in the FIFORX_TRIGGER field (USIM_FIFOS[12:9]) and in the FIFO_TX_TRIGGER field (USIM_FIFOS[5:2]).
4. USIMCONF2 register:
   a. Smart Card clock set at 13/4 MHz according to the CONFSCLKDIV field (USIMCONF2[3:2]) value set to 0 x 01 (field reset value)
   b. sim_io line in RX mode (because of the following ATR sequence): The TXNRX bit (USIMCONF2[1]) must be set to 0 for receive mode.
   c. ATR synchronous card bypass mode activated: The ATR_ASYN_BYPASS bit (USIMCONF2[4]) set to 1. The GSM standard requires that an activation sequence for asynchronous card always follow the activation sequence for synchronous cards. However, because asynchronous cards are used most often, the USIM controller enables bypassing of the activation sequence for the synchronous card; therefore, setting the ATR_ASYN_BYPASS bit saves 40,000 clock cycles.
5. USIMCONF3 register: Set the TDUSIM field value to 1 (USIM_CONF3[7:4]). The coefficient for contact activation/deactivation is thus set to $8xT_{ETU}$.
6. USIMCMD: Start the clock module with the MODULE_CLK_EN bit at 1 (USIMCMD[3]).
7. Wait for card insertion to detect the USIM_CD interrupt provided by the TWL3031 power IC through external pin ABB_IRQ.

When these steps are complete, the ATR sequence can start.

CAUTION

The USIM controller hardware module has no input for card detection/extraction (USIM_CD signal). Card plug/unplug information must be provided by the TWL3031 power IC. Thus, the LOCOSTO processor external pad ABB_IRQ is connected to the MPU interrupt handler for USIM card presence detection (USIM_CD).

24.3.1.3 PBIAS Cell Initialization

According to the Smart Card voltage, the PBIAS cell initialization sequence is different for 1.8-V and 2.8-V configurations.

Furthermore, a USIM software application 1.8-V PBIAS cell sequence can be started while waiting for an ATR response. If no answer is received from the Smart Card, the PBIAS cell voltage can be increased to 2.8 V while continuing to wait for an ATR response. In this case, the PBIAS cell must be software-programmed on the fly. For details about the PBIAS cell, see Chapter 18, *Configuration*.

24.3.2 Operational Mode

24.3.2.1 Automatic and Bypass Modes

Two operating modes, automatic and bypass, are provided during activation and deactivation sequences:

- Automatic mode: Card activation and deactivation sequences (start and stop commands from the MPU) are hardware-monitored by the USIM controller.

  The MPU software is responsible for:

  – Initiating a transmission sequence
  – Writing and reading characters in the USIM controller address space
  – Deactivating contacts in case of card rejection
  – Handling power management in idle mode (sim_clk activation/deactivation)
  – Resetting the card (deactivation/activation) in case of consecutive error detection
- Bypass (manual) mode: The interface signals (sim_clk, sim_rst, sim_pwrctrl, and sim_io) are directly under MPU software control during card activation and deactivation.

Note: Using the hardware sequencer of the USIM controller can automatically provide accurate timings to fulfill the ISO/GSM/3GPP standards and reduce timing constraints on the MPU.

The Smart Card activation process includes the following steps:

1. Connection of the contacts
2. Activation of the contacts (card powerup)
3. Reset of the card (cold reset)
4. ATR detection

The entire process can be automatically or manually executed based on the software setup of the appropriate configuration register (the CONFBYPASS bit of the USIMCONF1 register):

- If automatic execution mode is selected (CONFBYPASS = 0), the MPU initiates the start procedure (the CMD start-bit of the command register USIMCMD), and the entire sequence is executed by the hardware sequencer embedded in the USIM controller.
- If bypass (manual) mode is selected (CONFBYPASS = 1), the MPU directly controls the activation of signals such as sim_rst, sim_pwctrl, sim_clk, sim_io level, and direction. In this mode, software fulfills the Smart Card protocol timings by the setting bits in the appropriate configuration registers:

  – USIMCONF1 register: The SVCCLEV, SRSTLEV, and SCLKLEV bits control, respectively, the logic level on VCC, sim_rst, and sim_clk signals. The CONFSI/OLOW bit is used to force the sim_io

signal low. The CONF_SCLK_EN bit is used to enable the sim_clk signal.

– USIMCONF2 register: The TXNRX bit is used to control the sim_io direction.

Note: In automatic mode, the CONF_SCLK_EN, SRSTLEV, SVCCLEV, and CONFSI/OLOW bits of the USIMCONF1 register are not used. These bit fields are used only if CONFBYPASS = 1.

24.3.2.2 Setup for Protocol Configuration

24.3.2.2.1 Parity Check

Parity is automatically calculated by the USIM controller hardware with no software calculation. Thus, software users can only define parity check and enable/disable parity check protocol.

24.3.2.2.2 Configuration and Status Bits

When the CONFCHKPAR bit (USIMCONF2[0]) is set to 1, parity check is enabled. When the bit is reset to 0, the parity check on reception frame is disabled. If parity check is activated (CONFCHKPAR = 1) by software, the STARTRXPAR status bit (USIM_DRX[8]) checks whether the parity computed automatically by hardware is correct (STARTRXPAR = 0) or if an error has occurred (STARTRXPAR = 1).

The STATTXPAR status bit (USIMSTAT[1]) is also used to provide the status of the parity check for a transmitted byte. This read-only bit checks whether the parity is correct (STATTXPAR = 1) or if an error has occurred (STATTXPAR = 0).

24.3.2.2.3 Configuration and Status Bits for T = 1 Protocol Only

In addition to the previously described configuration and status bits, other settings are required for the T = 1 protocol.

The CONFEDC and CONFLRCCHECK bits (USIMCONF2[7] and USIMCONF2[6], respectively) software-define the parity check.

Checking of the EDC is based on CRC when setting the CONFEDC bit to 1. If this bit is reset to 0, checking of the EDC is based on LRC.

Hardware checks the EDC, if based on LRC. Otherwise (if CRC is used), EDC checking is done by software.

When setting the CONFLRCCHECK bit to 1, checking of LRC is enabled. When resetting this bit to 0, no check is performed.

If EDC checking is based on LRC (CONFEDC = 0), the STATLRC status bit (USIMSTAT[2]) enables checking if LRC is OK (STATLRC = 0) or if an error has occurred (STATLRC = 1).

24.3.2.2.4 Switching Between T = 1 and T = 0 Protocols

The transmit/receive protocol can be software-programmed using the CONFPROTOCOL bit (USIMCONF2[5]).

If the CONFPROTOCOL bit field is set to 1, the T = 1 protocol is applied between the USIM controller and the Smart Card. If CONFPROTOCOL is reset to 0, the T = 0 protocol is chosen.

Note: When changing the CONFPROTOCOL bit value, mask all the interrupts to prevent unexpected transitions by using the MASK_USIM_IT register (see Table 24-32). After switching protocols, remove all status interrupts by writing in the USIM_IT register (see Table 24-29).

24.3.2.2.5 Debugging USIM Controller Sequencers

As discussed in Section 24.2.3.2, *State-Machines*, the USIM controller includes three hardware sequencers scheduled on the ETU clock to implement the Smart Card interface protocol.

The DEBUG_REG register allows reading the different states of these FSMs to help software implementation and debug as follows:

- The MAIN_STATE_DEBUG field (DEBUG_REG[3:0]) gives the current state of the main state-machine.
- The TX_STATE_MACHINE field (DEBUG_REG[5:4]) gives the current state of the transmission state-machine.
- The RX_STATE_MACHINE field (DEBUG_REG[7:6]) gives the current state of the reception state-machine.

24.3.2.3 Data Transfer Configuration

24.3.2.3.1 Data Transfer Mode

Data can be transferred two ways:

- DMA mode: The DMA_MODE bit field (USIM_FIFOS[0]) is set to 1. DMA mode is adopted for both FIFO_TX and FIFO_RX.
- TIPB interrupts: The DMA_MODE bit field is reset to 0.

For details about DMA management, see Chapter 13, *Direct Memory Access*.

Note: Both data transfer methods use FIFO management.

24.3.2.3.2 Switching from Transmit to Receive Mode

Switching modes can be software-programmed by using the TXNRX configuration bit (USIMCONF2[1]). This bit controls the data line configuration; it must be set to 1 for transmit mode and reset to 0 for receive mode.

To allow immediate switching of the data line after transmitting the last character of the block, regardless of the inertial time of the MPU to react to an interrupt, the MPU can program the data line configuration in receive mode (TXNRX = 0) during the transmission of a character. Hardware delays the physical switching of the line until the end of the transmission.

Note: The USIMCONF2 register update can occur immediately after the writing of the last character of the block to transmit.

A status bit X_MODE (USIMSTAT[4]) gives the effective direction of the SIM_I/O line. Hardware resets this status bit to 0 in receive mode and sets it to 1 in transmit mode.

24.3.2.3.3 Sleep Mode Procedure

In transmit mode, the MPU can stop the USIM card clock (sim_clk) to save power.

Note: The clock-stop procedure is allowed only if the card can support the clock-stop mode and there is no ongoing data transmission between the USIM controller and the Smart Card.

24.3.2.3.4 Clock-Stop Command

The command bit CLOCK_STOP_CMD (USIMCMD[5]) activates and deactivates the clock-stop sequence. When this bit is set to 1, clock-stop mode is activated. The CLOCK_STOP_CMD bit must be reset to 0 to stop the clock-stop sequence.

24.3.2.3.5 Guard Time Value

According to the interface protocol detailed in Section 24.1.5.6, *Sleep Mode (Clock-Stop Mode)*, the USIM controller should wait at least 1860 clock cycles after receiving the last character, including the guard time added (TC_GUARD_TIME_ADD) to the response, before it switches off the clock. Guard time equals at least 744 clock cycles. This time corresponds to the waiting time before the USIM controller sends the first command after starting the clock, as follows:

744 clock cycles = TC_GUARD_TIME_ADD = 4096 clock cycles

The guard time value is processed by either hardware or software.

The control bit SOFT_TC_GUARD_TIME_ADD_EN (TC_GUARD_ TIME_ADD[13]) enables choosing between two options:

- If SOFT_TC_GUARD_TIME_ADD_EN is set to 1, the TC_GUARD_ TIME_ADD value is provided by software with the SOFT_TC_GUARD_TIME_ADD[12:0] field value (TC_GUARD_ TIME_ADD[12:0]). The default value for this field is 0 x 02E8, which corresponds to the lower value (744 clock cycles) allowed by the ISO7816-3 protocol.
- If SOFT_TC_GUARD_TIME_ADD_EN is reset to 0 (the default value), the TC_GUARD_TIME_ADD value is provided by hardware. In this case, TC_GUARD_TIME_ADD equals the fixed value 2 ETU.

Note: For software programming of guard time, the software does not program the SOFT_TC_GUARD_TIME_ADD field with a value lower than 0x02E8 (744 clock cycles). To fulfill the ISO7816-3 norm, SOFT_TC_ GUARD_TIME_ADD must be in the range [0 × 02E8;0 × 1000].

24.3.2.3.6 Baud Rates Supported by the USIM Controller

24.3.2.3.6.1 Transmission Factors

The ETU clock (F_{ETU}) and the oversampling clock (F_{SAM}) are calculated as follows:

$F_{SAM} = {}_{FCLK}/CONF\_SAM1\_DIV$

$F_{ETU} = F_{CLK}/CONF\_ETU\_DIV$

The CONF_ETU_DIV and CONF_SAM1_DIV factors can be defined either by hardware or software. The selection is done by the SOFT_NHARD_FIDI_PROG bit (CONF5_REG[8]). When this field is reset to 0, the USIM controller uses hardware coding; setting this bit to 1 activates the software mode.

- SOFT_NHARD_FIDI_PROG = 0:
  - The hardware values are generated from the binary values FI and DI given in the TA1 character of the ATR, and by the binary coding of the clock divider for the Smart Card clock (factor N), given in the TC1 character of the ATR. For details, see Section 24.3.2.3.6.2, *Hardware Baud Rate Generation*.
  - The CONFSCLKDIV[1:0] field (USIMCONF2[3:2]) is used for the sim_clk clock.
- SOFT_NHARD_FIDI_PROG = 1:
  - The CONF_ETU_DIV[15:0] field of the CONF_ETU_DIV register is used for the ETU period.
  - The CONF_SAM1_DIV[11:0] field of the CONF_SAM1_DIV register is used to software-program the ratio F/D of the oversampling clock.

24.3.2.3.6.2 Hardware Baud Rate Generation

Several clocks are used in the USIM controller:

- $_{CLK}$ = 13 MHz, provided by FCLK of the TIPB
- F_{SCK} = 13/N MHz (N = 1, 2, 4, or 8), clock given to the Smart Card
- F_{ETU} = D/F x F_{SCK}, ETU clock
- F_{SAM} = 8 x F_{ETU}, oversampling clock for start-bit detection in receive mode

The baud rate period of the transmission clock of the data bit between the Smart Card and the USIM controller is called the ETU, which is defined by two factors:

- The clock rate conversion factor F
- The bit rate adjustment factor D

The full coverage of the baud rates range defined in the ISO 7816-3 standard for Smart Card implies supporting more than 200 baud rate values from the whole of the combination offered by the clock rate conversion factor (Fi) and the bit rate adjustment factor (Di) values.

To support as many (F, D) pairs as mentioned in ISO/IEC 7816-3, F_{ETU} and F_{SAM1} are generated independently from the 13-MHz reference clock F_{13MHz} using the following formulas:

$T_{ETU\ approximated}$ = Round [(F/D) x N] x T_{13MHz}, approximated ETU clock

$T_{SAM\ approximated}$ = Round [(F/D) x (N/8)] x T_{13MHz}, approximate SAM clock

Note: The hardware baud rate generator is enabled only when SOFT_NHARD_FIDI_PROG = 0.

Despite the rounding of the ETU and SAM clock periods, the generation of the data samples to the Smart Card and the sampling of the data samples received from the Smart Card is acceptable if the timing requirements of the ISO 7816-3 standard are fulfilled:

- The accuracy of bit positioning in transmit mode must be equal to or greater than ± 0.2 x ETU.
- The accuracy of bit sampling in receive mode must be within a ± 0.3x ETU window versus the middle of the received bit.

To compute the ETU, the pair of (F/D) factors can equal the following values:

- Fd and Dd default values equal to (372/1)
- Fi and Di specific values as defined in TA1
- Fn and Dn negotiated values after a PPS

The default pair value (Fd/Dd) is used during the ATR. If the TA1 byte is part of the ATR, the subsequent communication uses either the specific pair value (Fi/Di) or an implicit pair value preprogrammed in the USIM controller, depending on the value of bit 5 of the TA2 byte.

If the TA2 byte is missing from the ATR, the USIM controller can initiate a PPS and propose to the card a pair value (Fn/Dn) with Fn and Dn in the range Fd to Fi and Dd to Di, respectively. If the card accepts the proposed pair value (Fn/Dn), this pair is used to define the ETU for subsequent data exchanges.

Table 24-10 provides the Fi clock rate conversion factor. Table 24-11 provides the Di bit rate adjustment factor.

Table 24-10. Fi Clock Rate Conversion Factor

FI	**0000**	**0001**	**0010**	**0011**	**0100**
Fi	372	372	558	744	1116
f (max) MHz	4	5	6	8	12
FI	**0101**	**0110**	**0111**	**1000**	**1001**
Fi	1488	1860	RFU	RFU	512
f (max) MHz	16	20	----	----	5

FI	1010	1011	1100	1101	111x
Fi	768	1024	1536	2048	RFU(1)
f (max) MHz	7,5	10	15	20	----

(1) RFU: reserved for future use

Table 24-11. Di Bit Rate Adjustment Factor

DI	0000	0001	0010	0011	0100
Di	RFU	1	2	4	8

DI	0101	0110	0111	1000	1001
Di	16	32	RFU	12	20

DI	1010	1011	1100	1101	111x
Di	RFU	RFU	RFU	RFU	RFU (1)

(1) RFU: reserved for future use

24.3.2.3.6.3 (F/D) Pairs Supported by the USIM Controller in Hardware Mode

This section is dedicated to the hardware generation of ETU and SAM clocks only when SOFT_NHARD_FIDI_PROG = 0.

The integer dividing ratios selected for the generation of the ETU and SAM clocks must cope with the time tolerances previously defined.

For each possible Smart Card system clock value (divided by 1, 2, 4, or 8), a matrix can be created with the DIV_SAM and DIV_ETU rounded ratios corresponding to each (Fi/Di) pairs, such as:

$F_{SAM} = F_{CLK}/DIV\_SAM$

$F_{ETU} = F_{CLK}/DIV\_ETU$

Table 24-12 through Table 24-19 list which (F,D) pairs can be used. The grayed values indicate invalid configurations for which FSCLK = F13MHz x (1/N) exceeds the fmax value specified in the ISO 7816-3 standard for a given Fi. The remaining values indicate the (F,D) pairs that can be supported by the USIM controller with the selected card clock frequency.

F_{SCLK} = 13/1 MHz (N = 1)

The divider factor (DIV_ETU) for ETU clock generation from a 13-MHz clock (N - 1) is shown in Table 24-12.

Table 24-12. T_{ETU} Approximated/$T_{13\ MHz}$ for N = 1

	$T_{ETU\ Approximated}/T_{13MHz}$							
D =	1	2	4	8	16	32	12	20
F =	N = 1							
372	372	186	93	47	23	12	31	19
558	558	279	140	70	35	17	47	28
744	744	372	186	93	47	23	62	37
1116	1116	558	279	140	70	35	93	56
1488	1488	744	372	186	93	47	124	74
1860	1860	930	465	233	116	58	155	93
512	512	256	128	64	32	16	43	26
768	768	384	192	96	48	24	64	38

Table 24-12. T_{ETU} Approximated/$T_{13\ MHz}$ for N = 1 (continued)

1024	1024	512	256	128	64	32	85	51
1536	1536	768	384	192	96	48	128	77
2048	2048	1024	512	256	128	64	171	102

The divider factor (DIV_SAM) for the start-bit sample clock generated from a 13-MHz clock (N = 1) is shown in Table 24-13.

Table 24-13. $T_{SAM\ Approximated}/T_{13\ MHz}$ for N = 1

	$T_{SAM\ approximated}/T_{13MHz}$							
D =	**1**	**2**	**4**	**8**	**16**	**32**	**12**	**20**
F =	**N = 1**							
372	47	23	12	6	33	1	4	2
558	70	35	17	9	4	2	6	3
744	93	47	23	12	6	3	8	6
1116	140	70	35	17	9	4	12	7
1488	186	93	47	23	12	6	16	9
1860	233	116	58	29	15	7	19	12
512	64	32	16	8	4	2	5	3
768	96	48	24	12	6	3	8	5
1024	128	64	32	16	8	4	11	6
1536	192	96	48	24	12	6	16	10
2048	256	128	64	32	16	8	21	13

The divider factor (DIV_ETU) for ETU clock generation from a 13-MHz clock (N = 2) is shown in Table 24-14.

Table 24-14. $T_{ETU\ Approximated}/T_{13\ MHz}$ for N = 2

	$T_{ETU\ approximated}/T_{13MHz}$							
D =	**1**	**2**	**4**	**8**	**16**	**32**	**12**	**20**
F =	**N = 2**							
372	744	372	186	93	47	23	62	37
558	1116	558	279	140	70	35	93	56
744	1488	744	372	186	93	47	124	74
1116	2232	1116	558	279	140	70	186	112
1488	2976	1488	744	372	186	93	248	149
1860	3720	1860	930	465	233	116	310	186
512	1024	512	256	128	64	32	85	51
768	1536	768	384	192	96	48	128	77
1024	2048	1024	512	256	128	64	171	102
1536	3072	1536	768	384	192	96	256	154
2048	4096	2048	1024	512	256	128	341	205

The divider factor (DIV_SAM) for the start-bit sample clock generated from a 13-MHz clock (N = 2) is shown in Table 24-15.

Table 24-15. $T_{SAM\ Approximated}/T_{13\ MHz}$ for N = 2

	$T_{SAM\ Approximated}/T_{13MHz}$							
D =	**1**	**2**	**4**	**8**	**16**	**32**	**12**	**20**
F =	**N = 2**							
372	93	47	23	12	6	3	8	6
558	140	70	35	17	9	4	12	7
744	186	93	47	23	12	6	16	9
1116	279	140	70	35	17	9	23	14
1488	372	186	93	47	23	12	31	19
1860	465	233	116	58	29	15	39	23
512	128	64	32	16	6	4	11	6
768	192	96	48	24	12	6	16	10
1024	256	128	64	32	16	8	21	13
1536	384	192	96	48	24	12	32	19
2048	512	256	128	64	32	16	43	26

The divider factor (DIV_ETU) for ETU clock generation from a 13-MHz clock (N = 4) is shown in Table 24-16.

Table 24-16. $T_{ETU\ Approximated}/T_{13\ MHz}$ for N = 4

	$T_{ETU\ Approximated}/T_{13MHz}$							
D =	**1**	**2**	**4**	**8**	**16**	**32**	**12**	**20**
F =	**N = 4**							
372	1488	744	372	186	93	47	124	74
558	2232	1116	558	279	140	70	186	112
744	2976	1488	744	372	186	93	248	149
1116	4464	2232	1116	558	279	140	372	223
1488	5952	2976	1488	744	372	186	296	298
1860	7440	3720	1860	930	465	233	620	372
512	2048	1024	512	256	128	64	171	102
768	3072	1536	768	384	192	96	256	154
1024	4096	2048	1024	512	256	128	341	205
1536	6144	3072	1536	768	384	192	512	307
2048	8192	4096	2048	1024	512	256	683	410

The divider factor (DIV_SAM) for the start-bit sample clock generated from a 13-MHz clock (N = 4) is shown in Table 24-17.

Table 24-17. $T_{SAM\ Approximated}/T_{13\ MHz}$ for N = 4

	$T_{SAM\ Approximated}/T_{13MHz}$							
D =	**1**	**2**	**4**	**8**	**16**	**32**	**12**	**20**
F =	**N = 4**							
372	186	93	47	23	12	6	16	9
558	279	140	70	35	17	9	23	14
744	372	186	93	47	23	12	31	19
1116	558	279	140	70	35	17	47	28
1488	744	372	186	93	47	23	62	37
1860	930	465	233	116	58	29	78	47
512	256	128	64	32	16	8	21	13
768	384	192	96	48	24	12	32	19

Table 24-17. $T_{SAM\ Approximated}/T_{13\ MHz}$ for N = 4 (continued)

1024	512	256	128	64	32	16	43	26
1536	768	384	192	96	48	24	64	38
2048	1024	512	256	128	64	32	85	51

The divider factor (DIV_ETU) for ETU clock generation from a 13-MHz clock (N = 8) is shown in Table 24-18.

Table 24-18. $T_{ETU\ Approximated}/T_{13\ MHz}$ for N = 8

	$T_{ETU\ Approximated}/T_{13MHz}$							
D =	**1**	**2**	**4**	**8**	**16**	**32**	**12**	**20**
F =	**N = 8**							
372	2976	1488	744	372	186	93	248	149
558	4464	2232	1116	558	279	140	372	223
744	5952	2976	1488	744	372	186	496	298
1116	8928	4464	2232	1116	558	279	744	446
1488	11904	5952	2976	1488	744	372	993	595
1860	14880	7440	3720	1860	930	465	1240	744
512	4096	2048	1024	512	256	128	341	205
768	6144	3072	1536	768	384	192	512	307
1024	8192	4096	2048	1024	512	256	683	410
1536	12288	6144	3072	1536	778	384	1024	614
2048	16384	8192	4096	2048	1024	512	1365	819

The divider factor (DIV_SAM) for the start-bit sample clock generated from a a 13-MHz clock (N = 8) is shown in Table 24-19.

Table 24-19. $T_{SAM\ Approximated}/T_{13\ MHz}$ for N = 8

	$T_{SAM\ Approximated}/T_{13MHz}$							
D =	**1**	**2**	**4**	**8**	**16**	**32**	**12**	**20**
F =	**N = 8**							
372	372	186	93	47	23	23	31	19
558	558	279	140	70	35	17	47	28
744	744	372	186	93	47	23	62	37
1116	1116	558	279	140	70	35	93	56
1488	1488	744	372	186	93	47	124	74
1860	1860	930	465	233	116	58	155	93
512	512	256	128	64	32	16	43	26
768	768	384	192	96	48	24	64	38
1024	1024	512	256	128	64	32	85	51
1536	1536	768	384	192	96	48	128	77
2048	2048	1024	512	256	128	64	171	102

24.3.2.4 FIFO Management

The USIM_FIFOS 16-bit register is dedicated to management of the two FIFOs (FIFO_RX and FIFO_TX). Each FIFO is 16 bytes deep.

The FIFO_ENABLE bit field (USIM_FIFOS[1]) allows access to both FIFO_RX and FIFO_TX. Setting this bit to 1 allows read/write access to both FIFOs. When reset to 0, read/write access to the FIFOs is forbidden.

24.3.2.4.1 FIFO_RX

Reception FIFO management is provided by the following fields:

- FIFORX_FULL bit (USIM_FIFOS[15]): When this read-only bit is hardware-set to 1, reception FIFO is full and the USIM_OV interrupt flag is set in the USIM_IT register.
- FIFORX_EMPTY bit (USIM_FIFOS[14]): When this read-only bit is hardware-set to 1, reception FIFO is empty.
- FIFORX_RESET bit (USIM_FIFOS[13]): Setting this bit to 1 resets all the FIFO_RX pointers.
- FIFORX_TRIGGER[3:0] field (USIM_FIFOS[12:9]) contains the value of the FIFO_RX trigger. When this value is reached in FIFO_RX, the USIM_RX interrupt flag is generated in the USIM_IT register. The FIFO trigger level can be modified on the flow during the ongoing reception phase. This allows the FIFO size to fit the expected number of characters to be received, removing the need to wait for the WWT interrupt when the transmission is complete.

24.3.2.4.2 FIFO_TX

Transmission FIFO management is provided by the following fields:

- FIFOTX_FULL bit (USIM_FIFOS[8]): When this read-only bit is hardware-set to 1, transmission FIFO is full.
- FIFOTX_EMPTY bit (USIM_FIFOS[7]): When this read-only bit is hardware-set to 1, transmission FIFO is empty.
- FIFOTX_RESET bit (USIM_FIFOS[6]): Setting this bit to 1 resets all the TX_FIFO pointers.
- FIFOTX_TRIGGER[3:0] field (USIM_FIFOS[5:2]) contains the value of the TX_FIFO trigger. When this value is reached in TX_FIFO, the USIM_TX interrupt flag is generated in the USIM_IT register.

24.3.2.5 Interrupt Handling

24.3.2.5.1 USIM_CD

The USIM_CD interrupt is provided by the TWL3031 power IC, which generates an external interrupt to the MPU interrupt handler to inform about any extraction or insertion of a Smart Card. This interrupt is provided by an external pin (ABB_IRQ) of the LOCOSTO device.

Texas Instruments provides a global solution by associating the LOCOSTO processor with the TWL3031 power IC. Contact your TI representative for information about the TWL3031 IC.

Note: The interrupt used for USIM card plug/unplug detection is not dedicated to this module, and other interrupts are generated by the TWL3031 IC on pin ABB_IRQ of the LOCOSTO device. When an interrupt is detected on the ABB_IRQ pad, software users must first read the TWL3031 IC registers to determine the cause of the interrupt. Then, the MPU acts according to the interrupt source; for the USIM_CD interrupt, the MPU initializes (if insertion) or deactivates (if extraction) the USIM controller. To understand card insertion and extraction, software users must read the interrupt register for IRQ_12 (MPU). For details, see Chapter 12, *Interrupt Handlers*.

24.3.2.5.2 USIM_IT

USIM_IT is an interrupt generated by the USIM controller hardware module. USIM_IT is mapped on the MPU interrupt handler (IRQ_24).

When an interrupt request occurs, the source is identified in the USIM controller register USIM_IT.

Read the USIM_IT register to identify the source of the interrupt:

- The TS_ERROR interrupt flag (USIM_IT[10]) is used only in the T = 1 protocol to indicate a decoding error on a TS character.

- The USIM_RESENT interrupt flag (USIM_IT[9]) is used only in the T = 0 protocol to indicate how many times a character is sent to the card without response. This interrupt flag is linked to the CONFRESENT[2:0] field of the USIMCONF2 register.
- The USIM_TOB, USIM_TOC, and USIM_EOB interrupt flags (USIM_IT[8], USIM_IT[7], and USIM_IT[6]) are used only in the T = 1 protocol to inform software users of a time-out block, time-out character, and end-of-block, respectively.
- The USIM_RX interrupt flag (USIM_IT[4]) informs software users when the FIFORX_TRIGGER[3:0] threshold is reached (USIM_FIFOS register).
- The USIM_TX interrupt flag (USIM_IT[3]) informs software users when the FIFOTX_TRIGGER[3:0] threshold is reached (USIM_FIFOS register).
- The USIM_OV interrupt flag (USIM_IT[2]) is used to inform software users that FIFO_RX is full.
- The USIM_WT bit (USIM_IT[1]) is an interrupt flag for character underflow.
- The USIM_NATR bit (USIM_IT[0]) is an interrupt flag to inform software users that the ATR has not been received.

USIM_IT is reset on write access to the dedicated register field. For example, to reset the USIM_OV interrupt flag in case of FIFO_RX overflow, software users must write 0 or 1 to the USIM_OV bit. This USIM_IT interrupt can be masked by using the MASK_USIM_IT register fields. All interrupt flags are masked by default.

24.3.2.6 Timer Software Configuration

24.3.2.6.1 Work Waiting Time (T = 1 Protocol Only)

The WWT value is used in the ATR sequence; WWT is defined as follows:

WWT = CONFWAITI x 960 x D x ETU

CONFWAITI can be software-programmed using the CONFWAITI[12:0] field of the CONF4_REG register. The default value of the CONFWAITI bit is 0x00A.

The WWT timer range for different bit rate adjustment factors (D) is listed in Table 24-20.

Table 24-20. WWT Timer (Min/Max) Range

D =	1		2		4		8	
F =	Min WI=1	Max WI=256	Min WI=0	Max WI=255	Min WI=0	Max WI=255	Min WI=0	Max WI=255
Don't care	960	244,800	1920	489,600	3840	979,200	7680	1,958,400

D =	16		32		12		20	
F =	Min WI=0	Max WI=255	Min WI=0	Max WI=255	Min WI=0	Max WI=255	Min WI=0	Max WI=255
Don't care	15,360	3,916,800	30,720	7,833,600	11,520	2,937,600	19,200	4,896,000

24.3.2.6.2 Character Guard Time (T = 0 Protocol Only)

The parameter N (additional guard time) is defined by the TC1 character of the received ATR. This parameter N of the CGT timer can be software-programmed. It corresponds to the CGT[7:0] field defined in the USIM_CGT[7:0] register. The default value of the CGT field is 0x0D, which corresponds to 13 ETU.

24.3.2.6.3 Character Waiting Time (T = 1 Protocol Only)

Character waiting time is defined as follows:

$$CWT = CWT\_value \times T_{etu}$$

with CWT_value = 11 + 2CWI (for the USIM card) CWT_value in the range [12,43]. CWT_value is software-configurable by the CWT[15:0] field defined in the USIM_CWT[15:0] register.

The default value of the CWT field is 8203 ETU (0 x 200B) for ISO 7816-3. The value written in this register corresponds to the 2_{CWI} term in the previous formula. The USIM controller automatically adds 11 to the value written to the register.

24.3.2.6.4 Block Waiting Time (T = 1 Protocol Only)

The BWT value is defined in the ATR received sequence; BWT is based on the BWI parameter from the first Tbi character (the 4 MSBs) of the ATR, after the first occurrence of T = 1 in TD_{i-1} for i > 2.

BWT is given by following formula:

$$BWT = (11 + 2_{CONFBWI} \times 960 \times 372 \times (D/F)) \text{ ETU}$$

The CONFBWI value is software-programmed in the BWT_LSB[15:0] and BWT_MSB[6:0] fields. The BWT_LSB field is defined in the USIM_BWT_LSB[15:0] register and the BWT_MSB field is defined in the USIM_BWT_MSB[6:0] register. The default value of both BWT_LSB and BWT_MSB is 0.

The BWT timer range for different clock rate conversion factors (F) and bit rate adjustment factors (D) is listed in Table 24-21.

Table 24-21. BWT Timer (Min/Max) Range

D =	1		2		4		8	
F =	Min BWI=1	Max BWI=9	Min BWI=0	Max BWI=9	Min BWI=0	Max BWI=9	Min BWI=0	Max BWI=9
372	971	491,531	1931	983,051	3851	1,966,091	7691	3,932,171
558	651	327,691	1291	655,371	2571	1,310,731	5131	2,62,451
744	491	245,771	971	491,531	1931	938,051	3851	1,966,091
1116	331	163,851	651	327,691	1291	655,371	2571	1,310,731
1488	251	122,891	491	245,771	971	491,531	1931	983,051
1060	203	98,315	395	196,619	779	393,227	1547	786,443
512	709	357,131	1406	714,251	2801	1,428,491	5591	2,856,971
768	476	238,091	941	476,171	1871	952,331	3731	1,904,651
1024	360	178,571	709	357,131	1406	714,251	2801	1,428,491
1536	244	119,051	476	238,091	941	476,171	1871	952,331
2048	185	89,291	360	178,571	709	357,131	1406	714,251
D =	16		32		12		20	
F =	Min BWI=0	Max BWI=9	Min BWI=0	Max BWI=9	Min BWI=0	Max BWI=9	Min BWI=0	Max BWI=9
372	15,371	7,864,331	30,731	15,728,651	11,531	5,898,251	19,211	9,830,411
558	10,251	5,242,891	20,491	10,485,771	7961	3,932,171	12,811	6,553,611
744	7691	3,932,171	15,371	7,864,331	5771	2,949,131	9611	4,915,211
1116	5131	2,621,451	10,251	5,242,391	3851	1,966,091	6411	3,276,811
1488	3851	1,966,091	7691	3,932,171	2891	1,474,571	4811	2,457,611
1060	3083	1,572,875	6155	3,145,739	2315	1,179,659	3851	1,966,091
512	11,171	5,713,931	22,331	11,427,851	8381	4,285,451	13,961	7,142,411
768	7451	3,809,291	14,891	7,618,571	5591	2,856,971	9311	4,761,611
1024	5591	2,856,971	11,171	5,713,931	4196	2,142,731	6986	3,571,211
1536	3731	1,904,651	7451	3,809,291	2801	1,428,491	4661	2,380,811
2048	2801	1,428,491	5591	2,856,971	2104	1,071,371	3499	1,785,611

24.3.2.7 Error Identification Process

As described in Section 24.2.7, *Error Reporting*, two types of errors can be detected:

- Protocol error integrates the four following errors:
  - No ATR received (see the TS_ERROR bit of the USIM_IT register)
  - Character overflow (see the USIM_NATR bit of the USIM_IT register)
  - Character underflow (see the USIM_WT bit of the USIM_IT register)
  - FIFO overflow (see the USIM_OV bit of the USIM_IT register)
- Time-out error concerns the two following errors:
  - Block time-out (see the USIM_TOB bit of the USIM_IT register)
  - Character time-out (see the USIM_TOC bit of the USIM_IT register)

24.4 USIM Controller Registers

Table 24-22 is the instance summary.

Table 24-22. Instance Summary

Module Name	MPU Base Address	Size
USIM	0xFFFF A800	2K bytes

USIM is TIPB strobe 0 peripheral; therefore, it is accessible by both the MPU and the DMA controller.

Table 24-23 lists the USIM controller registers. Table 24-24 through Table 24-44 describe the register bits.

Table 24-23. USIM Controller Registers

Register	Description	Type	Register Width (Bits)	Offset
USIMCMD	USIM control and command	RW	16	0x00
USIMSTAT	USIM status	R	16	0x02
USIMCONF1	USIM configuration 1	RW	16	0x04
USIMCONF2	USIM configuration 2	RW	16	0x06
USIMCONF3	USIM configuration 3	RW	16	0x08
USIM_IT	USIM interrupt	RW	16	0x0A
USIM_DRX	USIM received byte	R	16	0x0C
USIM_DTX	USIM transmitted byte	RW	16	0x0E
USIM_MASK_IT	USIM mask interrupt	RW	16	0x10
USIM_FIFOS	USIM RX/TX FIFO management	RW	16	0x12
USIM_CGT	USIM character guard time	RW	16	0x14
USIM_CWT	USIM character waiting time	RW	16	0x16
USIM_BWT_LSB	USIM block waiting time LSB	RW	16	0x18
USIM_BWT_MSB	USIM block waiting time MSB	RW	16	0x1A
DEBUG_REG	USIM debug	R	16	0x1C
CONF_SAM1_DIV	SAM clock ratio configuration	RW	16	0x1E
CONF4_REG	USIM configuration 4	RW	16	0x20
ATR_CLK_PRD_NBS	Clock period before ATR receipt	R	16	0x22
CONF_ETU_DIV	ETU clock period configuration	RW	16	0x24
CONF5_REG	USIM configuration 5	RW	16	0x26
TC_GUARD_TIME_ADD	USIM guard time	RW	16	0x28

Table 24-24. USIM Control and Command Register (USIMCMD)

Address Offset	0x00		
Physical Address	MPU: 0xFFFF A800	**Instance**	USIM
Description	Control and command register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CLOCK_STOP_CMD	WARM_RESET_CMD	MODULE_CLK_EN	CMDSTART	CMDSTOP	CMDIFRST

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Read returns 0.	R	0x0
5	CLOCK_STOP_CMD	0: Clock-stop sequence deactivated 1: Clock-stop sequence activated This field has to be reset to 0 to stop the clock-stop mode.	RW	0
4	WARM_RESET_CMD	0: Warm reset sequence deactivated 1: Warm reset sequence activated This field must be reset to 0 to stop the warm reset mode	RW	0
3	MODULE_CLK_EN	0: Module clock deactivated 1: Module clock activated	RW	0
2	CMDSTART	0: Start procedure for USIM card deactivated 1: Start procedure for USIM card activated	RW	0
1	CMDSTOP	0: Stop procedure for USIM card deactivated 1: Stop procedure for USIM card activated	RW	0
0	CMDIFRST	0: No action 1: Software reset of USIM module	RW	0

Table 24-25. USIM Status Register (USIMSTAT)

Address Offset	0x02		
Physical Address	MPU: 0xFFFF A802	**Instance**	USIM
Description	Status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											X_MODE	CONFCODCONV	STATLRC	STATXPAR	Reserved

Bits	Field Name	Description	Type	Reset
15:5	Reserved	Read returns 0.	R	0x000
4	X_MODE	Status of I/O line effective direction 1: Transmission mode 0: Reception mode	R	1
3	CONFCODCONV	Status of the coding convention used by USIM card 0: Direct convention 1: Inverse convention	R	0
2	STATLRC	LRC check status on received block 0: LRC OK 1: LRC error This field is relevant only with T = 1 protocol and edc_type = LRC	R	0
1	STATXPAR	Status on parity check for transmitted byte 1: Parity OK 0: Parity error	R	1
0	Reserved	Read returns 0.	R	0

Table 24-26. USIM Configuration 1 Register (USIMCONF1)

Address Offset	0x04		
Physical Address	MPU: FFFF A804	**Instance**	USIM
Description	Configuration 1 register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CONF_SCLK_EN	SRSTLEV	SVCCLEV	CONFBYPASS	CONFSI/OLOW	SCLKLEV

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Read returns 0.	R	0x0
5	CONF_SCLK_EN	0: SIM clock disabled 1: SIM clock enabled This field is used only when CONFBYPASS = 1.	RW	0
4	SRSTLEV	0: Low logic level on SRST signal 1: High logic level on SRST signal This field is used only when CONFBYPASS = 1.	RW	0
3	SVCCLEV	0: Low logic level on SVCC signal 1: High logic level on SVCC signal This field is used only when CONFBYPASS = 1.	RW	0
2	CONFBYPASS	0: No action 1: Bypass hardware timers (tdusim) and start/stop sequence.	RW	0
1	CONFSI/OLOW	0: No action 1: Force SIO signal to low level. This field is used only when CONFBYPASS = 1.	RW	0
0	SCLKLEV	USIM clock idle level 0: Low level 1: High level	RW	0

Table 24-27. USIM Configuration 2 Register (USIMCONF2)

Address Offset	0x04		
Physical Address	MPU: 0xFFFF A806	**Instance**	USIM
Description	Configuration 2 register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CONFRESENT			CONFLRCCHECK	CONFEDC	CONFPROTOCOL	ATR_ASYN_BYPASS	CONFSCLKDIV		TXNRX	CONFCHKPAR

Bits	Field Name	Description	Type	Reset
15:11	Reserved	Read returns 0.	R	0x0
10:8	CONFRESENT	Number of automatic resents on parity error in T = 0 protocol This field is used only in T = 0 protocol.	RW	0
7	CONFLRCCHECK	0: LRC check disabled 1: LRC check enabled This field is used only in T = 1 protocol.	RW	0
6	CONFEDC	0: LRC control activated 1: CRC control activated This field is used only in T = 1 protocol.	RW	0
5	CONFPROTOCOL	0: T = 0 protocol activated 1: T = 1 protocol activated When changing this field, mask all interrupts to avoid unexpected transitions.	RW	0
4	ATR_ASYN_ BYPASS	0: Enable synchronous and asynchronous ATR checks. 1: Bypass ATR waiting sequence for synchronous USIM cards.	RW	0
3:2	CONFSCLKDIV	Frequency of the SIM card clock 00: 13MHz/2 01: 13MHz/4 10: 13MHz/8 11: 13MHz	RW	01
1	TXNRX	SI/O line direction (control bit) 0: Receive mode 1: Transmit	RW	1
0	CONFCHKPAR	0: Parity check on RW reception disabled 1: Parity check on RW reception enabled	RW	0

Table 24-28. USIM Configuration 3 Register (USIMCONF3)

Address Offset	0x08		
Physical Address	MPU: 0xFFFF A808	**Instance**	USIM
Description	Configuration 3 register for time/delay parameters		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TDUSIM				Reserved			

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x0
7:4	TDUSIM	Coefficient value for contact activation/deactivation in duration time unit = 8 _UnicodeEncodeError_ T_{etu}	RW	0x4
3:0	Reserved	Read returns 0.	R	0x0

Note: Activation/deactivation time step: TDUSIM = (CONFTDUSIM + 1) * 8 * T_{etu}

Table 24-29. USIM Interrupt Register (USIM_IT)

Address Offset	0x0A		
Physical Address	MPU: 0xFFFF A80A	**Instance**	USIM
Description	Interrupt register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					TS_ERROR	USIM_RESENT	USIM_TOB	USIM_TOC	USIM_EOB	Reserved	USIM_RX	USIM_TX	USIM_OV	USIM_WT	USIM_NATR

Bits	Field Name	Description	Type	Reset
15:11	Reserved	Read returns 0.	R	0x0
10	TS_ERROR	Pending interrupt status for TS_decode error (T = 1 protocol only) Read 0: No interrupt event pending Read 1: TS_decode error interrupt event pending Write 0 or 1: Status bit cleared	RW	0
9	USIM_RESENT	Pending interrupt status for CONFRESENT error (T = 0 protocol only) Read 0: No interrupt event pending Read 1: CONFRESENT error interrupt event pending Write 0 or 1: Status bit cleared	RW	0
8	USIM_TOB	Pending interrupt status for time-out block (BWT) (T = 1 protocol only) Read 0: No interrupt event pending Read 1: Time-out block interrupt event pending	RW	0
7	USIM_TOC	Pending interrupt status for Time-out character (CWT) (T = 1 protocol only) Read 0: No interrupt event pending Read 1: Time-out character interrupt event pending Write 0 or 1: Status bit cleared	RW	0
6	USIM_EOB	Pending interrupt status for end-of-block receive (T = 1 protocol only) Read 0: No interrupt event pending Read 1: End-of-block interrupt event pending Write 0 or 1: Status bit cleared	RW	0
5	Reserved	Read returns 0.	R	0
4	USIM_RX	Pending interrupt status for maximum number of characters waiting to be read (FIFORX_ TRIGGER threshold) Read 0: No interrupt event pending Read 1: FIFORX trigger interrupt event pending Write 0 or 1: Status bit cleared	RW	0
3	USIM_TX	Pending interrupt status for minimum number of characters waiting to be transmitted (FIFOTX_ TRIGGER threshold) Read 0: No interrupt event pending Read 1: FIFOTX trigger interrupt event pending Write 0 or 1: Status bit cleared	RW	0
2	USIM_OV	Pending interrupt status for receive FIFO overflow	RW	0

Bits	Field Name	Description		Type	Reset
		Read 0:	No interrupt event pending		
		Read 1:	FIFO_RX overflow interrupt event pending		
		Write 0 or 1:	Status bit cleared		
1	USIM_WT	Pending interrupt status for character underflow		RW	0
		Read 0:	No interrupt event pending		
		Read 1:	Character underflow event pending		
		Write 0 or 1:	Status bit cleared		
0	USIM_NATR	Pending interrupt status for no ATR		RW	0
		Read 0:	No interrupt event pending		
		Read 1:	No ATR interrupt event pending		
		Write 0 or 1:	Status bit cleared		
		Write 0 or 1:	Status bit cleared		

Table 24-30. USIM Received Byte Register (USIM_DRX)

Address Offset	0x0C		
Physical Address	MPU: 0xFFFF A80C	**Instance**	USIM
Description	Received byte register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							STATRXPAR	USIM_DRX							

Bits	Field Name	Description	Type	Reset
15:9	Reserved	Read returns 0.	R	0x00
8	STATRXPAR	Parity check for received byte	R	1
		1: Correct parity check		
		0: Wrong parity check		
7:0	USIM_DRX	Next data byte in the FIFO ready to be read	R	unknown

Table 24-31. USIM Transmitted Byte Register (USIM_DTX)

Address Offset	0x0E		
Physical Address	MPU: 0xFFFF A80E	**Instance**	USIM
Description	Transmitted byte register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								USIM_DTX							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7:0	USIM_DTX	Next data byte in the FIFO ready to be transmitted	W	unknown

Table 24-32. USIM Interrupt Mask Register (USIM_MASK_IT)

Address Offset	0x10		
Physical Address	MPU: 0xFFFF A810	**Instance**	USIM
Description	Interrupt mask register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				Reserved	MASK_USIM_TS_ ERROR	MASK_USIM_ RESENT	MASK_USIM_TOB	MASK_USIM_TOC	MASK_USIM_EOB	Reserved	MASK_USIM_RX	MASK_USIM_TX	MASK_USIM_OV	MASK_USIM_WT	MASK_USIM_NATR

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Read returns 0.	R	0x0
11	Reserved	Read returns 1.	R	1
10	MASK_USIM_TS_ ERROR	Interrupt mask for TS_DECODE error 0: TS_DECODE error interrupt unmasked 1: TS_DECODE error interrupt masked	RW	1
9	MASK_USIM_ RESENT	Interrupt mask for CONF_RESENT reached 0: CONF_RESENT interrupt unmasked 1: CONF_RESENT interrupt masked	RW	1
8	MASK_USIM_TOB	Interrupt mask for time-out block (BWT) (only in T = 1 protocol) 0: Time-out block interrupt unmasked 1: Time-out block interrupt masked	RW	1
7	MASK_USIM_TOC	Interrupt mask for time-out character (only in T = 1 protocol) 0: Time-out character interrupt unmasked 1: Time-out character interrupt masked	RW	1
6	MASK_USIM_EOB	Interrupt mask for end of block received (only in T = 1 protocol) 0: End-of-block interrupt unmasked 1: End-of-block interrupt masked	RW	1
5	Reserved	Read returns 1.	R	1
4	MASK_USIM_RX	Interrupt mask for characters waiting to be read 0: FIFORX_TRIGGER interrupt unmasked 1: FIFORX_TRIGGER interrupt masked	RW	1
3	MASK_USIM_TX	Interrupt mask for characters waiting to be transmitted 0: FIFOTX_TRIGGER interrupt unmasked 1: FIFOTX_TRIGGER interrupt masked	RW	1
2	MASK_USIM_OV	Interrupt mask for FIFO_RX overflow 0: FIFORX overflow interrupt unmasked 1: FIFORX overflow interrupt masked	RW	1
1	MASK_USIM_WT	Interrupt mask for character wait-time underflow 0: Character wait-time overflow interrupt unmasked 1: Character wait-time overflow interrupt masked	RW	1
0	MASK_USIM_NATR	Interrupt mask for no ATR 0: No ATR interrupt unmasked	RW	1

Bits	Field Name	Description	Type	Reset
		1: No ATR interrupt masked		

Table 24-33. USIM RX/TX Management Register (USIM_FIFOS)

Address Offset	0x12		
Physical Address	MPU: 0xFFFF A812	**Instance**	USIM
Description	RX/TX FIFO management register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFORX_FULL	FIFORX_EMPTY	FIFORX_RESET	FIFORX_ TRIGGER				FIFOTX_FULL	FIFOTX_EMPTY	FIFOTX_RESET	FIFOTX_TRIGGER				FIFO_ENABLE	DMA_MODE

Bits	Field Name	Description	Type	Reset
15	FIFORX_FULL	0: Reception FIFO is not full.	R	0
		1: Reception FIFO is full.		
14	FIFORX_EMPTY	0: Reception FIFO is not empty.	R	1
		1: Reception FIFO is empty.		
13	FIFORX_RESET	0: Remove the reset of all FIFORX pointers.	RW	0
		1: Reset of all FIFORX pointers.		
12:9	FIFORX_ TRIGGER	From 0x0 to 0xF: Value of FIFORX trigger	RW	0x0
8	FIFOTX_FULL	0: Transmission FIFO is not full.	R	0
		1: Transmission FIFO is full.		
7	FIFOTX_EMPTY	0: Transmission FIFO is not empty.	R	1
		1: Transmission FIFO is empty.		
6	FIFOTX_RESET	0: Remove the reset of all FIFOTX pointers.	RW	0
		1: Reset all FIFOTX pointers.		
5:2	FIFOTX_TRIGGER	From 0x0 to 0xF: Value of FIFOTX trigger	RW	0xF
1	FIFO_ENABLE	0: Read and write access to FIFOs is forbidden.	RW	0
		1: Read and write access to FIFOs is allowed.		
0	DMA_MODE	0: DMA mode disabled	RW	0
		1: DMA mode enabled for FIFO_TX and FIFO_RX		

Table 24-34. USIM Character Guard Time Register (USIM_CGT)

Address Offset	0x14		
Physical Address	MPU: 0xFFFF A814	**Instance**	USIM
Description	Character guard time register (used only for T = 0 protocol)		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CGT							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7:0	CGT	Character guard time value (in ETU)	RW	0xD
		Default value is 13ETU.		

Table 24-35. USIM Character Waiting Time Register (USIM_CWT)

Address Offset	0x16		
Physical Address	MPU: 0xFFFF A816	**Instance**	USIM
Description	Character waiting time register (used only for T = 1 protocol)		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CWT															

Bits	Field Name	Description	Type	Reset
15:0	CWT	Character Waiting Time value (in ETU) Number of ETU to be added to the minimum value of CWT (11 ETUs) Default value is 8203 (0x200B) ETU (ISO 7816-3 norm).	RW	0x200B

Note: The value written in this register must be the 2^{CWI} of the following formula:

CWT = 11 + 2CWI

USIM automatically adds 11 to the value written in this register.

Table 24-36. USIM Block Waiting Time LSB Register (USIM_BWT_LSB)

Address Offset	0x18		
Physical Address	MPU: 0xFFFF A818	**Instance**	USIM
Description	Block waiting time LSB register (used only for T = 1 protocol)		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWT_LSB															

Bits	Field Name	Description	Type	Reset
15:0	BWT_LSB	Least significant byte of block waiting time value (in ETU) Default value is 15371 (0x3C0B) ETU (ISO 7816-3 norm)	RW	0x3C0B

Note: Card clock frequency (F_{sclk}) is equal to 13/4 MHz. ETU clock is 372/F_{sclk}.

Table 24-37. USIM Block Waiting Time MSB Register (USIM_BWT_MSB)

Address Offset	0x1A		
Physical Address	MPU: 0xFFFF A81A	**Instance**	USIM
Description	Block waiting time MSB register (used only for T = 1 protocol)		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BWT_MSB							

Bits	Field Name	Description	Type	Reset
15:7	Reserved	Read returns 0.	R	0x000
6:0	BWT_MSB	Most significant byte of block waiting time value (in ETU)	RW	0x00

Table 24-38. USIM Debug Register (DEBUG_REG)

Address Offset	0x1C		
Physical Address	MPU: 0xFFFF A81C	**Instance**	USIM
Description	Debug register for reading states of different state-machines		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RX_STATE_ MACHINE		TX_STATE_MACHINE		MAIN_STATE_ DEBUG			

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0.	R	0x00
7:6	RX_STATE_ MACHINE	Current state of the reception state-machine: 00: IDLE state 01: RX_CHARACTER state 10: BLOCK_MGMT state	R	0
5:4	TX_STATE_MACHINE	Current state of the transmission state-machine: 00: IDLE state 01: TX_CHARACTER state 10: FIFO_EMPTY state	R	0
3:0	MAIN_STATE_ DEBUG	Current state of the main state-machine: 0000: NC_PWR_OFF state 0001: PWR_ON state 0010: SI/O_RX_HIGH state 0011: SCLK_ON state 0100: WAIT_SYNC_ATR state 0101: SRST_HIGH state 0110: WAIT_ASYNC_ATR state 0111: CLOCK_STOP state 1000: CLOCK_RESTART state 1001: WARM_RESET state 1010: DECODE_TS state 1011: DATA_RX_TX state 1100: SRST_LOW state 1101: SCLK_OFF state 1110: SI/O_TX_LOW state 1111: tdsim timer is counting	R	0

Note: This register is not resynchronized.

Table 24-39. SAM Clock Ratio Configuration Register (CONF_SAM1_DIV)

Address Offset	0x1E		
Physical Address	MPU: 0xFFFF A81E	**Instance**	USIM
Description	SAM_clock ratio configuration register (in software mode only)		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SAM1_DIV							

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Read returns 0.	R	0x0
11:0	SAM1_DIV	Ratio F/D of the SAM_CLOCK Unit is in 13 MHz period – 1. Used only in software mode. See CONF5_REG register field for hardware/software mode selection. Default value is 186 (0x0B9) at reset.	RW	0x0B9

Table 24-40. USIM Configuration 4 Register (CONF4_REG)

Address Offset	0x20		
Physical Address	MPU: 0xFFFF A820	**Instance**	USIM
Description	Configuration 4 register (used only for T = 0 protocol)		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CONFWAITI							

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Read returns 0.	R	0x0
12:0	CONFWAITI	Gives the Wi x Di x 1 Coefficient value for character underflow wait time. Unit is RX_time unit = 960 _UnicodeEncodeError_ D _UnicodeEncodeError_ T_{etu} with D = 1	RW	0x00A

Table 24-41. Clock Period Before ATR Receipt Register (ATR_CLK_PRD_NBS)

Address Offset	0x22		
Physical Address	MPU: 0xFFFF A822	**Instance**	USIM
Description	Gives the number of clock period before the receipt of ATR		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLOCK_ NUMBER_ BEFORE_ATR															

Bits	Field Name	Description	Type	Reset
15:0	CLOCK_ NUMBER_ BEFORE_ATR	Number of SIM clock periods before the receipt of the start-bit of the ATR Value is updated for each ATR.	R	0x0000

Table 24-42. ETU Clock Period Configuration Register (CONF_ETU_DIV)

Address Offset	0x24		
Physical Address	MPU: 0xFFFF A824	**Instance**	USIM
Description	Configuration register for the ETU clock period based on the 13MHz clock (used only in software mode)		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONF_ETU_DIV															

Bits	Field Name	Description	Type	Reset
15:0	CONF_ETU_DIV	Number of periods to calculate ETU_clock period Unit is in 13MHz period – 1. Used only in software mode.	R	0x05CF

Note: This register must be set to the correct value (FI = 372 and DI = 1) when a warm reset is done in software mode (see the CONF5_REG description). In hardware mode, setting is automatically performed.

Table 24-43. USIM Configuration 5 Register (CONF5_REG)

Address Offset	0x26		
Physical Address	MPU: 0xFFFF A826	**Instance**	USIM
Description	Configuration 5 register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							SOFT_NHARD_ FIDI_PROG	FI				DI			

Bits	Field Name	Description	Type	Reset
15:9	Reserved	Read returns 0.	R	0x00
8	SOFT_NHARD_ FIDI_PROG	Software/hardware FIDI program: 0: Software configuration enabled CONF_SAM1_DIV is used for sampling clock period. CONF_ETU_DIV is used for the ETU period. 1: Hardware configuration enabled CONFFI and DI factors are used for the F/D ratio. CONFSCLKDIV is used for the sim_clock.	RW	1
7:4	FI	Decoding of the FI value according to the TA1 character of the Smart Card ATR: 0000: 372 0001: 372 0010: 558 0011: 744 0100: 1116	RW	0x0

Bits	Field Name	Description	Type	Reset
		0101:1488		
		0110: 1860		
		1001:512		
		1010: 768		
		1011:1024		
		1100: 1536		
		1101: 2048		
		Others: 372		
3:0	DI	Decoding of the DI value according to the TA1 character of the Smart Card ATR:	RW	0x1
		0001: 1		
		0010: 2		
		0011: 4		
		0100: 8		
		0101:16		
		0110: 32		
		1000:12		
		1001: 20		
		Others: 1		

Table 24-44. USIM Guard Time Register (TC_GUARD_TIME_ADD)

Address Offset 0x28

Physical Address MPU: 0xFFFF 8828 **Instance** USIM

Description Clock stop procedure guard time register

Type RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		SOFT_TC_GUARD_TIME_ADD_EN	SOFT_TC_GUARD_TIME_ADD												

Field Name	Description	Type	Reset
Reserved	Read returns 0.	R	00
SOFT_TC_ GUARD_TIME_ ADD_EN	0: Guard time is hardware-programmed.	RW	0
	1: Guard time is software-programmed with TC_GUARD_TIME_ADD field.		
SOFT_TC_ GUARD_TIME_ ADD	Additional guard time value for clock-stop mode	RW	0x2E8

Note: The tg timer will be equal at:

- 1860 + SOFT_TC_GUARD_TIME_ADD when SOFT_TC_GUARD_TIME_ADD_EN = 1 (software program)
- 1860 + 2x(CONF_ETU_DIV/N) when SOFT_TC_GUARD_TIME_ADD_EN = 0 (hardware program)

Multichannel Serial Interface

This chapter introduces the multichannel serial interface (MCSI) of the LOCOSTO device.

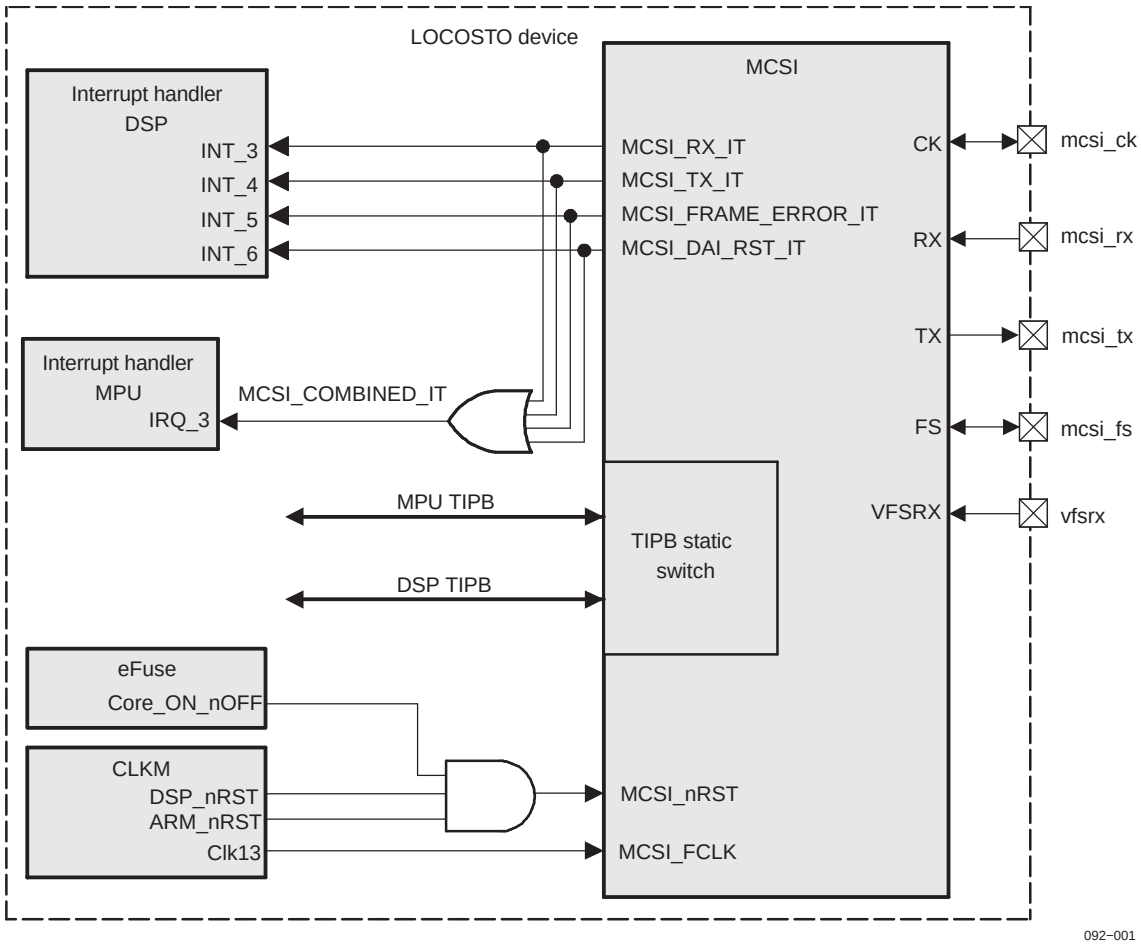
Topic **Page**

25.1 MCSI Overview

The multichannel serial interface (MCSI) has multichannel transmission and reception capability. The MCSI expands the parallel interface of an MPU or DSP to connect to external devices such as audio codecs and GSM system simulators.

Figure 25-1 shows an overview of the MCSI.

Figure 25-1. MCSI Overview



The MCSI on the LOCOSTO device provides full-duplex communication with master or slave clock control. All transmission parameters are configurable to cover the maximum number of operating conditions:

- Master or slave clock control (transmission clock and frame synchronization pulse)
- Programmable transmission clock frequency
- Single-channel or multichannel (x16) frame structure
- Programmable word length: 3 to 16 bits
- Full-duplex transmission
- Programmable frame configuration
- Continuous or burst transmission
- Normal or alternate framing
- Normal or inverted frame polarity
- Short or long frame pulse
- Programmable oversize frame length

- Programmable frame length
- Programmable interrupt occurrence time (TX and RX)
- Error detection with interrupt generation on wrong frame length
- GSM digital audio interface (DAI) operating modes (radio uplink, radio downlink, and acoustics). The DAI mode is a GSM test interface that is used to determine the routing of speech data for the devices being tested. In DAI mode, the MCSI is configured for direct connection to the GSM system simulator interface, including the reset system simulator (RSS) signal.

25.2 MCSI Environment

This section describes the MCSI environment in normal mode.

In DAI mode, the MCSI is directly connected to the GSM system simulator. The MCSI external connections in DAI mode with a GSM system simulator are not described in this section.

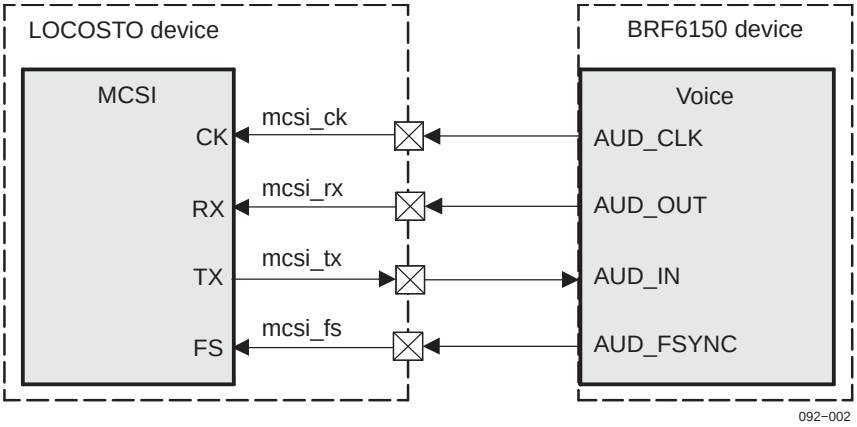
In a typical application, the LOCOSTO device is connected to a Bluetooth baseband integrated circuit (IC). In master mode, the Bluetooth device provides the clock and frame synchronization to the MCSI module.

The BRF6150 device implements an advanced solution for the Bluetooth protocol that interfaces easily to the LOCOSTO device. The BRF6300 device is pin-to-pin compatible and can be used instead of the BRF6150 device.

For more details on the BRF6150 and BRF6300 devices, contact your TI representative to obtain the device documentation.

Figure 25-2 shows the external connections between the LOCOSTO device and the BRF6150 device.

Figure 25-2. MCSI External Connections in Normal Mode

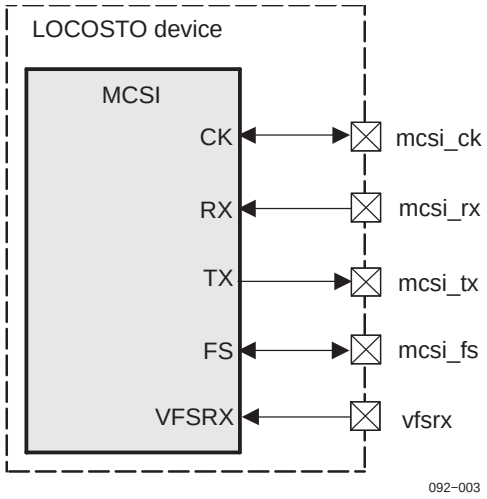


25.2.1 MCSI Functional Interface

25.2.1.1 MCSI Pins for Serial Interface

Figure 25-3 shows the signals of the MCSI module.

Figure 25-3. MCSI Interface Signals



25.2.2 MCSI Interface Description

Table 25-1. MCSI I/O Description

Signal Name	I/O	Description	Value at Reset
mcsi_ck	IO	Serial clock I/O	N/A(1)
mcsi_rx	I	Serial data input	N/A
mcsi_tx	O	Serial data output	0
mcsi_fs	IO	Frame synchronization I/O	N/A
vfsrx	I	Voice frame synchronization	N/A

(1) N/A: Not applicable

25.2.3 MCSI Serial Interface Protocol and Data Format

The MCSI protocol is a simple 4-wire serial communications interface.

The timing diagrams in Figure 25-4 through Figure 25-22 show the MCSI interface protocol with several functional modes. These timing diagrams are based on a positive clock polarity (MCSI.MCSI_MAIN_PARAM_REG[4] CLK_POL bit = 0) transmit on rising edge/receive on falling edge.

Figure 25-4. Single-Channel/Alternate Long Framing

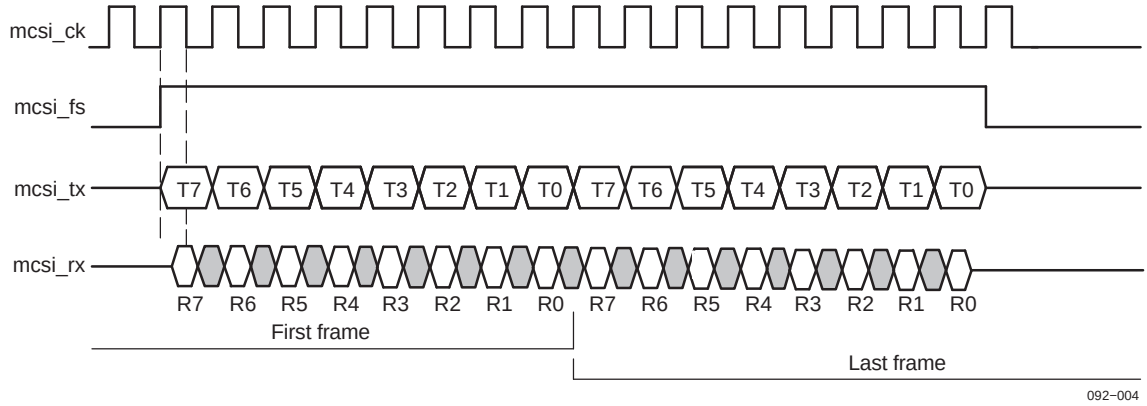
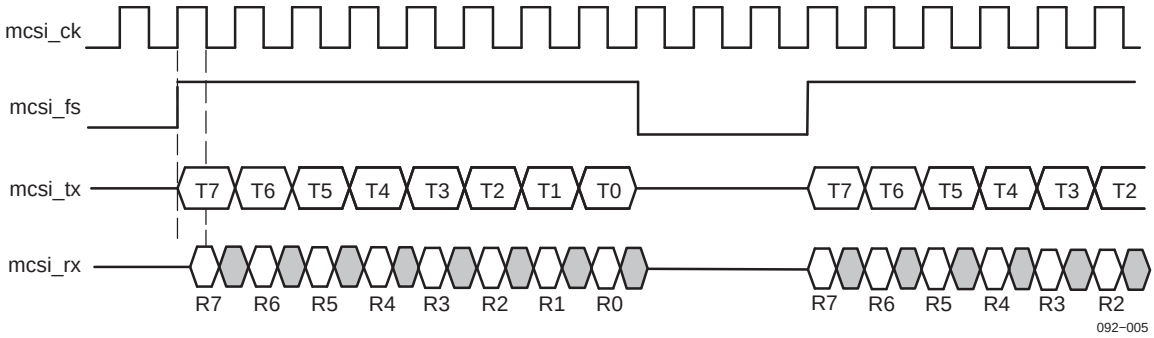
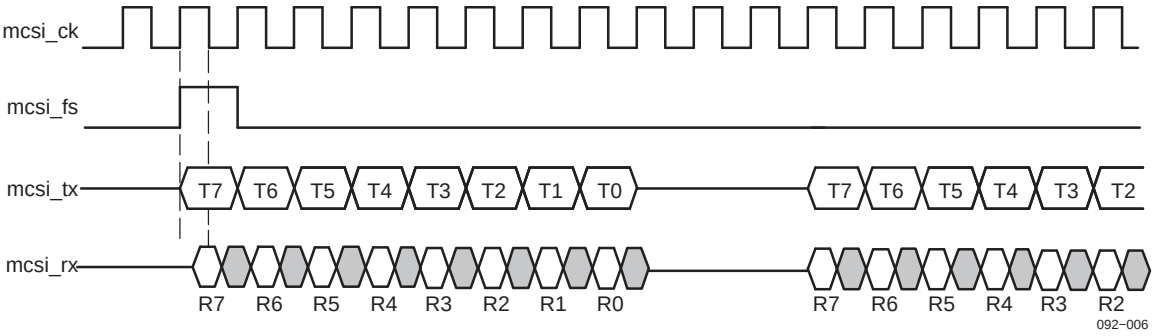


Figure 25-5. Single-Channel/Alternate Long Framing/Burst



NOTE: With MCSI.MCSI_OVER_CLOCK_REG = 0x0003

Figure 25-6. Single-Channel/Alternate Short Framing/Continuous/Burst



NOTE: With MCSI.MCSI_OVER_CLOCK_REG = 0x0003

Figure 25-7. Multichannel/Normal Short Framing/TX Channel4 Disable

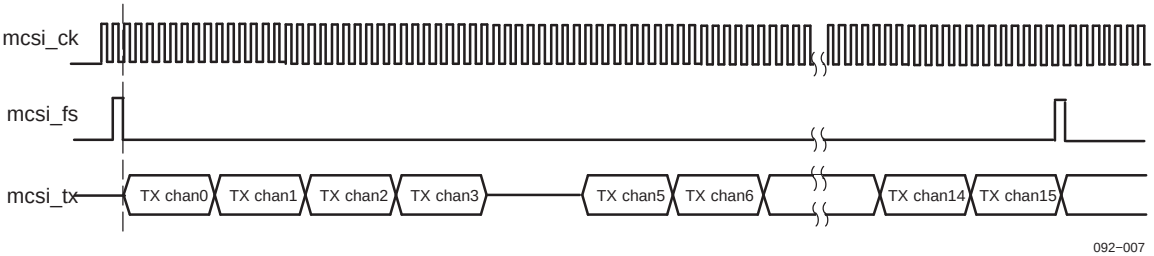
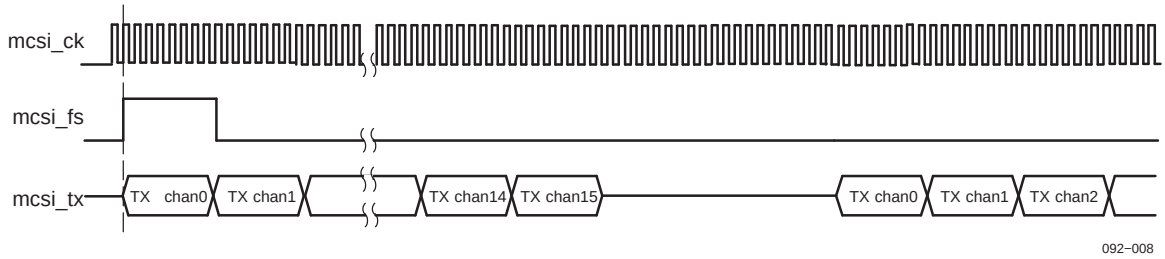
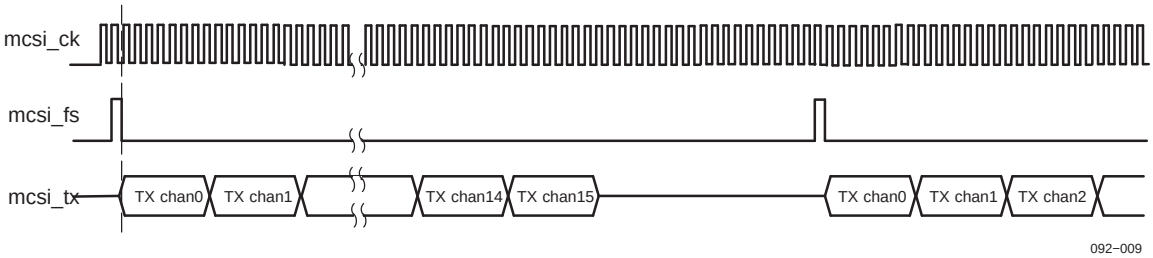


Figure 25-8. Multichannel/Alternate Long Framing/Continuous/Burst



NOTE: With MCSI.MCSI_OVER_CLOCK_REG = 0x0013

Figure 25-9. Multichannel/Normal Short Framing/Burst



NOTE: With MCSI.MCSI_OVER_CLOCK_REG = 0x0013

Figure 25-10. Single-Channel/Normal Short Framing

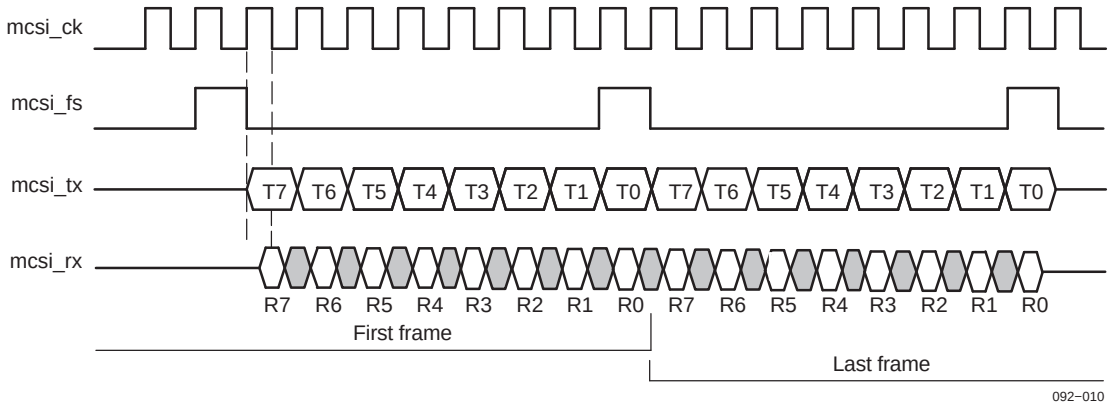
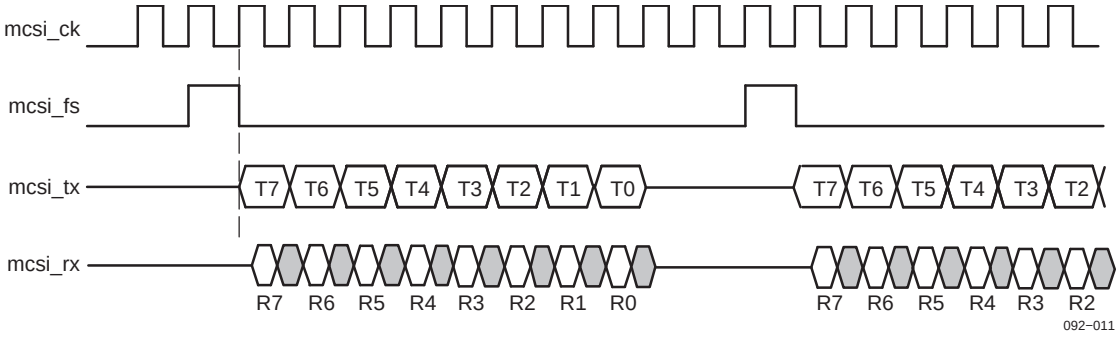


Figure 25-11. Single-Channel/Normal Short Framing/Burst



NOTE: With MCSI.MCSI_OVER_CLOCK_REG = 0x0003

Figure 25-12. Single-Channel/Normal Long Framing

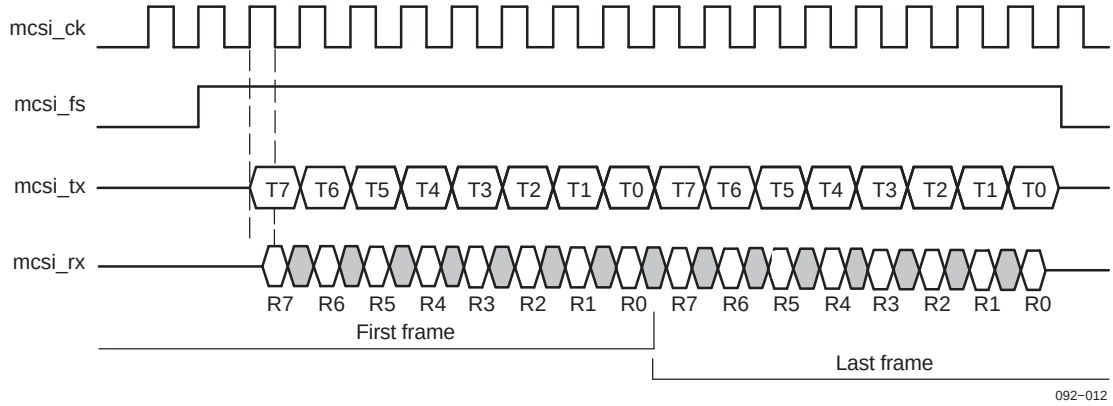
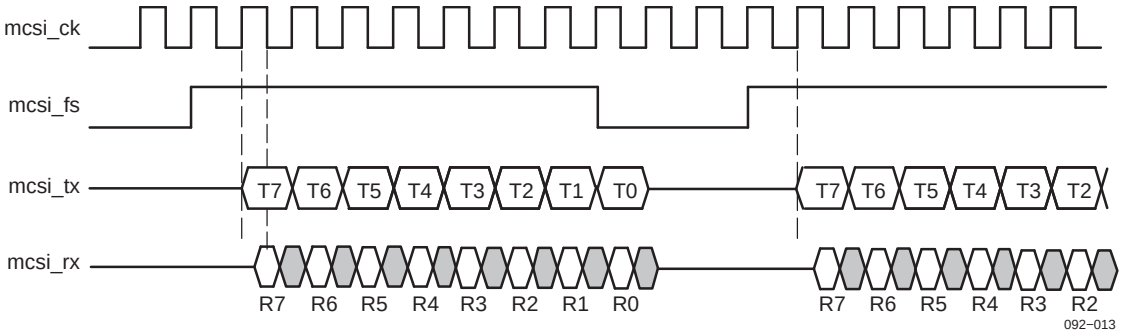


Figure 25-13. Single-Channel/Normal Long Framing/Burst



NOTE: With MCSI.MCSI_OVER_CLOCK_REG = 0x0003

Figure 25-14. Single-Channel/Normal Long Framing/Continuous

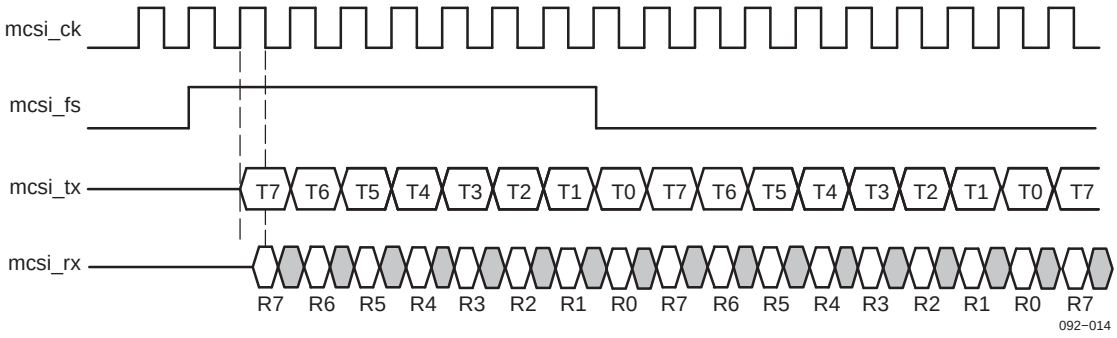


Figure 25-15. Single-Channel/Alternate Short Framing

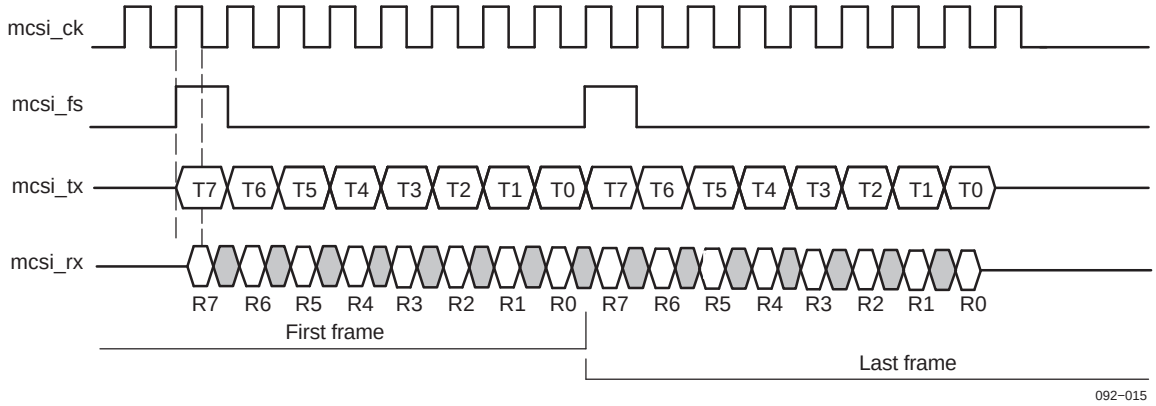
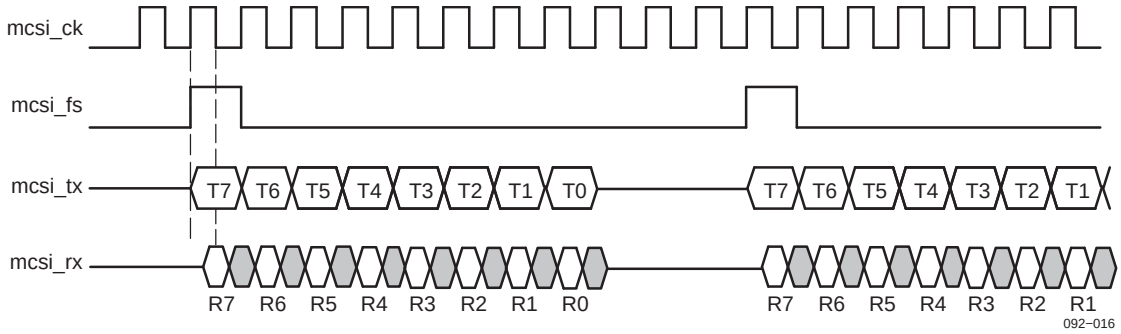
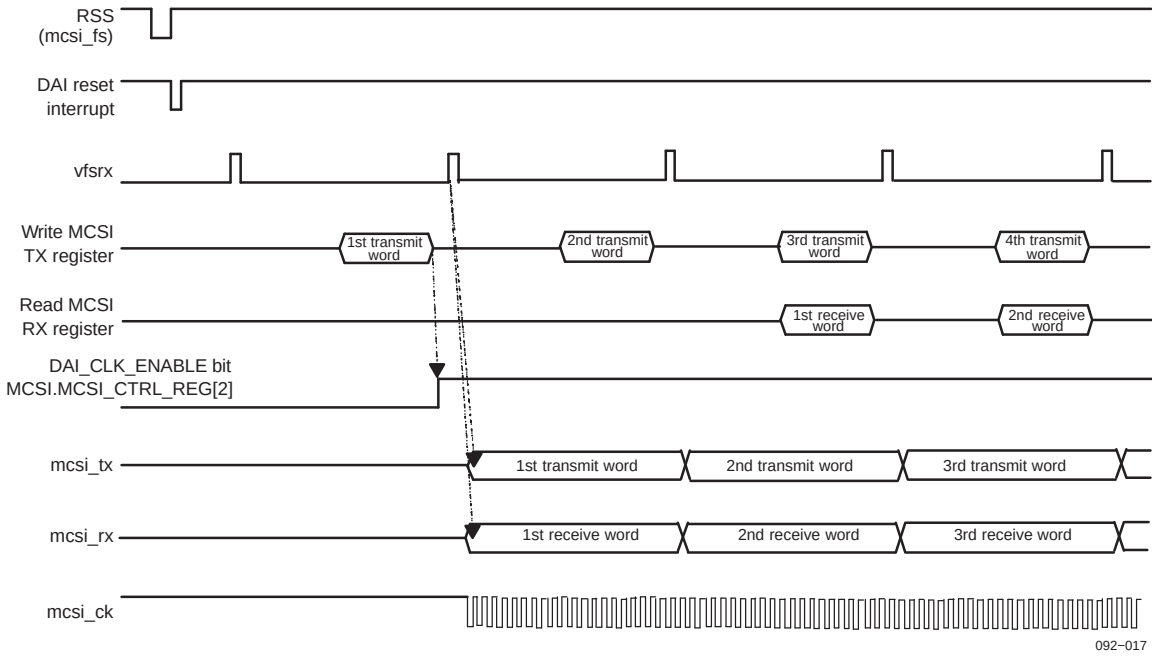


Figure 25-16. Single-Channel/Alternate Short Framing/Burst



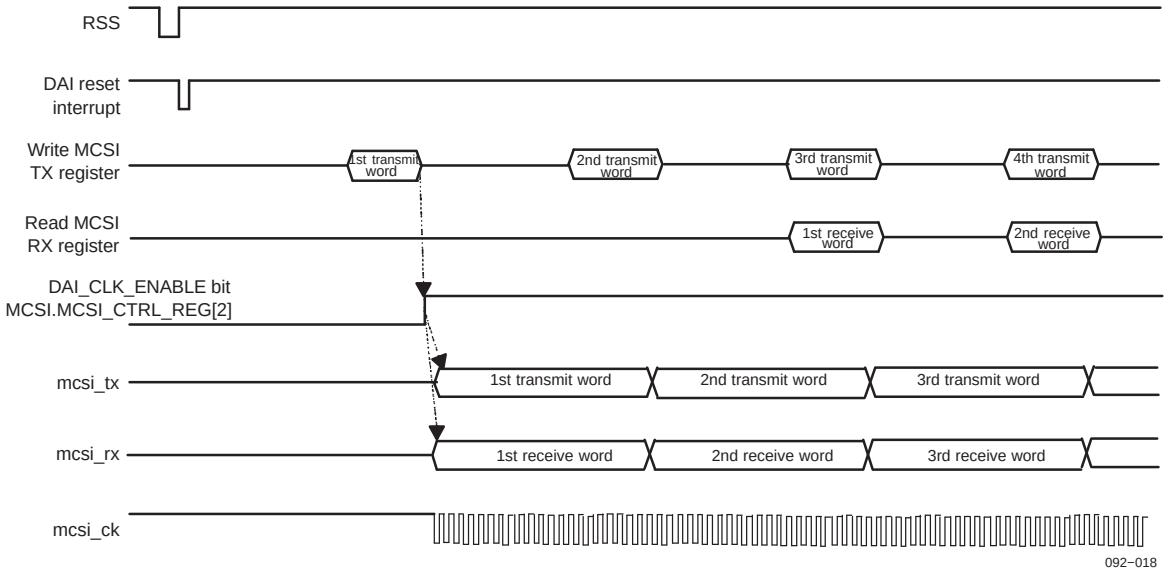
NOTE: With MCSI.MCSI_OVER_CLOCK_REG = 0x0003

Figure 25-17. DAI Acoustic Mode (With Voice Interface Synchro Frame)



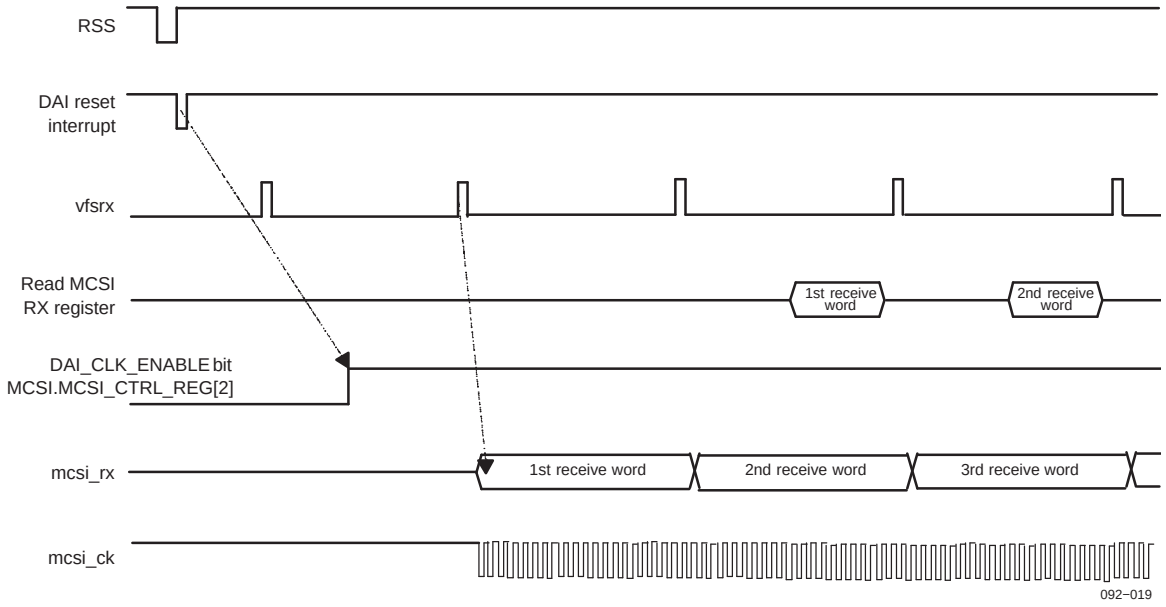
NOTE: To launch the transmission, the DAI_CLK_ENABLE bit must be set to 1 by the software after the first write in the TX register.

Figure 25-18. DAI Acoustic Mode (Without Voice Interface Synchro Frame)



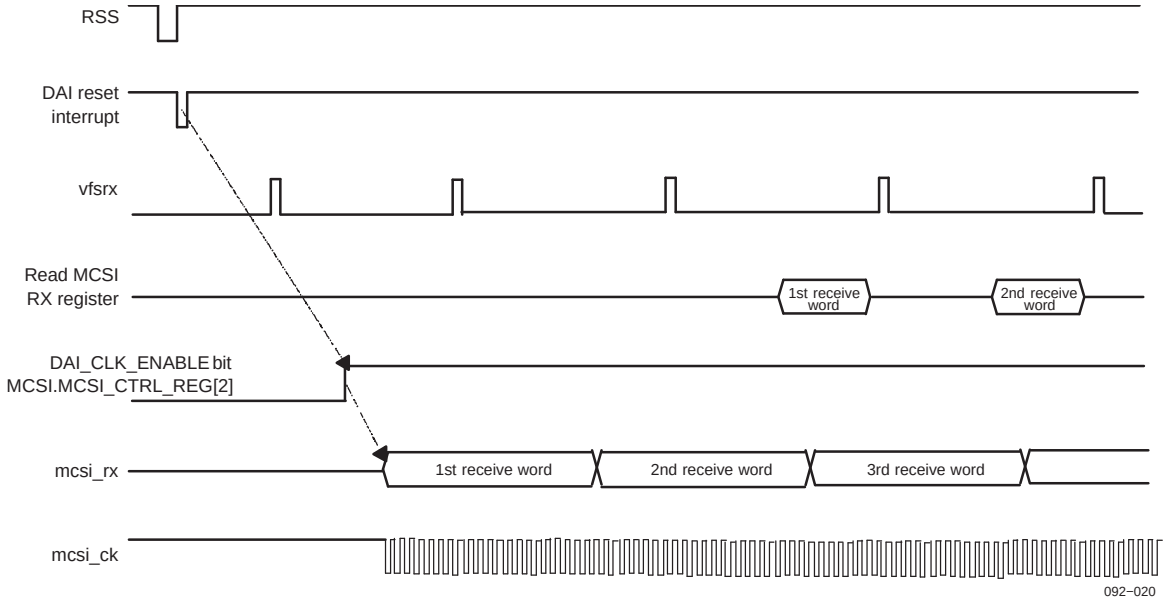
NOTE: To launch the transmission, the DAI_CLK_ENABLE bit must be set to 1 by the software after the first write in the TX register.

Figure 25-19. DAI Radio Uplink Mode (With Voice Interface Synchro Frame)



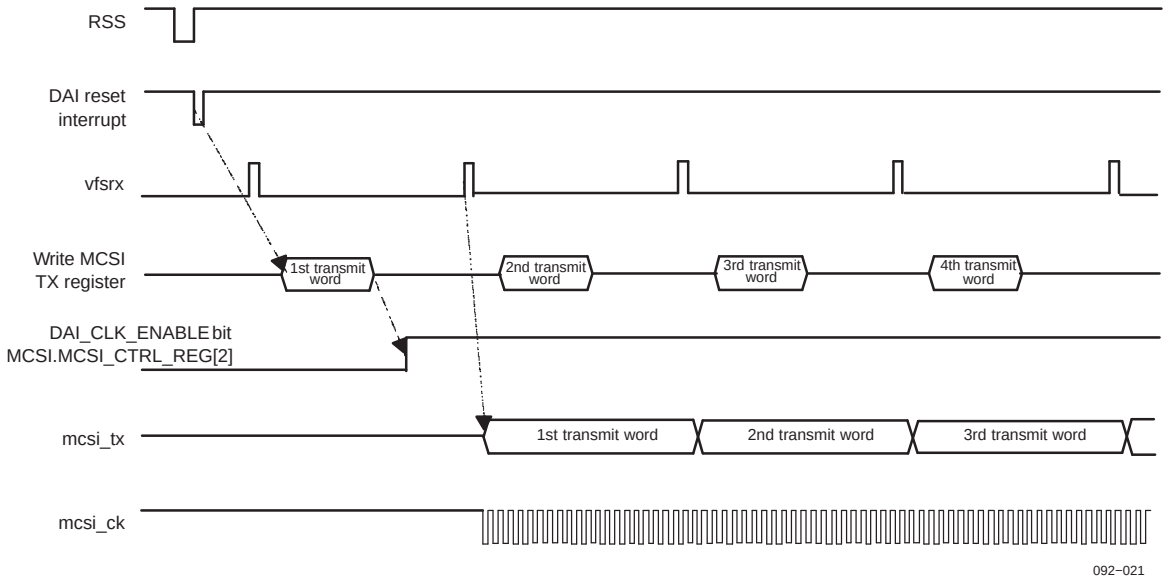
NOTE: To enable DAI activity, the DAI_CLK_ENABLE bit must be set to 1 by the software when the DAI reset interrupt is received.

Figure 25-20. DAI Radio Uplink Mode (Without Voice Interface Synchro Frame)



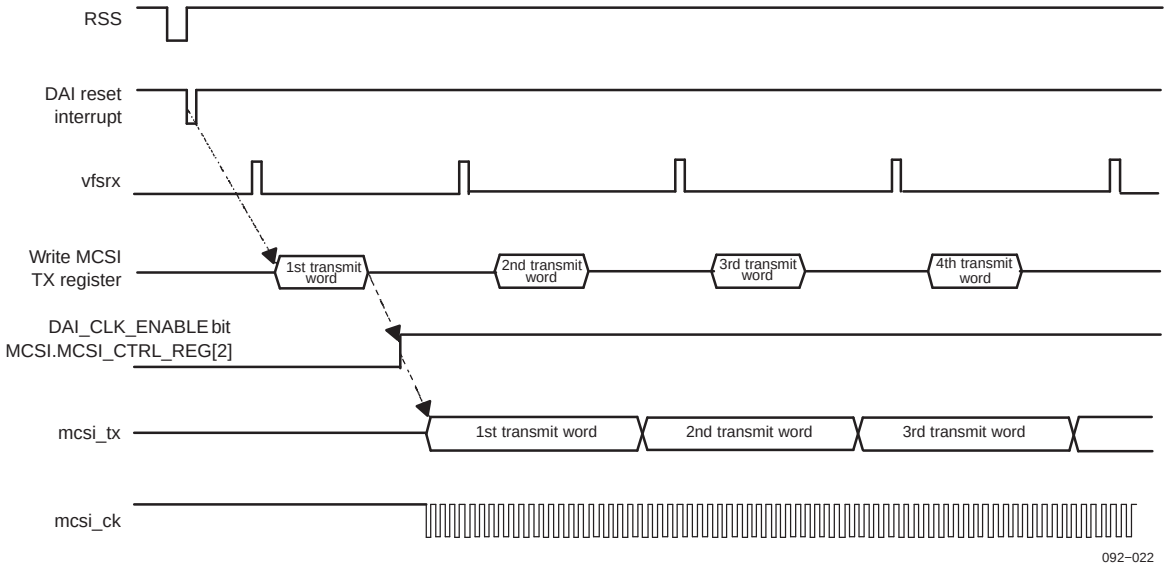
NOTE: To enable DAI activity, the DAI_CLK_ENABLE bit must be set to 1 by the software when the DAI reset interrupt is received.

Figure 25-21. DAI Radio Downlink Mode (With Voice Interface Synchro Frame)



NOTE: To launch the transmission, the DAI_CLK_ENABLE bit must be set to 1 by the software after the first write in the TX register.

Figure 25-22. DAI Radio Downlink Mode (Without Voice Interface Synchro Frame)

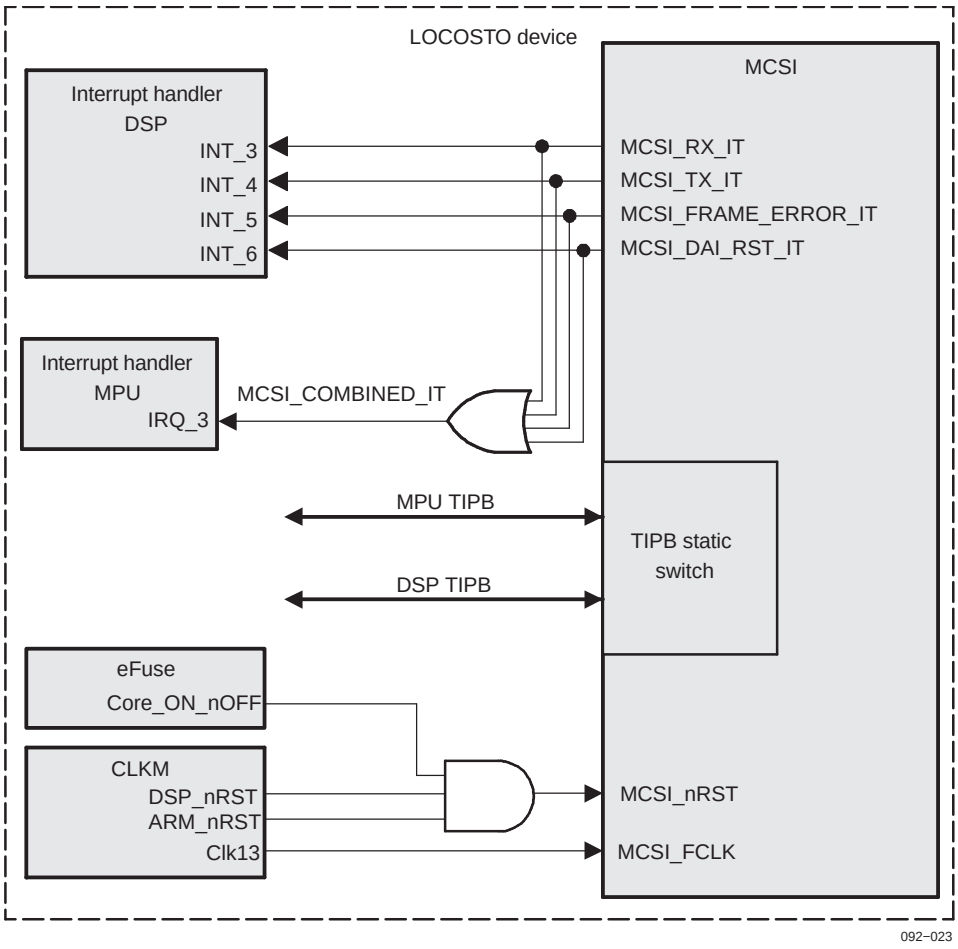


NOTE: To launch the transmission, the DAI_CLK_ENABLE bit must be set to 1 by the software after the first write in the TX register.

25.3 MCSI Integration

Figure 25-23 shows the MCSI module integration in the LOCOSTO device.

Figure 25-23. MCSI Integration



For more information about the TIPB switch, see Chapter 5, *Interconnect*.

25.3.1 Clock and Reset Scheme

The MCSI module receives its reset and clocks from the CLKM module. Table 25-2 lists the MCSI clocks and resets.

Table 25-2. MCSI Clocks and Resets

Module	Functional Clock	Interface Clock	Resets
MCSI	MCSI_FCLK	MCSI_ICLK	Hardware: DSP_nRST ARM_nRST Core_ON_nOFF Software: SOFT_RESET bit MCSI.MCSI_CTRL_REG[1]

25.3.1.1 MCSI Clocks

Table 25-3 lists the clock domains of the timer and watchdog modules.

Table 25-3. Clock Description

Type	Name	Source	Frequency	Description
Functional	MCSI_FCLK	Clk13	13 MHz	Clock provided by the CLKM; used inside the MCSI module to generate the serial clock output (mcsi_ck)
Interface	MCSI_ICLK	TIPB clock (strobe signal)	52 MHz	Clock fed by the TIPB; used to trigger access to the MCSI TIPB static switch

25.3.1.2 MCSI Reset Scheme

25.3.1.2.1 Hardware Reset

Global reset of the module is performed by the DSP_nRST or the ARM_nRST hardware reset signals depending on whether the MCSI is controlled by the DSP or the MPU, respectively. The Core_ON_nOFF hardware reset signal resets the entire chip, including the MCSI module. For more details on the reset signals, see Chapter 6, *Power, Reset, and Clock Management*.

25.3.1.2.2 Software Reset

The MCSI module can also be software reset by using the SOFT_RESET bit MCSI.MCSI_CTRL_REG[1]. For a complete description of the MCSI.MCSI_ CTRL_REG register, see Section 25.6, *Register Manual*.

To reset the MCSI module, set the SOFT_RESET bit to 1.

Note: This software reset is limited to the control and status registers (MCSI.MCSI_CTRL_REG and MCSI.MCSI_STATUS_REG), the internal state-machine, and the PISO and SIPO logic. The parameters registers are not affected by this software reset.

On the software reset, the MCSI clock is disabled (CLK_ENABLE bit MCSI.MCSI_CTRL_REG[0]), thus halting the execution of any current operating mode.

25.3.2 Hardware Requests

25.3.2.1 Static Switch Interface

The MCSI module is accessible via the MPU or the DSP shared TIPB. The selection between the DSP and the MPU shared TIPB is provided by a TIPB static switch with status and configuration registers listed in Table 25-4. For a complete description of these registers, see Chapter 5, *Interconnect*. By default, the MCSI module is controlled by the DSP-shared TIPB.

Table 25-4. MCSI TIPB Static Switch Physical Addresses

Register Name	Type	Register Width (Bits)	MPU or DSP Physical Address
MCSI_TSW_DSP_CONF	RW	16	0x7CA0
MCSI_TSW_DSP_STA	RW	16	0x7CA1
MCSI_TSW_MPU_CONF	RW	16	0xFFFF 8820
MCSI_TSW_MPU_STA	RW	16	0xFFFF 8824

25.3.2.2 Interrupt Requests

Table 25-5 shows interrupt mapping from the MCSI_COMBINED_IT is an interrupt to the MPU subsystem interrupt handler. MCSI_TX_IT, MCSI_RX_IT, and MCSI_FRAME_ERROR_IT are interrupts to the DSP subsystem interrupt handler. MCSI_DAI_RST_IT is an interrupt to the DSP subsystem interrupt handler.

For more details on the MPU and DSP interrupt handlers, see Chapter 12, *Interrupt Handlers*.

Table 25-5. MCSI Interrupt Names and MPU/DSP IRQ Mapping

Interrupt Name	Mapping	Comments	Domain
MCSI_COMBINED_IT	IRQ_3	MCSI combined interrupt corresponding to a logic OR between the four MCSI interrupts	MPU
MCSI_TX_IT	INT_3	MCSI receive interrupt	DSP
MCSI_RX_IT	INT_4	MCSI transmit interrupt	DSP
MCSI_FRAME_ERROR_IT	INT_5	MCSI frame error interrupt	DSP
MCSI_DAI_RST_IT	INT_6	MCSI DAI reset interrupt	DSP

CAUTION

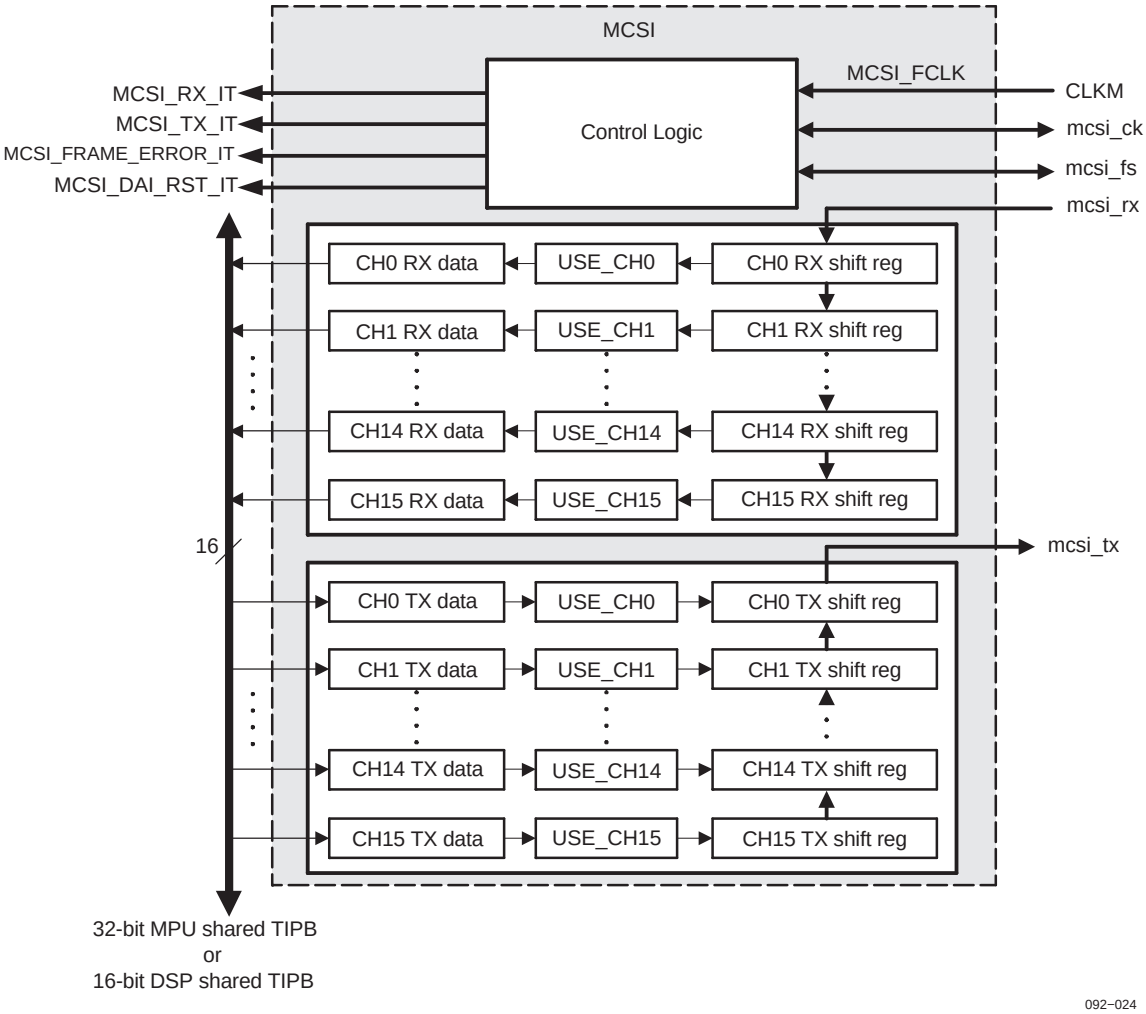
- It is the responsibility of the user to select the DSP or the MPU to handle these interrupts.
- INT_6 is shared between the MCSI and the C-port. For more details, see Section 12.2.2.1, *Shared Level-Sensitive Interrupts*, in Chapter 12, *Interrupt Handlers*.

25.4 MCSI Functional Description

25.4.1 Block Diagram

Figure 25-24 shows the main blocks of the MCSI interface module.

Figure 25-24. MCSI Block Diagram



25.4.2 Configuration Parameters

The configuration parameters can be modified only if the MCSI is disabled (CLK_ENABLE bit MCSI.MCSI_CTRL_REG[0] = 0).

25.4.2.1 Slave/Master Control

Using the control bit, the interface can be configured in one of two ways:

- In master mode with the transmission clock and the frame synchronization pulse generated by the interface
- In slave mode with the transmission clock and the frame-synchronization pulse generated from an external device

Control bit: MCSI_MODE bit MCSI.MCSI_MAIN_PARAM_REG[6]

- 1: Master
- 0: Slave

25.4.2.2 Single-Channel/Multichannel

The frame structure can be either single-channel-based (one channel per frame) or multichannel-based (with the number of channels fixed at 16).

Control bit: MULTI_SINGLE bit MCSI.MCSI_MAIN_PARAM_REG[7]

- 1: Multichannel
- 0: Single-channel

25.4.2.3 Short/Long Framing

The frame-synchronization pulse duration can be either short (with a pulse duration equal to the bit duration) or long (with a pulse duration equal to the channel duration).

The long frame is active only during transmission on channel 0.

Control bit: FSYNCH_SIZE bit MCSI.MCSI_MAIN_PARAM_REG[8]

- 1: Long
- 0: Short

25.4.2.4 Normal/Alternate Frame Synchronization

The frame-synchronization pulse mode is either normal (with the frame synchronization pulse starting 1 bit before channel 0) or alternate (with the frame-synchronization pulse starting with the first bit of channel 0).

Control bit: FSYNCH_MODE bit MCSI.MCSI_MAIN_PARAM_REG[9]

- 1: Alternate
- 0: Normal

25.4.2.5 Continuous/Burst Mode

The frame mode is either continuous (with one frame-synchronization pulse at the first frame) or burst (with one frame-synchronization pulse at each frame).

Control bit: CONT_BURST bit MCSI.MCSI_MAIN_PARAM_REG[5]

- 1: Continuous
- 0: Burst

25.4.2.6 Normal/Inverted Clock

The polarity of the clock can be either normal (with writing on the positive edge clock and reading on the negative edge clock) or inverted (with writing on the negative edge clock and reading on the positive edge clock).

Control bit: CLK_POL bit MCSI.MCSI_MAIN_PARAM_REG[4]

- 1: Inverted
- 0: Normal

25.4.2.7 Normal/Inverted Frame Synchronization

The polarity of the frame-synchronization pulse can be either normal (with a positive pulse) or inverted (with a negative pulse).

Control bit: FSYNCH_POL bit MCSI.MCSI_MAIN_PARAM_REG[10]

- 1: Inverted
- 0: Normal

25.4.2.8 Channel Used

To enable a channel in multimode, set the USE_CHn bit for the desired channel n in MCSI.MCSI_CHAN_USED_REG register.

25.4.2.9 Word Size

To choose the size of the word, set its size minus one into the main parameters registers.

Control field: WORD_ SIZE[3:0] field MCSI.MCSI_MAIN_PARAM_REG[3:0]

With 0x2 ≤ WORD_SIZE field ≤ 0xF

The MCSI transmits and receives the most significant bit (MSB) first. For example, if WORD_SIZE field equals 0xB, the upper 12 bits of the TX registers are transmitted, the upper 12 bits of the RX registers contain the received data, and the lower 4 bits are zeros.

25.4.2.10 Frame Size

To add any overhead bits at the end of each frame, set the number of desired overhead bits in the MCSI.MCSI_OVER_CLOCK_REG register.

Control field: OVER_CLOCK[9:0] field MCSI.MCSI_OVER_CLOCK_ REG[9:0]

With 0x000 ≤ OVER_CLOCK field ≤ 0x3FF

25.4.2.11 Transmission Clock Frequency

In master mode, the clock frequency is derived from the 13-MHz system clock and can be programmed from 6.35 kHz to 6.5 MHz.

The clock frequency of the mcsi_ck signal is calculated with the following formula:

Freq (mcsi_ck) = 13MHz / CLK_FREQ field

Control field: CLK_FREQ field MCSI.MCSI_CLOCK_FREQ_REG[10:0]

With 0x002 ≤ CLK_FREQ field ≤ 0x7FF

Note: If CLK_FREQ = 0x000 or 0x001, the 13-MHz functional clock is divided by 2 as if CLK_FREQ = 0x002.

25.4.2.12 DAI Mode

The interface can be configured in one of the following modes:

- Normal mode with MCSI full-duplex transmission
- Radio uplink mode with DAI uplink transmission
- Radio downlink mode with DAI downlink transmission
- Acoustic mode with DAI full-duplex transmission

Control field: DAI_CONFIG field MCSI.MCSI_MAIN_PARAM_REG[13:12]

- 11: Acoustic
- 10: Radio uplink
- 01: Radio downlink
- 00: Normal (no DAI)

25.4.3 Interrupt Generation

Four physical interrupts are available for real-time management of the MCSI by the MPU or DSP:

- MCSI_RX_IT: Data receive interrupt
- MCSI_TX_IT: Data transmit interrupt
- MCSI_FRAME_ERROR_IT: Frame duration error interrupt
- MCSI_DAI_RST_IT: System simulator reset interrupt

Note: These four previous interrupt requests are merged on the same interrupt line, MCSI_COMBINED_IT, connected to the MPU interrupt handler.

MCSI_RX_IT, MCSI_TX_IT, and MCSI_FRAME_ERROR_IT are maskable with dedicated programmable control bits of the interrupt register MCSI.MCSI_INTR_REG.

- MCSI_RX_IT is masked when MCSI.MCSI_INTR_REG[8] MASK_IT_RX bit = 0.
- MCSI_TX_IT is masked when MCSI.MCSI_INTR_REG[9] MASK_IT_TX bit = 0.
- MCSI_FRAME_ERROR_IT is masked when MCSI.MCSI_INTR_ REG[10] MASK_IT_ERR bit = 0.

MCSI_DAI_RST_IT is not maskable. It is activated as soon as a GSM RSS is detected when the MCSI is configured in DAI mode. In all other modes, this interrupt is inactive.

Each interrupt is associated with a flag bit in the MCSI.MCSI_STATUS_REG register that is set to 1 when the interrupt is generated. To acknowledge the interrupt and release the corresponding physical signal, the MPU or DSP must write 1 at the bit location in the status register. The interrupt/flag bit associations are:

- MCSI_RX_IT: RX_READY bit MCSI.MCSI_STATUS_REG[2]
- MCSI_TX_IT: TX_READY bit MCSI.MCSI_STATUS_REG[4]
- MCSI_FRAME_ERROR_IT: FRAME_ERR bit MCSI.MCSI_STATUS_ REG[0]
- MCSI_DAI_RST_IT: DAI_READY bit MCSI.MCSI_STATUS_REG[6]

25.4.3.1 Receive Interrupt

The receive interrupt is generated every frame after the completion of the reception of a data word:

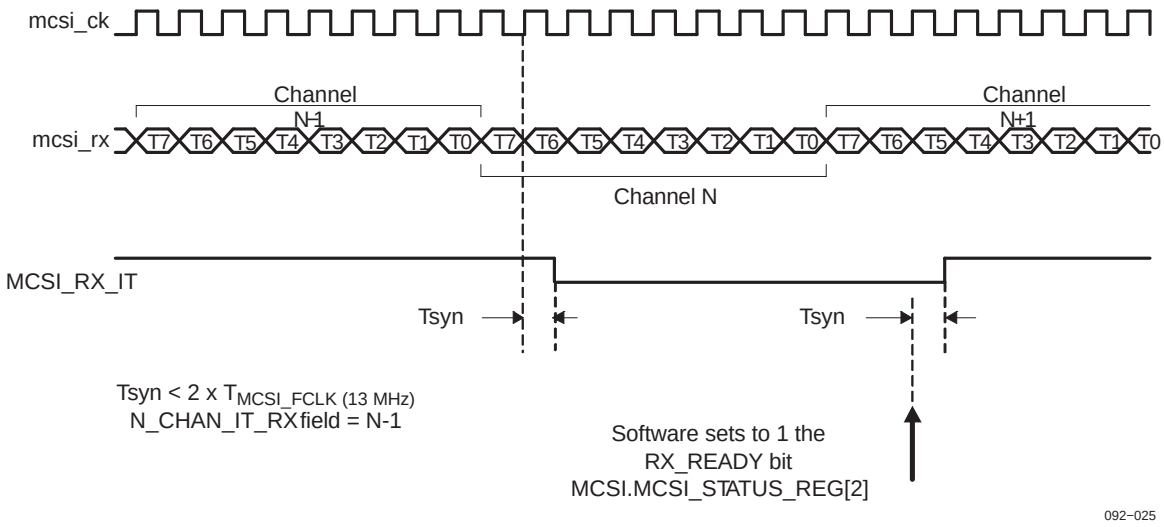
- In single-channel mode, the interrupt is generated one clock period plus a synchronization delay Tsyn after the reception of the word.
- In multichannel mode, the interrupt is generated one clock period plus a synchronization delay Tsyn after the reception of the word of the channel whose number is defined by the MCSI.MCSI_INTR_REG[3:0] N_CHAN_IT_RX field.

Figure 25-25 shows the receive interrupt timing diagram.

Note: If MCSI is in slave mode, the interface clock must be driven after the generation of a valid data receive interrupt. The interface clock must not be gated before this receive interrupt because the interrupt is generated on the MCSI interface clock (MCSI_ICLK).

For more information about the interface clock, see Section 25.3.1.1, *MCSI Clocks*.

Figure 25-25. Receive Interrupt Timing Diagram



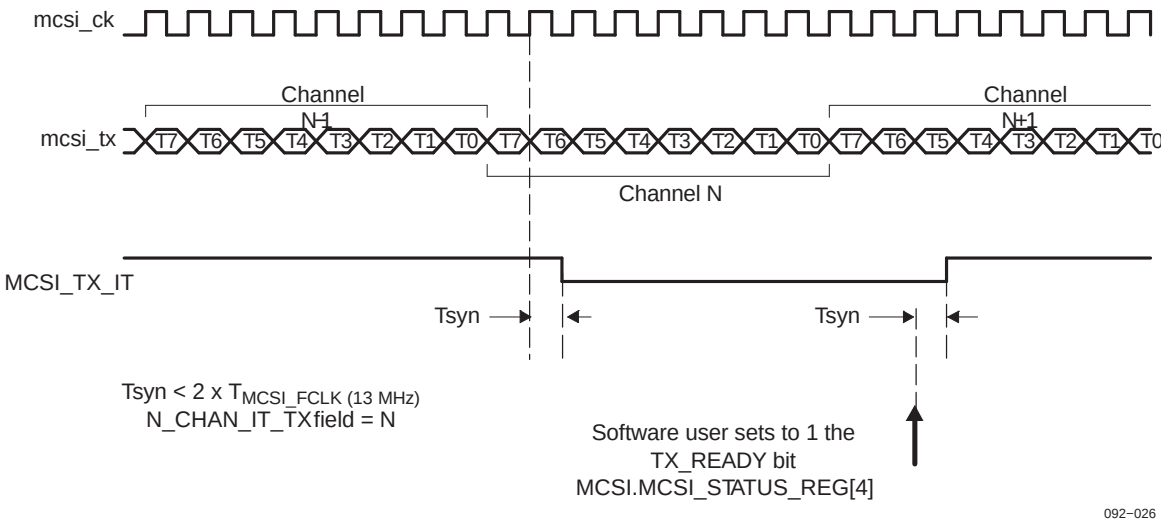
25.4.3.2 Transmit Interrupt

The transmit interrupt is generated every frame after the start of the transmission of a data word.

- In single-channel mode, the interrupt is generated one clock period plus a synchronization delay Tsyn after the beginning of the transmission of the word.
- In multichannel mode, the interrupt is generated one clock period plus a synchronization delay Tsyn after the transmission of the word of the channel whose number is defined by the MCSI.MCSI_INTR_REG[7:4] N_CHAN_IT_TX field.

Figure 25-26 shows the transmit interrupt timing diagram.

Figure 25-26. Transmit Interrupt Timing Diagram



25.4.3.3 Frame-Duration Error Interrupt

The frame-duration error interrupt is generated only when:

- The interface is configured in burst mode; that is, CONT_BURST bit MCSI.MCSI_MAIN_PARAM_REG[5] is set to 0.
- The frame duration is smaller or longer than the expected value.
  Expected frame duration = [(channels number) x (WORD_SIZE field)] + OVER_CLOCK field.

The MCSI.MCSI_OVER_CLOCK[9:0] OVER_CLOCK field is the oversize period in clock periods.

If the frame duration is longer than the expected value, the interrupt is generated one clock period after the number of the oversize clock periods plus a synchronization delay Tsyn, as defined in the OVER_CLOCK field.

If the frame duration is smaller than the expected value, the interrupt is generated one clock period plus a synchronization delay Tsyn after the occurrence of the next frame pulse (first active edge).

Figure 25-27 shows the frame duration error when the frame duration is longer than the expected value.

Figure 25-27. Frame Duration Error Too Many (Long)

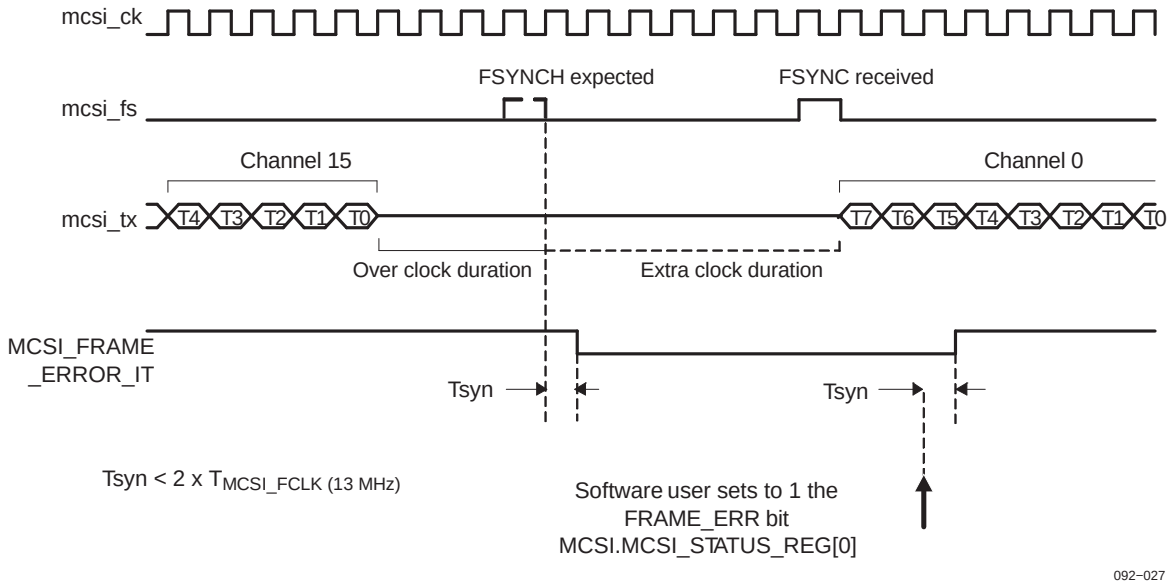
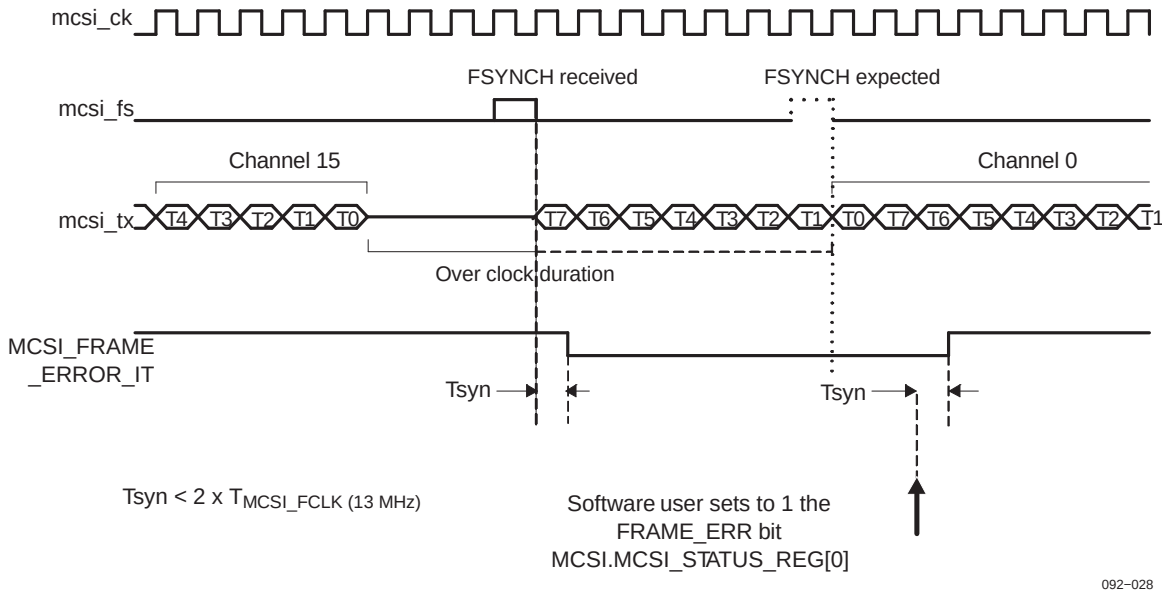


Figure 25-28 shows the frame duration error when the frame duration is shorter than the expected value.

Figure 25-28. Frame Duration Error Too Few (Short)

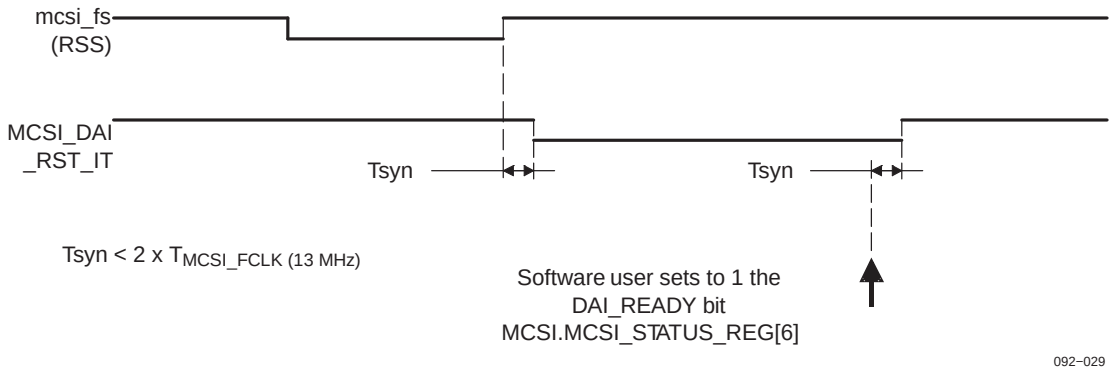


25.4.3.4 System Simulator Reset Interrupt

The system simulator reset interrupt (MCSI_DAI_RST_IT) is generated on the detection of a rising edge on the RSS signal (mcsi_fs) plus a synchronization delay Tsyn. This interrupt is not generated if the DAI mode is disabled (MCSI.MCSI_MAIN_PARAM_REG[13:12] DAI_CONFIG field set to 00).

Figure 25-29 shows the RSS interrupt.

Figure 25-29. System Simulator Reset Interrupt



25.4.4 System Power Management

The functional clock MCSI_FCLK of the MCSI module is stopped only in deep sleep mode to reduce power consumption.

For further information, see Chapter 6, *Power, Reset, and Clock Management*.

25.5 MCSI Programming Model

25.5.1 MCSI Software Reset

The MCSI software reset is activated with the SOFT_RESET bit MCSI.MCSI_CTRL_REG[1] of the control register. On the software reset, the MCSI reference clock is disabled, thus halting the execution of any current operating mode.

Note: This reset is limited to the control and status registers, the internal state machine and the PISO and SIPO logic. The parameters registers are not affected by this software reset.

25.5.2 MCSI Normal Mode Configuration

25.5.2.1 Start Sequence

A typical sequence to start the interface is:

1. MCSI configuration:
   a. Write 0x0000 in the MCSI.MCSI_CTRL_REG register to remove the write protection on the control registers.
   b. Program the MCSI.MCSI_MAIN_PARAM_REG register with the parameters required by your application.
   c. Program the MCSI.MCSI_INTR_REG register to select the RX and TX channel numbers and to mask interrupts, if required.
   d. In multichannel mode only, select the channel to use in the MCSI.MCSI_CHAN_USED_REG register.
   e. In master mode only, select the clock ratio in the MCSI.MCSI_CLOCK_FREQ_REG register.
   f. Select the oversized frame dimension in the MCSI.MCSI_OVER_CLOCK_REG register.
2. Transmit data loading for selected channels: Write data to be transmitted in MCSI.TXi_REG with the channel index $0 \leq i \leq 15$.
3. Enable MCSI clock: Set the CLK_ENABLE bit MCSI.MCSI_CTRL_ REG[0] to 1.

25.5.2.2 Stop Sequence

A typical sequence to stop the interface is:

1. Disable the MCSI clock: Set the CLK_ENABLE bit MCSI.MCSI_CTRL_REG[0] to 0.
   The status register keeps its content even after the transmission stops. The control registers can now be modified.
2. Software reset: Set the SOFT_RESET bit MCSI.MCSI_CTRL_REG[1] to 1.
   The software reset initializes the status register.

25.5.3 MCSI DAI Mode Configuration

For a DAI test, the MCSI interface must be configured to comply with the GSM 11.10 standard. For more details on DAI protocol, see the *ETSI GSM11.10 Standard* at www.etsi.org.

25.5.3.1 DAI Configuration

The sequence for DAI configuration is:

1. Write 0x0000 in the MCSI.MCSI_CHAN_USED_REG register.
2. Write 0x007D in the MCSI.MCSI_CLOCK_FREQ_REG register.
3. Write 0x0000 in the MCSI.MCSI_OVER_CLOCK_REG register.
4. Write 0x0000 in the MCSI.MCSI_INTR_REG register.
5. If radio uplink mode; otherwise, go to step 6:

a. With audio frame synchro:
   Slave mode: Write 0x2A1C in the MCSI.MCSI_MAIN_PARAM_REG register.
   Master mode: Write 0x2A5C in the MCSI.MCSI_MAIN_PARAM_REG register.
b. Without audio frame synchro:
   Slave mode: Write 0x221C in the MCSI.MCSI_MAIN_PARAM_REG register.
   Master mode: Write 0x225C in the MCSI.MCSI_MAIN_PARAM_REG register.

6. If radio downlink mode; otherwise, go to step 7:
   a. With audio frame synchro:
      Slave mode: Write 0x1A1C in the MCSI.MCSI_MAIN_PARAM_REG register.
      Master mode: Write 0x1A5C in the MCSI.MCSI_MAIN_PARAM_REG register.
   b. Without audio frame synchro:
      Slave mode: Write 0x121C in the MCSI.MCSI_MAIN_PARAM_REG register.
      Master mode: Write 0x125C in the MCSI.MCSI_MAIN_PARAM_REG register.
7. If radio acoustic mode:
   a. With audio frame synchro:
      Slave mode: Write 0x3A1C in MCSI.MCSI_MAIN_PARAM_REG register.
      Master mode: Write 0x3A5C in MCSI.MCSI_MAIN_PARAM_REG register.
   b. Without audio frame synchro:
      Slave mode: Write 0x321C in MCSI.MCSI_MAIN_PARAM_REG register.
      Master mode: Write 0x325C in MCSI.MCSI_MAIN_PARAM_REG register.

CAUTION

The programming of the DAI mode must be executed in two steps: first in DAI slave mode, and then in DAI master mode. This prevents the MCSI interface from detecting a pseudo-RSS when switching from slave mode to master mode, because the frame-synchro line is monitored by the interface in normal master mode (potential level change during mode switching, depending on the previous value of the FSYNC_POL bit MCSI.MCSI_MAIN_PARAM_REG[10].

25.5.3.2 Start Sequence

The sequence to start the interface is:

1. On RSS at LOW level:
   a. The MCSI clock is disabled, but the CLK_ENABLE bit MCSI.MCSI_CTRL_REG[0] retains its value (either 0 or 1).
   b. Set the DAI_CLK_ENABLE bit MCSI.MCSI_CTRL_REG[2] to 0.
   c. All registers remain unchanged (no software reset is needed).
2. On RSS at HIGH level:
   a. The MCSI clock is re-enabled if the CLK_ENABLE bit MCSI.MCSI_CTRL_REG[0] is set to 1.
   b. The MCSI_DAI_RST_IT interrupt is generated.
3. In radio downlink mode or radio acoustic mode; otherwise, go to step 4:
   a. Write data to transmit in the TX0_REG register.
   b. Write 0x0005 in the MCSI.MCSI_CTRL_REG register as soon as a first data word is loaded in the transmit word register to enable MCSI clock and DAI interface activity.
4. In radio uplink mode: Write 0x0005 in the MCSI.MCSI_CTRL_REG register as soon as possible to enable the MCSI clock and DAI interface activity.

Note: In DAI mode, the MCSI clock is disabled when RSS is low, regardless of the value of the CLK_ENABLE bit.

25.5.3.3 Stop Sequence

A typical sequence to stop the interface is as follows:

1. Disable the MCSI clock: Set the MCSI.MCSI_CTRL_REG[0] CLK_ENABLE bit and the MCSI.MCSI_CTRL_REG[2] DAI_CLK_ ENABLE bit to 0.
   The status register keeps its content even after the transmission stops.
   The control registers can now be modified.
2. Perform a software reset: Set the MCSI.MCSI_CTRL_REG[1] SOFT_ RESET bit to 1.
   The software reset initializes the status register.

25.5.4 Interrupt Programming Guide

At MCSI module reset, MCSI_RX_IT, MCSI_TX_IT, and MCSI_FRAME_ ERROR_IT are masked. MCSI_DAI_RST_IT cannot be masked.

To validate an interrupt:

If in multichannel mode, the RX and TX interrupts can be configured to occur in a dedicated channel of the frame [1_UnicodeEncodeError_16].

- Select the channel number by setting MCSI.MCSI_INTR_REG[3:0] N_CHAN_IT_RX field for MCSI_RX_IT.
- Select the channel number by setting MCSI.MCSI_INTR_REG[7:4] N_CHAN_IT_TX field for MCSI_TX_IT.

Unmask the interrupt:

1. Set to 1 the MASK_IT_RX bit MCSI.MCSI_INTR_REG[8] for MCSI_RX_IT.
2. Set to 1 the MASK_IT_TX bit MCSI.MCSI_INTR_REG[9] for MCSI_TX_IT.
3. Set to 1 the MASK_IT_ERR bit MCSI.MCSI_INTR_REG[10] for MCSI_FRAME_ERROR_IT.

On interrupt occurrence:

1. Read the FRAME_ERR bit MCSI.MCSI_STATUS_REG[0] for MCSI_FRAME_ERROR_IT occurrence.
2. Read the RX_READY bit MCSI.MCSI_STATUS_REG[2] for MCSI_RX_IT occurrence.
3. Read the RX_OVF read-only bit MCSI.MCSI_STATUS_REG[3] for RX character overflow.
4. Read the TX_READY bit MCSI.MCSI_STATUS_REG[4] for MCSI_TX_IT occurrence.
5. Read the TX_UF read-only bit MCSI.MCSI_STATUS_REG[5] for TX character underflow.
6. Read the DAI_READY bit MCSI.MCSI_STATUS_REG[6] for RSS detection (active only in DAI mode).

Then, to release the interrupt signal and reset the corresponding status bits:

1. Set to 1 the FRAME_ERR bit MCSI.MCSI_STATUS_REG[0] for MCSI_FRAME_ERROR_IT release.
2. Set to 1 the RX_READY bit MCSI.MCSI_STATUS_REG[2] for MCSI_RX_IT release.
3. Set to 1 the TX_READY bit MCSI.MCSI_STATUS_REG[4] for MCSI_TX_IT release.

CAUTION

To resume an interrupt, the status bit of this interrupt must be set to 1 in the status register. Thus, after reading the status bit of the interrupt and its associated error type bit (if any), the software user must write a 1 in the corresponding location of the status bit to release the interrupt signal and reset the status bit and the error type bit (if any).

25.5.5 Setup Example for Communications μ-Law Interface Using Interrupts

An example of communication μ-Law interface setup using interrupts is presented in this section.

MCSI configuration:

- Write 0x0000 in the MCSI.MCSI_CTRL_REG register to disable MCSI for setup.
- Write 0x0007 in the MCSI.MCSI_MAIN_PARAM_REG register for MCSI configuration:
  - Bit 10 (0b0): Positive polarity for frame
  - Bit 9 (0b0): Normal synchronization mode
  - Bit 8 (0b0): Short framing
  - Bit 7 (0b0): Single channel
  - Bit 6 (0b0): Slave mode
  - Bit 5 (0b0): Burst mode
  - Bit 4 (0b0): Positive edge for clock
  - Bit 3_UnicodeEncodeError_0 (0b0111): 8-bit data
- Write 0x0700 in the MCSI.MCSI_INTR_REG register to enable all interrupts.
- Write 0x0000 in the MCSI.MCSI_OVER_CLOCK_REG register.
- Write 0x0001 in the MCSI.MCSI_CTRL_REG register to start MCSI.

Transmit data loading (on the MCSI_TX_IT interrupt): Write data to transmit in the TX_REG register.

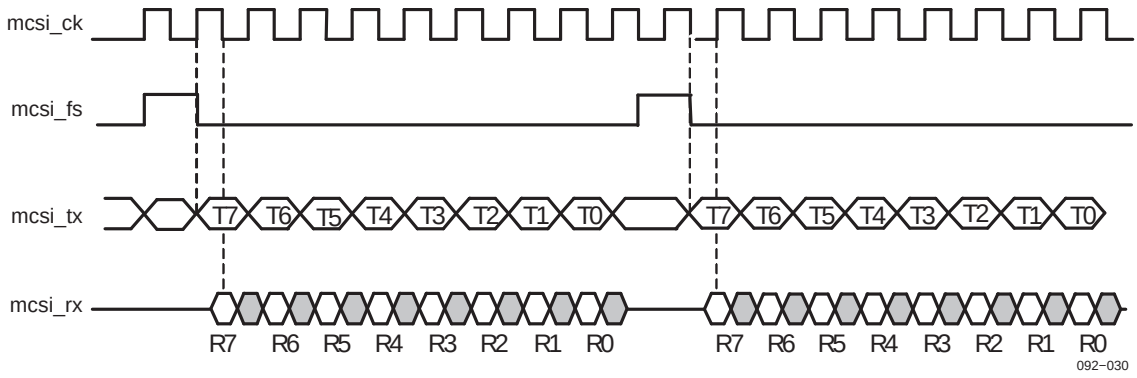
Received data loading (on the MCSI_RX_IT interrupt): Read received data in the RX_REG register.

Stop MCSI:

- Write 0x0000 in MCSI.MCSI_CTRL_REG register for disabling the MCSI clock.
- Write 0x0002 in MCSI.MCSI_CTRL_REG register for resetting.

Figure 25-30 shows the timing for the communication μ-Law interface interrupts waveform example.

Figure 25-30. Communication μ-Law Interface Interrupts Waveform Example



25.6 MCSI Register Manual

Table 25-6 throughTable 25-8 show the base address and address space for the MCSI module instances when accessed by the MPU or the DSP.

Table 25-6. Instance Summary for MPU

Module Name	MPU Base Address	Size
MCSI	0xFFFF 7800	2K bytes

Table 25-7. Instance Summary for DSP Data Page

Module Name	DSP Base Address	Size
MCSI	0x0000 7800	256 bytes

Table 25-8. Instance Summary for DSP XIO Page

Module Name	DSP Base Address	Size
MCSI	0x0000 0800	4K bytes

25.6.1 MCSI Register Mapping Summary

Table 25-9 lists the MCSI registers, including type, width, and address offsets.

Table 25-9. MCSI Registers Address Offset

Register Name	Type	Register Width (Bits)(1)	DSP Offset	MPU Offset
MCSI.MCSI_CTRL_REG	RW	16	0x00	0x00
MCSI.MCSI_MAIN_ PARAM_REG	RW	16	0x01	0x02
MCSI.MCSI_INTR_REG	RW	16	0x02	0x04
MCSI.MCSI_CHAN_USED_ REG	RW	16	0x03	0x06
MCSI.MCSI_OVER_CLOCK_ REG	RW	16	0x04	0x08
MCSI.MCSI_CLOCK_FREQ_ REG	RW	16	0x05	0x0A
MCSI.MCSI_STATUS_REG	RW	16	0x06	0x0C
TX0_REG	RW	16	0x20	0x40
TX1_REG	RW	16	0x21	0x42
TX2_REG	RW	16	0x22	0x44
TX3_REG	RW	16	0x23	0x46
TX4_REG	RW	16	0x24	0x48
TX5_REG	RW	16	0x25	0x4A
TX6_REG	RW	16	0x26	0x4C
TX7_REG	RW	16	0x27	0x4E
TX8_REG	RW	16	0x28	0x50
TX9_REG	RW	16	0x29	0x52
TX10_REG	RW	16	0x2A	0x54
TX11_REG	RW	16	0x2B	0x56
TX12_REG	RW	16	0x2C	0x58
TX13_REG	RW	16	0x2D	0x5A
TX14_REG	RW	16	0x2E	0x5C

(1) The MCSI registers are limited to 16-bit accesses. 32-bit and 8-bit data accesses generate an error. For further details, see Chapter 5, *Interconnect*.

Table 25-9. MCSI Registers Address Offset (continued)

Register Name	Type	Register Width (Bits)(1)	DSP Offset	MPU Offset
TX15_REG	RW	16	0x2F	0x5E
RX0_REG	R	16	0x30	0x60
RX1_REG	R	16	0x31	0x62
RX2_REG	R	16	0x32	0x64
RX3_REG	R	16	0x33	0x66
RX4_REG	R	16	0x34	0x68
RX5_REG	R	16	0x35	0x6A
RX6_REG	R	16	0x36	0x6C
RX7_REG	R	16	0x37	0x6E
RX8_REG	R	16	0x38	0x70
RX9_REG	R	16	0x39	0x72
RX10_REG	R	16	0x3A	0x74
RX11_REG	R	16	0x3B	0x76
RX12_REG	R	16	0x3C	0x78
RX13_REG	R	16	0x3D	0x7A
RX14_REG	R	16	0x3E	0x7C
RX15_REG	R	16	0x3F	0x7E

25.6.2 MCSI Register Description

Table 25-10 throughTable 25-18 describe the individual bit fields of the MCSI registers.

Table 25-10. MCSI_CTRL_REG

Address Offset	0x00		
Physical address	MPU: 0xFFFF 7800 DSP: 0x0000 7800	**Instance**	MCSI
Description	Activity control register		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													DAI_CLK_ENABLE	SOFT_RESET	CLK_ENABLE

Bits	Field Name	Description	Type	Reset
15:3	RESERVED	Reserved. Write has no effect on this field. Read returns 0s.	R	0x0000
2	DAI_CLK_ENABLE	Enables DAI interface activity: 0: Disable 1: Enable	RW	0

Bits	Field Name	Description	Type	Reset
1	SOFT_RESET(1)	Asynchronous reset of MCSI module: 0: Disable 1: Enable	RW	0
0	CLK_ENABLE	Enables MCSI module clock: 0: Disable 1: Enable	RW	0

(1) The software reset is applied as long as the MCSI software reset bit is set to 1. A software reset disables the MCSI module (the CLK_ENABLE bit is cleared) and initializes the status register (MCSI.MCSI_STATUS_REG), but does not modify the other registers.

Table 25-11. MCSI_MAIN_PARAM_REG(1)

Address Offset	MPU: 0x02 DSP: 0x01		
Physical address	MPU: 0xFFFF 7802 DSP: 0x0000 7801	**Instance**	MCSI
Description	Activity control register		
Type	RW		

(1) This register is write-protected if the MCSI module is enabled with CLK_ENABLE bit MCSI.MCSI_CTRL_REG[0] set to 1.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		DAI_CONFIG		DAI_SYNC_REQ	FSYNCH_POL	FSYNCH_MODE	FSYNCH_SIZE	MULTI_SINGLE	MCSI_MODE	CONT_BURST	CLK_POL	WORD_SIZE			

Bits	Field Name	Description	Type	Reset
15:14	RESERVED	Do not write any value in this field.	RW	00
13:12	DAI_CONFIG	DAI mode selection: 00: Normal (no DAI) 01: Radio downlink 10: Radio uplink 11: Acoustic mode	RW	0
11	DAI_SYNC_REQ	Synchronization with audio frame: 0: No synchro mode 1: Synchro mode	RW	0
10	FSYNCH_POL	Frame-synchronization pulse polarity: 0: Positive 1: Negative	RW	0
9	FSYNCH_MODE	Frame-synchronization pulse position: 0: Normal 1: Alternate	RW	0
8	FSYNCH_SIZE	Frame-synchronization pulse shape: 0: Short 1: Long	RW	0
7	MULTI_SINGLE	Frame structure: 0: Single 1: Multi	RW	0
6	MCSI_MODE	Interface transmission mode: 0: Slave 1: Master	RW	0

Bits	Field Name	Description	Type	Reset
5	CONT_BURST	Frame mode: 0: Burst 1: Continuous	RW	0
4	CLK_POL	Clock edge selection: 0: Positive 1: Negative	RW	0
3:0	WORD_SIZE	Word size in bit number _UnicodeEncodeError_1 2 ≤ WORD_SIZE ≤ 15 The word size is WORD_SIZE + 1. (2 for 3 bits and 15 for 16 bits)	RW	000

Table 25-12. MCSI_INTR_REG[1]

Address Offset	MPU: 0x04 DSP: 0x02		
Physical address	MPU: 0xFFFF 7804 DSP: 0x000 7802	**Instance**	MCSI
Description	Interrupt mask register		
Type	RW		

(1) This register is write-protected if the MCSI module is enabled with CLK_ENABLE bit MCSI.MCSI_CTRL_REG[0] set to 1.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					MASK_IT_ERR	MASK_IT_TX	MASK_IT_RX	N_CHAN_IT_TX				N_CHAN_IT_RX			

Bits	Field Name	Description	Type	Reset
15:11	RESERVED	Reserved. Write has no effect on this field. Read returns 0s.	R	00000
10	MASK_IT_ERR	Mask of frame duration error interrupt (active at 0)	RW	0
9	MASK_IT_TX	Mask of transmit interrupt (active at 0)	RW	0
8	MASK_IT_RX	Mask of receive interrupt (active at 0	RW	0
7:4	N_CHAN_IT_TX	Channel number for transmit interrupt generation with 0 ≤ N_CHAN_IT_TX ≤ 15	RW	0000
3:0	N_CHAN_IT_RX	Channel number for receive interrupt generation with 0 ≤ N_CHAN_IT_RX ≤ 15	RW	0000

Table 25-13. MCSI_CHAN_USED_REG[1][2]

Address Offset	MPU: 0x06 DSP: 0x03		
Physical address	MPU: 0xFFFF 7806 DSP: 0x0000 7803	**Instance**	MCSI
Description	Channel selection register		
Type	RW		

(1) This register is write-protected if the MCSI module is enabled with CLK_ENABLE bit MCSI.MCSI_CTRL_REG[0] set to 1.
(2) USE_CHi bit selects channel [i] for data transmission (active high).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USE_CH15	USE_CH14	USE_CH13	USE_CH12	USE_CH11	USE_CH10	USE_CH9	USE_CH8	USE_CH7	USE_CH6	USE_CH5	USE_CH4	USE_CH3	USE_CH2	USE_CH1	USE_CH0

Bits	Field Name	Description	Type	Reset
15	USE_CH15	Selects channel 15 for data transmission (active high)	RW	0
14	USE_CH14	Selects channel 14 for data transmission (active high)	RW	0
13	USE_CH13	Selects channel 13 for data transmission (active high)	RW	0
12	USE_CH12	Selects channel 12 for data transmission (active high)	RW	0
11	USE_CH11	Selects channel 11 for data transmission (active high)	RW	0
10	USE_CH10	Selects channel 10 for data transmission (active high)	RW	0
9	USE_CH9	Selects channel 9 for data transmission (active high)	RW	0
8	USE_CH8	Selects channel 8 for data transmission (active high)	RW	0
7	USE_CH7	Selects channel 7 for data transmission (active high)	RW	0
6	USE_CH6	Selects channel 6 for data transmission (active high)	RW	0
5	USE_CH5	Selects channel 5 for data transmission (active high)	RW	0
4	USE_CH4	Selects channel 4 for data transmission (active high)	RW	0
3	USE_CH3	Selects channel 3 for data transmission (active high)	RW	0
2	USE_CH2	Selects channel 2 for data transmission (active high)	RW	0
1	USE_CH1	Selects channel 1 for data transmission (active high)	RW	0
0	USE_CH0	Selects channel 0 for data transmission (active high)	RW	0

Table 25-14. MCSI_OVER_CLOCK_REG[(1)]

Address Offset	MPU: 0x08 DSP: 0x04		
Physical address	MPU: 0xFFFF 7808 DSP: 0x0000 7804	**Instance**	MCSI
Description	Oversized frame dimension register		
Type	RW		

(1) This register is write-protected if the MCSI module is enabled with CLK_ENABLE bit MCSI.MCSI_CTRL_REG[0] set to 1.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						OVER_CLOCK									

Bits	Field Name	Description	Type	Reset
15:10	RESERVED	Reserved. Write has no effect on this field. Read returns 0s.	R	0x00
9:0	OVER_CLOCK	Overhead clock periods in frame duration (0 ≤ OVER_CLOCK field ≤ 0x3FF)	RW	0x000

Table 25-15. MCSI_CLOCK_FREQ_REG[1][2]

Address Offset	MPU: 0x0A DSP: 0x05
Physical address	MPU: 0xFFFF 780A **Instance** MCSI DSP: 0x0000 7805
Description	Clock frequency register
Type	RW

(1) This register is used only in master mode (MCSI_MODE bit MCSI.MCSI_MAIN_PARAM_REG[6] set to 1) when the interface generates the serial clock.
(2) This register is write-protected if the MCSI module is enabled with CLK_ENABLE bit MCSI.MCSI_CTRL_REG[0] set to 1.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CLK_FREQ							

Bits	Field Name	Description	Type	Reset
15:11	RESERVED	Reserved. Write has no effect on this field. Read returns 0s.	R	0x00
10:0	CLK_FREQ	Division factor of 13-MHz reference clock In master mode, this register defines the transmission baud rate from a frequency ratio based on a 13-MHz reference clock. The transmission clock frequency can be programmed from 6.35 kHz to 6.5 MHz. Clock frequency =13 MHz/CLK_FREQ with 0x002 ≤ CLK_FREQ ≤ 0x7FF **Note:** If CLK_FREQ = 0x000 or 0x001, the 13-MHz functional clock is divided by 2 as if CLK_FREQ = 0x002.	RW	0x00000

Table 25-16. MCSI_STATUS_REG[1][2]

Address Offset	MPU: 0x0C DSP: 0x06
Physical address	MPU: 0xFFFF 780C **Instance** MCSI DSP: 0x0000 7806
Description	Interface status register
Type	RW

(1) This register is cleared by a software reset with the SOFT_RESET bit.
(2) To clear an interrupt on the MCSI, the DSP or MPU must write to the MCSI status register with the bit corresponding to the interrupt set to 1. The MCSI status register has a 2-cycle latency when written to, so the interrupt line is cleared two cycles after a write. The interrupt routine must be at least two cycles long to prevent clearing the interrupt handler before the interrupt line is cleared.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								DAI_READY	TX_UF	TX_READY	RX_OVF	RX_READY	ERROR_TYPE	FRAME_ERR	

Bits	Field Name	Description	Type	Reset
15:7	RESERVED	Reserved. Write has no effect on this field. Read returns 0s.	R	0x00
6	DAI_READY	Flag for system simulator reset detection: 0: No detect 1: Detect	RW	0

Bits	Field Name	Description	Type	Reset
5	TX_UF	Transmit underflow: 0: No underflow 1: Underflow	R	0
4	TX_READY	Flag for transmit interrupt occurrence: 0: No TX interrupt 1: TX interrupt	RW	0
3	RX_OVF	Receive overflow: 0: No overflow 1: Overflow	R	0
2	RX_READY	Flag for receive interrupt occurrence: 0: No RX interrupt 1: RX interrupt	RW	0
1	ERROR_TYPE	Too short (few) or too long (many) frame status: 0: Short 1: Long	R	0
0	FRAME_ERR	Error flag when wrong frame duration: 0: Correct (No MCSI_FRAME_ERROR_IT interrupt) 1: Bad (MCSI_FRAME_ERROR_IT interrupt)	RW	0

Table 25-17. TXi_REG With 0 ≤ i ≤ 15[(1)]

Address Offset	MPU: From 0x40 (TX0_REG) to 0x5E (TX15_REG)		
	DSP: From 0x20 (TX0_REG) to 0x2F (TX15_REG)		
Physical address	TX0_REG:	**Instance**	MCSI
	MPU: 0xFFFF 7840		
	DSP: 0x0000 7820		
	TX15_REG:		
	MPU: 0xFFFF 785E		
	DSP: 0x0000 782F		
Description	Transmit word registers		
Type	RW		

(1) The MCSI transmits the MSB first. For example, if the WORD_SIZE field equals 0xB, the upper 12 bits of the MCSI.TXi_REG register are transmitted.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0

Bits	Field Name	Description	Type	Reset
15	B15	Bit 15 of the data word to be transmitted	RW	Undefined
14	B14	Bit 14 of the data word to be transmitted	RW	Undefined
13	B13	Bit 13 of the data word to be transmitted	RW	Undefined
12	B12	Bit 12 of the data word to be transmitted	RW	Undefined
11	B11	Bit 11 of the data word to be transmitted	RW	Undefined
10	B10	Bit 10 of the data word to be transmitted	RW	Undefined
9	B9	Bit 9 of the data word to be transmitted	RW	Undefined
8	B8	Bit 8 of the data word to be transmitted	RW	Undefined
7	B7	Bit 7 of the data word to be transmitted	RW	Undefined
6	B6	Bit 6 of the data word to be transmitted	RW	Undefined
5	B5	Bit 5 of the data word to be transmitted	RW	Undefined

Bits	Field Name	Description	Type	Reset
4	B4	Bit 4 of the data word to be transmitted	RW	Undefined
3	B3	Bit 3 of the data word to be transmitted	RW	Undefined
2	B2	Bit 2 of the data word to be transmitted	RW	Undefined
1	B1	Bit 1 of the data word to be transmitted	RW	Undefined
0	B0	Bit 0 of the data word to be transmitted	RW	Undefined

Table 25-18. RXi_REG With 0 ≤ i ≤ 15[(1)]

Address Offset	MPU: From 0x60 (RX0_REG) to 0x7E (RX15_REG) DSP: From 0x30 (RX0_REG) to 0x3F (RX15_REG)		
Physical address	RX0_REG: MPU: 0xFFFF 7860 DSP: 0x0000 7830 RX15_REG: MPU: 0xFFFF 787E DSP: 0x0000 783F	**Instance**	MCSI
Description	Received word registers		
Type	R		

(1) The MCSI receives the MSB first. For example, if the WORD_SIZE field equals 0xB, the upper 12 bits of the MCSI.RXi_REG register are transmitted, and the lower 4 bits are zeros.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0

Bits	Field Name	Description	Type	Reset
15	B15	Bit 15 of the received data word	R	Undefined
14	B14	Bit 14 of the received data word	R	Undefined
13	B13	Bit 13 of the received data word	R	Undefined
12	B12	Bit 12 of the received data word	R	Undefined
11	B11	Bit 11 of the received data word	R	Undefined
10	B10	Bit 10 of the received data word	R	Undefined
9	B9	Bit 9 of the received data word	R	Undefined
8	B8	Bit 8 of the received data word	R	Undefined
7	B7	Bit 7 of the received data word	R	Undefined
6	B6	Bit 6 of the received data word	R	Undefined
5	B5	Bit 5 of the received data word	R	Undefined
4	B4	Bit 4 of the received data word	R	Undefined
3	B3	Bit 3 of the received data word	R	Undefined
2	B2	Bit 2 of the received data word	R	Undefined
1	B1	Bit 1 of the received data word	R	Undefined
0	B0	Bit 0 of the received data word	R	Undefined

Master/Slave SPI

This chapter describes the master/slave serial port interface (MSSPI) module of the LOCOSTO device.

Topic **Page**

26.1 MSSPI Overview

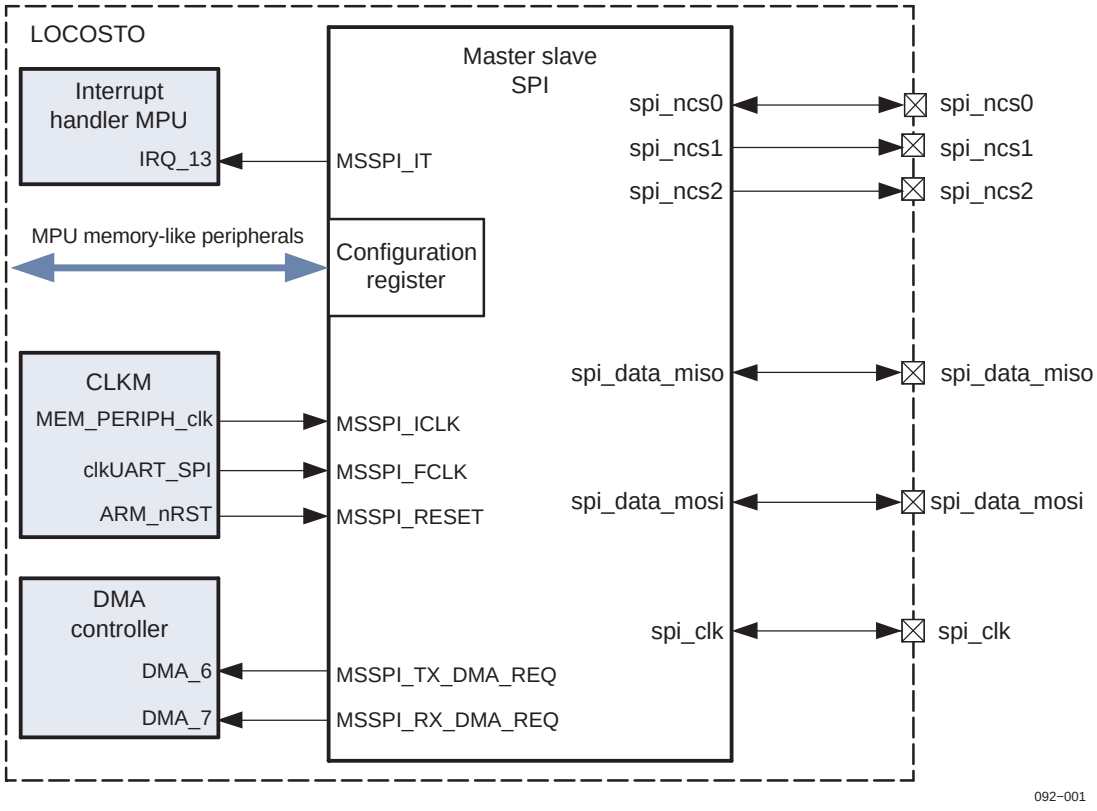
The master/slave serial port interface (MSSPI) module complies with the serial port interface (SPI) standard. It is a bidirectional, 4-line interface consisting of:

- A clock, used to shift data in and out
- Multiple device enable
- Multiple data input
- Multiple data output

The LOCOSTO device has one MSSPI module. It is based on a looped shift register, thus allowing both transmit and receive modes. It can operate in master or in slave mode using microprocessor unit (MPU) or direct memory access (DMA) protocol.

In master mode, the SPI provides up to three chip-selects: spi_ncs0, spi_ncs1, and spi_ncs2. In slave mode, spi_ncs0 as the SPI module input chip-select.

Figure 26-1. MSSPI Overview



26.1.1 Main Features

- Complies with the SPI standard
- Serial clock with programmable frequency, polarity, and phase
- Wide selection of SPI word lengths from 1 to 32 bits
- Master or slave mode:
  - Full duplex
  - Transmit only/receive only/transmit and receive modes
  - Flexible input/output (I/O) port controls
- Two DMA requests

- Single interrupt line for interrupt source events
- Maximum SPI data rate is 26M bits/s

26.1.2 Signals Input/Output Descriptions

Table 26-1 lists the MSSPI I/O descriptions.

Table 26-1. MSSPI I/O Descriptions

Signal Name	I/O[1]	Description	Value at Reset
spi_clk	I/O	MSSPI serial clock	Unknown
spi_data_mosi	I/O	MSSPI serial data, master-out slave-in	Unknown
spi_data_miso	I/O	MSSPI serial data, master-in slave-out	Unknown
spi_ncs0	I/O	MSSPI chip-select 0	High
spi_ncsi	O	MSSPI chip-select i output, i=[1:2]	High

(1) I = Input, O = Output

26.1.3 MSSPI Environment

Figure 26-2 through Figure 26-5 show how to interface the MSSPI module to external devices in master/slave and single-/full-duplex modes.

26.1.3.1 MSSPI in Master Mode Functional Interface

Figure 26-2 and Figure 26-3 show two master mode cases.

Figure 26-2. MSSPI Master Mode (Full-Duplex)

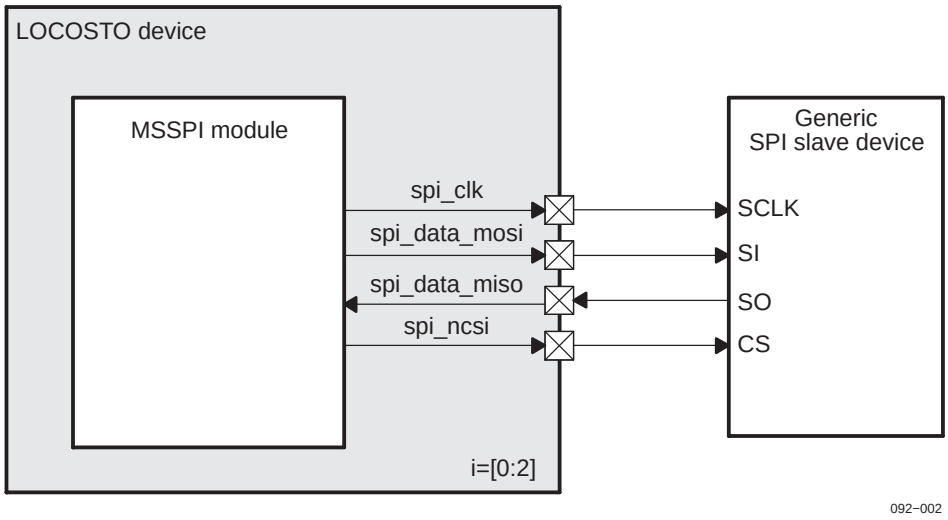


Figure 26-3. MSSPI Master Mode (Transmit-Only)

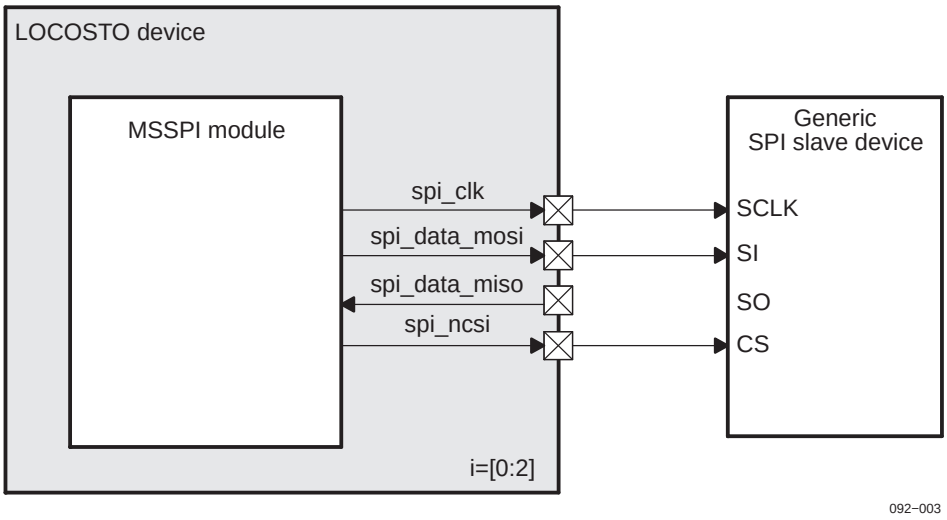


Table 26-2 lists the master mode interface descriptions.

Table 26-2. MSSPI Master Mode I/O Descriptions

Signal Name	I/O[1]	Description	Value at Reset
spi_clk	O	MSSPI serial clock	Unknown
spi_data_mosi	O	MSSPI serial data master-out	Unknown
spi_data_miso	I	MSSPI serial data master-in	Unknown
spi_ncsi	O	MSSPI chip-select i output, i=[0:2]	High

(1) I = Input, O = Output

26.1.3.2 MSSPI in Slave Mode Functional Interface

In this mode, only the spi_ncs0 chip-select can be used (see Table 26-3).

Figure 26-4 and Figure 26-5 show two slave mode cases.

Figure 26-4. MSSPI Slave Mode (Full-Duplex)

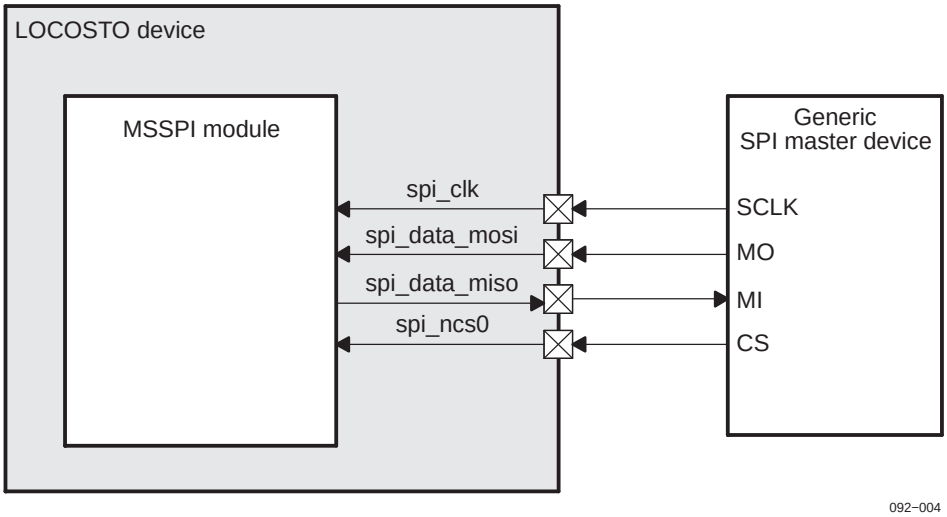


Figure 26-5. MSSPI Slave Mode (Transmit-Only)

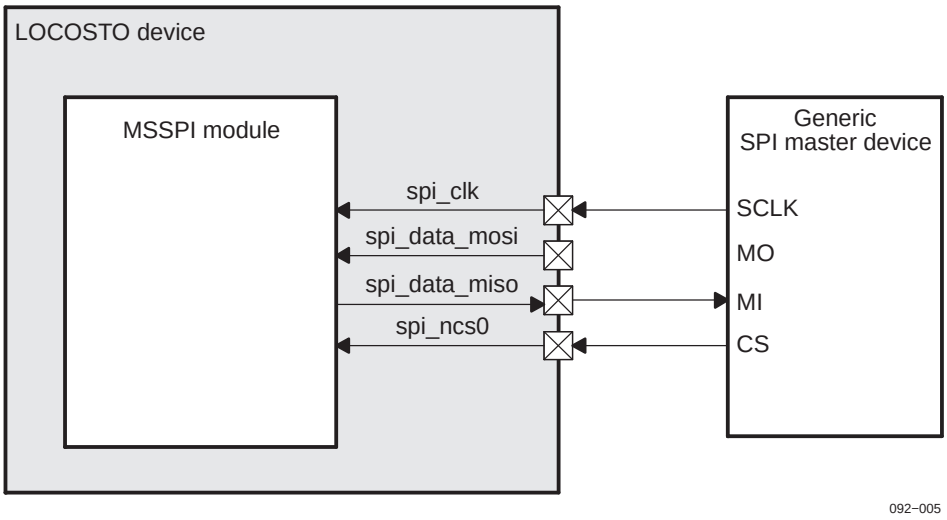


Table 26-3. MSSPI Slave Mode I/O Description

Signal Name	I/O[(1)]	Description	Value at Reset
spi_clk	I	MSSPI serial clock	Unknown
spi_data_mosi	I	MSSPI serial data slave-in	Unknown
spi_data_miso	O	MSSPI serial data slave-out	Unknown
spi_ncs0	I	MSSPI enable	High

(1) I = Input, O = Output

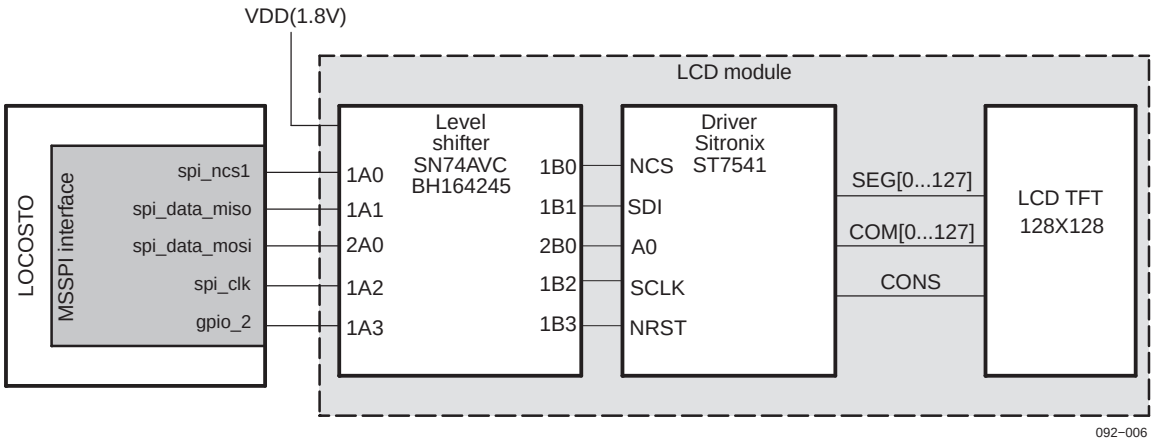
26.1.3.3 Secondary LCD Implementation

A secondary liquid crystal display (LCD) is physically placed on top of a clamshell mobile phone and can be connected to the LOCOSTO MSSPI interface. This LCD is the three-five system (TFS) LCD. The module consists of a transflective positive-image FSTN 128 x 128 black and white monochrome LCD, with a flex interface for insertion in a ZIF, JAE FF0115SA2, or equivalent.

An integral LED light guide is also provided. Yellow and green LEDs are on the customer printed circuit board (PCB). The LCD must be connected to the Sitronix ST7541 driver, MSSPI 4 lines interface. The secondary LCD (TFS LCD and Sitronix driver) must have 0.5 mA/2.8 V supplied by the external 2.8-V LDO using level shifters to connect to the LOCOSTO device.

Figure 26-6 shows the secondary LCD implementation.

Figure 26-6. LCD Secondary Implementation



26.1.4 MSSPI Protocol and Data Format

The MSSPI module offers the flexibility of modifying the following parameters to adapt to the device features:

- Word length:

  The MSSPI module supports any word ranging from 1 to 32 bits long (bits MSSPI.SPI_CTRL[6:2]). A word length can be changed between transmissions to allow the master device to communicate with peripheral slaves that have different requirements. The word length must be programmed when the MSSPI module is configured in both master and slave modes.
- SPI enable (spi_ncsi):

  The polarity of the SPI enable signals is controlled by the CE field MSSPI.SPI_SET2 register. The spi_ncsi signals may be active high or low.
- Programmable SPI clock:
  - Baud rate:

    In master mode, the baud rate of the MSSPI serial clock is programmable by changing the PTV field MSSPI.SPI_SET1[4:1].
  - Polarity and phase:

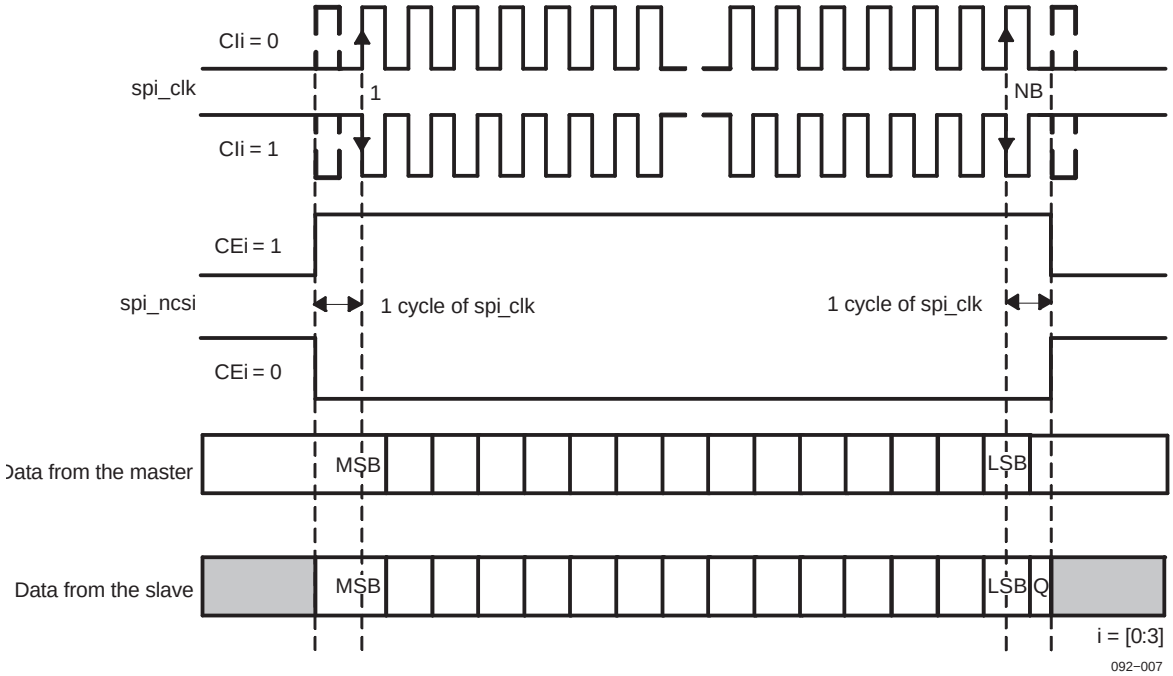
    The polarity and phase of the SPI serial clock are configurable per chip-select to offer four combinations. The polarity is controlled by the CI field MSSPI.SPI_SET2[3:0]. The phase is controlled by the CP field MSSPI.SPI_SET2[13:10]. Even in slave mode, the software selects the correct combination according to the device.

The serial interface becomes active when the spi_clk signal is activated.

- Transmission of NB bits with CPi = 0:

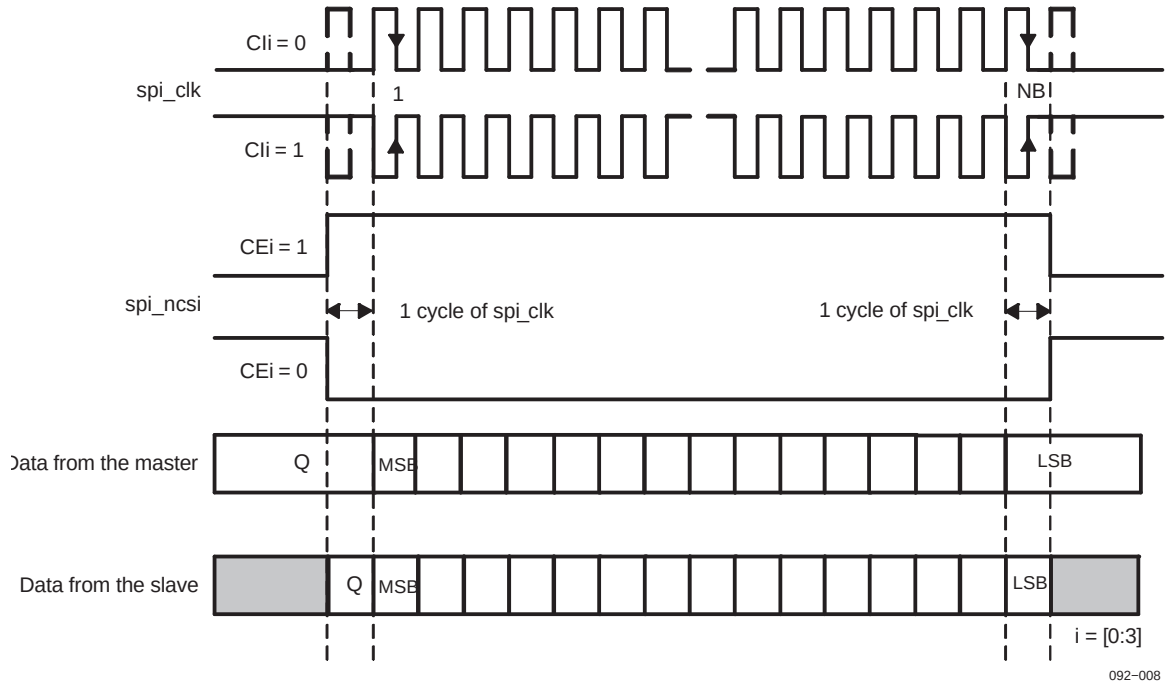
  When CPi = 0, the first edge (rising or falling) of spi_clk is used to capture the data and the second edge (falling or rising) is used to shift the data, as shown in Figure 26-7.

Figure 26-7. Transmission With CPi = 0



- Transmission of NB bits with CPi = 1:
  When CPi = 1, the first edge (rising or falling) of spi_clk is used to shift the data and the second edge (falling or rising) is used to capture the data, as shown in Figure 26-8.

Figure 26-8. Transmission With CPi = 1



26.1.5 Clocking, Reset, and Power-Management Scheme

26.1.5.1 Clocks

Table 26-4 lists the characteristics of the clock.

- The MSSPI operates from a variable functional clock of 52-, 48-, or 13-MHz generated by DPLL, APLL, and DCXO, respectively, through the clock and reset management (CLKM) module (see Chapter 6, *Power, Reset, and Clock Management*, for details). The functional clock is fed into a divider that allows division from 1 to 4096.
- The interface clock, MSSPI_ICLK, triggers access to the MSSPI controller registers for reading or writing (see Chapter 6, *Power, Reset, and Clock Management*, for details).

Table 26-4. Clocks

Type	Name	Source	Frequency	Description
Functional	MSSPI_FCLK	ClkUART_SPI	52, 48, or 13 MHz	Clock into the module to generate Tx and Rx shift
Interface	MSSPI_ICLK	MEM_PERIPH_clk	52 MHz	Clock bus to interface MSSPI

26.1.5.2 Power Management and Power Domain

At the system level, power-reduction techniques can be applied by shutting down certain internal clock and power domains of the device.

- Sleep mode: The module clock is stopped internally by resetting the EN_CLK bit MSSPI.SPI_SET1[0] bit. Because most registers are clocked using these clocks, this greatly reduces power consumption.
- Autoidle: The MSSPI.SPI_SCR[0] AUTOIDLE bit can be set to save power. This bit controls the internal open-core protocol (OCP) clock activity:
  - When this bit is cleared, the internal clock is free-running.
  - When this bit is set, the internal OCP clock becomes inactive if the OCP command is in idle state.

26.1.5.3 Hardware and Software Resets

The MSSPI has hardware and software resets. Table 26-5 lists the characteristics of each reset.

- Global reset or hardware reset of the MSSPI can be performed by activating the ARM_nRST (see Chapter 6, *Power, Reset, and Clock Management*, for details).
- The software reset can be performed by setting the SOFTRESET bit MSSPI.SPI_SCR[1]. For details, see Section 26.3.

Table 26-5. Hardware and Software Resets

Type	Name	Source	Activation	Domain
Hardware	MSSPI_RESET	ARM_nRST	0	Global reset
Software	SOFTRESET	SPI_SCR register	1	Module reset

26.1.6 Hardware Requests

26.1.6.1 DMA Requests

Table 26-6 lists the MSSPI DMA request mapping.

Table 26-6. MSSPI DMA Requests

DMA Request Name	Mapping	Description
MSSPI_TX_DMA_REQ	DMA_6	Request generated for MSSPI transmit
MSSPI_RX_DMA_REQ	DMA_7	Request generated for MSSPI received

26.1.6.2 Interrupt Requests

Table 26-7 lists a mapping of the interrupt generated by the MSSPI and its destination. One interrupt, MSSPI_IT, is generated by five maskable events to drive IRQ_13 on the MPU interrupt handler.

Table 26-7. MSSPI Module Interrupt Mapping

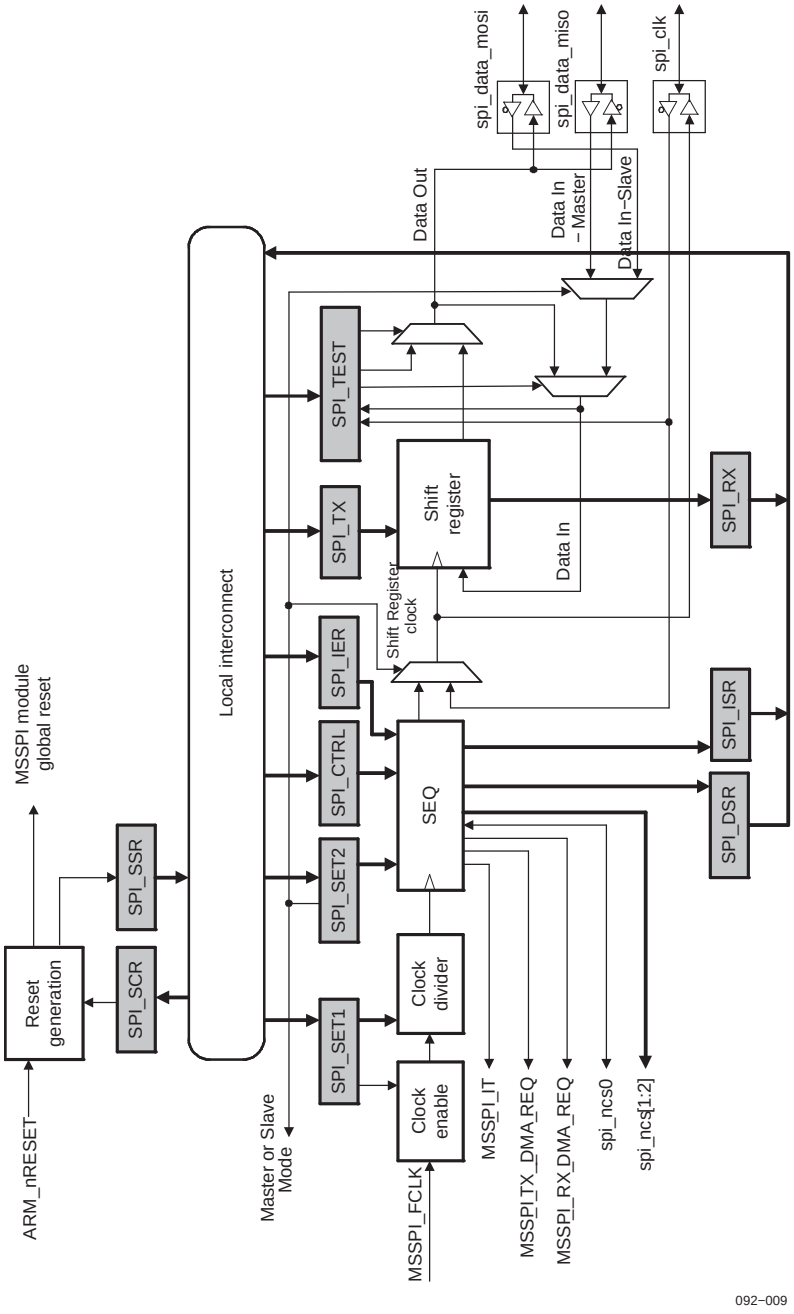
Interrupt Name	Mapping	Domain	Description
MSSPI_IT	IRQ_13	MPU	MSSPI module interrupt to MPU interrupt handler

26.2 MSSPI Functional Description

26.2.1 Block Diagram

Figure 26-9 is the MSSPI block diagram.

Figure 26-9. MSSPI Block Diagram



A read transaction is always simultaneous with a write transaction because the internal shift register is based on a loop (FIFO principle). However, the concurrent write process can be a dummy write if there is no data to be transmitted.

Depending on the mode selection (master or slave mode), the shift register clock can be derived internally from the MSSPI_FCLK or can come directly from the spi_clk input signal.

- In master mode, the functional clock (MSSPI_FCLK) is divided according to the PTV field (MSSPI.SPI_SET1[4:1]) to provide the internal shift register clock and the serial clock (spi_clk). The shift register clock and spi_clk frequencies equal MSSPI_FCLK/2^{PTV}.
- In slave mode, the shift register clock comes from the serial clock (spi_clk) provided by the external master device.

CAUTION

PTV = 0 is forbidden when MSSPI_FCLK = 48 or 52 MHz.

The transfer of a packet starts when the transmit clock is generated.

The received/transmitted data packet is shifted in/shifted out on the rising or falling edge of the shift register clock, depending on the polarity and phase configurations.

The loading of the packet is then validated on deactivation of the chip-select signal (rising or falling edge).

The MSSPI.SPI_ISR register is updated at the end of a transaction in MPU and DMA modes (master or slave), and an interrupt (MSSPI_IT) can be generated, depending on the value of the SPI_IER bits.

In MPU protocol, the interrupt must not be masked; thus, the SPI can issue an interrupt and notify the host that the transaction is finished.

26.2.2 Master Mode

26.2.2.1 Master Mode Features

The MSSPI master mode supports communication with up to three independent SPI devices. MSSPI initiates a data transfer on the data lines (spi_data_mosi and spi_data_miso) and generates clock (spi_clk) and control (spi_ncsi) signals.

When connected to multiple external devices, the MSSPI exchanges data with one SPI slave device at a time through two main modes:

- Two-data-pins interface mode (transmit and receive mode)
- Single-data-pin interface mode (recommended for half-duplex transmission)

The MSSPI is in master mode when the MSSPI.SPI_SET2[15] MODE bit is set.

26.2.2.2 Master Transmit and Receive Mode (Full-Duplex)

In full-duplex transmission, data is transmitted (shifted out serially on spi_data_mosi) and received (shifted in serially on spi_data_miso) simultaneously on separate data lines.

Access to the shift registers is based on the states of the SPI_TX transmit register and the SPI_RX receive registers.

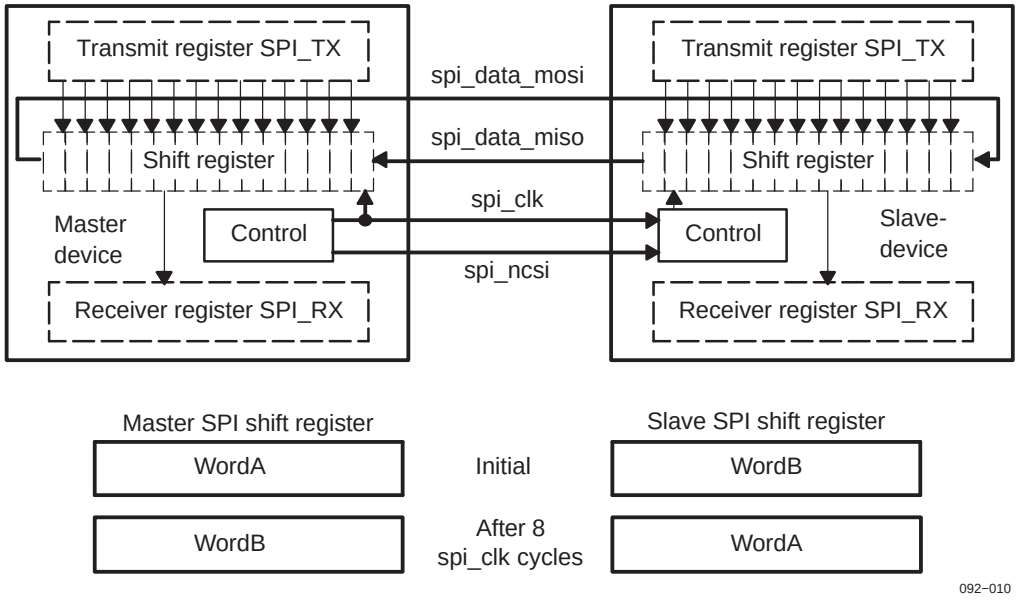
Both read and write transactions start when the MSSPI.SPI_CTRL[0] RD bit is set. This bit is automatically reset by the hardware after one cycle of spi_clk.

The serial clock (spi_clk) synchronizes shifting and sampling of the information on the two serial data lines (spi_data_mosi and spi_data_miso). Each time a bit is transferred out from the master device, a bit is transferred in from the slave device.

A read-end (RE) interrupt is generated exactly one spi_clk cycle plus two cycles of the interface clock (MSSPI_ICLK) after the last bit of the transaction is received, if MSK0 is set in the interrupt-enable register (MSSPI.SPI_IER[0]). The receive data can be read in receive register MSSPI.SPI_RX.

Figure 26-10 shows an example of a full-duplex, 1-byte data exchange between a master device (on the left) and a slave device (on the right).

Figure 26-10. MSSPI Full-Duplex Transmission Example

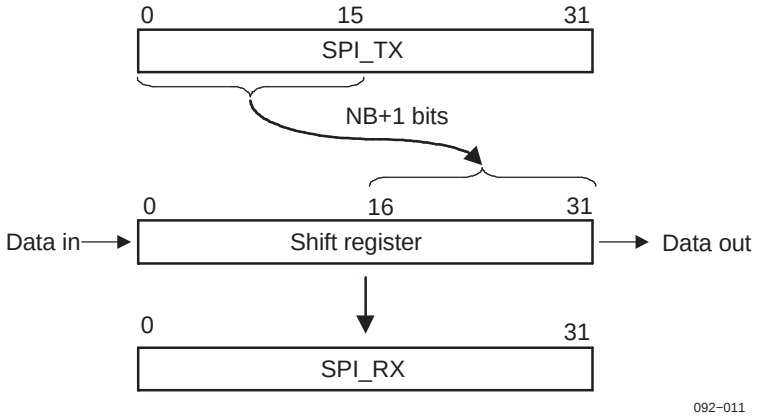


After eight cycles of the spi_clk serial clock, WordA is transferred from the master to the slave. At the same time, WordB is transferred from the slave to the master.

26.2.2.2.1 Transmit Register

The number of bits in the word to be transmitted is programmed through the MSSPI.SPI_CTRL[6:2] NB bit. The transmit register (SPI_TX) data loading must be completed according to the transmitted word bit length and in the correct register access sequence. Figure 26-11 is an example of a correct transmission.

Figure 26-11. Transmission Example



The SPI_TX register is considered updated (transmit register full with new data) when the most-significant bit (MSB) part of the transmitted word is written. If the number of bits of the transmitted word is not aligned on a byte boundary, the unused bit values are considered don't care.

- $0 \le NB \le 7$: MSB = SPI_TX[7:0]
- $8 \le NB \le 15$: MSB = SPI_TX[15:8]
- $16 \le NB \le 23$: MSB = SPI_TX[23:16]
- $24 \le NB \le 31$: MSB = SPI_TX[31:24]

The MSSPI.SPI_TX register is a 32-bit-wide register that is 8-bit-, 16-bit-, or 32-bit-addressable. Partial register updates with successive 8-bit or 16-bit accesses can be used to load the transmit register. The least-significant bit (LSB) part must be updated before the MSB part of the transmitted word.

26.2.2.2.2 Receive Register

The number of bits in the word to be received is programmed through the MSSPI.SPI_CTRL[6:2] NB bit. The receive register (SPI_RX) data reads must be completed according to the received word bit length and in the correct register access sequence.

The SPI_RX register is considered empty when the MSB part of the received word is read. If the number of bits of the received word is not aligned on a byte boundary, the unused bits are read as an undefined value.

The SPI_RX register is a 32-bit-wide register that is 8-bit-, 16-bit-, or 32-bit-addressable. Partial register reads with successive 8-bit or 16-bit accesses can be used to read the receive register. The LSB part must be read before the MSB part of the received word.

26.2.2.3 Master Transmit-Only Mode

The master transmit-only mode prevents the local host from reading the receive register (minimizing data movement) when only transmission is meaningful.

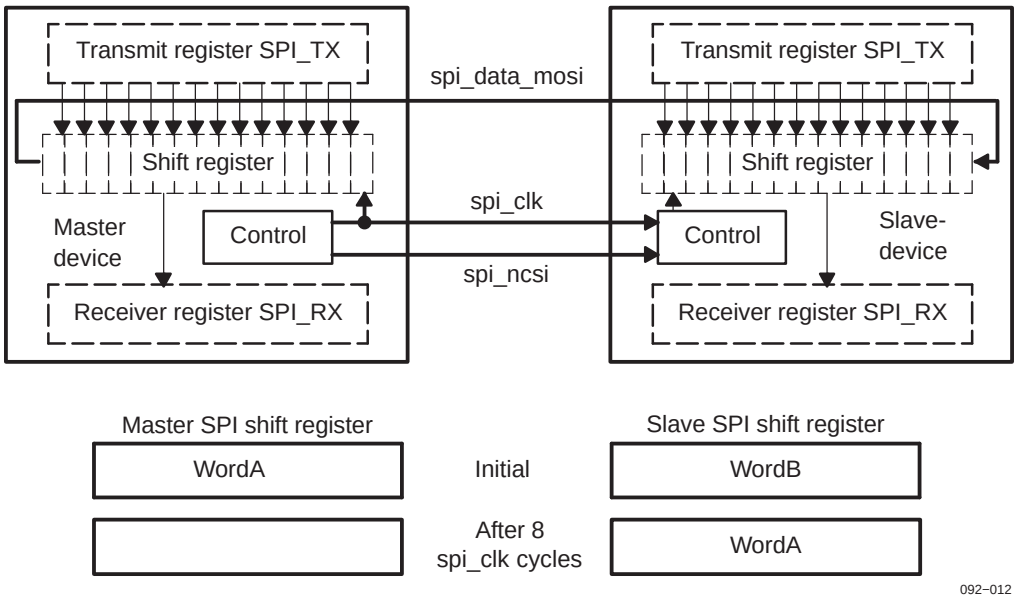
In transmit-only mode, the transmission starts when the MSSPI.SPI_CTRL[1] WR bit is set. This bit is automatically reset by the hardware after one spi_clk cycle.

In master transmit-only mode, a full state of the receive register does not prevent transmission, and the receive register is always overwritten with the new word. This event is not significant when only transmission is meaningful. Therefore, the MSSPI.SPI_ISR[2] RX_OVERFLOW bit is never set in this mode.

A write-end (WE) interrupt is generated exactly one spi_clk cycle plus two cycles of the interface clock (MSSPI_ICLK) after the last bit of the transaction is transmitted, if the MSSPI.SPI_IER[1] MSK1 bit is set.

Figure 26-12 shows an example of a full-duplex, 1-byte data exchange between a master device (on the left) and a slave device (on the right).

Figure 26-12. MSSPI Half-Duplex Transmission (Master Transmit-Only Example)



After eight cycles of the spi_clk serial clock, the WordA transfer from the master to the slave is complete.

26.2.3 Slave Mode

26.2.3.1 Slave Mode Features

The MSSPI is in slave mode when the MSSPI.SPI_SET2[15] MODE bit is reset.

The MSSPI can be connected to only one external master device (only one chip-select input signal spi_ncs0).

In slave mode, the MSSPI initiates a data transfer on the data lines (spi_data_mosi and spi_data_miso) when it is selected by the active control signal (spi_ncs0) and receives the serial clock spi_clk from the external master device.

The end of a transaction is detected based on the clock-enable field, MSSPI.SPI_SET2[7:5].

CAUTION

In slave mode, all clock-enable (CEi) bits must have the same value, depending on the active level of the master-enable signal. The clock-phase (CPi) and the clock-invert (CIi) bits must also have the same values, depending on the master clock configuration (polarity and phase).

There must also be a write access to the MSSPI.SPI_SET2 register during TX/RX to avoid losing information.

26.2.3.2 Slave Transmit and Receive Mode (Full-Duplex)

In slave transmit and receive mode, the transmitter register (SPI_TX) must be loaded before the MSSPI is selected by an external SPI master device. The read and write processes must also be activated. This bit is reset after one cycle of the incoming spi_clk. The WR bit does not affect behavior; the WR bit is reset like the RD bit, if it was previously set.

After enabling chip-select 0 (spi_ncs0), transmission and reception proceed with interrupt and DMA request events.

On completion of a serial word transfer, the received data is transferred to the receive register.

An RE interrupt is generated exactly one spi_clk cycle plus two cycles of the interface clock (MSSPI_ICLK) after the last bit of the transaction is received, if the MSSPI.SPI_IER[0] MSK0 bit is set. Then the receive data can be read in receive register SPI_RX.

26.2.3.3 Slave Transmit-Only Mode

Slave transmit-only mode prevents the local host from reading the receiver register, thus minimizing data movement when only the transmission is meaningful.

The slave transmit-only mode is programmable when the MSSPI.SPI_CTRL[1] WR bit is set.

A WE interrupt is generated exactly one spi_clk cycle plus two cycles of the interface clock (MSSPI_CYCLE) after the last bit of the transaction is transmitted, if the MSSPI.SPI_IER[1] MSK1 bit is set.

26.2.4 Overflow/Underflow Interrupt

In slave mode, whether the functional mode is MPU or DMA, the receive register (MSSPI.SPI_RX) can overflow and/or the transmit register (MSSPI.SPI_TX) can underflow. In either case, an interrupt is generated (if it is unmasked) and sent to the host. The host must take the correct action according to the received interrupt.

26.2.4.1 Overflow Interrupt Generation

To generate an overflow interrupt, the MSSPI must be in the following state:

- The SPI is configured in slave mode: MSSPI.SPI_SET2[15] = 0.
- The enable for overflow interrupt is active: MSSPI.SPI_IER[2] = 1.
- The receive register (MSSPI.SPI_RX) has not been read between two receptions.

  To release the interrupt (MSSPI_IT) activated by the RX_OVERFLOW bit, the RX_OVERFLOW status bit must be cleared by writing 1 to the SPI_ISR[2] bit.

26.2.4.2 Underflow Interrupt Generation

To generate an underflow interrupt, the SPI must be in the following state:

- The SPI is configured in slave mode: MSSPI.SPI_SET2[15] = 0.
- The enable for underflow interrupt is active: MSSPI.SPI_IER[3] = 1.
- The transmit register (SPI_TX) has not been updated between two transmissions.

To release the interrupt (MSSPI_IT) activated by the TX_UNDERFLOW bit, the TX_UNDERFLOW status bit must be cleared by writing 1 to the MSSPI.SPI_ISR[3] bit.

26.2.5 Test Mode

Activate test mode by setting the MSSPI.SPI_TEST[0] TMODE bit. This enables the following features:

- The data-out pin is fed back to the data-in pin.
- The data-out, data-in, and MSSPI_FCLK pins can be controlled and observed.

When test mode is not active, reading the MSSPI.SPI_TEST register always returns 0.

When test mode is activated, it is possible to:

- Force the data-out pin to read the value set in the MSSPI.SPI_TEST[2] WTV bit. This allows control of the data-out pin.
- Read the data-in pin (assuming that the input signal is a static signal).
- Force the clock output signal spi_clk (in master mode only). This provides control of the clock-output pin.
- Read the value of the clock-input signal spi_clk (in slave mode only).
- Read the values of the spi_ncsi pins.

For details about test mode, see Table 26-21.

26.3 MSSPI Programming Model

26.3.1 Reset

Before accessing or using the MSSPI, the local host must ensure that the internal reset is released:

1. The MPU sets the MSSPI.SPI_SCR[1] SOFTRESET bit. This bit is automatically reset by hardware. A read always returns 0.
2. The internal reset is completed when the MSSPI.SPI_SSR[0] RESETDONE bit takes the value 0x1.

26.3.2 Operation Mode

This section describes the MPU and DMA protocols. Select one of these protocols by programming the MSSPI.SPI_SET1[5] DMA_EN bit. For correct behavior, the following sequence must occur:

In the MPU protocol, the interrupt request must be enabled. Thus, the MSSPI can issue an interrupt to inform the host that the transaction is complete.

26.3.2.1 MPU Transmit Protocol in Master Mode

1. The initiator writes to the setup registers (MSSPI.SPI_SET1 and MSSPI.SPI_SET2).
2. The initiator configures the word length and the index of the addressed device in the control register (MSSPI.SPI_CTRL[6:2] and MSSPI.SPI_CTRL[9:7]).
3. The initiator writes the data to transmit in the transmit register (SPI_TX).
4. The initiator sets the MSSPI.SPI_CTRL[1] WR bit.
   When the WR bit is set:
   - The transmit register (SPI_TX) is copied to the shift register (MSSPI.SPI_SR).
   - The device-enable (spi_ncsi) goes low if CEi = 0 in MSSPI.SPI_SET2, or high if CEi = 1.
   - The serial clock (spi_clk) is activated and transmission begins.
   - The WR bit is reset one spi_clk cycle later.

   When transmission is complete:
   - The device-enable (spi_ncsi) goes high if CEi = 0 in MSSPI.SPI_SET2, or low if CEi = 1.
   - The WE bit is set in the interrupt status register (MSSPI.SPI_ISR).
   - A WE interrupt is generated after one spi_clk cycle plus two cycles of the MSSPI_ICLK clock, if MSK1 is set in the interrupt-enable register (SPI_IER).
5. The interrupt request is released when the initiator clears the SPI_ISR[1] WE bit.

26.3.2.2 MPU Receive/Transmit Protocol in Master Mode

1. The initiator writes to the setup registers (MSSPI.SPI_SET1 and MSSPI.SPI_SET2).
2. The initiator configures the word length and the index of the addressed device in the control register (MSSPI.SPI_CTRL[9:7] and MSSPI.SPI_CTRL[9:7]).
3. The initiator writes to the transmit register (SPI_TX) (optional). This is necessary only when transmission is performed simultaneously with reception.
4. The initiator sets the MSSPI.SPI_CTRL[0] RD bit.
   When the RD bit is set:
   - The transmit register (SPI_TX) is copied to the shift register (MSSPI.SPI_SR).
   - The device-enable (spi_ncsi) goes low if CEi = 0 in MSSPI.SPI_SET2, or high if CEi = 1.
   - The serial clock (spi_clk) is activated and transmission and reception begin.
   - The RD bit is reset one spi_clk cycle later. The WR bit does not affect behavior; the WR bit is reset like the RD bit, if it was previously set.

   When reception is complete:
   - The device-enable (spi_ncsi) goes high if CEi = 0 in MSSPI.SPI_SET2, or low if CEi = 1.
   - The RE bit is set in the interrupt status register (MSSPI.SPI_ISR).
   - The shift register (SPI_SR) is copied to the receive register (SPI_RX).
   - An RE interrupt is generated after one spi_clk cycle plus two cycles of the MSSPI_ICLK clock if

MSK0 is set in the interrupt-enable register (SPI_IER).

5. The interrupt request is released when the initiator reads the receive register (SPI_RX) and clears the SPI_ISR[0] RE bit.

26.3.2.3 MPU Transmit Protocol in Slave Mode

1. The initiator writes to the setup registers (MSSPI.SPI_SET1 and MSSPI.SPI_SET2).
2. The initiator configures the word length in the control register (SPI_CTRL[6:2]).
3. The initiator writes to the transmit register (MSSPI.SPI_TX).
4. The initiator sets the MSSPI.SPI_CTRL[1] WR bit.
   When the WR bit is set:
   - The transmit register (SPI_TX) is copied to the shift register (MSSPI.SPI_SR).
   - If CEi = 0 in MSSPI.SPI_SET2, transmission begins when the slave-device-enable (spi_ncs0) goes low, or high if CEi = 1, and the serial clock (spi_clk) is activated.
   - The WR bit is reset one spi_clk cycle later.

   When transmission is complete:
   - The device-enable (spi_ncs0) goes high if CEi = 0 in MSSPI.SPI_SET2, or low if CEi = 1.
   - The MSSPI.SPI_ISR[1] WE bit is set.
   - A WE interrupt is generated after one cycle of spi_clk plus two cycles of the MSSPI_ICLK clock, if MSK1 is set in the interrupt-enable register (MSSPI.SPI_IER).
5. The interrupt request is released when the initiator clears the MSSPI.SPI_ISR[1] WE bit.

26.3.2.4 MPU Receive/Transmit Protocol in Slave Mode

1. The initiator writes to the setup registers (MSSPI.SPI_SET1 and MSSPI.SPI_SET2).
2. The initiator configures the word length in the control register (MSSPI.SPI_CTRL[6:2]).
3. The initiator writes to the transmit register (SPI_TX) (optional). This is necessary only when transmission is performed simultaneously with reception.
4. The initiator sets the MSSPI.SPI_CTRL[0] RD bit.
   When the RD bit is set:
   - The transmit register (SPI_TX) is copied to the shift register (MSSPI.SPI_SR).
   - If CEi = 0 in SPI_SET2, transmission and reception begin when the slave-device-enable (spi_ncs0) goes low, or high if CEi = 1, and the serial clock (spi_clk) is activated.
   - The RD bit is reset one spi_clk cycle later. The WR bit does not affect behavior; the WR bit is reset like the RD bit, if it was previously set.

   When reception is complete:
   - The device-enable (spi_ncs0) goes high if CEi = 0 in SPI_SET2, or low if CEi = 1.
   - The MSSPI.SPI_ISR[0] RE bit is set.
   - The shift register is copied to the receive register (SPI_RX).
   - An RE interrupt is generated after one spi_clk cycle plus two cycles of the MSSPI_ICLK clock, if MSK0 is set in the interrupt-enable register (MSSPI.SPI_IER).
5. The interrupt request is released when the MPU reads the receive register (SPI_RX) and clears the MSSPI.SPI_ISR[0] RE bit.

26.3.2.5 DMA Protocol

The DMA_EN bit field of the MSSPI.SPI_SET1 register must be set (MSSPI.SPI_SET1[5] = 0x1) to enable DMA transfer.

26.3.2.6 DMA Transmit Protocol in Master Mode

1. The MPU writes to the setup registers (MSSPI.SPI_SET1 and MSSPI.SPI_SET2).
2. The MPU configures the word length and the index of the addressed device in the control register (MSSPI.SPI_CTRL[6:2] and MSSPI.SPI_CTRL[9:7]).

3. The MPU sets the MSSPI.SPI_CTRL[1] WR bit.
   When DMA_EN is set, a transmit DMA request is generated when the WR bit is set.
4. The DMA writes the data to the transmit register (SPI_TX). When the data is written:
   - The transmit DMA request is cleared.
   - The TX_EMPTY status bit is reset in the MSSPI.SPI_DSR register.
   - The transmit register (SPI_TX) is copied to the shift register (MSSPI.SPI_SR).
   - The device-enable (spi_ncsi) goes low if CEi = 0 in SPI_SET2, or high if CEi = 1.
   - The serial clock (spi_clk) is activated and transmission begins.
5. When transmission is complete:
   - The device-enable (spi_ncsi) goes high if CEi = 0 in SPI_SET2, or low if CEi = 1.
   - The TX_EMPTY status bit is set in the data-status register (MSSPI.SPI_DSR).
   - A transmit DMA request is generated.
   - Another transmission can begin (Step 4 → Step 5 → Step 4 → Step 5 →…).

To stop the process, the MPU must reset the MSSPI.SPI_CTRL[1] WR bit.

26.3.2.7 DMA Receive Protocol in Master Mode

1. The MPU writes to the setup registers (MSSPI.SPI_SET1 and MSSPI.SPI_SET2).
2. The MPU configures the word length and the index of the addressed device in the control register (MSSPI.SPI_CTRL[6:2] and MSSPI.SPI_CTRL[9:7]).
3. The MPU sets the MSSPI.SPI_CTRL[0] RD bit.
   With DMA_EN is set, when the RD bit is set:
   - The device-enable (spi_ncsi) goes low if CEi = 0 in MSSPI.SPI_SET2, or high if CEi = 1.
   - The serial clock (spi_clk) is activated and reception begins.
4. When reception is complete:
   - The device-enable (spi_ncsi) goes high if CEi = 0 in MSSPI.SPI_SET2, or low if CEi = 1.
   - The RX_FULL status bit is set in the data-status register (MSSPI.SPI_DSR).
   - The shift register (SPI_SR) is copied to the receive register (SPI_RX).
   - A receive DMA request is generated.
5. When the DMA reads the receive register (SPI_RX):
   - The device-enable (spi_ncsi) goes low if CEi = 0 in MSSPI.SPI_SET2, or high if CEi = 1.
   - The RX_FULL status bit is reset in the data-status register (SPI_DSR).
   - The receive DMA request is released.
   - Another reception can begin (Step 4 → Step 5 → Step 4 → Step 5 →…).

To stop the process, the MPU must reset the RD bit in the control register (MSSPI.SPI_CTRL[0]).

26.3.2.8 DMA Transmit and Receive Protocol in Master Mode

1. The MPU writes to the setup registers (MSSPI.SPI_SET1 and MSSPI.SPI_SET2).
2. The MPU configures the word length and the index of the addressed device in the control register (MSSPI.SPI_CTRL[6:2] and MSSPI.SPI_CTRL[9:7]).
3. The MPU sets the RD and WR bits of the control register (MSSPI.SPI_CTRL[0] and MSSPI.SPI_CTRL[1]).
   When the DMA_EN bit is set, and the RD and WR bits are set, a transmit DMA request is generated.
4. The DMA writes the data in the transmit register (SPI_TX). When the data is written:
   - The transmit DMA request is released.
   - The TX_EMPTY status bit is reset in the data-status register (MSSPI.SPI_DSR).
   - The transmit register (SPI_TX) is copied to the shift register (MSSPI.SPI_SR).
   - The device-enable (spi_ncsi) goes low if CEi = 0 in MSSPI.SPI_SET2, or high if CEi = 1.
   - The serial clock (spi_clk) is activated and transmission and reception begin.
5. When transmission and reception are complete:
   - The device-enable (spi_ncsi) goes high if CEi = 0 in MSSPI.SPI_SET2, or low if CEi = 1.
   - The shift register (SPI_SR) is copied to the receive register (SPI_RX).
   - RX_FULL and TX_EMPTY are set in the data-status register (MSSPI.SPI_DSR).
   - A receive DMA request is generated.

6. When the DMA reads the receive register (SPI_RX):
   - The RX_FULL status bit is reset in the data-status register (SPI_DSR).
   - The receive DMA request is released.
   - A transmit DMA request is generated.
   - Another transmission and reception can begin (Step 4 → Step 5 → Step 6 → Step 4 → Step 5 → Step 6 →…).

To stop the process, the MPU must reset the MSSPI.SPI_CTRL[0] RD bit and the SPI_CTRL[1] WR bit.

26.3.2.9 DMA Transmit Protocol in Slave Mode

1. The MPU writes to the setup registers (MSSPI.SPI_SET1 and MSSPI.SPI_SET2).
2. The MPU configures the word length in the control register (MSSPI.SPI_CTRL[6:2]).
3. The MPU sets the MSSPI.SPI_CTRL[1] WR bit.
   When the DMA_EN bit is set, and the WR bit is set, a transmit DMA request is generated.
4. The DMA writes the data in the transmit register (SPI_TX). When the data is written:
   - The transmit DMA request is released.
   - The TX_EMPTY status bit is reset in the data-status register (SPI_DSR).
   - The transmit register (SPI_TX) is copied to the shift register (SPI_SR).
   - If CEi = 0 in SPI_SET2, the transmission starts when the slave device-enable (nSPEN0) goes low, or high if CEi = 1, and the serial clock (spi_clk) is activated.
5. When transmission is complete:
   - The TX_EMPTY status bit is set in the data-status register (SPI_DSR).
   - A transmit DMA request is generated.
   - Another transmission can begin (Step 4 → Step 5 → Step 4 → Step 5→…).

To stop the process, the MPU must reset the MSSPI.SPI_CTRL[1] WR bit.

26.3.2.10 DMA Receive Protocol in Slave Mode

1. The MPU writes to the setup registers (MSSPI.SPI_SET1 and MSSPI.SPI_SET2).
2. The MPU configures the word length in the control register (MSSPI.SPI_CTRL[6:2]).
3. The MPU sets the MSSPI.SPI_CTRL[0] RD bit.
   - If CEi = 0 in MSSPI.SPI_SET2 and the RD bit is set, the reception starts when the slave device-enable (spi_ncs0) goes low, or high if CEi = 1, and the serial clock (spi_clk) is activated.
4. When reception is complete:
   - The device-enable (spi_ncs0) goes high if CEi = 0 in MSSPI.SPI_SET2, or low if CEi = 1.
   - The RX_FULL bit is set in the data-status register (MSSPI.SPI_DSR).
   - The shift register (MSSPI.SPI_SR) is copied to the receive register (SPI_RX).
   - A receive DMA request is generated.
5. When the DMA reads the receive register (SPI_RX):
   - The RX_FULL status bit is reset in the data-status register (SPI_DSR).
   - The receive DMA request is released.
   - Another reception can begin (Step 4 → Step 5 → Step 4 → Step 5 →…).

To stop the process, the MPU must reset the MSSPI.SPI_CTRL[0] RD bit.

26.3.2.11 DMA Transmit and Receive Protocol in Slave Mode

1. The MPU writes to the setup registers (MSSPI.SPI_SET1 and MSSPI.SPI_SET2).
2. The MPU configures the word length in the control register (MSSPI.SPI_CTRL[6:2]).
3. The MPU sets the MSSPI.SPI_CTRL[0] RD bit and the MSSPI.SPI_CTRL[1] WR bit.
   When the DMA_EN bit is set, and the RD and the WR bits are set, a transmit DMA request is generated.
4. The DMA writes the data in the transmit register (SPI_TX). When the data is written:
   - The transmit DMA request is released.
   - The TX_EMPTY status bit is reset in the data-status register (SPI_DSR).

- The transmit register (SPI_TX) is copied to the shift register (MSSPI.SPI_SR).
- If CEi = 0 in MSSPI.SPI_SET2, the transmission and reception start when the slave device-enable (spi_ncs0) goes low, or high if CEi = 1, and the serial clock (spi_clk) is activated.

5. When transmission and reception are complete:
   - The shift register (MSSPI.SPI_SR) is copied to the receive register (SPI_RX).
   - The RX_FULL and TX_EMPTY bits are set in the data-status register (SPI_DSR).
   - A receive DMA request is generated.
6. When the DMA reads the receive register (SPI_RX):
   - The RX_FULL status bit is reset in the data-status register (MSSPI.SPI_DSR).
   - The receive DMA request is released.
   - A transmit DMA request is generated.
   - Another transmission and reception can begin (Step 4 → Step 5 → Step 6 → Step 4 → Step 5 → Step 6 →…).

To stop the process, the MPU must reset the MSSPI.SPI_CTRL[0] RD bit and the MSSPI.SPI_CTRL[1] WR bit.

26.4 MSSPI Register Manual

26.4.1 Instance Summary

Table 26-8 shows the base address and address space for the MSSPI module.

Table 26-8. Instance Summary

Module Name	MPU Base Address	Size
MSSPI	0x09E0 0000	1M byte

26.4.2 MSSPI Register Mapping Summary

Table 26-9 lists the MSSPI registers.

Table 26-9. MSSPI Registers Summary

Register Name	Type	Register Width (Bits)	Offset
SPI_REV	R	32	0x00
SPI_SCR	RW	32	0x10
SPI_SSR	R	32	0x14
SPI_ISR	RW	32	0x18
SPI_IER	RW	32	0x1C
SPI_SET1	RW	32	0x24
SPI_SET2	RW	32	0x28
SPI_CTRL	RW	32	0x2C
SPI_DSR	R	32	0x30
SPI_TX	RW	32	0x34
SPI_RX	R	32	0x38
SPI_TEST	RW	32	0x3C

26.4.3 Register Descriptions

Table 26-10 through Table 26-21 describe the individual bit fields of the MSSPI registers.

Table 26-10. Identification Register Bit Description (SPI_REV)

Address Offset	0x00		
Physical address	0x09E0 0000	**Instance**	MSSPI
Description	This register contains the hard-coded SPI RTL revision number.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REV							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	REV	IP revision The four LSBs indicate a minor revision. The four MSBs indicate a major revision. Example: 0x10 for 1.0, 0x21 for 2.1	R	See (1)

(1) TI internal data

Table 26-11. System Configuration Register Bit Description (SPI_SCR)

Address Offset	0x10		
Physical address	0x09E0 0010	**Instance**	MSSPI
Description	This register allows controlling various parameters of the OCP interface.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														SOFTRESET	AUTOIDLE

Bits	Field Name	Description		Type	Reset
31:2	Reserved	Read returns 1.		R	0x0000001
1	SOFTRESET	Software reset		R/W	0x0
		0x0:	Normal mode		
		0x1:	The module is reset. Set this bit to 1 to trigger a module reset. This bit is automatically reset by the hardware. Read returns 0.		
0	AUTOIDLE	Internal OCP clock-gating strategy		R/W	0x0
		0x0:	OCP clock is free-running.		
		0x1:	Automatic OCP clock-gating strategy is applied based on OCP interface activity. It takes one more OCP clock to access a register.		

Table 26-12. System Status Register Bit Description (SPI_SSR)

Address Offset	0x14		
Physical address	0x09E0 0014	**Instance**	MSSPI
Description	This register provides status information about the reset.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															RESETDONE

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Read returns 0.		0x00000000
0	RESETDONE	Internal reset monitoring	R	0x0
		0x0: Internal module reset is ongoing.		
		0x1: Reset complete		

Note: Before accessing or using the MSSPI module, the local host must ensure that internal reset is released by reading this register.

Table 26-13. Interrupt Status Register Bit Description (SPI_ISR)

Address Offset	0x18		
Physical address	0x09E0 0018	**Instance**	MSSPI
Description	This register is used to qualify the interrupt.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											WAKEUP	TX_UNDERFLOW	RX_OVERFLOW	WE	RE

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Read returns 0.		0x00000000
4	WAKEUP	Wakeup—active high	R/W	0x0
3	TX_UNDERFLOW	Transmit underflow—active high	R/W	0x0
2	RX_OVERFLOW	Receive overflow—active high	R/W	0x0
1	WE	Write-end—active high	R/W	0x0
		The serialization is done.		
0	RE	Read-end—active high	R/W	0x0
		Receive register is updated.		

Note: To release the interrupt, users must write 1 to the corresponding status bit. Writing 0 has no effect.

Table 26-14. Interrupt-Enable Register Bit Description (SPI_IER)

Address Offset	0x1C		
Physical address	0x09E0 001C	**Instance**	MSSPI
Description	This register allows enabling/disabling of module internal sources of interrupt on an event-by-event basis.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											MSK4	MSK3	MSK2	MSK1	MSK0

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Read returns 0.		0x00000000
4	MSK4	Enable interrupt on wakeup	R/W	0x0
		0x0: Interrupt disabled		
		0x1: Interrupt enabled		
3	MSK3	Enable interrupt on TX underflow	R/W	0x0
		0x0: Interrupt disabled		
		0x1: Interrupt enabled		
2	MSK2	Enable interrupt on RX overflow	R/W	0x0
		0x0: Interrupt disabled		
		0x1: Interrupt enabled		
1	MSK1	Enable interrupt for write cycle	R/W	0x0
		0x0: Interrupt disabled		
		0x1: Interrupt enabled		
0	MSK0	Enable interrupt for read/write cycle	R/W	0x0
		0x0: Interrupt disabled		
		0x1: Interrupt enabled		

Table 26-15. Set up SPI 1 Register Bit Description (SPI_SET1)

Address Offset	0x24		
Physical address	0x09E0 0024	**Instance**	MSSPI
Description	This register is dedicated to the configuration of the serial port interface.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										DMA_EN	PTV				EN_CLK

Bits	Field Name	Description	Type	Reset
31:6	Reserved	Read returns 0.	R	0x0000000
5	DMA_EN	Defines the protocol	R/W	0x0
		0x0: MPU protocol		
		0x1: DMA protocol		
4:1	PTV	The prescale time value (2^{PTV}) is used to generate the internal shift register clock in master mode.	R/W	0x0
		The scale values supported are:		
		0x0: 1 (Not allowed when MSSPI_FCLK = 48 or 52 MHz)		
		0x1: 2		
		0x2: 4		
		0x3: 8		
		0x4: 16		
		0x5: 32		
		0x6: 64		
		0x7: 128		
		0x8: 256		

Bits	Field Name	Description	Type	Reset
		0x9: 512		
		0xA: 1024		
		0xB: 2048		
		Others: 4096		
0	EN_CLK	SPI functional clock enable	R/W	0x0
		0x0: The clock is shut off.		
		0x1: The clock is running.		

Note: A write access to this register must not be performed during a transaction.

Table 26-16. Set up SPI 2 Register Bit Description (SPI_SET2)

Address Offset	0x28		
Physical address	0x09E0 0028	**Instance**	MSSPI
Description	This register is dedicated to the configuration of the serial port interface.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MODE	Reserved		CP			Reserved		CE			Reserved		CI		

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0.	R	0x0000
15	MODE	0x0: Slave mode	R/W	0x0
		0x1: Master mode		
14:13	Reserved	Reserved for future compatibility	R/W	0x0
12:10	CP	Clock phase	R/W	0x00
		The shift register clock begins toggling at:		
		0x0: The middle of the data transfer		
		0x1: The beginning of the data transfer		
		CP[0] qualifies the access on device 0.		
		CP[1] qualifies the access on device 1.		
		CP[2] qualifies the access on device 2.		
9:8	Reserved	Reserved for future compatibility	R/W	0x0
7:5	CE	Clock enable	R/W	0x00
		Active level of the shift register enable		
		0x0: The active level is low.		
		0x1: The active level is high.		
		CE[0] qualifies the access on device 0.		
		CD[1] qualifies the access on device 1.		
		CE[2] qualifies the access on device 2.		
4:3	Reserved	Reserved for future compatibility	R/W	0x0
2:0	CI	Clock invert	R/W	0x00
		Inactive edge of the shift register clock		
		0x0: The inactive state of the clock is low.		
		0x1: The inactive edge of the clock is high.		
		CI[0] qualifies the access on device 0.		
		CI[1] qualifies the access on device 1.		

Bits	Field Name	Description	Type	Reset
		CI[2] qualifies the access on device 2.		

Table 26-17. Control Register Bit Description (SPI_CTRL)

Address Offset	0x2C		
Physical address	0x09E0 002C	**Instance**	MSSPI
Description	This register is dedicated to the configuration of the serial port interface.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								AD			NB			WR	RD

Bits	Field Name	Description	Type	Reset
31:9	Reserved	Read returns 0.	R	0x000000
8:7	AD	Index of the addressed device	R/W	0x0
		0x0: Enable device 0		
		0x1: Enable device 1		
		0x2: Enable device 2		
		Others: Undefined		
6:2	NB	Number of bits to receive/transmit	R/W	0x00
		0x0: 1 bit to receive/transmit		
		0x1: …		
		0x1F: 32 bits to receive/transmit		
1	WR	Write process activation—active high	R/W	0
0	RD	Read and write process activation—active high	R/W	0

Table 26-18. Data-Status Register Bit Description (SPI_DSR)

Address Offset	0x30		
Physical address	0x09E0 0030	**Instance**	MSSPI
Description	This register provides status information about the transmit and receive registers.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														TX_EMPTY	RX_FULL

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Read returns 0.	RW	0x00000000
1	TX_EMPTY	Shift register is empty—active high. This bit is cleared when the transmit register (SPI_TX) has been written.	R	0
0	RX_FULL	Receive register is full—active high. This bit is cleared when the receive register (SPI_RX) has been read.	R	0

Table 26-19. Transmit Register Bit Description (SPI_TX)

Address Offset	0x34		
Physical address	0x09E0 0034	**Instance**	MSSPI
Description	This register contains a single SPI word to transmit on the serial link, whatever the length of the SPI word.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI_TX																															

Bits	Field Name	Description	Type	Reset
31:0	SPI_TX	Data to transmit	R/W	0x00000000

Notes:

1. The bits to transmit must be aligned on the LSB of the SPI_TX register.
2. The number of bits of the word to be transmitted is programmed through the NB bit field in the SPI_CTRL register.
3. If the number of bits to transmit is 8, SPI_TX[7] is transmitted first, then SPI_TX[6], and so on.
4. If the number of bits to transmit is 32, SPI_TX[31] is transmitted first, then SPI_TX[30], and so on.

Table 26-20. Receive Register Bit Description (SPI_RX)

Address Offset	0x38		
Physical address	0x09E0 0038	**Instance**	MSSPI
Description	This register contains a single SPI word received through the serial link, whatever the length of the SPI word.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI_RX																															

Bits	Field Name	Description	Type	Reset
31:0	SPI_RX	Received data	R/W	0x00000000

Table 26-21. Test Register Bit Description (SPI_TEST)

Address Offset	0x3C		
Physical address	0x09E0 003C	**Instance**	MSSPI
Description	This register actives the test mode.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								RTSPEN		RCV	WCV	RTV	WTV	FDO	TMODE

Bits	Field Name	Description	Type	Reset
31:9	Reserved	Read returns 0.	R	0x000000
8:6	RTSPEN	Bits are directly connected to spi_ncsi outputs when test mode is enabled. RTSPEN[0] represents the value of spi_ncs0. RTSPEN[1] represents the value of spi_ncs1. RTSPEN[2] represents the value of spi_ncs2.	R	0x0
5	RCV	Bit is directly connected to the spi_clk input. It enables testing the connectivity of the spi_clk in feedback mode only.	R	0x0
4	WCV	Bit enables forcing the spi_clk output (in master mode only), providing control of the spi_clk output.	R/W	0x0

Bits	Field Name	Description	Type	Reset
3	RTV	Bit is directly connected to the data-in pin for test monitoring. It assumes the input signal is static.	R	0x0
2	WTV	Bit enables forcing the data-out pin value (for testing).	R/W	0x0
1	FDO	Control bit enables forcing the data-out pin to read the WTV bit field value. This provides control of the data-out pin.	R/W	0x0
0	TMODE	Test mode enable—active high	R/W	0x0

General Purpose Interface

This chapter describes the general-purpose interface of the LOCOSTO device.

Topic **Page**

27.1 General-Purpose Interface Overview

The general-purpose interface of the LOCOSTO device combines three general-purpose input/output (GPIO) banks: GPIO, GPIO1, and GPIO2.

Each GPIO module provides 16 general-purpose pins with input and output capabilities. Therefore, the general-purpose interface supports up to 48 (3 x 16) pins.

Each pin can be configured for the following applications:

- Data input/output
- Asynchronous interrupt generation upon the detection of external events
- Reference pin debouncing

Some outputs of a GPIO module are reserved for buzzer and light control. The GPIO module provides the following:

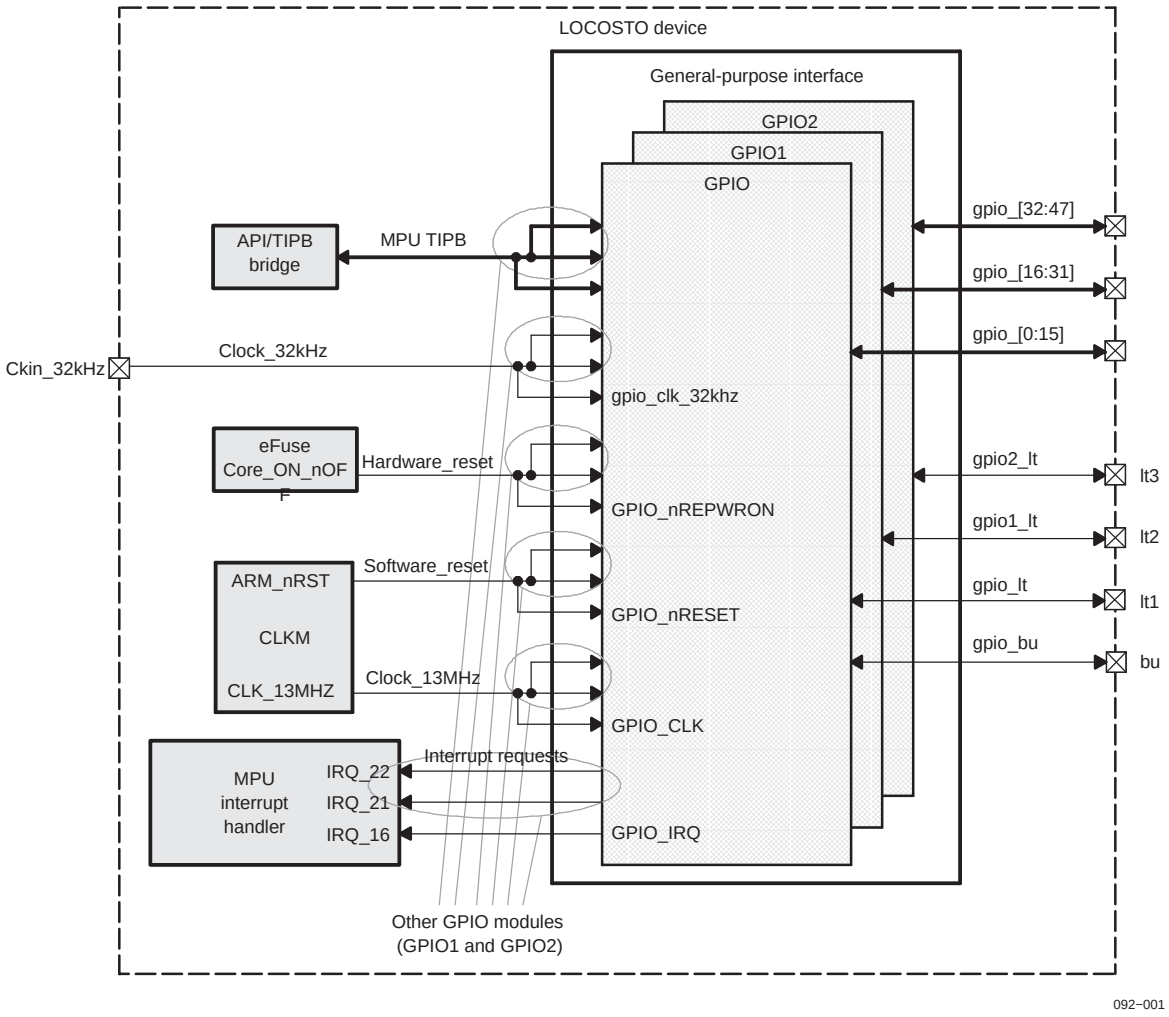
- An LT signal to control the power level of a backlight
- A BU signal to control the power level and tone of a buzzer

Note: Only one BU signal is available in the LOCOSTO device, whereas three LT signals can be used, depending on the pin multiplexing selection.

These modules do not include pad control (pullup/pulldown control, open drain feature), which is part of the device configuration. The pad configuration is software-controlled by the configuration registers in the configuration module. For further information, see Chapter 18, *Configuration*.

Figure 27-1 shows an overview of the general-purpose interface of the LOCOSTO device.

Figure 27-1. General_UnicodeEncodeError_Purpose Interface Overview



The output pins bu, lt1, lt2, and lt3 are accessible with the appropriate configuration mode of each pad. For further details about the correct mode to use, see Chapter 18, *Configuration*.

Each GPIO module includes the following global features:

- One asynchronous interrupt request processed by an interrupt generation on the detection of external events on each channel. This interrupt request is mapped on the microprocessor unit (MPU) interrupt handler.
- Data input (capture) and data output (drive)
- Two GPIO outputs dedicated to light and buzzer control. These outputs are controlled by pulse width modulation (PWM) to tune the power level, and are controlled by the internal timer to control the frequency level.
- Each GPIO can be chosen as the debouncing reference pin. The debouncing is applied to only one pin at a time.

Each general-purpose channel has the following features:

- The I/O control register (GPIOn.IO_CTNL_REG) controls the input and output capability for each pin.
- The output line level reflects the value written in the output data register (GPIOn.OUTPUT_REG) through the MPU TI peripheral bus (TIPB).
- The input line value is sampled into the input data register (GPIOn.INPUT_LATCH) and can be read through the MPU TIPB.
- The input line can be used through transition detectors to generated interrupts. The transition (high-to-low or low-to-high) can be configured in the interrupt level register

(GPIOn.INTERRUPT_LEVEL_REG).

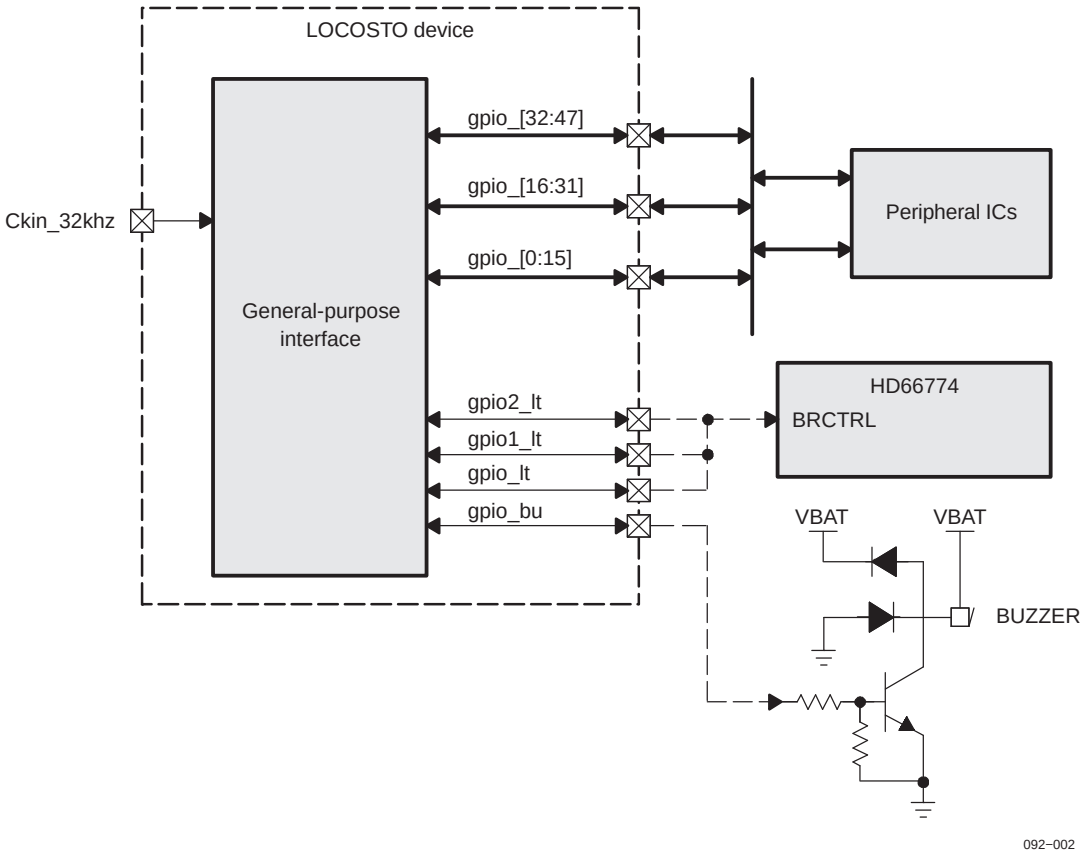
The module provides an alternative to the atomic test and set operation for the GPIOn.OUTPUT_REG register. For this register, the module implements the set and clear protocol register update (see Section 27.5.1, *Set and Clear Output Data Register*).

27.2 General-Purpose Interface Environment

The general-purpose interface combines three GPIO modules for a flexible, user-programmable, GPIO controller. The general-purpose interface implements functions that are not implemented with the dedicated controllers in the LOCOSTO device and that require simple input and/or output signals controlled by software. The general-purpose interface allows a variety of custom connections and expands the I/O capabilities of the system to external devices.

Figure 27-2 shows an overview of a typical application system using the general-purpose interface.

Figure 27-2. General-Purpose Interface Typical Application System Overview



Note: HD66774: A color TFT LCD driver from RENESAS

The general-purpose interface can physically connect the LOCOSTO device to peripheral integrated circuits (ICs). One specific use of the general-purpose interface is the control of the light and the buzzer.

Light and Buzzer Control

A PWM and registers are implemented within each GPIO module to control the power level and tone of the light and the buzzer.

These registers control the following:

- The light and buzzer enable
- The power level of the light and the buzzer
- The cycles of the buzzer

From the parameters contained in the control registers, each PWM provides these signals:

- A signal LT to control the power level of the light
- A signal BU to control the power level and tone of the buzzer

Note: Only the BU signal that comes from the GPIO instance is available in the LOCOSTO device; other BU signals are not linked. Depending on the pin multiplexing selection, the three LT signals can be used.

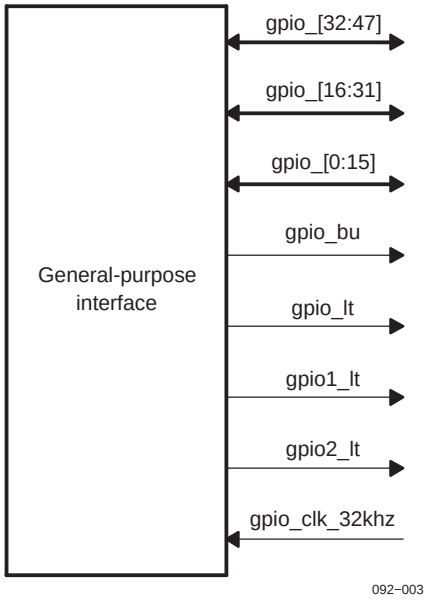
For further details, see Chapter 29, *Light and Buzzer Control*.

27.2.1 General-Purpose Interface Functional Interfaces

27.2.1.1 Basic General-Purpose Interface Pins

Figure 27-3 shows the functional interface signals in the general-purpose interface.

Figure 27-3. General-Purpose Interface Functional Interface Signals



27.2.1.2 General-Purpose Interface Functional Interface Description

Table 27-1 lists the I/Os of the general-purpose interface.

Table 27-1. I/O Description

Signal Name	I/O	Description	Reset Value
gpio_clk_32khz	I	External clock	Unknown
gpio_bu	O	Buzzer control from the PWM of the GPIO instance	0
gpio_lt	O	Light control from the PWM of the PWM of the GPIO instance	0
gpio1_lt	O	Light control from the PWM of the GPIO1 instance	0
gpio2_lt	O	Light control from the PWM of the GPIO2 instance	0
gpio_[47:0]	I/O	GPIO signals	Input until software configuration

Note: I = Input, O = Output

For further information on pin multiplexing, see Chapter 18, *Configuration*.

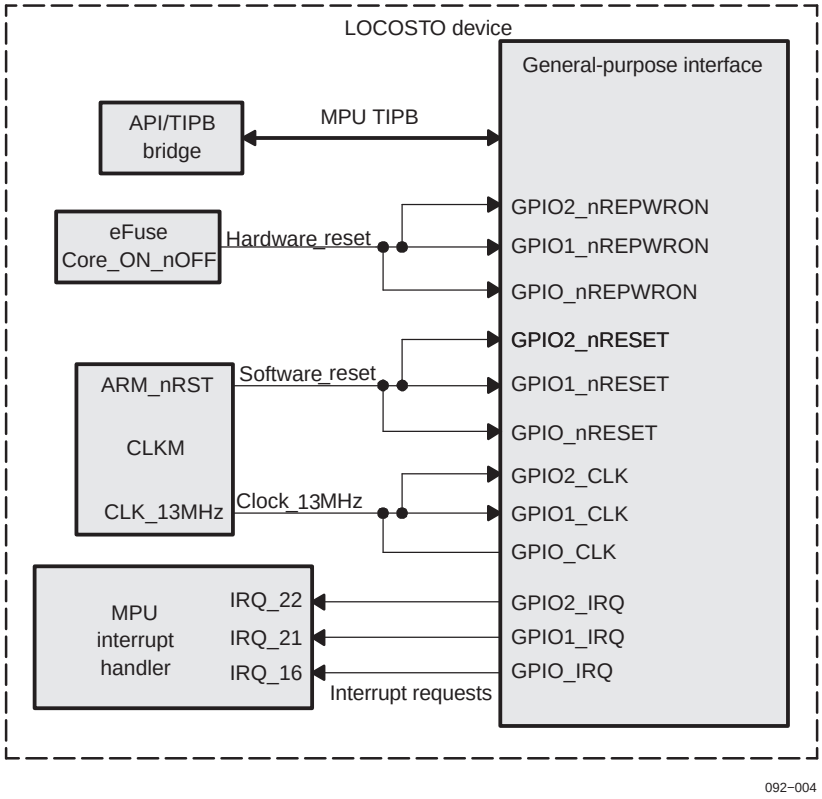
27.3 General-Purpose Interface Integration

27.3.1 Description

This section describes the integration of the GPIO modules within the LOCOSTO device.

Figure 27-4 shows the integration of the general-purpose interface, including the interrupt handler, interrupt requests, the clock manager (CLKM) module, resets, and interconnect.

Figure 27-4. General-Purpose Interface integration



27.3.2 Clocking, Reset, and Power-Management Scheme

27.3.2.1 Clocking

Each GPIO module uses the following clocks:

- Debouncing clock: This clock is used for the debounce cell logic. This cell samples the input line and uses a programmed delay to filter the input level. The debouncing clock can have two frequencies: 13 MHz and 32 kHz. The frequency is controlled by the CLOCK_DEBOUNCING bit of the debouncing mode register (GPIOn.JOGDIAL_MODE_REG). The 13-MHz clock comes from the CLKM module; the 32-kHz clock is an external clock that comes from the TWL3031 device.
- Functional clock: This is a 13-MHz clock. It comes from the CLKM module and is used throughout the GPIO module (except within the debounce cell logic when the 32-kHz clock has been selected).
- Interface clock: This is a 52-MHz clock. It is fed by the TIPB and is used to clock the data exchanges between the MPU TIPB and the internal logic (configuration registers).

The three clocks (32-kHz, 13-MHz, and 52-MHz) are provided by the general-purpose interface and are distributed within the module.

Table 27-2 lists the general-purpose interface clocks.

Table 27-2. Clock Description

Type	Name	Source	Frequency	Description
Debounce	gpio_clk_32khz	Clkin_32kHz	32 kHz	The 32-kHz clock is an external clock that comes from the TWL3031 device. It is distributed directly to the modules.
Functional/ debounce	GPIO_CLK GPIO1_CLK GPIO2_CLK	CLKM	13 MHz	The 13-MHz clock is generated by the digital radio processor (DRP) and is managed by the CLKM module.
Interface	GPIO_ICLK GPIO1_ICLK GPIO2_ICLK	TIPB	52 MHz	This clock is fed by the TIPB.

For further details, see Chapter 6, *Power, Reset, and Clock Management*.

27.3.2.2 Power Management

To reduce the dynamic power consumption, the software can program the CLKM module to selectively cut off clocks to various parts of the device. The LOCOSTO device offers three power-saving modes: running, big sleep, and deep sleep.

- Running and big sleep modes: All clocks are running. The GPIO module runs normally; interrupts can be generated on the configuration and external signals.
- Deep sleep mode: Only the Clk32K clock is running. The asynchronous interrupt generation is still active, but the rest of the module is off.

For further details, see Chapter 6, *Power, Reset, and Clock Management*.

27.3.2.3 Power Domain

The general-purpose interface belongs to the only power domain on the device, the core power domain.

For further details, see Chapter 6, *Power, Reset, and Clock Management*.

27.3.2.4 Reset

Hardware can reset the general-purpose interface. There is no dedicated software reset bit in the GPIO module registers.

Table 27-3 describes the hardware input reset.

Table 27-3. Reset Description

Type	Name	Source	Activation	Domain
Hardware	GPIO_nREPWRON	core_ON_nOFF (eFuse)	Active low	Core hardware reset that resets the whole module
Hardware	GPIO_nRESET	ARM_nRST(CLKM)	Active low	MPU hardware reset that resets the whole module

For further details, see Chapter 6, *Power, Reset, and Clock Management*.

27.3.3 Hardware Requests

27.3.3.1 DMA Requests

The three GPIO modules do not handle direct memory access (DMA) requests.

27.3.3.2 Interrupt Requests

All interrupt sources (the 16 input GPIO channels) are merged to generate one interrupt request in each GPIO module. Therefore, the general-purpose interface has three interrupt lines (one interrupt line per GPIO module instance).

The asynchronous interrupt request lines are active according to their respective interrupt mask registers (GPIOn.INTERRUPT_MASK_REG).

The three asynchronous interrupt lines are mapped on the MPU interrupt handler.

Table 27-4 lists the interrupt requests that are driven out from the general-purpose interface to the MPU interrupt handler.

Table 27-4. Interrupt Request Description

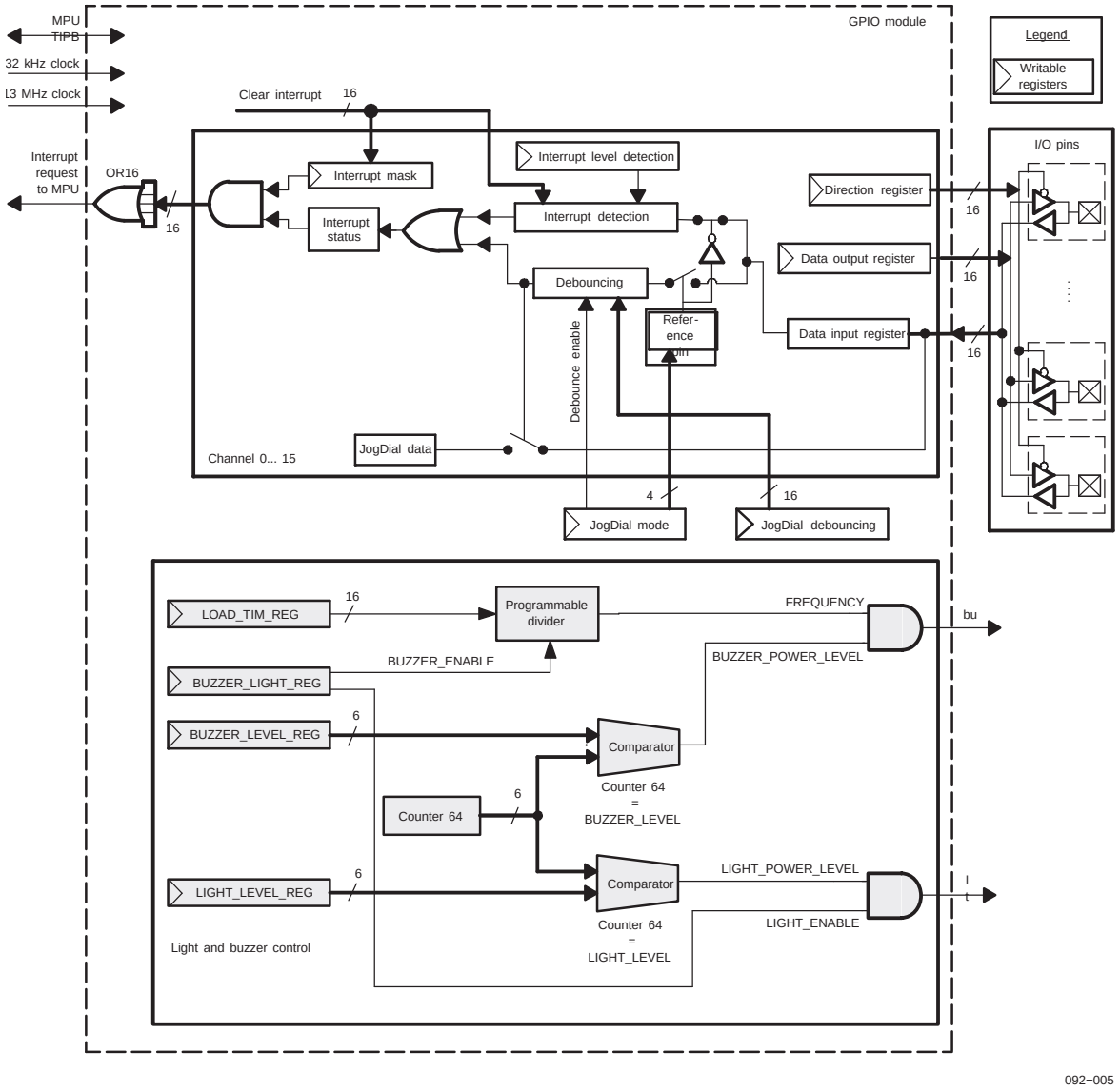
Type	Name	Source	Destination	Description
Interrupt request	GPIO_IRQ GPIO1_IRQ GPIO2_IRQ	Instance GPIO Instance GPIO1 Instance GPIO2	MPU interrupt handler	This interrupt request is sent asynchronously on the MPU interrupt handler.

27.4 General-Purpose Interface Functional Description

27.4.1 Block Diagram

Figure 27-5 shows the block diagram of the general-purpose interface, including the configuration registers and the main functional paths.

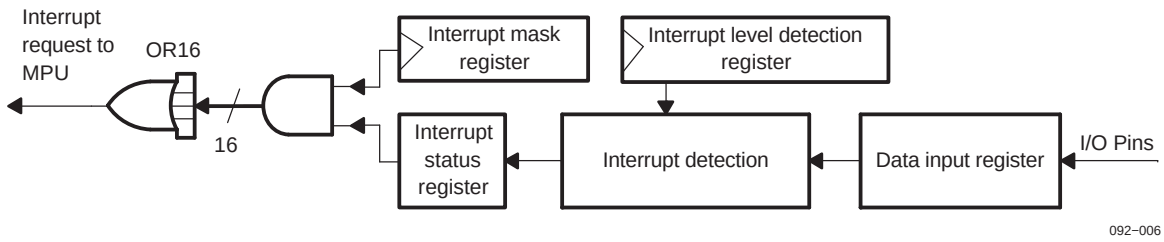
Figure 27-5. General-Purpose Interface Block Diagram



- The asynchronous path is used to generate an asynchronous interrupt request when a transition event is detected on any input GPIO. The transition (high-to-low or low-to-high) can be configured in the interrupt level register. The asynchronous interrupt request line to the MPU interrupt handler is active according to the interrupt mask register. In deep sleep mode, this path is always active to wake up the MPU. Figure 27-6 shows the asynchronous path.

Note: Interrupt requests can be detected only on edge, not on level.

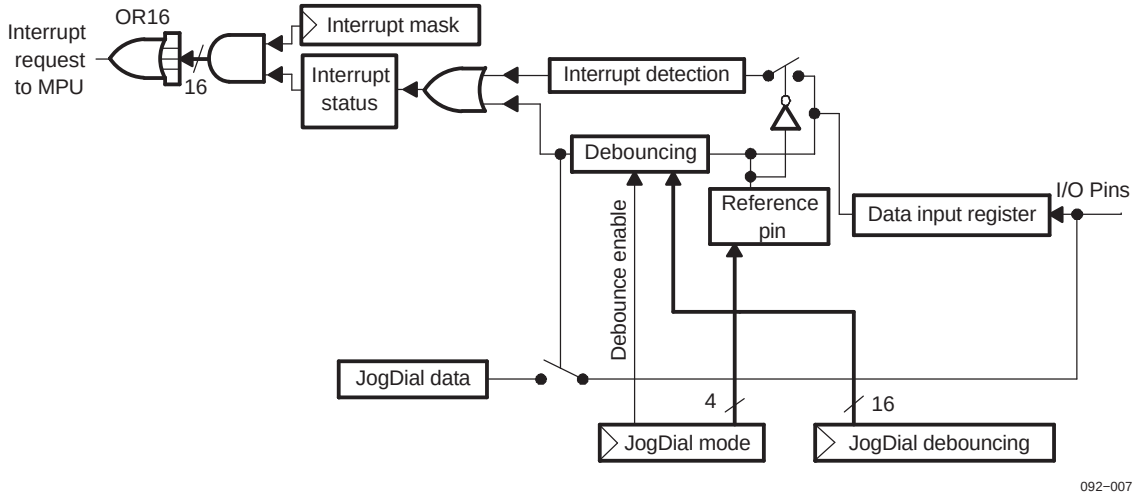
Figure 27-6. Asynchronous Path



- The debouncing path is used to apply a debounce filter at one GPIO input. When this path is active on a pin, the asynchronous path is enabled. The debounce path is synchronous with the debouncing clock (32 kHz or 13 MHz). The debouncing can be configured by programming the JogDial mode and the debouncing time registers. The data results of the debouncing are stored in the JogDial data register. This path is not active in deep sleep when the 13-MHz clock is selected. Figure 27-7 shows the debouncing path.

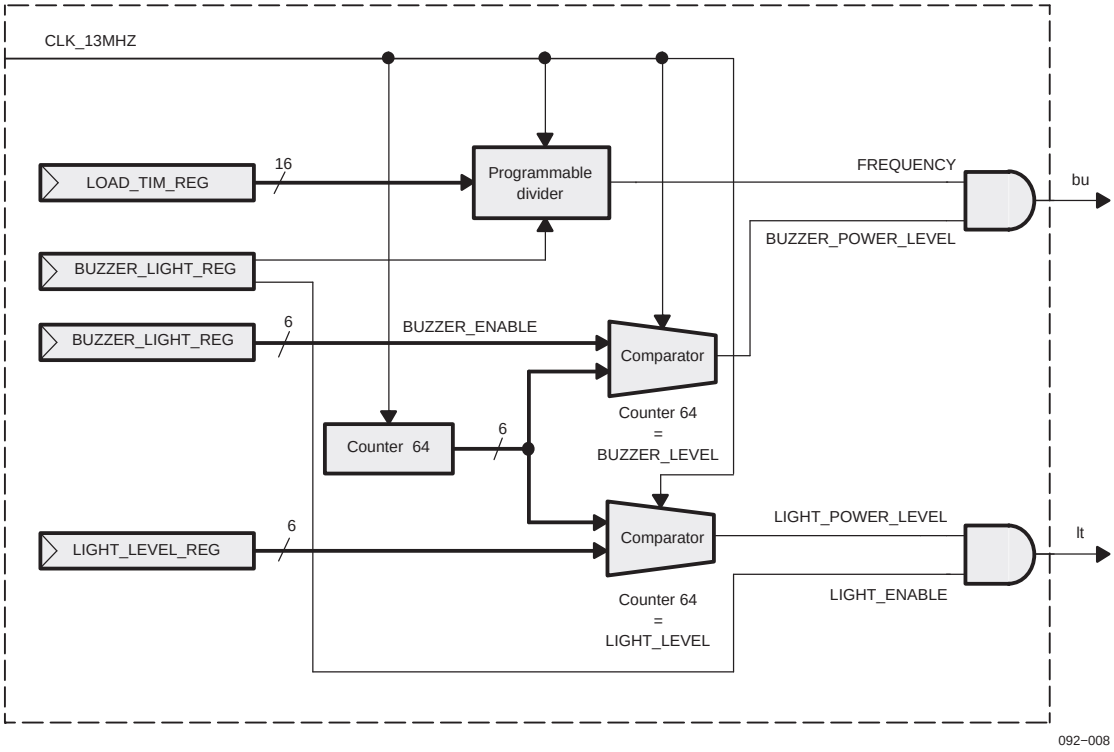
Note: The data in the JogDial data register has some delayed data from the data input register. This delay is set by the specified debouncing time.

Figure 27-7. Debouncing Path



- Some outputs of a GPIO module are reserved for buzzer and light control. For that function, the PWM generates two signals (LT and BU) to control the power level and tone of the light and the buzzer (see Figure 27-8).

Figure 27-8. GPIO Light and Buzzer Control



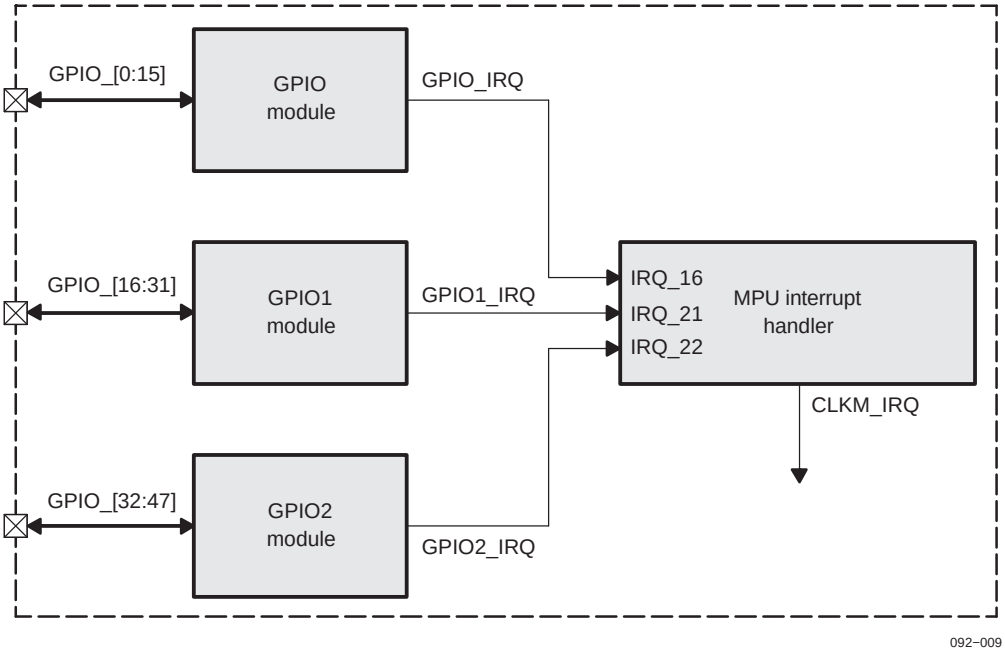
For further details, see Chapter 29, *Light and Buzzer Control*.

27.4.2 Interrupt Features

27.4.2.1 Interrupt Request Generation

The general-purpose interface has three interrupt lines (one interrupt line per GPIO module instance). The three interrupt signals (GPIOn_IRQ) are used by the MPU interrupt handler. Figure 27-9 shows the interrupt request.

Figure 27-9. Interrupt Request Description



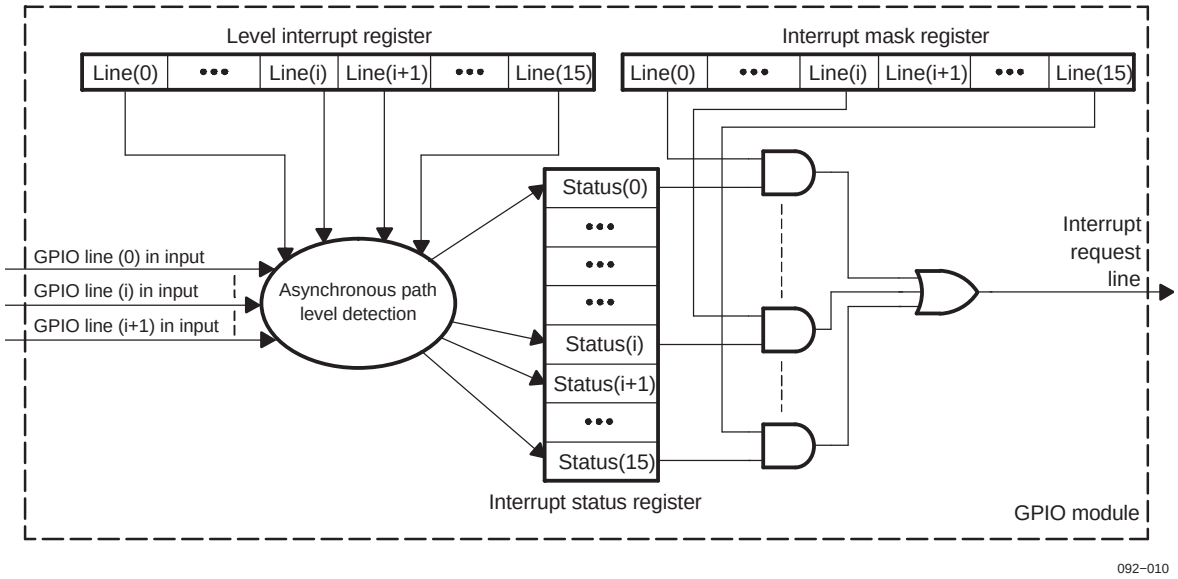
Asynchronous interrupt requests from each channel are processed by an interrupt generation submodule to be used by the MPU interrupt handler. This submodule controls its own asynchronous interrupt request line and has its own interrupt mask register (GPIOn.INTERRUPT_MASK_REG) and interrupt status register (GPIOn.INTERRUPT_STATUS_REG). The interrupt mask register selects the channel(s) considered for the interrupt request generation; the interrupt status register determines which channel(s) has activated the interrupt request. Event detection on GPIO channels is reflected in the interrupt status register independent of the interrupt mask register content. The interrupt status register can be read by the MPU to determine which input pin requires servicing.

When the GPIO configuration registers are set to unmask the interrupt generation (see Section 27.5.2, *Interrupt*), an asynchronous path samples the transition on the input GPIO. When an event matches the programmed settings, the corresponding bit in the interrupt status register is set to 1 and the interrupt line is activated (depending on the interrupt mask register).

The asynchronous interrupt request line is mapped on the MPU interrupt handler.

Figure 27-10 shows an overview of the interrupt request generation.

Figure 27-10. Interrupt Request Generation



27.4.2.2 Interrupt Line Release

When the MPU interrupt handler in the LOCOSTO device receives an interrupt request issued by the GPIO module, it can read the corresponding interrupt status register (GPIOn.INTERRUPT_STATUS_REG) to determine which GPIO input triggered the interrupt request.

After servicing the interrupt request, the processor resets the status bit and releases the interrupt line by writing 1 in the appropriate bit of the GPIOn.SOFT_CLEAR_REG register. This clears the corresponding bit of the interrupt status register. If an interrupt request is still pending (all bits in the interrupt status register not masked by the interrupt mask register are not cleared), the interrupt line is reasserted.

27.4.3 Debouncing

Debouncing is applied to only one GPIO input at a time. This cell uses a programmed delay to sample the input line and filter the input level. When the debounce is enabled, the interrupt level detection is not functional on the selected pin. To configure debouncing, see Section 27.5.4, *Debouncing*.

Note: When debouncing is enabled, the minimum pulse width on the GPIO input to trigger is set by the specified debouncing time.

27.4.4 Light and Buzzer Control

From the parameters contained in the light and buzzer control configuration registers, the PWM provides the following signals:

- BU controls the power level and the tone of a buzzer.
- LT controls the power level of a light.

For more information on the programming model of the configuration registers, see Section 27.5.5, *Light and Buzzer Control*.

For a functional description of the light and buzzer control, see Chapter 29, *Light and Buzzer Control*.

27.5 General-Purpose Interface Programming Model

27.5.1 Set and Clear Output Data Register

27.5.1.1 Description

The GPIO module implements the set and clear protocol register update for the GPIOn.OUTPUT_REG register. This protocol is an alternative to the atomic test and set operations and consists of writing operations at dedicated addresses (one address for setting bits, and one address for clearing bits). To clear or set a bit, write 1; write 0 at unaffected bits. The register can be accessed in two ways:

- Standard: Full register read and write operations at the primary register address
- Set and clear: Separate addresses to set and clear the bit register. Writing 1 at these addresses sets or clears the corresponding bit into the equivalent register; writing 0 has no effect.

Therefore, for this register, three addresses are defined for one unique physical register. Reading these addresses has the same effect and returns the register value.

27.5.1.2 Clear Instruction

27.5.1.2.1 Clear Register Address

A write operation in the clear data output register (GPIOn.CLEAR_ OUTPUT_REG) clears the corresponding bit in the data output register when the written bit is 1; a written 0 has no effect.

A read of the clear data output register returns the value of the data output register.

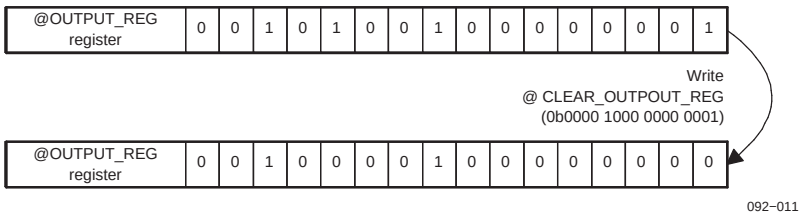
27.5.1.2.2 Clear Instruction Example

In this example, assume the data output register contains this binary value: 0b0010 1001 0000 0001. The intent is to clear bits 0 and 11.

With the clear instruction feature, write 0b0000 1000 0000 0001 at the address of the clear data output register. After this write operation, a reading of the data output register returns 0b0010 0001 0000 0000. Bits 0 and 11 have been cleared.

Figure 27-11 shows an example of a clear instruction.

Figure 27-11. Write @CLEAR_OUTPUT_REG Register Example



27.5.1.3 Set Instruction

27.5.1.3.1 Set Register Address

A write operation in the set data output register (GPIOn.SET_OUTPUT_REG) sets the corresponding bit in the data output register when the written bit is 1; a written 0 has no effect.

A read of the set data output register returns the value of the data output register.

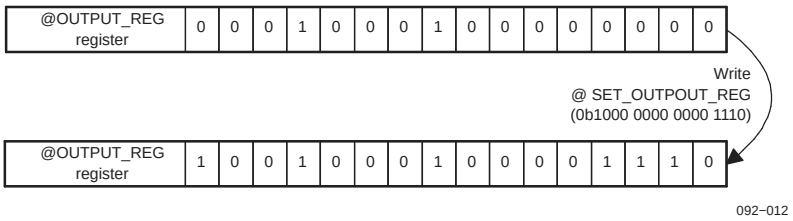
27.5.1.3.2 Set Instruction Example

In this example, assume the data output register contains this binary value: 0b0001 0001 0000 0000. The intent is to set bits 1, 2, 3, and 15.

With the set instruction feature, write 0b1000 0000 0000 1110 at the address of the set data output register. After this write operation, a reading of the data output register returns 0b1001 0001 0000 1110. Bits 1, 2, 3, and 15 have been set.

Figure 27-12 shows an example of a set instruction.

Figure 27-12. Write @SET_OUTPUT_REG Register Example



27.5.2 Interrupt

27.5.2.1 Related Configuration Registers

The interrupt configuration registers are as follows:

- Interrupt mask register (GPIOn.INTERRUPT_MASK_REG)
  The interrupt mask register allows masking of the excepted transition on the input GPIO from generating an interrupt request on the line. The interrupt mask register is programmed synchronously with the interface clock.
  This register is accessed with a direct read/write operation.
- Interrupt status register (GPIOn.INTERRUPT_STATUS_REG)
  The interrupt status register determines which input GPIO pin triggered the interrupt line request. This register can be read by the MPU to determine which input pin requires servicing.
  When a bit is set to 1 in this register, the corresponding GPIO pin is requesting the interrupt. To reset a bit in this register, use the alternate clear protocol register update feature with the soft clear register (GPIOn.SOFT_CLEAR_REG). Writing 1 to the interrupt status register cannot generate an interrupt, however.
  This register is accessed with a direct write operation.

Notes:

- During the reset phase, this register is set to 0. After the reset phase, if there is any connection, the register is undefined.
- The clearing of an interrupt status register bit (GPIOn.INTERRUPT_ STATUS_REG[bit]) must be done before unmasking this bit in the interrupt mask register.

- Interrupt level register (GPIOn.INTERRUPT_LEVEL_REG)
  The interrupt level register allows the defining of when an interrupt request should occur. The interrupt can be generated from either a high-to-low transition or a low-to-high transition. The interrupt level register is programmed synchronously with the interface clock.
  This register is accessed with a direct read/write operation.
- Interrupt status soft clear register (GPIOn.SOFT_CLEAR_REG)
  This register allows the resetting of the interrupt status register. A write operation in the soft clear register clears the corresponding bit in the interrupt status register when the written bit is 1; a written 0 has no effect.

A read of the clear data output register returns the value of the data output register.

This register is accessed with a direct write operation and is programmed synchronously with the interface clock.

27.5.2.2 Description

For the GPIO module to generate an interrupt request to the MPU interrupt handler at a defined transition occurring on a GPIO pin (interrupt source), the GPIO configuration registers must be programmed as follows:

1. Configure the GPIO pin as an input by the I/O control register (write 1 in the corresponding bit of the GPIOn.IO_CTNL_REG register).
2. Unmask the interrupts for the GPIO channel in the interrupt mask register (write 0 in the corresponding bit of the GPIOn.INTERRUPT_MASK_REG register).
3. Select the expected event(s) on the GPIO input to trigger the interrupt request in the interrupt level register (GPIOn.INTERRUPT_LEVEL_REG).

CAUTION

After servicing the interrupts, the status bit in the interrupt status register (GPIOn.INTERRUPT_STATUS_REG) must be reset and the interrupt line must be released (by writing 1 in the corresponding bit of the soft clear register) before masking an interrupt for the GPIO channel in the interrupt mask register (GPIOn.INTERRUPT_ MASK_REG). This is done to avoid unexpected interrupts when masking an interrupt for the GPIO channel.

27.5.3 Data I/O

The I/O control register (GPIOn.IO_CTNL_REG) controls the output or input capability for each pin. At reset, all the GPIO-related pins are configured as input. This register is not used within the module. Its only function is to carry the configuration of the pad. When the application uses a pin as an output and does not want interrupt generation from this pin, the application must properly configure the interrupt mask register (GPIOn.INTERRUPT_MASK_REG). For debouncing-only, mask all pins except the reference pin.

When a pin is configured as an output (the desired bit is set to 0 in the GPIOn.IO_CTNL_REG), the value of the corresponding bit in the data output register (GPIOn.OUTPUT_REG) is driven on the corresponding GPIO pin. Data is written to the data output register synchronously with the interface clock. This register can be accessed with direct read/write operations.

When a pin is configured as an input (the desired bit is set to 1 in GPIOn.IO_CTNL_REG), the state of the input can be read from the corresponding bit in the input data register (GPIOn.INPUT_LATCH). The input data register is sampled asynchronously.

27.5.4 Debouncing

27.5.4.1 Programming the Debouncing Configuration Registers

To apply the debounce filter on one GPIO pin, program the GPIO configuration registers as follows:

1. Configure the GPIO pin as an input by the I/O control register (write 1 in the corresponding bit of the GPIOn.IO_CTNL_REG register).
2. Unmask the interrupts from the GPIO channel in the interrupt mask register.
3. Enable debouncing (set to 1 the SET_JOGDIAL_MODE bit of the GPIOn.JOGDIAL_MODE_REG register).
4. Select a reference pin to debounce (enter the number of the selected pin in the GPIOn.JOGDIAL_MODE_REG[4:1] field).
5. Select the expected event on the GPIO input to start the debounce delay on the

DEBOUNCING_EDGE bit of the GPIOn.JOGDIAL_MODE_REG register.

6. Select debouncing (set the CLOCK_DEBOUNCING bit of the JOGDIAL_MODE_REG register to the correct value).
7. Set the value of the GPIOn.JD_DEBOUNCING_REG register to define the debouncing time (see Section 27.5.4.3,).

27.5.4.2 Changing the Debouncing Clock

To change the debouncing clock from 32 kHz to 13 MHz or from 13 MHz to 32 kHz, disable the debouncing (set to 0 the SET_JOGDIAL_MODE bit of the GPIOn.JOGDIAL_MODE_REG register).

Example:

Assume the GPIOn.JOGDIAL_MODE_REG register contains the binary value 0bXXXX XXXX X0XX XXX1. The debouncing is enabling, and the debouncing clock is running at 32 kHz. To change the debouncing, do the following:

1. Write 0bXXXX XXXX X1XX XXX0 in GPIOn.JOGDIAL_MODE_REG to disable the debouncing and change the clock.
2. Write 0bXXXX XXXX X1XX XXX1 in GPIOn.JOGDIAL_MODE_REG to restart the debouncing.

27.5.4.3 Debouncing Time

The debouncing value register (GPIOn.JD_DEBOUNCING_REG) is used to set the debouncing time for the input line selected in the GPIO module. The debouncing cell is running with the debouncing clock (32 kHz or 13 MHz). This register represents the number of clock cycles to use.

One cycle is 30 _UnicodeEncodeError_s long for the 32-kHz clock; it is 76.9 ns for the 13-MHz clock.

The following formula describes the required input stable time to be propagated to the debounced output:

time = JD_DEBOUNCING_REG x Frequency ± Frequency/2

Where: 0 ≤ JD_DEBOUNCING_REG value ≤ 2^16 = 65536

27.5.4.4 Debouncing Reset

The debouncing system is set to 0 when any of the following conditions occur:

- The GPIO_nRESET = 0.
- The debouncing finishes (JD_DEBOUNCING_REG = Counter value).
- A glitch occurs on the debouncing reference pin during the debouncing.
- The field PINS_MODE_REG of the JOGDIAL_MODE_REG register is cleared.

27.5.5 Light and Buzzer Control

CAUTION

The GPIOn.BUZZER_LIGHT_REG[0] BUZZER bit and the GPIOn.BUZZER_LIGHT_REG[1] LIGHT bit must be set to 0 to load a new value in the following registers:

- GPIOn.LOAD_TIM_REG[15:0] LOAD_TIM_REG field
- GPIOn.LIGHT_LEVEL_REG[5:0] LIGHT_LEVEL_REG field
- GPIOn.BUZZER_LEVEL_REG[5:0] BUZZER_LEVEL_REG field

27.5.5.1 Related Configuration Registers

- Buzzer and light control register (GPIOn.BUZZER_LIGHT_REG)

  The buzzer and light control register controls the enabling of the buzzer and light.

  This register is accessed with a direct read/write operation and is programmed synchronously with the interface clock.
- Light power level register (GPIOn.LIGHT_LEVEL_REG)

  The light power level register controls the power level of the light.

  This register is accessed with a direct read/write operation and is programmed synchronously with the interface clock.
- Buzzer power level register (GPIOn.BUZZER_LEVEL_REG)

  The buzzer power level register controls the power level of the buzzer.

  This register is accessed with a direct read/write operation and is programmed synchronously with the interface clock.
- Load timer register (GPIOn.LOAD_TIM_REG)

  The load timer register defines the cycles of the buzzer.

  This register is accessed with a direct read/write operation and is programmed synchronously with the interface clock.

For further details, see Chapter 29, *Light and Buzzer Control*.

27.6 General-Purpose Interface Register Manual

This section summarizes the hardware interface for the GPIO, including each module instance, the module register map, and the bit definitions for each bit field.

Table 27-5 lists the base address and address space for the GPIO module instances. Table 27-6 through Table 27-8 list the GPIO registers.

Table 27-5. Instance Summary

Module Name	Base Address	Size
GPIO	0xFFFE 4800	2K bytes
GPIO1	0xFFFE 5000	2K bytes
GPIO2	0xFFFE 5800	2K bytes

27.6.1 GPIO Register Mapping Summary

Table 27-6. GPIO Register Summary

Register Name	Type	Register Width (Bits)	Offset
INPUT_LATCH	R	16	0x00
OUTPUT_REG	R/W	16	0x02
IO_CNTL_REG	R/W	16	0x04
GPIO_CNTL_REG	R/W	16	0x06
LOAD_TIM_REG	R/W	16	0x08
SET_OUTPUT_REG	R/W	16	0x0A
CLEAR_OUTPUT_REG	R/W	16	0x0C
BUZZER_LIGHT_REG	R/W	16	0x0E
LIGHT_LEVEL_REG	R/W	16	0x10
BUZZER_LEVEL_REG	R/W	16	0x12
JOGDIAL_MODE_REG	R/W	16	0x14
INTERRUPT_LEVEL_REG	R/W	16	0x16
INTERRUPT_MASK_REG	R/W	16	0x18
JD_DEBOUNCING_REG	R/W	16	0x1A
JOGDIAL_DIR_REG	R	16	0x1C
INTERRUPT_STATUS_REG	R	16	0x1E
PULL_REG	R	16	0x20
SOFT_CLEAR_REG	R/W	16	0x22

Table 27-7. GPIO1 Register Summary

Register Name	Type	Register Width (Bits)	Offset
INPUT_LATCH	R	16	0x00
OUTPUT_REG	R/W	16	0x02
IO_CNTL_REG	R/W	16	0x04
GPIO_CNTL_REG	R/W	16	0x06
LOAD_TIM_REG	R/W	16	0x08
SET_OUTPUT_REG	R/W	16	0x0A
CLEAR_OUTPUT_REG	R/W	16	0x0C
BUZZER_LIGHT_REG	R/W	16	0x0E
LIGHT_LEVEL_REG	R/W	16	0x10
BUZZER_LEVEL_REG	R/W	16	0x12

Table 27-7. GPIO1 Register Summary (continued)

Register Name	Type	Register Width (Bits)	Offset
JOGDIAL_MODE_REG	R/W	16	0x14
INTERRUPT_LEVEL_REG	R/W	16	0x16
INTERRUPT_MASK_REG	R/W	16	0x18
JD_DEBOUNCING_REG	R/W	16	0x1A
JOGDIAL_DIR_REG	R	16	0x1C
INTERRUPT_STATUS_REG	R	16	0x1E
PULL_REG	R	16	0x20
SOFT_CLEAR_REG	R/W	16	0x22

Table 27-8. GPIO2 Register Summary

Register Name	Type	Register Width (Bits)	Offset
INPUT_LATCH	R	16	0x00
OUTPUT_REG	R/W	16	0x02
IO_CNTL_REG	R/W	16	0x04
GPIO_CNTL_REG	R/W	16	0x06
LOAD_TIM_REG	R/W	16	0x08
SET_OUTPUT_REG	R/W	16	0x0A
CLEAR_OUTPUT_REG	R/W	16	0x0C
BUZZER_LIGHT_REG	R/W	16	0x0E
LIGHT_LEVEL_REG	R/W	16	0x10
BUZZER_LEVEL_REG	R/W	16	0x12
JOGDIAL_MODE_REG	R/W	16	0x14
INTERRUPT_LEVEL_REG	R/W	16	0x16
INTERRUPT_MASK_REG	R/W	16	0x18
JD_DEBOUNCING_REG	R/W	16	0x1A
JOGDIAL_DIR_REG	R	16	0x1C
INTERRUPT_STATUS_REG	R	16	0x1E
PULL_REG	R	16	0x20
SOFT_CLEAR_REG	R/W	16	0x22

27.6.2 Register Description

Table 27-9 through Table 27-26 describe the individual register bits.

Table 27-9. INPUT_LATCH

Address Offset	0x00		
Physical Address	0xFFFE 4800	**Instance**	GPIO
	0xFFFE 5000		GPIO1
	0xFFFE 5800		GPIO2
Description	This register reflects the current value of the pins.		
Type	R		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INPUT_LATCH															

Bits	Field Name	Description	Type	Reset
15:0	INPUT_LATCH	0: Input value is 0. 1: Input value is 1.	R	Reflects input pins

Table 27-10. OUTPUT_REG

Address Offset	0x02		
Physical Address	0xFFFE 4802	**Instance**	GPIO
	0xFFFE 5002		GPIO1
	0xFFFE 5802		GPIO2
Description	This register stores values to send on the output pin.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUTPUT_REG															

Bits	Field Name	Description	Type	Reset
15:0	OUTPUT_REG	0: Output value is 0. 1: Output value is 1.	R	0xFFFF

Table 27-11. IO_CNTL_REG

Address Offset	0x04		
Physical Address	0xFFFE 4804	**Instance**	GPIO
	0xFFFE 5004		GPIO1
	0xFFFE 5804		GPIO2
Description	This register is used to configure the GPIO pins for either input or output.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IO_CNTL_REG															

Bits	Field Name	Description	Type	Reset
15:0	IO_CNTL_REG	0: Output 1: Input	R	0xFFFF

Table 27-12. GPIO_CNTL_REG

Address Offset	0x06		
Physical Address	0xFFFE 4806	**Instance**	GPIO
	0xFFFE 5006		GPIO1
	0xFFFE 5806		GPIO2
Description	This register is used in the emulation mode.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CLOCK_ENBLE	Reserved	SOFT	FREE	Reserved	

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Reserved	R	Undefined
5	CLOCK_ENBLE	GPIO 13-MHz clock enable	R/W	0x0
4	Reserved	Reserved	R	Undefined
3	SOFT	Soft bit This bit is used with the FREE bit to determine the state of the peripheral when a breakpoint is encountered. This bit is used in the emulation mode. 0: The peripheral halts immediately, either retaining or discarding the current state. 1: The peripheral stops after completion of the current task.	R/W	0x0
2	FREE	Free bit This bit is used with the SOFT bit to determine the state of the peripheral when a breakpoint is encountered. This bit is used in the emulation mode. 0: The SOFT bit selects the emulation mode. 1: The peripheral clock runs free regardless of the SOFT bit.	R/W	0x0
1:0	Reserved	Reserved	R	Undefined

Table 27-13. LOAD_TIM_REG

Address Offset	0x08		
Physical Address	0xFFFE 4808	**Instance**	GPIO
	0xFFFE 5008		GPIO1
	0xFFFE 5808		GPIO2
Description	This register contains the value that is loaded when the timer passes through 0 or when it starts.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD_TIM_REG															

Bits	Field Name	Description	Type	Reset
15:0	LOAD_TIM_REG	Defines the cycles of the buzzer	R/W	0x0000

Table 27-14. SET_OUTPUT_REG

Address Offset	0x0A		
Physical Address	0xFFFE 480A	**Instance**	GPIO
	0xFFFE 500A		GPIO1
	0xFFFE 580A		GPIO2
Description	This register allows the setting of specific bits of OUTPUT_REG to 1 without writing in OUTPUT_REG.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SET															

Bits	Field Name	Description	Type	Reset
15:0	SET	0: No change on OUTPUT_REG[bit] 1: Set the OUTPUT_REG[bit] to 1.	R/W	0x0000

Table 27-15. CLEAR_OUTPUT_REG

Address Offset	0x0C		
Physical Address	0xFFFE 480C	**Instance**	GPIO
	0xFFFE 500C		GPIO1
	0xFFFE 580C		GPIO2
Description	This register allows the setting of specific bits of OUTPUT_REG to 0 without writing in OUTPUT_REG.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLEAR															

Bits	Field Name	Description	Type	Reset
15:0	CLEAR	0: No change on OUTPUT_REG[bit] 1: Set the OUTPUT_REG[bit] to 0.	R/W	0x0000

Table 27-16. BUZZER_LIGHT_REG

Address Offset	0x0E		
Physical Address	0xFFFE 480E	**Instance**	GPIO
	0xFFFE 500E		GPIO1
	0xFFFE 580E		GPIO2
Description	This register is used to start or stop the buzzer and/or the light.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														BUZZER	LIGHT

Bits	Field Name	Description	Type	Reset
15:2	Reserved	Reserved	R	Undefined
1	BUZZER	0: Stop buzzer 1: Start buzzer	R/W	0x0
0	LIGHT	0: Stop light 1: Start light	R/W	0x0

Table 27-17. LIGHT_LEVEL_REG

Address Offset	0x10		
Physical Address	0xFFFE 4810	**Instance**	GPIO
	0xFFFE 5010		GPIO1
	0xFFFE 5810		GPIO2
Description	This register contains the value for the light power level.		
Type	R/W		

Table 27-17. LIGHT_LEVEL_REG (continued)

Write Latency	Not relevant

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								LIGHT_LEVEL_REG							

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Reserved	R	Undefined
5:0	LIGHT_LEVEL_REG	0x00: Level 0 (no light) 0x01: Level 1 0x1F: Level 63	R/W	0x1F

Table 27-18. BUZZER_LEVEL_REG

Address Offset	0x12		
Physical Address	0xFFFE 4812	**Instance**	GPIO
	0xFFFE 5012		GPIO1
	0xFFFE 5812		GPIO2
Description	This register contains the value for the buzzer power level.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BUZZER_LEVEL_REG							

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Reserved	R	Undefined
5:0	BUZZER_LEVEL_REG	0x00: Level 0 (no light) 0x01: Level 1 0x1F: Level 63	R/W	0x1F

Table 27-19. JOGDIAL_MODE_REG

Address Offset	0x14		
Physical Address	0xFFFE 4814	**Instance**	GPIO
	0xFFFE 5014		GPIO1
	0xFFFE 5814		GPIO2
Description	This register is used to configure the debouncing filter of a pin.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CLOCK_DEBOUNCING	DEBOUNCING_EDGE	PIN_MODE_REG					SET_JOGDIAL_MODE

Bits	Field Name	Description	Type	Reset
15:7	Reserved	Reserved	R	Undefined
6	CLOCK_DEBOUNCING	0: 32 kHz 1: 13 MHz	R/W	0x0
5	DEBOUNCING_EDGE	0: High-to-low transition 1: Low-to-high transition	R/W	0x0
4:1	PIN_MODE_REG	Choose reference pin to debounce	R/W	0x0
0	SET_JOGDIAL_MODE	0: No debouncing (or reset) 1: Debouncing	R/W	0x0

Table 27-20. INTERRUPT_LEVEL_REG

Address Offset	0x16		
Physical Address	0xFFFE 4816	**Instance**	GPIO
	0xFFFE 5016		GPIO1
	0xFFFE 5816		GPIO2
Description	This register is used to define the edge on which an interrupt should occur when the GPIO input pin(s) moved.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTERRUPT_LEVEL_REG															

Bits	Field Name	Description	Type	Reset
15:0	INTERRUPT_LEVEL_REG	0: High-to-low transition 1: Low-to-high transition	R	0x0000

Table 27-21. INTERRUPT_MASK_REG

Address Offset	0x18		
Physical Address	0xFFFE 4818	**Instance**	GPIO
	0xFFFE 5018		GPIO1
	0xFFFE 5818		GPIO2
Description	This register is used to mask a specific input from generating an interrupt request.		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTERRUPT_MASK_REG															

Bits	Field Name	Description	Type	Reset
15:0	INTERRUPT_MASK_REG	0: Unmasked 1: Masked	R	0xFFFF

Table 27-22. JD_DEBOUNCING_REG

Address Offset	0x1A		
Physical Address	0xFFFE 481A	**Instance**	GPIO
	0xFFFE 501A		GPIO1
	0xFFFE 581A		GPIO2
Description	This register controls debouncing time (the value is global for all ports).		
Type	R/W		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JD_DEBOUNCING															

Bits	Field Name	Description	Type	Reset
15:0	JD_DEBOUNCING	Input debouncing time value	R	0x0000

Table 27-23. JOGDIAL_DIR_REG

Address Offset	0x1C		
Physical Address	0xFFFE 481C	**Instance**	GPIO
	0xFFFE 501C		GPIO1
	0xFFFE 581C		GPIO2
Description	This register contains the INPUT_LATCH data loaded at the end of debouncing.		
Type	R		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JOGDIAL_REG															

Bits	Field Name	Description	Type	Reset
15:0	JOGDIAL_REG	Load gpio_in data at the end of debouncing	R	Undefined

Table 27-24. INTERRUPT_STATUS_REG

Address Offset	0x1E		
Physical Address	0xFFFE 481E	**Instance**	GPIO
	0xFFFE 501E		GPIO1
	0xFFFE 581E		GPIO2
Description	This register contains the current interrupts status of each GPIO pin.		
Type	R		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTERRUPT_STATUS_REG															

Bits	Field Name	Description		Type	Reset
15:0	INTERRUPT_STATUS_REG	0:	No interrupt request	R	Undefined
		1:	Interrupt request		

Table 27-25. PULL_REG

Address Offset	0x20		
Physical Address	0xFFFE 4820	**Instance**	GPIO
	0xFFFE 5020		GPIO1
	0xFFFE 5820		GPIO2
Description	Reserved		
Type			
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Bits	Field Name	Description	Type	Reset
15:0	Reserved	Reserved	R	Undefined

Table 27-26. SOFT_CLEAR_REG

Address Offset	0x22		
Physical Address	0xFFFE 4822	**Instance**	GPIO
	0xFFFE 5022		GPIO1
	0xFFFE 5822		GPIO2
Description	This register allows to clear the interrupt status register.		
Type	RW		
Write Latency	Not relevant		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SOFT_CLEAR															

Bits	Field Name	Description		Type	Reset
15:0	SOFT_CLEAR	0:	No change on INTERRUPT_STATUS_REG[bit]	R/W	0x0000
		1:	Set the INTERRUPT_STATUS_REG[bit] to 0.		

C-port

This chapter describes the codec port (C-port) module of the LOCOSTO device.

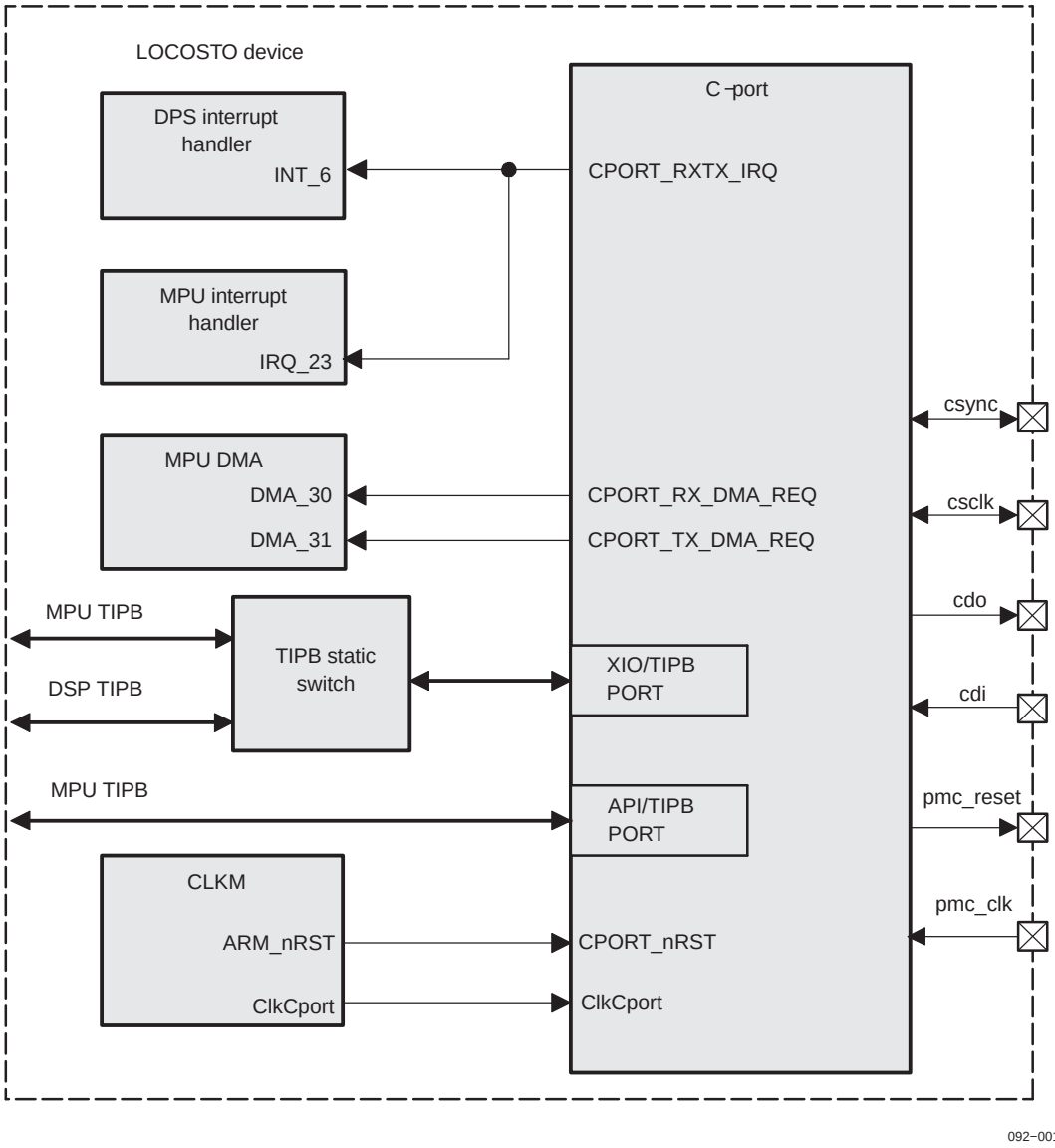
Topic **Page**

28.1 C-port Overview

The LOCOSTO device includes a codec port (C-port) interface module dedicated to audio applications.

Figure 28-1 shows an overview of the C-port module in the LOCOSTO device.

Figure 28-1. C-port Interface Overview



The C-port interface can be configured to support several industry standard serial interface protocols. These protocols include the following:

- Inter-IC sound (I2S) mode
- 16-bit pulse code modulation (PCM) mode: Least significant bit (LSB)-justified format or standard format
- AC'97 Revision 1.x without sample rate converter mode

The C-port interface can be configured in slave or master mode.

The C-port has three interfaces:

- External interface: A C-port interface to connect an external codec
- Internal interface: An XIO/TIPB port interface connected to either the digital signal processor (DSP) or microprocessor unit (MPU) TI peripheral bus (TIPB); used for C-port module configuration and data transfer.
- Internal interface: An API/TIPB port interface connected to the MPU TIPB; used for direct memory access (DMA) operations.

The C-port interface includes two independent DMA channels to read or write audio data into the system memory. Each channel has a FIFO for data flow control with buffered data registers that allow a continuous data stream for DMA operations.

The C-port interface is a buffered serial port interface (4-word-deep FIFO for transmission and 4-word-deep FIFO for reception. A word consists of 16 bits).

During transmission, when the TX FIFO is almost empty, an emission interrupt (CPORT_RXTX_IRQ) is sent to the MPU or DSP interrupt handler, if the interrupt is not masked. A DMA transmission request (CPORT_TX_DMA_REQ) is also sent to the DMA, if the DMA request is not masked.

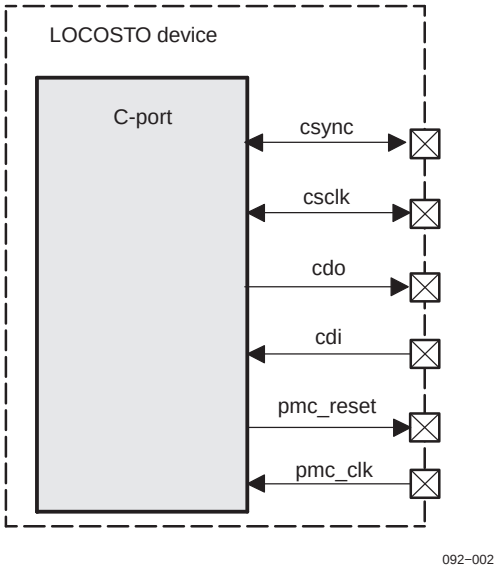
During reception, when the RX FIFO is almost full, a reception interrupt (CPORT_RXTX_IRQ) is sent to the MPU or DSP interrupt handler, if the interrupt is not masked. A DMA reception request (CPORT_RX_DMA_REQ) is also sent to the DMA, if the DMA request is not masked.

28.2 C-port Environment

The C-port interface is a serial interface used to transfer data between the DSP or MPU and a codec device. The serial protocol and formats supported are the AC'97 1.0 modes, PCM modes, and I2S modes.

Figure 28-2 shows the general environment of the C-port interface.

Figure 28-2. C-port Interface Environment



An external companion chip can be provided for serial audio interface. Texas Instruments provides a global solution with a LOCOSTO device and an audio device: the TWL3031 IC. For more details on the TWL3031 device, contact your TI representative.

Notes:

- In master mode, the serial and synchro clocks are generated using a divider of an external clock supplied by the slave codec. In the LOCOSTO device, the C-port clock frequency is fixed at 13 MHz by default. To configure the C-port interface in master mode with a slave codec, apply a clock generator to the external input clock pin (pmc_clk) at the correct frequency (11.2896 MHz in I2S and PCM modes, and 12.288 MHz or 24.576 MHz in AC'97 mode). The pmc_clk and pmc_reset signals are used only in master mode.
- In slave mode, the serial and synchro clocks are generated through the internal 13-MHz ClkCport clock.

The pmc_reset, pmc_clk, and cdi signals are multiplexed with the gpio_1, gpio_2, and gpio_4 signals, respectively.

The csync, csclk, and cdo signals are available in the following modes:

- Primary mode 0: For direct access on these signals
- Multiplexed mode: For csync (input mode only) and csclk (input mode only). In this case, the csync and csclk signals are multiplexed with the mcsi_fs and mcsi_ck signals, respectively. The cdo signal is also multiplexed with the mcsi_tx signal. For more details, see Chapter 18, *Configuration*.

28.2.1 I2S Mode Functional Interfaces

In I2S mode, the C-port interface can be configured as either a master or a slave I2S link serial interface to the I2S codec device. The I2S link serial interface is a time division multiplexed (TDM) slot-based serial interface used to transfer audio data and command/status data to the codec device.

Figure 28-3 shows how the C-port master interface connects to an I2S slave codec IC. Figure 28-4 shows how the C-port slave interface connects to an I2S master codec IC (TWL3031 case).

Figure 28-3. Connection of the C-port Master Interface to an I2S Slave Codec IC

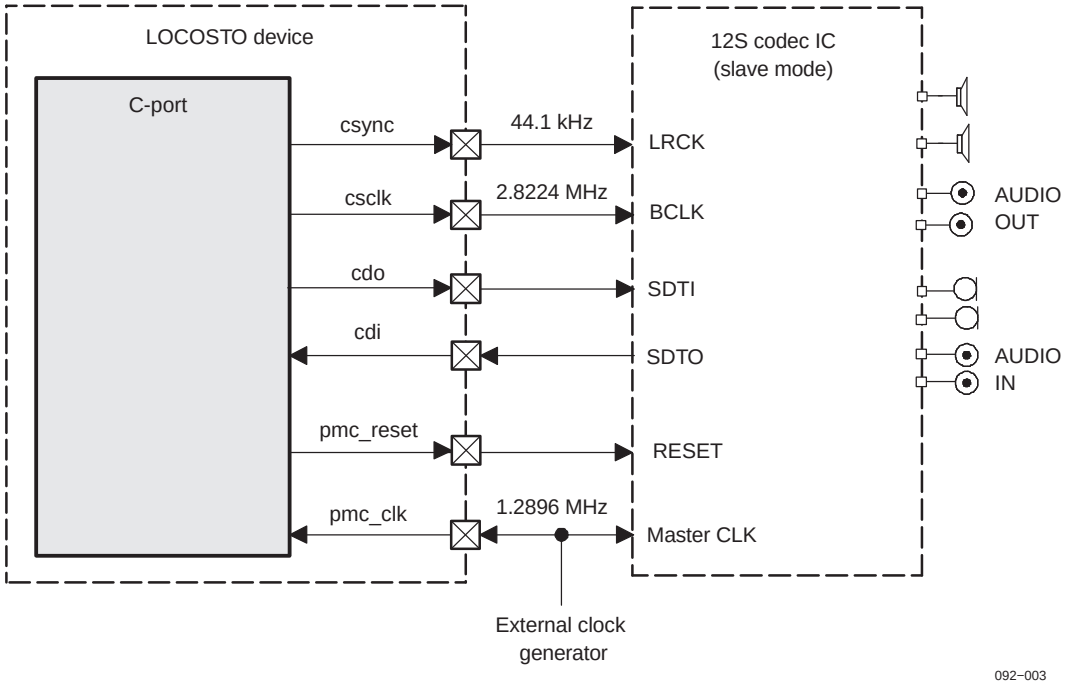
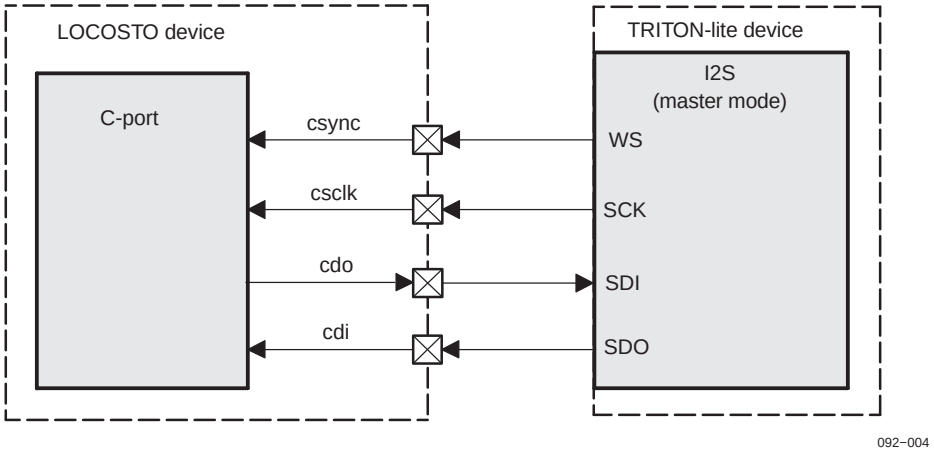


Figure 28-4. Connection of the C-port Slave Interface to an I2S Master Codec IC (TWL3031 case)



The TWL3031 device uses the LOCOSTO 13-MHz clock to generate sampling frequency and serial port clock

Note: Only I2S slave mode for the C-port module is supported if the LOCOSTO device is associated with the TWL3031 companion chip. For more details on the TWL3031 IC, contact your TI representative.

28.2.1.1 C-port Pins for I2S Mode

Figure 28-5 shows the I2S master mode interface signal. Figure 28-6 shows the I2S slave mode interface signal.

Figure 28-5. I2S Master Mode Interface Signal

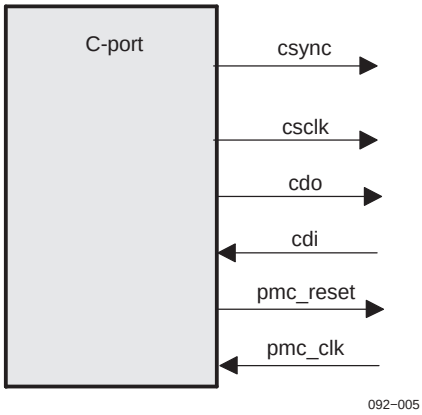
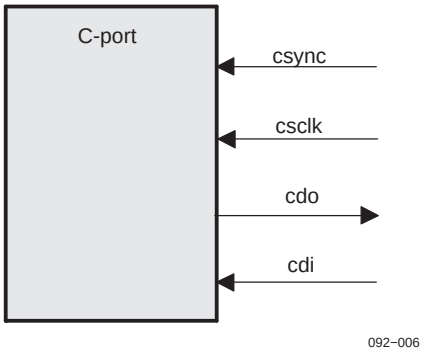


Figure 28-6. I2S Slave Mode Interface Signal



28.2.1.2 I2S Mode Interface Description

Table 28-1 lists the I2S mode inputs and outputs (I/Os).

Table 28-1. I2S Mode I/O Description

Signal Name	I/O	Description	Reset Value
csync	O (master mode)	C-port interface frame synchro	0 (master mode)
	I (slave mode)		Unknown (slave mode)
csclk	O (master mode)	C-port interface serial clock	0 (master mode)
	I (slave mode)		Unknown (slave mode)
cdo	O	C-port interface serial data output	0
cdi	I	C-port interface serial data input	Unkown
pmc_reset	O (master mode only)	C-port interface output reset	0
pmc_clk	I (master mode only)	C-port interface input master clock (provided by an external clock generator)	Unkown

Note: I = Input, O = Output

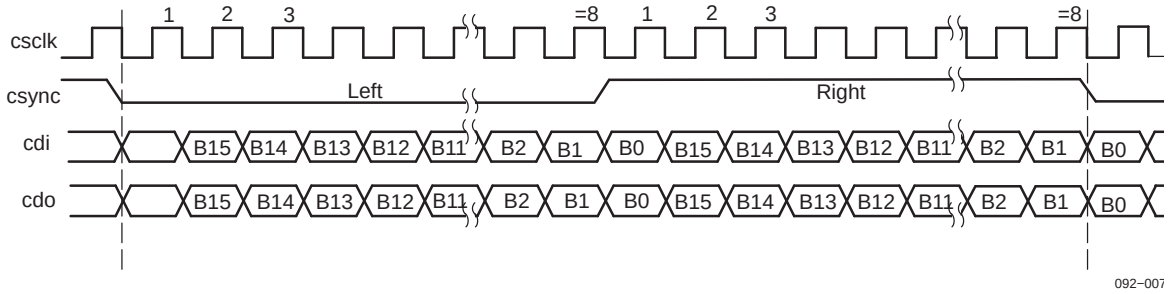
28.2.1.3 I2S Mode Protocol and Data Format

In this mode, the C-port interface is configured as a bidirectional full-duplex serial interface with two time slots per frame: time slot 0 is used for the left channel audio data, and time slot 1 is used for the right channel audio data.

The csync signal has a 50 percent duty cycle. The csync signal is low for the left channel time slot and high for the right channel time slot. In addition, the csync signal is synchronous to the falling edge of the csclk signal. The serial data is shifted out on the falling edge of csclk and shifted in on the rising edge of csclk. There is a one csclk cycle delay from the edge of the csync before the most significant bit (MSB) of the data is shifted out for both the left and right channels.

For the I2S mode of the C-port interface, there is a 16-bit transmit and a 16-bit receive shift register for each cdo and cdi signal, respectively. The interface pads the unused bits automatically with zeros. Serial data is transmitted in twos complement with the MSB first. Figure 28-7 shows the I2S serial interface format.

Figure 28-7. I2S Serial Interface Format



The transmitter always sends the MSB of the next word one clock period after the csync changes.

The valid data bits and time slot length setting must not be less than 16 bits. In case of higher values, only the 16 MSBs of each left and right sample are taken into account; the input sample LSBs are discarded, and the output sample LSBs are filled with zeros.

For more details on the I2S protocol, see the I2S bus specification on www.philips.com.

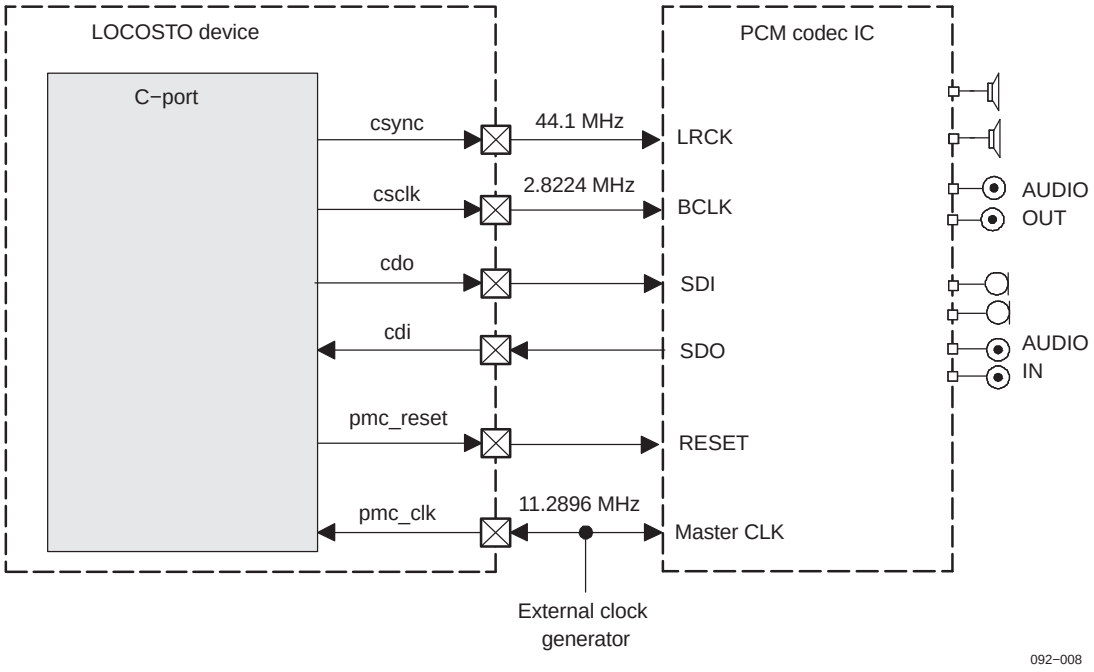
The registers dedicated to the AC'97 protocol are not used. The C-port valid time slot registers (CPORT.CPTVSLL and CPORT.CPTVSLH) must be left at their reset value. The content of the C-port address and data registers (CPORT.CPTTADR, CPORT.CPTDATL, and CPORT.CPTATH) is unknown.

28.2.2 PCM Mode Functional Interfaces

In PCM mode, the C-port interface can be configured as a PCM link serial interface to the PCM codec device. The LOCOSTO device is the master in PCM mode. The PCM link serial interface is a TDM slot-based serial interface used to transfer audio data and command/status data to the codec device.

Figure 28-8 shows the C-port PCM mode.

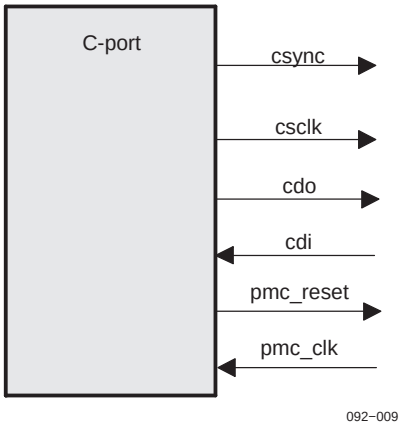
Figure 28-8. C-port PCM Mode



28.2.2.1 C-port Pins for PCM Mode

Figure 28-9 shows the interface signal for PCM mode.

Figure 28-9. PCM Mode Interface Signal



28.2.2.2 PCM Mode Interface Description

Table 28-2 lists the inputs and outputs for the PCM mode.

Table 28-2. PCM Mode I/O Description

Signal Name	I/O	Description	Reset Value
csync	O	C-port interface frame synchro	0
csclk	O	C-port interface serial clock	0
cdo	O	C-port interface serial data output	0
cdi	I	C-port interface serial data input	Unkown

Table 28-2. PCM Mode I/O Description (continued)

Signal Name	I/O	Description	Reset Value
pmc_reset	O	C-port interface output reset	0
pmc_clk	I	C-port interface input master clock (provided by an external clock generator)	Unkown

Note: I = Input, O = Output

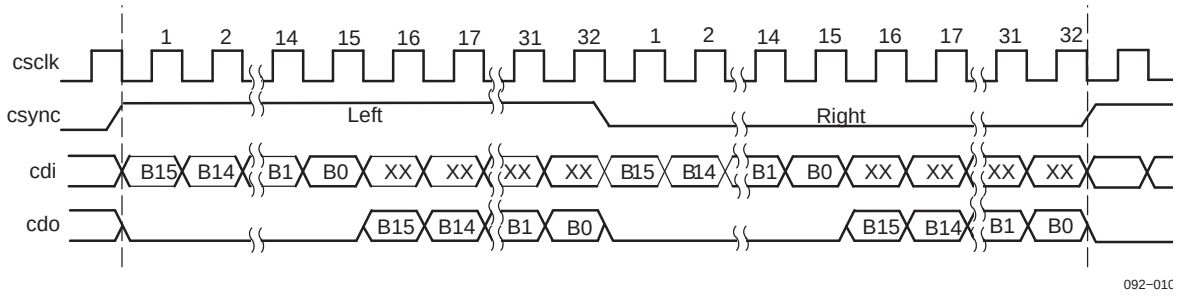
28.2.2.3 PCM Mode Protocol and Data Format

In this mode, the C-port interface is configured as a bidirectional full-duplex serial interface with two time slots per frame: the time slot 0 is used for the left channel audio data, and the time slot 1 for the right channel audio data.

The csync signal has a 50 percent duty cycle. The csync signal is high for the left channel time slot and low for the right channel time slot. In addition, the csync signal is synchronous to the falling edge of the csclk signal. The serial data is shifted out on the falling edge of csclk and shifted in on the rising edge of csclk. There is no csclk cycle delay from the edge of the csync before the MSB of the data is shifted out for both the left and right channels.

For the PCM mode of the C-port interface, there is a 16-bit transmit and a 16-bit receive shift register for each cdo and cdi signal, respectively. Figure 28-10 shows that unlike the I2S mode, when there is a data in, there is no data out, and vice versa.

Figure 28-10. PCM Serial Interface Format



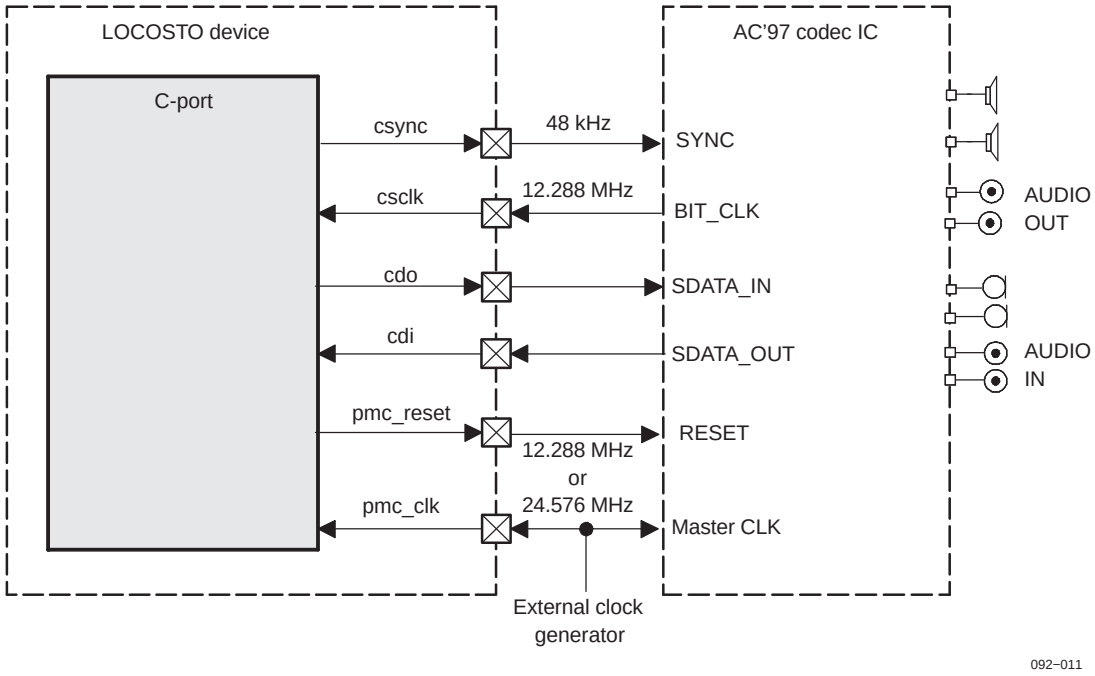
As for the I2S mode, the valid data bits and time slot length setting must not be less than 16 bits. In the case of PCM mode with higher valid serial data bit values, only the 16 MSBs of each left and right sample are taken into account; the input sample LSBs are discarded, and the output sample LSBs are filled with zeros.

The registers dedicated to the AC'97 protocol are not used. The C-port valid time slot registers (CPORT.CPTVSLL and CPORT.CPTVSLH) must be left at their reset value. The content of the C-port address and data registers (CPORT.CPTTADR, CPORT.CPTDATL, and CPORT.CPTATH) is unknown.

28.2.3 AC'97 Mode Functional Interfaces

In AC'97 mode, the C-port interface can be configured as an ac link serial interface to the AC'97 codec device (see Figure 28-11). The LOCOSTO device is the master in AC'97 mode. The ac link serial interface is a TDM slot-based serial interface used to transfer audio data and command/status data to the codec device.

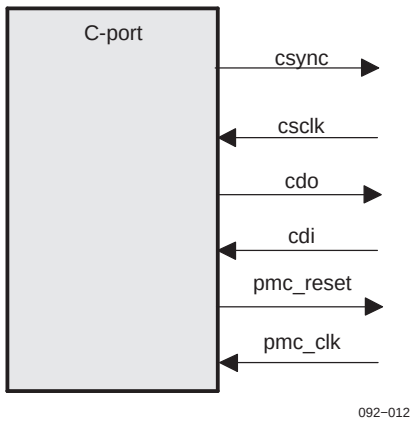
Figure 28-11. C-port AC'97 Mode



28.2.3.1 C-port Pins for AC'97 Mode

Figure 28-12 shows the interface signal for the AC'97 mode.

Figure 28-12. AC'97 Mode Interface Signal



28.2.3.2 AC'97 Mode Interface Description

Table 28-3 lists the I/Os for the AC'97 mode.

Table 28-3. AC'97 Mode I/O Description

Signal Name	I/O	Description	Reset Value
csync	O	C-port interface frame synchro	0
csclk	I	C-port interface serial clock	Unknown
cdo	O	C-port interface serial data output	0
cdi	I	C-port interface serial data input	Unkown

Table 28-3. AC'97 Mode I/O Description (continued)

Signal Name	I/O	Description	Reset Value
pmc_reset	O	C-port interface output reset	0
pmc_clk	I	C-port interface input master clock (provided by an external clock generator)	Unkown

Note: I = Input, O = Output

28.2.3.3 AC'97 Mode Protocol and Data Format

In this mode, the C-port interface is configured as a bidirectional full-duplex serial interface with a fixed rate of 48 kHz. Each 48-kHz frame is divided into 13 time slots, with the use of each time slot predefined by the AC'97 specification (see Table 28-4).

Table 28-4. AC'97 Audio Frame

Time Slot	Out	In
0	Tag	Tag
1	Command address port	Status address port
2	Command data port	Status data port
3	PCM playback left channel	PCM record left channel
4	PCM playback right channel	PCM record right channel
5	Optional modem line 1 DAC	Optional modem line 1 ADC
6	Optional PCM center	Optional dedicated microphone record data
7	Optional PCM left surround	Reserved
8	Optional PCM right surround	Reserved
9	Optional LFE DACs	Reserved
10	Optional modem line 2 DAC	Optional modem line 2 ADC
11	Optional modem headset DAC	Optional modem headset ADC
12	Optional modem general-purpose input/output (GPIO) control	Optional modem GPIO status

For more details on the AC'97 specification, contact your TI representative.

Each time slot, except for slot 0, is composed of 20 serial clock cycles; time slot 0 has 16 serial clock cycles. The serial clock (csclk), referred to as the BIT_CLK for AC'97 modes, is set to 12.288 MHz.

The csclk is derived from the codec master system clock (pcm_clk) input, according to the following formula:

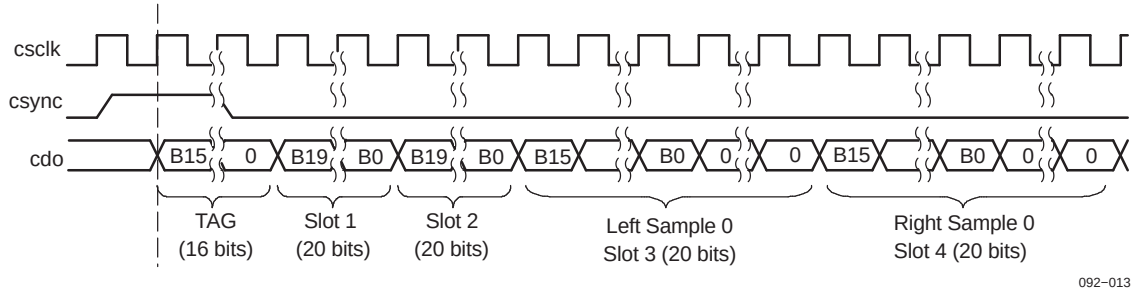
- If pcm_clk = 12.288 MHz, then csclk = pcm_clk = 12.288 MHz.
- If pcm_clk = 24.576 MHz, then csclk = pcm_clk / 2 = 12.288 MHz.

The csync signal is generated by dividing-down the serial bit clock (csclk):

- 12.288 MHz / 256 = 48 kHz

Figure 28-13 shows the AC'97 serial interface format.

Figure 28-13. AC'97 Serial Interface Format



The start of all audio sample packets (audio frames) transferred over the ac link is synchronized to the rising edge of the csync signal. Data is transitioned on the ac link on every rising edge of the csclk signal, and is subsequently sampled on the receiving side of the ac link immediately following the falling edge of the csclk signal. This configuration occurs if the CPORT.CPCFR3[5] CLKBP bit is set to 0. If the CLKBP bit is set to 1 (the reset value), the cdo signal is generated with the negative edge of the csclk signal, and the cdi signal is sampled with the positive edge of the csclk signal.

28.2.3.3.1 Tag Phase

The portion of the audio frame where the csync signal is high, shifted one csclk forward. It is the duration of time slot 0. The time slot 0 has only 16 bits. Each bit of time slot 0 conveys a valid tag for its corresponding time slot within the current audio frame (1 → valid data). The tag bits are set/read through the CPORT.CPTVSLL and CPORT.CPTVSLH (C-port interface valid time slots low and high bytes, respectively) registers.

28.2.3.3.2 Data Phase

The portion of the audio frame where the csync signal is low, shifted one csclk forward. Each audio frame has 12 outgoing and 12 incoming data streams with 20-bit sample resolution (with only 16 bits for valid data). Outgoing and incoming data streams are MSB-justified (MSB first). Unused LSBs are stuffed with zeros.

28.2.3.3.3 Command/Status Address and Data Slot Management

The command/status address on slot 1 is sent to the external code device/status through the CPORT.CPTTADR (C-port interface address) register. The operation (read for status, and write for command) is also set in this register.

The command/status data words on slot 2 are exchanged through the CPORT.CPTDATL and CPORT.CPTDATH (C-port interface data register low and high bytes, respectively) registers. Data written to these registers is sent to the external codec upon a command write operation; the status received from the external codec can be read in these registers after a status read operation.

The transmit and receive FIFO status is given in the CPORT.STATUS register.

28.2.3.3.4 Audio Samples Exchange

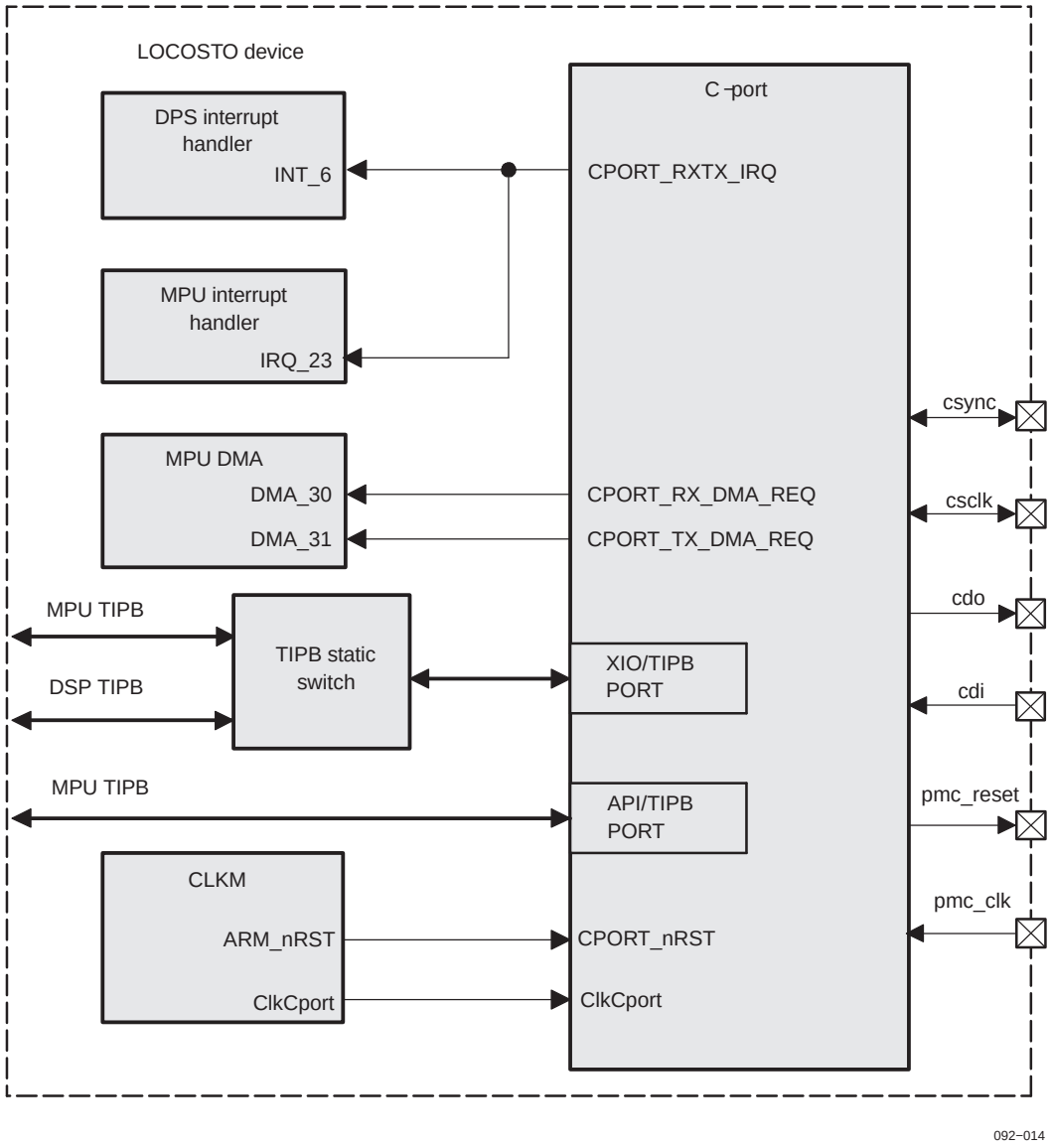
The C-port FIFO only supports stereo data. Only slot 3 (left sample) and slot 4 (right sample) contain valid audio samples. Only the 16 MSBs of each audio sample are taken into account; the input sample LSBs are discarded, and the output sample LSBs are filed with zeros.

28.3 C-port Integration

This section describes the C-port interface integration inside the LOCOSTO device.

Figure 28-14 shows the C-port integration with interrupt handlers, DMA subsystem, clock manager (CLKM), and interconnect.

Figure 28-14. C-port Integration



28.3.1 Clocking, Reset, and Power-Management Scheme

28.3.1.1 Clocks

The C-port module receives its clocks from the CLKM module.

Table 28-5 lists the C-port module clocks.

Table 28-5. C-port Clocks

Type	Name	Source	Frequency	Description
Functional (slave) and interface	ClkCport	CLKM	13 MHz	C-port 13-MHz functional clock (slave mode) and interface clock
Functional (master)	pmc_clk	External to LOCOSTO device	11.2896 MHz (I2S, PCM modes), 12.288 MHz, or 24.576 MHz (AC'97 mode)	C-port pmc_clk functional clock (master mode)

For more details on the CLKM module, see Chapter 6, *Power, Reset, and Clock Management*.

28.3.1.2 Power Management

The ClkCport clock can be gated to reduce power consumption.

The C-port functional clock can be disabled with the CPORT.CNTL_CLK[4] CLK_EN bit. Resetting this bit to 0 disables the ClkCport clock, thus saving power. The reset value of this bit is 1 (that is, the ClkCport clock is enabled by default).

For a description of the CNTL_CLK register, see Chapter 6, *Power, Reset, and Clock Management*.

28.3.1.3 Hardware and Software Reset

The C-port module receives its resets from the CLKM module.

Table 28-6 lists the C-port module resets.

Table 28-6. C-port Resets

Type	Name	Source	Frequency	Description
Hardware	ARM_nRST	CLKM	Active low	MPU hardware reset that resets the whole module
Software	SW_RESET bit	CPORT.CTRL[0] register	Active high	Software reset

The software reset is activated when the CPORT.CTRL[0] SW_RESET bit is set to 1, and all C-port registers are reset. The reset value of this bit is 0.

28.3.2 Hardware Requests

28.3.2.1 DMA Requests

The C-port DMA request mapping is shown in Table 28-7. For more details on DMA, see Chapter 13, *Direct Memory Access*.

Table 28-7. C-port DMA Mapping

DMA Request Name	Mapping	Description
CPORT_RX_DMA_REQ	DMA_30	C-port receive DMA request
CPORT_TX_DMA_REQ	DMA_31	C-port transmit DMA request

28.3.2.2 Interrupt Requests

The interrupt mapping from the C-port interface to the MPU and DSP interrupt handlers is shown in Table 28-8.

Table 28-8. C-port Interrupt Names and MPU/DSP IRQ Mapping

Interrupt Name	Mapping	Description	Domain
CPORT_RXTX_IRQ	IRQ_23	C-port receive interrupt C-port transmit interrupt	MPU
CPORT_RXTX_IRQ	INT_6	C-port receive interrupt C-port transmit interrupt	DSP

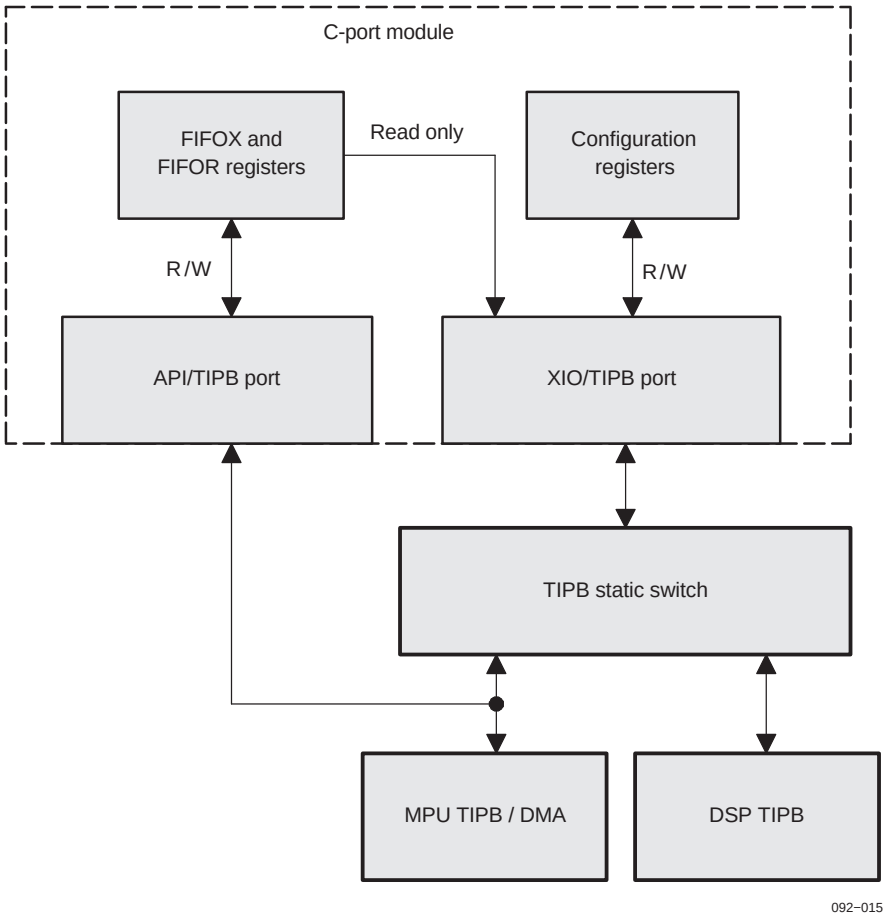
Notes:

1. The user must select the DSP or the MPU to handle these interrupts. If the C-port module is controlled by the MPU (the C-port interrupt is being generated to both the MPU and the DSP), the user must mask the DSP interrupt at the DSP interrupt handler; and vice versa.
2. INT_6 connected to the DSP interrupt handler is shared between the MCSI and the C-port. It is also the MCSI interrupt for DAI mode. For more details, see Section 12.2.2.1, *Shared Level-Sensitive Interrupts*, in Chapter 12, *Interrupt Handlers*.
3. CPORT_RXTX_IRQ is common for transmit and receive interrupts.

28.3.2.3 Interconnect Interface

The C-port module interconnects to other modules through two ports: API/TIPB and XIO/TIPB. Figure 28-15 shows the access ports in the C-port module.

Figure 28-15. C-port Access Ports



The MPU/DMA can access the C-port module through the following:

- XIO/TIPB port: At 0xFFFFD000, to initialize the configuration registers and read the CPORT.FIFOX and CPORT.FIFOR registers. This port access is shared between the MPU and the DSP through a TIPB static switch.
- API/TIPB port: At 0xFFFFD800, to read the CPORT.FIFOR register and write the CPORT.FIFOX register

The DSP can access only the XIO/TIPB port at 0x7C00:

- FIFO registers only in read access
- Configuration registers in read/write

The C-port XIO/TIPB port interfaces with the internal interconnect through a TIPB static switch. This static switch is accessible through the MPU/DMA or the DSP TIPB. The selection between the DSP and the MPU TIPB is provided by a TIPB static switch with status and configuration registers (see Table 28-9). For a complete description, see Chapter 5, *Interconnect*. By default, the C-port XIO/TIPB port is controlled by the DSP TIPB.

Table 28-9. C-port TIPB Static Switch Physical Addresses

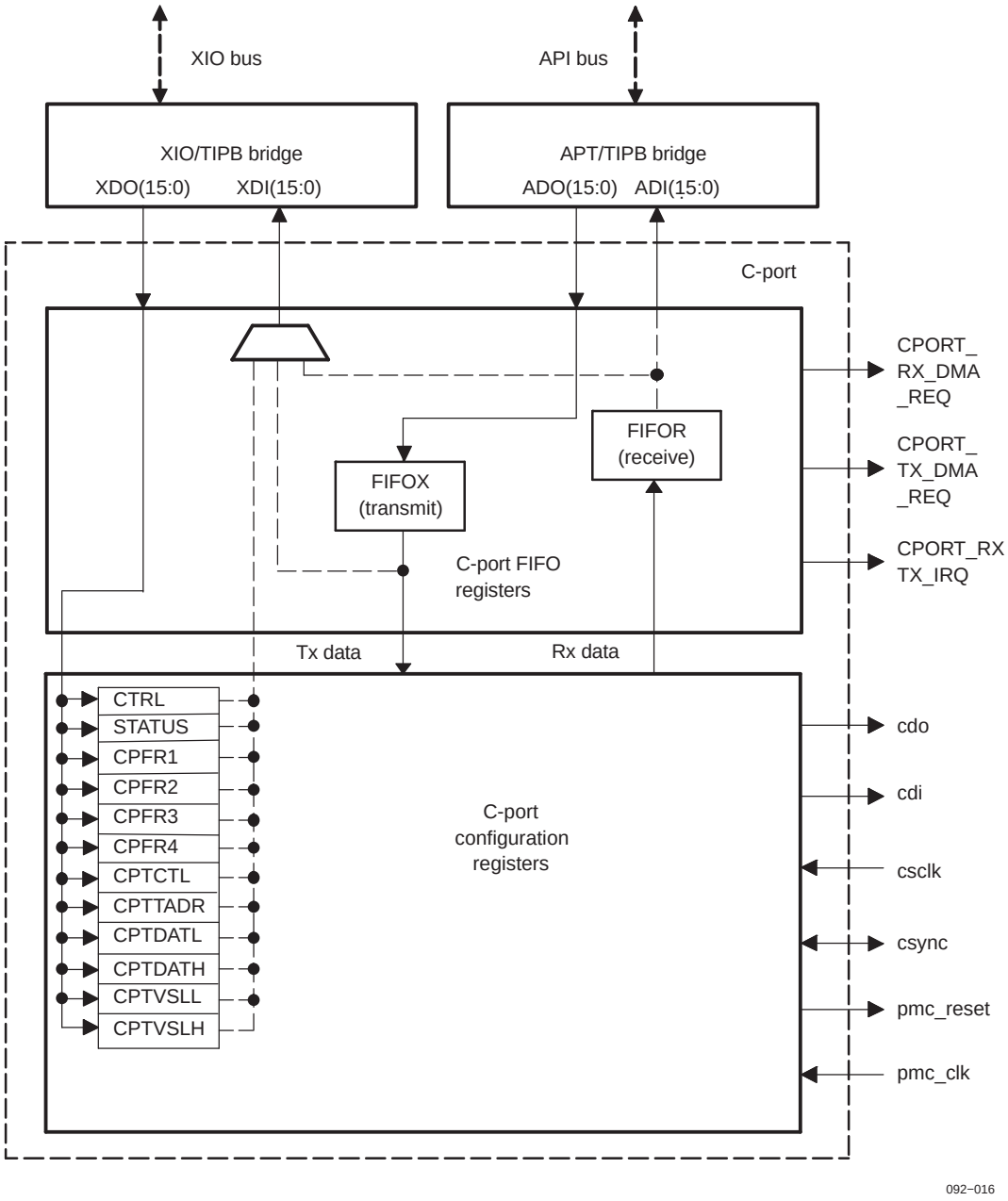
Register Name	Type	Register Width (Bits)	MPU or DSP Physical Address
CPORT_TSW_DSP_CONF	R/W	16	0x7CC0
CPORT_TSW_DSP_STA	R/W	16	0x7CC1
CPORT_TSW_MPU_CONF	R/W	16	0xFFFF8840
CPORT_TSW_MPU_STA	R/W	16	0xFFFF8844

28.4 C-port Functional Description

28.4.1 Block Diagram

Figure 28-16 shows the main hardware blocks of the C-port interface module. The blocks are discussed in detail in the following sections.

Figure 28-16. C-port Block Diagram



28.4.2 TIPB Bus Interface

The C-port module includes two ports: API/TIPB and XIO/TIPB.

28.4.2.1 API/TIPB Port

This port allows the MPU/DMA to read the RX FIFO content (CPORT.FIFOR register) directly and to write in the TX FIFO (CPORT.FIFOX register).

Only these two FIFO registers are accessible through this port.

The MPU can access the C-port configuration registers only through the XIO/TIPB port.

28.4.2.2 XIO/TIPB Port

This port allows both the MPU/DMA and the DSP to access the C-port configuration registers. The selection is performed through a TIPB static switch. By default, the XIO/TIPB port is connected to the DSP.

The MPU/DMA and DSP can also access the two FIFO registers (FIFOX and FIFOR), but only in read-only mode.

28.4.2.3 Audio Data Interface

The audio data transfer can be executed in two ways: one is to use DMA (with the CPORT_TX_DMA_REQ and CPORT_RX_DMA_REQ signals), and the other is to use an interrupt (TIPB interrupts, CPORT_RXTX_IRQ signal). Both methods use the FIFO. DMA or interrupt requests can be masked through the CPORT.CTRL register. For more details, see the programming guide in Section 28.5.1.5.

28.4.3 Clock Selection

The C-port clock configuration differs depending on whether the C-port module is in slave mode or master mode.

The clock selection is programmed with the CPORT.CTRL[1] EXT_MCLK_EN bit.

28.4.3.1 Slave Mode

In slave mode, the C-port functional clock is internally generated.

The 13-MHz ClkCport clock is supplied by the CLKM module.

28.4.3.1.1 Master Mode

In master mode, the serial (csclk) and synchro (csync, running at 44.1 kHz) clocks are generated using a divider of the C-port clock that must be fixed at 11.2896 MHz to achieve the desired frequency. In the LOCOSTO device, the C-port functional clock is fixed at 13 MHz in slave mode by default. To configure the C-port interface in master mode with a slave code, apply to the external input clock pin (pmc_clk) a clock generator at the correct frequency (11.2896 MHz in I2S and PCM modes, and 12.288 MHz or 24.576 MHz in AC'97 mode).

28.4.4 C-port Main Configurations

The serial interface is basically a TDM time slot-based scheme. The serial format is programmed by setting the number of time slots per codec frame and the number of serial clock cycles (or bits) per time slot. The interface in all modes (I2S, PCM, and AC'97) is bidirectional and full-duplex. For some modes, both audio data and command/status data are transferred via the serial interface.

In I2S mode, the serial data I/O signal is delayed one serial clock (csclk) cycle in the reference to the leading edge of the csync signal.

The following are the main C-port configurations:

- Operation mode: AC'97, I2S, or PCM
- Number of time slots per codec frame: 1 to 32
- Number of serial clock cycles per time slot 0: 8, 16, 32, or others

- Number of data bits per audio time slot: 8, 16, 18, 20, 24, or 32
- Number of serial clock cycles for all slots other than slot 0: 8, 16, 18, 20, 24, or 32
- The time slots to be used for command/status address and data
- The serial clock (csclk) frequency in relation to the C-port functional clock (pmc_clk in master mode and ClkCport in slave mode)
- Data delay in the reference to the leading edge of the csync signal: no delay or one cycle delay
- Data serial output state: Tristate or enable
- The csync and cdo signals are generated with the negative/positive edge of the csclk signal; the cdi signal is sampled with the positive/negative edge of the csclk signal.
- Active state of the csync signal: Low or high
- The csync length is equal or not to the time slot 0 length.
- The direction of the C-port clock csclk: Input or output
- The direction of the C-port interface frame sync csync: Input or output
- Time slot format: Time slot 1 for address, and time slot 2 for data or other format

28.4.5 FIFO

Both receive and transmit audio paths integrate a FIFO (RX FIFO and TX FIFO).

The following are the main features of the FIFO:

- 4-word storage capacity (not configurable); a word consists of 16 bits
- Dynamically programmable trigger level: 1 to 4 words (for both RX FIFO and TX FIFO)
- Automatic flow control
- Flag FIFO full, FIFO empty, FIFO almost full, and FIFO almost empty trigger reached

The RX FIFO and the TX FIFO are synchronized with the 13-MHz ClkCport clock.

The generation of the receive or transmit interrupt CPORT_RXTX_IRQ is related to the trigger level of the RX FIFO or the TX FIFO, respectively.

For more information, see the programming guide in Section 28.5.4.

28.4.6 Interrupt Controller

The interrupt for the C-port protocol management is mapped on the MPU and DSP IRQ line. The CPORT_RXTX_IRQ interrupt request is associated with the CPORT.CTRL register.

The main features of this interrupt controller include the following:

- Mask RX IT or TX IT or both by using the CPORT.CTRL register
- Enable and disable receive or transmit interrupts with the CPORT.CPTCTL register
- Generate two interrupt signals (IRQ) to the MPU and DSP and have a status in the CPORT.CPTCTL register

Note: The user must select the DSP or MPU to handle these interrupts. Because the CPORT interrupt is generated to both the MPU and DSP, if the C-port module is controlled by the MPU, the software user must mask the DSP interrupt at the DSP interrupt handler, and vice versa if the module is controlled by the DSP.

The INT_6 interrupt connected to the DSP interrupt handler is shared between the MCSI and the C-port. It is also the MCSI interrupt for DAI mode. For more information, see Section 12.2.2.1, *Shared Level-Sensitive Interrupts*, in Chapter 12.

28.4.7 Register Block

The register block gathers the configuration and status registers accessible in read/write access by the DSP and MPU, but only the FIFO registers are accessible in read/write access by the MPU.

The registers are as follows:

- The general configuration register (CPORT.CTRL, CPORT.CPTCTL, CPORT.CPCFR1, CPORT.CPCFR2, CPORT.CPCFR3, and CPORT.CPCFR4)
- The configuration registers for AC'97 mode (CPORT.CPTTADR, CPORT.CPTDATL, and CPORT.CPTATH)
- The status register (CPORT.STATUS
- The FIFO registers (CPORT.FIFOR and CPORT.FIFOX)

The CPORT.CTRL, CPORT.CPCFR1, CPORT.CPCFR2, CPORT.CPCFR3, and CPORT.CPCFR4 registers are used for operation mode configuration.

These registers can be updated only if the interface is not running (that is, the CPORT.CPTCTL[3] CPEN bit is set to 0).

For a complete description of the C-port module registers, see Section 28.6.2.

28.4.8 Error Reporting

The software must ensure that when the RX FIFO is full (see the status flag in the CPORT.STATUS register), no more audio data is received by the C-port interface. This is done to avoid a receive overflow on the RX FIFO.

28.5 C-port Programming Guide

28.5.1 C-port Setup for Typical Configuration

28.5.1.1 C-port Pin Multiplexing Choice

The pmc_reset, pmc_clk, and cdi signals are multiplexed with the gpio_1, gpio_2, and gpio_4 pins, respectively. To select these signals, program the pin multiplexing configuration registers. For more details, see Chapter 18, *Configuration*.

The csync, csclk, and cdo signals are available in the following modes

- Primary mode 0: For direct access on these signals. No software program is required in this case.
- Multiplexed mode: For the csync (input mode only), csclk (input mode only), and cdo signals. To select these signals, program the pin multiplexing configuration registers. For more details, see Chapter 18, *Configuration*.

28.5.1.2 Software Reset

Software reset can be performed on the C-port module by using the CPORT.CTRL[0] SW_RST bit. Setting this bit to 1 resets all C-port registers.

28.5.1.3 Clock Configuration

The CPORT.CTRL[1] EXT_MCLK_EN bit controls the C-port functional clock selection:

- If this bit is set to 0 (reset value), the functional clock is the 13-MHz ClkCport clock.
- If this bit is set to 1, the functional clock uses an external chip clock from the pmc_clk pin.

The external clock pmc_clk input is required only for master mode. In master mode, the C-port uses the external clock as a functional clock and a generator clock for the synchro and serial clocks. The external clock must be a free-running clock.

28.5.1.4 Interrupts Setup

The CPORT_RXTX_IRQ is a common IRQ line for transmit and receive.

This IRQ generation can be masked using the following bits:

- CPORT.CTRL[2] X_INT_MASK bit. Setting this bit to 1 masks the transmit IRQ.
- CPORT.CTRL[3] R_INT_MASK bit. Setting this bit to 1 masks the receive IRQ.

By default, both IRQ lines are masked.

The CPORT_RXTX_IRQ line is connected to both the DSP and MPU interrupt handlers. The software must select the appropriate interrupt handler to manage the C-port module.

The DSP C-port IRQ line (INT_6) is shared between the MCSI and C-port modules. It is also used as the MCSI DAI mode interrupt. For the programming guide on shared IRQs between modules, see Section 12.2.2.1, *Shared Level- Sensitive Interrupts*, in Chapter 12.

28.5.1.5 DMA Setup

The C-port module generates two DMA requests: CPORT_RX_DMA_REQ (receive) and CPORT_TX_DMA_REQ (transmit):

- CPORT_TX_DMA_REQ can be masked with the CPORT.CTRL[10] XDMA_REQ_MASK bit. Setting this bit to 1 masks the transmit DMA request.
- CPORT_RX_DMA_REQ can be masked with the CPORT.CTRL[11] RDMA_REQ_MASK bit. Setting this bit to 1 masks the receive DMA request.

By default, both DMA requests are masked.

For details on the DMA programming guide, see Chapter 13, *Direct Memory Access*.

28.5.1.6 FIFO Setup

The RX and TX FIFO are both four words deep. A word consists of 16 bits.

The value of the threshold for generating a transmit, a receive interrupt, or a DMA request is programmable. If the DMA interface is used, these thresholds must be set according to MPU DMA burst configuration.

- The threshold value of the transmit FIFO (TX FIFO) is fixed by the CPORT.CTRL[6:4] X_THRESHOLD field. The maximum value allowed is 3. The values from 0b100 to 0b111 are reserved. By default, the threshold for the TX FIFO is set to 1.

Note: A threshold set to 1 is equal to a transfer of three samples sent by frame.

The threshold value of the receive FIFO (RX FIFO) is fixed by the CPORT.CTRL[9:7] R_THRESHOLD field. The maximum value allowed is 4. The value 0b000 and values from 0b101 to 0b111 are reserved. By default, the threshold for TX FIFO is set to 1, which is the minimum threshold level allowed for the RX FIFO.

28.5.1.7 C-port Module Enable

The CPORT.CPTCTL[3] CPEN bit is used to enable the C-port interface. By setting this bit to 1, the internal clock is not gated and the C-port interface is running.

The CPORT.CTRL, CPORT.CPCFR1, CPORT.CPCFR2, CPORT.CPCFR3, and CPORT.CPCFR4 configuration registers can be updated only if the interface is not running (that is, the CPORT.CPTCTL[3] CPEN bit is set to 0).

28.5.2 C-port Operational Mode (I2S, PCM, or AC'97)

28.5.2.1 Operation Mode Configuration

The CPORT.CPCFR1[2:0] MODE field is used to select the C-port interface mode: AC'97, I2S, or PCM. Table 28-10 lists the C-port interface configuration.

Table 28-10. Operation Mode Configuration for C-Port

Register	Name	Bit 2	Bit 1	Bit 0	Mode
CPCFR1	MODE	0	0	0	Reserved
		0	0	1	PCM mode
		0	1	0	AC'97 mode
		0	1	1	Reserved
		1	0	0	I2S master mode
		1	0	1	I2S slave mode
		1	1	0	Reserved
		1	1	1	Reserved

28.5.2.2 Number of Time Slots per Codec Frame

The CPORT.CPCFR1[7:3] MTSL field is used to program the number of time slots per audio frame. By default, one time slot is programmed per audio frame (that is, the MTSL field is set to 0b00001).

From 1 to 32 time slots (that is, MTSL is set to 0b11111) can be programmed per audio frame.

28.5.2.3 Number of Serial Clock Cycles for Time Slot 0

The CPORT.CPCFR2[7:6] TSLOL field is used to program the number of serial clock (csclk) cycles for time slot 0. The programmed values can be as follows:

- TSLOL set to 0b01: 8 csclk cycles for time slot 0
- TSLOL set to 0b10: 16 csclk cycles for time slot 0
- TSLOL set to 0b11: 32 csclk cycles for time slot 0
- By default, the TSLOL field is set to 0b00, which is the same csclk cycle number as programmed in the CPORT.CPCFR2[2:0] TSLL field. For more information, see Section 28.5.2.5, *Number of Serial Clock Cycles for all Slots Other Than Slot 0*.

28.5.2.4 Number of Data Bits per Audio Time Slot

The CPORT.CPCFR2[5:3] BPTSL field is used to program the number of data bits per audio time slot. lists the programmed values.

Table 28-11. Configuration of Data Bits Number per Audio Data Time Slot

Register	Name	Bit 5	Bit 4	Bit 5	Data Bits per Time Slot
CPCFR2	BPTSL	0	0	0	8
		0	0	1	16
		0	1	0	18
		0	1	1	20
		1	0	0	24
		1	0	1	32
		1	1	0	Reserved
		1	1	1	Reserved

By default, the BPTSL field is set to 0b001, which is 16 data bits per audio time slot.

28.5.2.5 Number of Serial Clock Cycles for all Slots Other Than Slot 0

The CPORT.CPCFR2[2:0] TSLL field is used to program the number of serial clock (csclk) cycles for all time slots except time slot 0. Table 28-12 lists the programmed values.

Table 28-12. Configuration of Serial Clock Cycles Number for all Slots other than Slot 0

Register	Name	Bit 2	Bit 1	Bit 0	csclk Cycles for all Time Slots Except Time Slot 0
CPCFR2	TSLL	0	0	0	8
		0	0	1	16
		0	1	0	18
		0	1	1	20
		1	0	0	24
		1	0	1	32
		1	1	0	Reserved
		1	1	1	Reserved

By default, the TSLL field is set to 0b001, which is 16 csclk cycles for all time slots except time slot 0.

28.5.2.6 Data Delay

The CPORT.CPCFR3[7] DDLY bit is used to program one csclk cycle delay on the serial data output (cdo) and input (cdi) signals in the reference to the rising edge of the csync signal.

Setting the DDLY bit to 1 (reset value) programs one cycle delay. By setting this bit to 0, no delay is fixed.

28.5.2.7 Tristate Data Serial Output

The CPORT.CPCFR3[6] TRSEN bit is used to drive the data serial output (cdo signal) state to a tristate when the audio frame is not valid.

Setting the TRSEN bit to 1 enables the tristate. By setting this bit to 0 (reset value), the data serial output is still in active state when the audio frame is not valid, and the cdo signal is driven to output.

CAUTION

The software user must perform a software reset (by using the CPORT.CTRL[0] SW_RESET bit) before modifying the value of the CPORT.CPCFR3[6] TRSEN bit.

28.5.2.8 Frame Sync, Serial Data Output, and Serial Data Input Signals

The CPORT.CPCFR3[5] CLKBP bit is used to program the C-port frame sync (csync), the serial data output (cdo), and the serial data input (cdi) for the serial port clock (csclk). Table 28-13 lists the programming values.

Table 28-13. Generation of CPT_SYNC, CPT_SDO, and CPT_SDI Signals

Register	Name	Bit 5	Generation of csync signal at	Generation of cdo signal at	Sample of cdi signal at
CPCFR3	CLKBP	0	Positive edge of csclk signal	Positive edge of csclk signal	Negative edge of csclk signal
		1	Negative edge of csclk signal	Negative edge of csclk signal	Positive edge of csclk signal

By default, the CLKBP bit is set to 1.

28.5.2.9 Active State of the C-port Interface Frame Sync (csync) Output Signal

The CPORT.CPCFR3[4] CSYNCP bit is used to program the polarity of the csync output signal.

When this bit is set to 1 (the reset value), the csync signal is active high; when it is set to 0, csync is active low.

In I2S mode, when the csync signal is active high, the left sample of the stereo path is sent first (time slot 0), and then the right sample is sent. When the csync signal is active low, the right sample is sent first, and then the left sample is sent.

28.5.2.10 Length of the C-port Interface Frame Sync (csync) Output Signal

The CPORT.CPCFR3[3] CSYNCL bit is used to program the length of the csync signal active state.

When this bit is set to 1 (the reset value), the length of the csync signal is the same as the length of time slot 0. When this bit is set to 0, the length of csync is fixed by hardware and differs from the length of time slot 0.

28.5.2.11 Direction of the C-port Clock (csclk)

The CPORT.CPCFR3[1] CSCLKD bit is used to program the direction of the C-port serial clock (csclk).

When this bit is set to 1 (the reset value), the csclk signal is set as an input. When this bit is set to 0, the csclk signal is set as an output.

28.5.2.12 Direction of the C-port Interface Frame Sync (csync)

The CPORT.CPCFR3[0] CSYNCD bit is used to program the direction of the C-port frame synchronization (csync).

When this bit is set to 1 (the reset value), the csync signal is set as an input. When this bit is set to 0, the csync signal is set as an output.

28.5.2.13 Time Slot Format

The CPORT.CPCFR4[7:4] ATSL field is used to program the format of the time slot: time slot 1 for address, and time slot 2 for data and others.

In I2S or PCM mode, the ATSL field must be left at its reset value (0b0000).

In AC'97 mode, the ATSL field must be set to 0001b, which results in time slot 1 being used for the address, and time slot 2 being used for the data.

28.5.2.14 Divide by B Value

The CPORT.CPCFR4[2:0] DIVB field is used to program the csclk signal divider.

By setting this field to 0b000 (the reset value), the divider is disabled.

By setting this field from 0b001 to 0b111, the software user enables from divide-by-2 to divide-by-8.

28.5.3 C-port Operational Mode (AC'97 Mode Only)

28.5.3.1 AC'97 Configuration Registers Setting

The AC'97 registers are configured according to the following procedure:

1. Program the CPORT.CPCFR1 register with 0x62 value:
   a. Select AC'97 mode: CPORT.CPCFR1[2:0] MODE = 0b010
   b. 13 time slots per audio frame: CPORT.CPCFR1[7:3] MTSL = 0b01100
2. Program the CPORT.CPCFR2 register with 0x8B value:
   a. 16 serial clock cycles for time slot 0: CPORT.CPCFR2[7:6] TSOL = 0b10
   b. 16 data bits per audio time slot: CPORT.CPCFR2[5:3] BPTSL = 0b001 (only the 16 MSBs are used)
   c. 20 serial clock (csclk) cycles for all time slots except time slot 0: CPORT.CPCFR2[2:0] TSLL = 0b011
3. Program the CPORT.CPCFR3 register with 0x9A value:
   a. One csclk cycle delay of the serial data output and input signals in the reference to the leading edge of the csync signal: CPORT.CPCFR3[7] DDLY = 1
   b. Data serial output is enabled when the audio frame data are not valid: CPORT.CPCFR3[6] TRSEN = 0.

CAUTION

The software user must perform a software reset (by using the CPORT.CTRL[0] SW_RESET bit) before modifying the value of the CPORT.CPCFR3[6] TRSEN bit.

c. The csync signal is generated with the positive edge of the csclk signal. The cdo signal is also generated with the positive edge of the csclk signal, and the cdi signal is sampled with the negative edge of the csclk signal: CPORT.CPCFR3[5] CLKBP = 0.
d. The polarity of the C-port interface frame sync (csync) output signal is active high: CPORT.CPCFR3[4] CSYNCP = 1.
e. The length of the C-port interface frame sync (csync) output signal is the same number of csclk cycles as time slot 0: CPORT.CPCFR3[3] CSYNCL = 1.
f. The direction of the C-port interface serial clock (csclk) signal is an input of the LOCOSTO device: CPORT.CPCFR3[1] CSCLKD = 1.
g. The direction of the C-port interface frame sync (csync) signal is an output from the device: CPORT.CPCFR3[0] CSYNCD = 0.

4. Program the CPORT.CPCFR4 register with 0x10 value:
   a. Time slot 1 is used for the address, and time slot 2 is used for the data: CPORT.CPCFR4[7:4] ASTL = 0b0001.
   b. No divider on master clock (pmc_clk) to generate csclk transmit clock (pmc_clk = 12.288 MHz, 256 bits per frame: frame synchro frequency = 48 kHz). CPORT.CPCFR4[2:0] DIVB = 0b000. If the 24.576-MHz external master clock is used, the divide-by-2 must be enabled with CPORT.CPCFR4[2:0] DIVB = 0b001.

28.5.3.2 Address, Data, and Valid Time Slot Registers

To send audio data to the AC'97 codec, both of the following must occur:

- Time slots 3 and 4 must be validated with the CPORT.CPTVSLL and CPORT.CPTVSLH registers.
- The CPORT.CPTVSLH[7] VF bit must be set to 1 to indicate that the current audio frame contains at least one time slot with valid data.

To send control data, the dedicated time slots do not have to be validated, because the C-port hardware module performs it automatically when the CPORT.CPTVSLL[0] CDTS bit is set to 1 (that is, the time slot 0 value is processed by hardware).

28.5.4 FIFO Management

28.5.4.1 Trigger Level

The receive FIFO (RX FIFO) and the transmit FIFO (TX FIFO) each have a 4-word storage capacity. A word consists of 16 bits.

If the DMA interface is used, these thresholds must be set according to the MPU DMA burst configuration.

The trigger level for both RX FIFO and TX FIFO is programmable as follows:

1. Program the CPORT.CTRL[9:7] RTHRESHOLD field from 0b001 to 0b100 to fix the threshold value. Other values are reserved.
2. Program the CPORT.CTRL[6:4] XTHRESHOLD field from 0b000 to 0b011 to fix the threshold value. Other values are reserved.

Note: An XTHRESHOLD field set to 1 is equal to a transfer of three samples sent by the frame.

The generation of the receive or transmit interrupt CPORT_RXTX_IRQ and of the CPORT_TX_DMA_REQ and CPORT_RX_DMA_REQ DMA requests is related to the trigger level of the RX FIFO or TX FIFO.

28.5.4.2 FIFO Status Flags

The following status flags are available for the software to poll the FIFO status on the RX and TX FIFOs:

- Receive FIFO:
  - The CPORT.STATUS[5] RFIFO_FULL status bit gives a receive FIFO full status.
  - The CPORT.STATUS[4] RFIFO_EMPTY status bit gives a receive FIFO empty status.

  - The CPORT.STATUS[3] RFIFO_ALMOST_FULL status bit gives a receive FIFO almost full status.
- Transmit FIFO:
  - The CPORT.STATUS[2] XFIFO_FULL status bit gives a transmit FIFO full status.
  - The CPORT.STATUS[1] XFIFO_EMPTY status bit gives a transmit FIFO empty status.
  - The CPORT.STATUS[0] XFIFO_ALMOST_FULL status bit gives a transmit FIFO almost full status.

28.5.5 Interrupt Management

28.5.5.1 Enable Interrupts

The CPORT_RXTX_IRQ is a common interrupt line for both transmit and receive data.

- Receive data: Set the CPORT.CPTCTL[6] RXIE bit to 1 to enable the C-port receive data register full interrupt.
- Transmit data: Set the CPORT.CPTCTL[4] TXIE bit to 1 to enable the C-port transmit data register empty interrupt.

28.5.5.2 Interrupt Status Flags

The following status flags are available for the software to poll receive data register full and transmit data register empty events:

- Receive data register full: The CPORT.CPTCTL[7] RXF status bit is set to 1 by hardware when a new data value is received into the receive data register from the codec device. To clear this status bit, write 0.
- Transmit data register empty: The CPORT.CPTCTL[5] TXE status bit is set to 1 by hardware when the data value in the transmit data register is sent to the codec device. To clear this status bit, write 0.

28.5.6 Power Saving

28.5.6.1 Save Clock Gating

To save power, disable the ClkCport clock by setting the CPORT.CNTL_CLK[4] CLK_EN bit to 0. By default, the reset value of this bit is 1 (that is, the ClkCport clock is enabled by default).

Note: This clock must be running for any register read/write access.

28.5.6.2 Disable the C-port Interface Module

To save power, if the C-port interface module is not used, disable the module by setting the CPORT.CPTCTL[3] CPEN bit to 0.

28.6 Register Manual

Table 28-14 lists the instance summary for the C-port registers.

Table 28-14. Instance Summary

Module Name	MPU TIPB		DISP TIPB	
	MPU Base Address	Size	DSP Base Address	Size
C-port registers	0xFFFF D000	2K bytes	0x7CC0	256 bytes
C-port (FIFO registers only)	0xFFFF D800	2K bytes	—	—

28.6.1 C-port Register Mapping Summary

Table 28-15 lists the offset address for the C-port registers, and Table 28-16 lists the offset address for the C-port FIFO registers.

Table 28-15. C-port Registers Offset Address (Base Address: 0xFFFF D000 or 0x7C00)

Register Name	Type	Register Width (Bits)	MPU Offset	DSP Offset
FIFOX	R	16	0x00	0x00
FIFOR	R	16	0x02	0x01
CPCFR1	R/W	16	0x04	0x02
CPCFR2	R/W	16	0x06	0x03
CPCFR3	R/W	16	0x08	0x04
CPCFR4	R/W	16	0x0A	0x05
CPTCTL	R/W	16	0x0C	0x06
CPTADR	R/W	16	0x0E	0x07
CPTDATL	R/W	16	0x10	0x08
CPTDATH	R/W	16	0x12	0x09
CPTVSLL	R/W	16	0x14	0x0A
CPTVSLH	R/W	16	0x16	0x0B
CTRL	R/W	16	0x18	0x0C
STATUS	R	16	0x1A	0x0D

CAUTION

The C-port registers are limited to 16-bit access. 32-bit and 8-bit accesses are forbidden. For further details, see Chapter 5, *Interconnect*.

Table 28-16. C-port FIFO Registers Offset Address (Base Address: 0xFFFF D800)

Register Name	Type	Register Width (Bits)	MPU Offset
FIFOX	R/W	16	0x00
FIFOR	R	16	0x02

CAUTION

The C-port registers are limited to 16-bit access. 32-bit and 8-bit accesses are forbidden. For further details, see Chapter 5, *Interconnect*.

28.6.2 C-port Register Description

Table 28-17 through Table 28-31 describe the individual C-port registers.

Table 28-17. FIFOX (XIO/TIPB Port)

Address Offset	0x00		
Physical Address	MPU: 0xFFFF D000	**Instance**	C-port
	DSP: 0x7C00		
Description	FIFO transmit register (read-only mode)		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFOX															

Bits	Field Name	Description	Type	Reset
15:0	FIFOX	Data to transmit	R	0x0000

Table 28-18. FIFOX (API/TIPB Port)

Address Offset	0x00		
Physical Address	MPU: 0xFFFF D800	**Instance**	C-port
Description	FIFO transmit register (read-write mode)		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFOX															

Bits	Field Name	Description	Type	Reset
15:0	FIFOX	Data to transmit	R	0x0000

Table 28-19. Table 2: FIFOR

Address Offset	MPU: 0x02		
	DSP: 0x01		
Physical Address	MPU : 0xFFFF D002 0xFFFF D802	**Instance**	C-port
	DSP: 0x7C01		
Description	FIFO receive register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFOR															

Bits	Field Name	Description	Type	Reset
15:0	FIFOR	Received data	R	0x0000

Table 28-20. CPCFR1

Address Offset	MPU: 0x04		
	DSP: 0x02		
Physical Address	MPU: 0xFFFF D004	**Instance**	C-port
	DSP: 0x7C02		
Description	C-port interface configuration register 1		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MTSL					MODE		

Bits	Field Name	Description		Type	Reset
15:8	Reserved	Reserved		R	0x000000
7:3	MTSL	The number of time slot bits are set to program the number of time slots per audio frame:		R/W	0x01
		0b00000:	1 time slot per frame		
		0b00001:	2 time slots per frame		
		0b11111:	32 time slots per frame		
2:0	MODE	The mode select bits are set to program the C-port interface.		R/W	0x4
		0b000:	Reserved		
		0b001:	PCM mode		
		0b010:	AC'97 mode		
		0b100:	I2S master mode		
		0b101:	I2S slave mode		
		Others:	Reserved		

Note: The CPORT.CPCFR1 register is used for operation mode configuration and can be updated only if the C-port interface is not running (that is, the CPORT.CPCTL[3] CPEN bit is set to 0).

Table 28-21. CPCFR2

Address Offset	MPU: 0x06		
	DSP: 0x03		
Physical Address	MPU: 0xFFFF D006	**Instance**	C-port
	DSP: 0x7C03		
Description	C-port interface configuration register 2		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TSLOL		BPTSL			TSLL		

Bits	Field Name	Description		Type	Reset
15:8	Reserved	Reserved		R	0x000000
7:6	TSLOL	The time slot 0 length bits are set to program the number of serial clock (csclk) cycles for time slot 0.		R/W	0x0
		0b00:	Same csclk cycles for time slot 0 as the TSLL field (CPORT.CPCFR2[2:0])		
		0b01:	8 csclk cycles for time slot 0		
		0b10:	16 csclk cycles for time slot 0		
		0b11:	32 csclk cycles for time slot 0		
5:3	BPTSL	The data bits per time slot are set by MPU/DSP to program the number of data bits per audio time slot. This value is not used for the secondary communication address and data time slots.		R/W	0x1
		0b000:	8 data bits per time slot		
		0b001:	16 data bits per time slot		
		0b010:	18 data bits per time slot		
		0b011:	20 data bits per time slot		
		0b100:	24 data bits per time slot		
		0b101:	32 data bits per time slot		

Bits	Field Name	Description	Type	Reset
		Others: Reserved		
2:0	TSLL	The time slot length bits are set to program the number of serial clock (csclk) cycles for all time slots except time slot 0. 0b000: 8 csclk cycles for time slot 0b001: 16 csclk cycles for time slot 0b010: 18 csclk cycles for time slot 0b011: 20 csclk cycles for time slot 0b100: 24 csclk cycles for time slot 0b101: 32 csclk cycles for time slot Others: Reserved	R/W	0X1

Note: The CPORT.CPCFR2 register is used for operation mode configuration and can be updated only if the C-port interface is not running (that is, the CPORT.CPCTL[3] CPEN bit is set to 0).

Table 28-22. CPCFR3

Address Offset	MPU: 0x08 DSP: 0x04		
Physical Address	MPU: 0xFFFF D008 DSP: 0x7C04	**Instance**	C-port
Description	C-port interface configuration register 3		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DDLY	TRSEN	CKLB	CSYNCP	CSYNCL	Reserved	CSCLKD	CSYNCD

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	0x000000
7	DDLY	The data delay is set to program a one-csclk cycle delay of the serial data output and input signals in the reference to the leading edge of the csync signal. 0: No delay 1: One csclk cycle delay	R/W	0x1
6	TRSEN	The tristate enable is set to program the hardware to tristate data serial output when the audio frame data are not valid. 0: Data serial output is enabled (tristate disabled) 1: Tristate enabled	R/W	0x0
5	CKLB	The C-port interface frame sync (csync) output signal and the C-port interface serial data output (cdo) signal and the input (cdi) signal. When this bit is set to 1, the csync signal is generated with the negative edge of the C-port interface serial clock (csclk) signal. Also, the cdo signal is generated with the negative edge of the csclk signal, and the cdi signal is sampled with the positive edge of the csclk signal. When this bit is cleared to 0, the csync signal is generated with the positive edge of the csclk signal. Also, the cdo signal is generated with the positive edge of the csclk signal, and the cdi signal is sampled with the negative edge of the csclk signal.	R/W	0x1
4	CSYNCP	The csync polarity bit is set to 1 to program the polarity of the C-port interface frame sync (csync) output signal to be active high. 0: Active low 1: Active high	R/W	0x0

Bits	Field Name	Description	Type	Reset
3	CSYNCL	The csync length bit is set to 1 to program the length of the C-port interface frame sync (csync) output signal to be the same number of csclk cycles as time slot 0.	R/W	0x1
2	Reserved	Reserved	R/W	0x0
1	CSCLKD	The csclk direction bit is set to 1 to program the direction of the C-port interface serial clock (csclk) signal as an input to the LOCOSTO device. When cleared to 0, the direction of the csclk signal is an output from the LOCOSTO device.	R/W	0x1
0	CSYNCD	The csync direction bit is set to 1 to program the direction of the C-port interface frame sync (csync) signal as an input to the LOCOSTO device. When cleared to 0, the direction of the csync signal is an output from the LOCOSTO device.	R/W	0x1

Note: The CPORT.CPCFR3 register is used for operation mode configuration and can be updated only if the C-port interface is not running (that is, the CPORT.CPCTL[3] CPEN bit is set to 0).

CAUTION

The software user must perform a software reset (by using the CPORT.CTRL[0] SW_RESET bit) before modifying the value of the CPORT.CPCFR3[6] TRSEN bit.

Table 28-23. CPCFR4

Address Offset	MPU: 0x0A DSP: 0x05		
Physical Address	MPU: 0xFFFF D00A DSP: 0x7C05	**Instance**	C-port
Description	C-port interface configuration register 4		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ASTL				Reserved	DIVB		

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	0x000000
7:4	ASTL	The command/status address/data time slot bits are set to program the time slots to be used for the secondary communication address and data values. For the AC'97 mode of operation, this value must be set to 0b0001, which results in time slot 1 being used for the address and time slot 2 being used for the data. For the I2S or PCM modes of operation, this value must be left to its reset value (0b0000).	R/W	0x0
3	Reserved	Reserved	R/W	0x0
2:0	DIVB	Divide-by-B value. The divide-by-B control bits are set to program the csclk signal divider. 0b000: Divider is disabled. 0b001: Divide by 2 0b010: Divide by 3	R/W	0x0

Bits	Field Name	Description	Type	Reset
		0b011: Divide by 4		
		0b100: Divide by 5		
		0b101: Divide by 6		
		0b110: Divide by 7		
		0b111: Divide by 8		

Note: The CPORT.CPCFR4 register is used for operation mode configuration and can be updated only if the C-port interface is not running (that is, the CPORT.CPCTL[3] CPEN bit is set to 0).

Table 28-24. CPTCTL

Address Offset	MPU: 0x0C		
	DSP: 0x06		
Physical Address	MPU: 0xFFFF D00C	**Instance**	C-port
	DSP: 0x7C06		
Description	C-port interface control register		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RXF	RXIE	TXE	TXIE	CPEN	Reserved		CRST

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	0x000
7	RXF	The receive data register full (RXF) bit is set to 1 by hardware when a new data value is received into the receive data register from the codec device. This bit is cleared to 0 by hardware when the MPU/DSP writes a 1.	R	0x0
6	RXIE	The receive interrupt enable (RXIE) bit is set to 1 to enable the C-port module to receive data register full interrupt.	R/W	0x0
5	TXE	The transmit data register empty (TXE) bit is set to 1 by hardware when the data value in the transmit data register is sent to the codec device. This bit is cleared to 0 by hardware when the MPU/DSP writes a 1.	R	0x1
4	TXIE	The transmit interrupt enable (TXIE) bit is set to 1 to enable the C-port transmit data register empty interrupt.	R/W	0x0
3	CPEN	The C-port enable bit is used to control the C-port interface. 0: Disabled (internal clock is not gated) 1: Enabled (internal clock is gated)	R/W	0x0
2:1	Reserved	Reserved	R/W	0x0
0	CRST	The codec reset bit is used to control the C-port interface reset (pmc_reset) output signal. When this bit is set to 1, the pmc_reset signal is active high. When this bit is cleared to 0, the pmc_reset signal is active low.	R/W	0x0

Table 28-25. CPTADR

Address Offset	MPU: 0x0E		
	DSP: 0x07		
Physical Address	MPU: 0xFFFF D00E	**Instance**	C-port
	DSP: 0x7C07		
Description	C-port interface address register		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								R_W	A6_A0						

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	0x00
7	R_W	The command/status read/write control bit value is set to program the type of secondary communication transaction to be done. This bit must be set to 1 for a read transaction and cleared to 0 for a write transaction.	R/W	0x0
6:0	A6_A0	The command/status address value is set to program the codec device control/status register address to be accessed during the read or write transaction.	R/W	0x00

Table 28-26. CPTDATL

Address Offset	MPU: 0x10 DSP: 0x08		
Physical Address	MPU: 0xFFFF D010 DSP: 0x7C08	**Instance**	C-port
Description	C-port interface data register (low byte)		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								D7_D0							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	0x00
7:0	D7_D0	The command/status data value is set to be transmitted to the codec device for write transactions. The command/status data value is updated by hardware with the status data received from the codec device for read transactions.	R/W	0x00

Table 28-27. CPTDATH

Address Offset	MPU: 0x12 DSP: 0x09		
Physical Address	MPU: 0xFFFF D012 DSP: 0x7C09	**Instance**	C-port
Description	C-port interface data register (high byte)		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								D15_D8							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	0x00
7:0	D15_D8	The command/status data value is set to be transmitted to the codec device for write transactions. The command/status data value is updated by hardware with the status data received from the codec device for read transactions.	R/W	0x00

Table 28-28. CPTVSLL

Address Offset	MPU: 0x14 DSP: 0x0A		
Physical Address	MPU: 0xFFFF D014 DSP: 0x7C0A	**Instance**	C-port

Table 28-28. CPTVSLL (continued)

Description	C-port interface valid time slots register (low byte)
Type	R/W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VTSL							CDTS

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	0x00
7:1	VTSL	The valid time slot bits are set to 1 to define which time slots in the audio frame contain valid data. Bits 7 to 3 correspond to time slots 8 to 12.	R/W	0x00
0	CDTS	The command data start bit is set to send the command data on the next frame. Write 1 to send command data on the next frame. This bit is automatically cleared by hardware when the command data is sent.	R/W	0x0

Table 28-29. CPTVSLH

Address Offset	MPU: 0x16 DSP: 0x0B		
Physical Address	MPU: 0xFFFF D016 DSP: 0x7C0B	**Instance**	C-port
Description	C-port interface valid time slots register (high byte)		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VF	Reserved		VTSL				

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Reserved	R	0x000000
7	VF	The valid frame bit is set to 1 to indicate that the current audio frame contains at least one time slot with valid data.	R/W	0x0
6:5	Reserved	Reserved	R/W	0x0
4:0	VTSL	The valid frame bit is set to 1 to indicate that the current audio frame contains at least one time slot with valid data. Bits 4 to 0 correspond to time slots 3 to 7.	R/W	0x00

Table 28-30. CTRL

Address Offset	MPU: 0x18 DSP: 0x0C		
Physical Address	MPU: 0xFFFF D018 DSP: 0x7C0C	**Instance**	C-port
Description	C-port interface control register		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RDMA_REQ_MASK	XDMA_REQ_MASK	RTHRESHOLD		XTHRESHOLD				R_INT_MASK	X_INT_MASK	EXT_MCLK_EN	SW_RESET

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Reserved	R	0x00
11	RDMA_REQ_MASK	1 to mask receive DMA request	R/W	0x1
10	XDMA_REQ_MASK	1 to mask transmit DMA request	R/W	0x1
9:7	RTHRESHOLD	Receive FIFO threshold value 0b000: Reserved 0b001: 1 word (minimum) 0b010: 2 words 0b011: 3 words 0b100: 4 words (maximum) Other values: Reserved	R/W	0x1
6:4	XTHRESHOLD	Fix transmit FIFO threshold value 0b000: 1 word (minimum) 0b001: 2 words 0b010: 3 words 0b011: 4 words (maximum) Other values: Reserved	R/W	0x1
3	R_INT_MASK	1 to mask receive interrupt request	R/W	0x1
2	X_INT_MASK	1 to mask transmit interrupt request	R/W	0x1
1	EXT_MCLK_EN	1 to enable external clock input (pmc_clk) required for master mode	R/W	0x0
0	SW_RESET	1 to activate the C-port software reset	R/W	0x0

Note: The CPORT.CTRL register is used for operation mode configuration and can be updated only if the C-port interface is not running (that is, the CPORT.CPCTL[3] CPEN bit is set to 0).

Table 28-31. STATUS

Address Offset	MPU: 0x1A DSP: 0x0D		
Physical Address	MPU: 0xFFFF D01A DSP: 0x7C0D	**Instance**	C-port
Description	C-port interface status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RFIFO_FULL	RFIFO_EMPTY	RFIFO_ALMOST_FULL	XFIFO_FULL	XFIFO_EMPTY	XFIFO_ALMOST_FULL

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Reserved	R	0x000
5	RFIFO_FULL	Receive FIFO full status	R	0x0
4	RFIFO_EMPTY	Receive FIFO empty status	R	0x1
3	RFIFO_ALMOST_FULL	Receive FIFO almost full status	R	0x0

Bits	Field Name	Description	Type	Reset
2	XFIFO_FULL	Transmit FIFO full status	R	0x1
1	XFIFO_EMPTY	Transmit FIFO empty status	R	0x1
0	XFIFO_ALMOST_FULL	Transmit FIFO almost empty status	R	0x1

Light and Buzzer Control

This chapter introduces the light and buzzer control of the LOCOSTO device. The LED pulse generator (LPG) module, the pulse-width light (PWL) module, the pulse-width tone (PWT) module, and the light and buzzer part of the general-purpose input/output (GPIO) module are discussed.

Topic **Page**

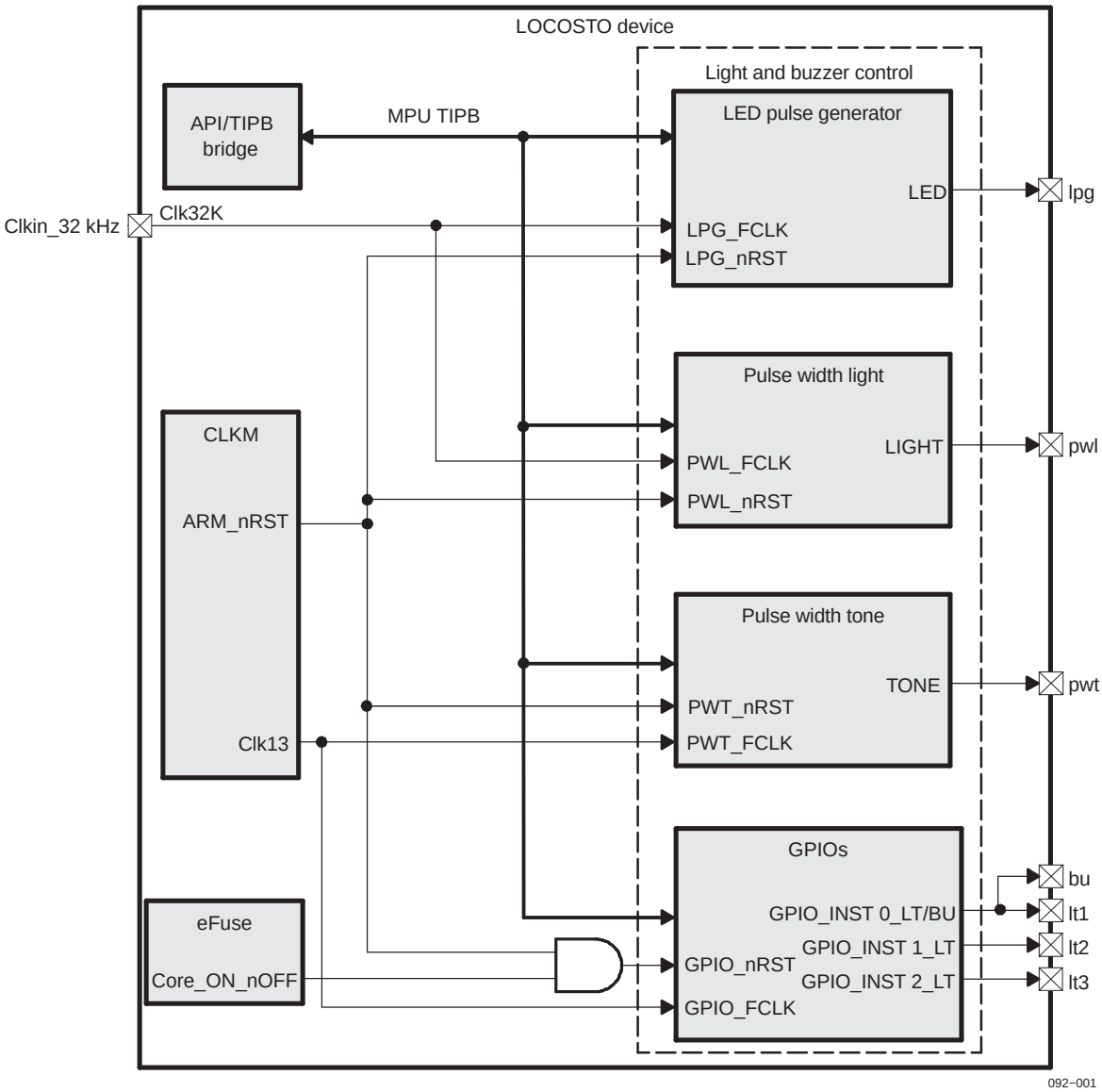
29.1 Light and Buzzer Control Overview

The light and buzzer control has four modules:

- The LED pulse generator (LPG) module: Generates a signal to control a blinking indication LED.
- The pulse-width tone (PWT) module: Generates a modulated frequency signal for an external buzzer.
- The pulse-width light (PWL) module: Controls the power level of an LCD or a keypad backlight.
- The light and buzzer part of five general-purpose inputs/outputs(GPIOs). Some GPIO outputs can be configured as follows:
  - In light mode to control the power level of a backlight
  - In buzzer mode to control the power level and tone of a buzzer

Figure 29-1 is an overview of the light and buzzer control part.

Figure 29-1. Light and Buzzer Module Highlights



29.1.1 LED Pulse Generator

The blinking period is programmable between 125 ms and 3 s. Also, the on duration is programmable, and the LED can be permanently switched on.

The LPG module has the following features:

- Divider generating a 256-Hz clock
- TI peripheral bus (TIPB) control interface
- 8-bit register to control the entire LPG block
- Three blink-frequency control bits (LPG.LCR_REGISTER[2:0] PERCTRL field)
- Three pulse-width control bits (LPG.LCR_REGISTER[5:3] onCTRL field)
- Programmable counter with comparator for the PWM
- Synchronous control logic for the output and the counter

29.1.2 Pulse-Width Tone (PWT)

The PWT output frequency is programmable between 349 Hz and 5276 Hz with 12 half-tone frequencies per octave. The volume of this signal is also programmable.

The PWT module has the following features:

- TIPB control interface
- Four dividers with the coefficients 101/107, 49/55, 50/63, and 80/127 to generate each note particularity
- Four dividers with the coefficient 1/2 and a mux to select the octave
- 6-bit register to control tone frequency
- 6-bit register to control tone volume
- 2-bit register for testing and CLK_EN bit GCR_REGISTER[0]
- 5-bit counter and comparator to create volume pulse
- Divider 1/154 to obtain the final frequency

29.1.3 Pulse-Width Light (PWL)

The PWL voltage-level control technique, based on a 4096-bit random sequence, decreases the spectral power at the modulator harmonic frequencies. The logic operates from the 32-kHz reference clock to be independent of the system activity mode.

29.1.4 General-Purpose Input/Output

The GPIO module contains a PWM configurable in two exclusive modes:

- In light mode, the PWM is configured to control the spectral power at the modulator harmonic frequencies.
- In buzzer mode, the PWM is configured to control the power level, whereas a timer controls constant frequency tones.

29.2 Light and Buzzer Module Environment

A typical application of the LPG, PWL, and GPIO modules in light mode is to control an LCD illumination. Another typical application of these modules is to control an LED.

A typical application of the PWL module and the GPIO module in light mode is to control the backlight of an LCD device:

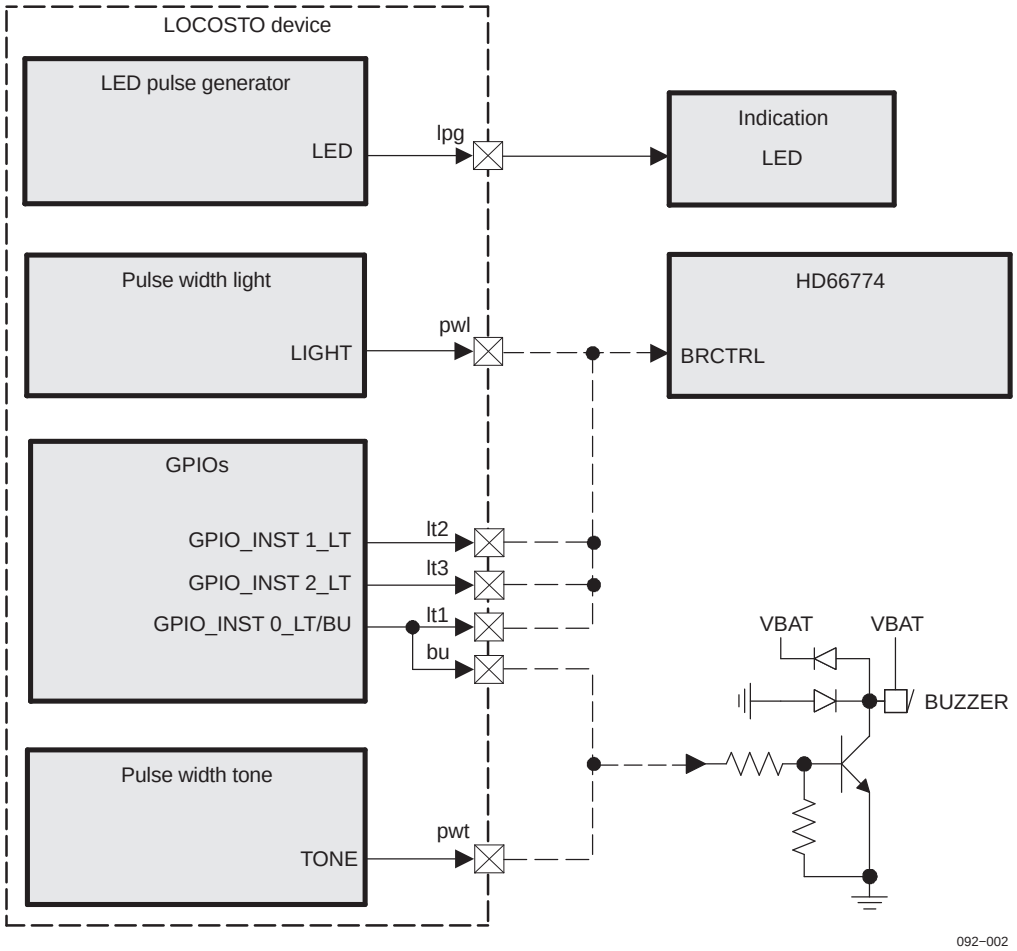
- HD66774: A color TFT LCD driver from RENESAS
- HD66772: A color TFT LCD source from RENESAS
- LPH8754-2: An active-matrix TFT LCD module from Phillips

Only the connector interface for the backlight control is described for these devices.

The typical application of the PWT module and the GPIO module in buzzer mode is to control the audio volume and the tone of a buzzer.

Figure 29-2 shows the typical application for the light and buzzer control modules.

Figure 29-2. Typical Light and Buzzer Application



The lt or pwl signal controls the brightness of the illumination. For the same frequency, the variation of the brightness is controlled by the variation of the duty cycle.

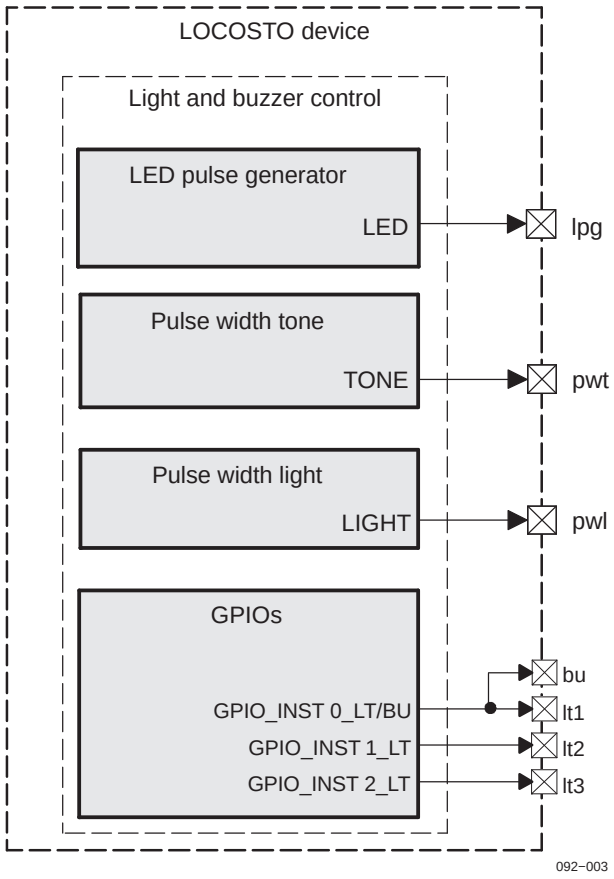
The audio volume of the buzzer is also controlled by the variation of the duty cycle, but the frequency of the pwt or bu signal can be modified to create different tones.

The lpg signal commands an indication LED. The blinking frequency and the length of time the LED remains on are controlled independently.

29.2.1 Light and Buzzer Pins

Figure 29-3 shows the interface signal of the light and buzzer modules.

Figure 29-3. Light and Buzzer Module Interface Signals



29.2.2 Light and Buzzer Module Input/Output

Table 29-1 lists the I/O of the light and buzzer modules.

Table 29-1. I/O Description

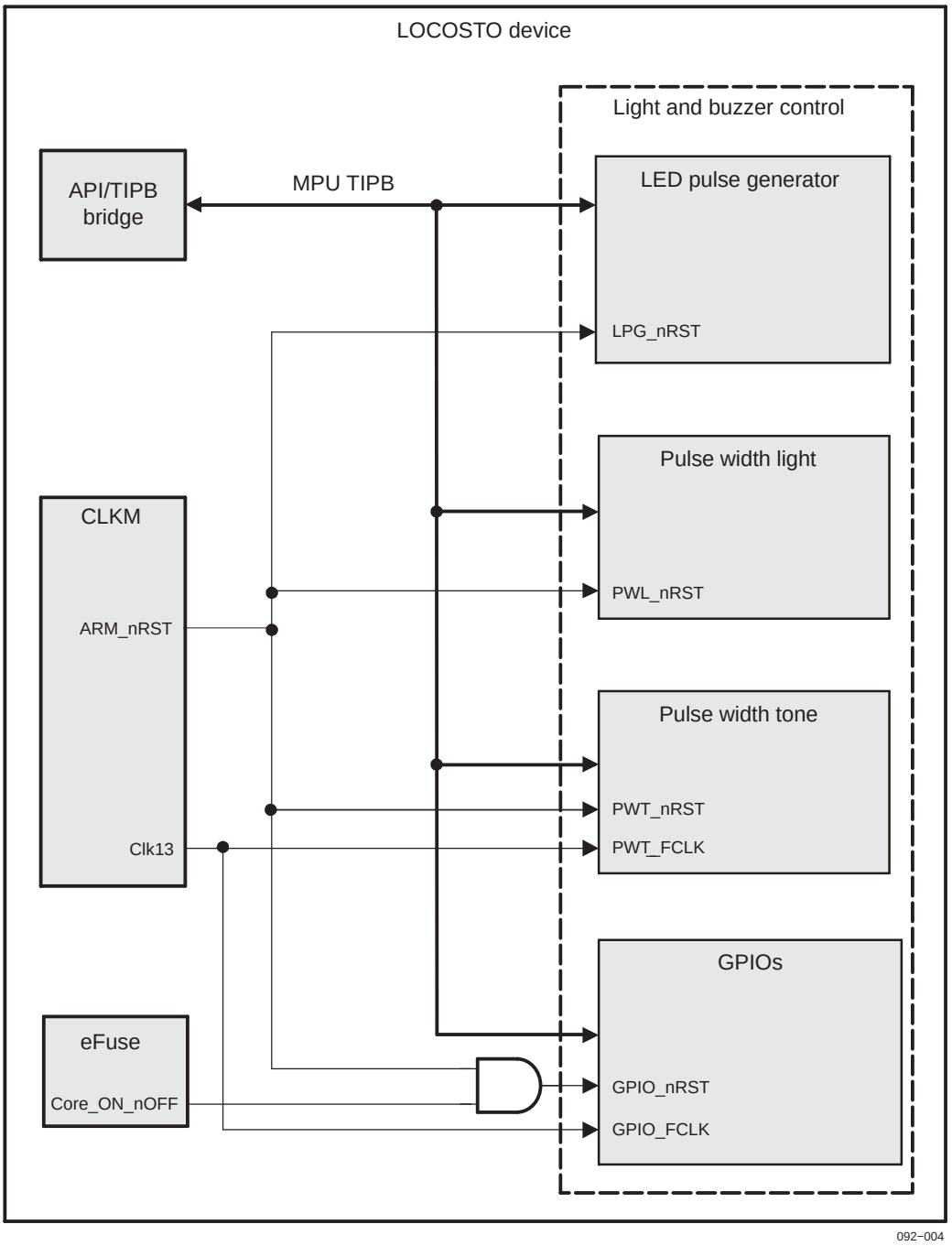
Signal Names	I/O[(1)]	Description	Reset Value
lpg	O	Control LED pulse signal	0
pwt	O	Control tone buzzer signal	0
pwt	O	Control pseudo-noise PWM light signal	0
lt1	O	Control PWM light signal 1	0
lt2	O	Control PWM light signal 2	0
lt3	O	Control PWM light signal 3	0
bu	O	Control PWM buzzer signal	0

(1) I = Input, O = Output

29.3 Light and Buzzer Module Integration

Figure 29-4 shows the module integration and specifies the hardware connections.

Figure 29-4. Light and Buzzer Module Integration



29.3.1 Clocking, Reset, and Power-Management Scheme

29.3.1.1 Clocks

29.3.1.1.1 Module Clocks

Table 29-2 lists the clock domains of the light and buzzer control modules.

Table 29-2. Clock Description

Type	Name	Source	Frequency	Description
Functional	LPG_FCLK PWL_FCLK	Clk32K	32 kHz	The 32-kHz clock comes from the Triton Lite device IC. It is directly distributed to the modules.
Functional	PWT_FCLK GPIO_FCLK	Clk13	13 MHz	The 13-MHz clock is generated by the DRP and managed by the clock manager module.
Interface	LPG_ICLK PWT_ICLK PWL_ICLK GPIO_ICLK	TIPB	52 MHz	Clock fed by the TIPB

For more detail, see Chapter 6, *Power, Reset, and Clock Management*.

29.3.1.1.2 Power Management

- PWT: The PWT input clock comes from the 13-MHz CLKM clock. This clock is stopped in deep-sleep mode to reduce power consumption. When the PWT module is not used, the PWT_FCLK clock can be switched off to reduce power consumption in any mode.
- LPG: The LPG input clock comes from the 32-kHz crystal oscillator because it must work even if the system is in deep-sleep mode. The internal clock of the LPG runs with 256 Hz; therefore, power consumption by this block can be neglected. The LPG_FCLK clock, however, should be switched off if the LPG is not used.
- PWL: The PWL input clock comes from the 32-kHz crystal oscillator. This oscillator is not stopped in deep-sleep mode. When the PWL module is not used, the PWL_FCLK clock can be switched off to reduce power consumption.

For more information, see Chapter 6, *Power, Reset, and Clock Management*. For GPIO module power-management details, see Chapter 27, *General-Purpose Interface.*

29.3.1.2 Resets

29.3.1.2.1 Hardware Reset

Table 29-3 lists the hardware input resets.

Table 29-3. Hardware Reset Domain

Name	Source	Domain
LPG_nRST PWT_nRST PWL_nRST GPIO_nRST	ARM_nRST (CLKM)	All module registers are cleared and all outputs are set to low level asynchronously.
GPIO_nRST	Core_on_nOFF	The GPIO registers are cleared and the outputs are set to low level asynchronously.

For more information, see Chapter 6, *Power, Reset, and Clock Management*.

29.3.1.2.2 Software Reset

Table 29-4 lists the software input resets.

Table 29-4. Software Reset Description

Bit Name	Register Name	Description
LPGRES	LPG.LCR_REGISTER[6]	LPG counter reset The LPG.LCR_REGISTER register is not reset by a software reset.

For more information, see Chapter 6, *Power, Reset, and Clock Management*.

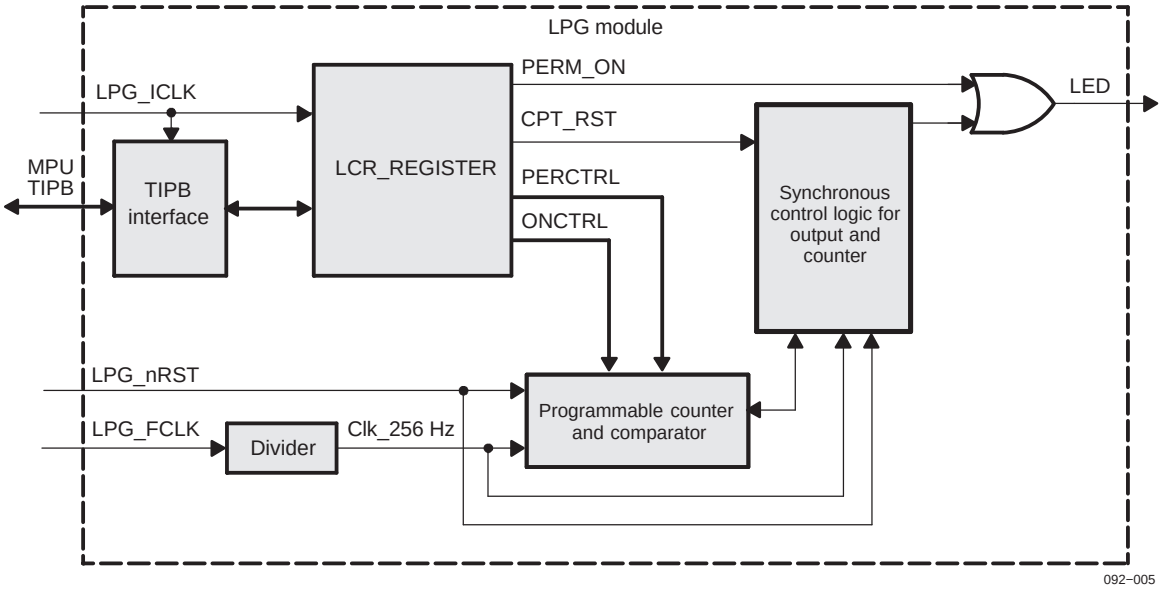
29.4 Light and Buzzer Functional Description

29.4.1 LPG Module

29.4.1.1 Block Diagram

Figure 29-5 is a block diagram of the LPG module.

Figure 29-5. LPG Block Diagram

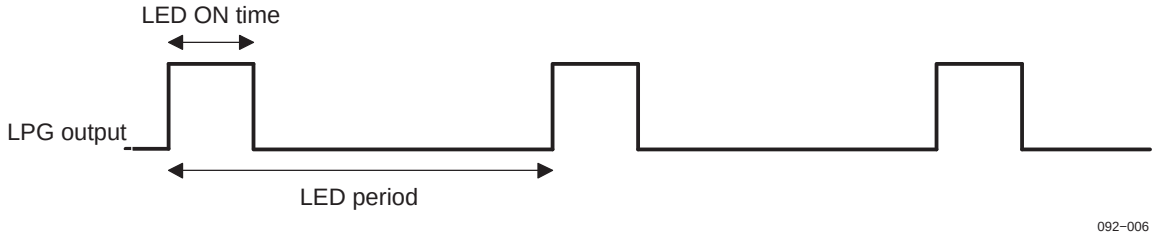


The Clk_256 Hz signal is a 256-Hz clock obtained by a division of the Clk32K LPG_FCLK.

29.4.1.2 LPG Description

Figure 29-6 is a chronogram of the LPG output signal.

Figure 29-6. LPG Output Signal Chronogram



29.4.1.2.1 Software Reset

Software reset is performed by the LPGRES bit LPG.LCR_REGISTER[6].

By setting the LPGRES bit to 0, the user resets the entire PWM circuit (but not the LPG.LCR_REGISTER register) and switches off the LED.

29.4.1.2.2 LED Blinking-Period Configuration

The LED blinking period is programmed with the PERCTRL field LPG.LCR_REGISTER[2:0].

Table 29-5 lists the PERCTRL field values.

Table 29-5. LED Blinking Period

PERCTRL[2:0]	LED Period	Number of Clk_256Hz Cycles
000	125 ms	32
001	250 ms	64
010	500 ms	128
011	1 s	256
100	1.5 s	384
101	2 s	512
110	2.5 s	640
111	3 s	768

29.4.1.2.3 LED On Time Configuration

The length of time the LED remains on is determined with the onCTRL field LPG.LCR_REGISTER[5:3].

Table 29-6 lists the onCTRL field values.

Table 29-6. LED On Time

onCTRL[2:0]	LED On Time	Number of Clk_256Hz Cycles
000	3.889 ms	1
001	7.789 ms	2
010	15.59 ms	4
011	31.39 ms	8
100	46.59 ms	12
101	62.59 ms	16
110	78.39 ms	20
111	93.59 ms	24

The LED can be forced permanently on with the PERM_on bit LPG.LCR_REGISTER[7]. By setting the PERM_on bit to 1, the user switches on the LED permanently and independently of the PWM circuit.

29.4.1.2.4 Power Management

To save power, the LPG_FCLK clock can be disabled by using the CLK_EN bit LPG.PMR_REGISTER[0].

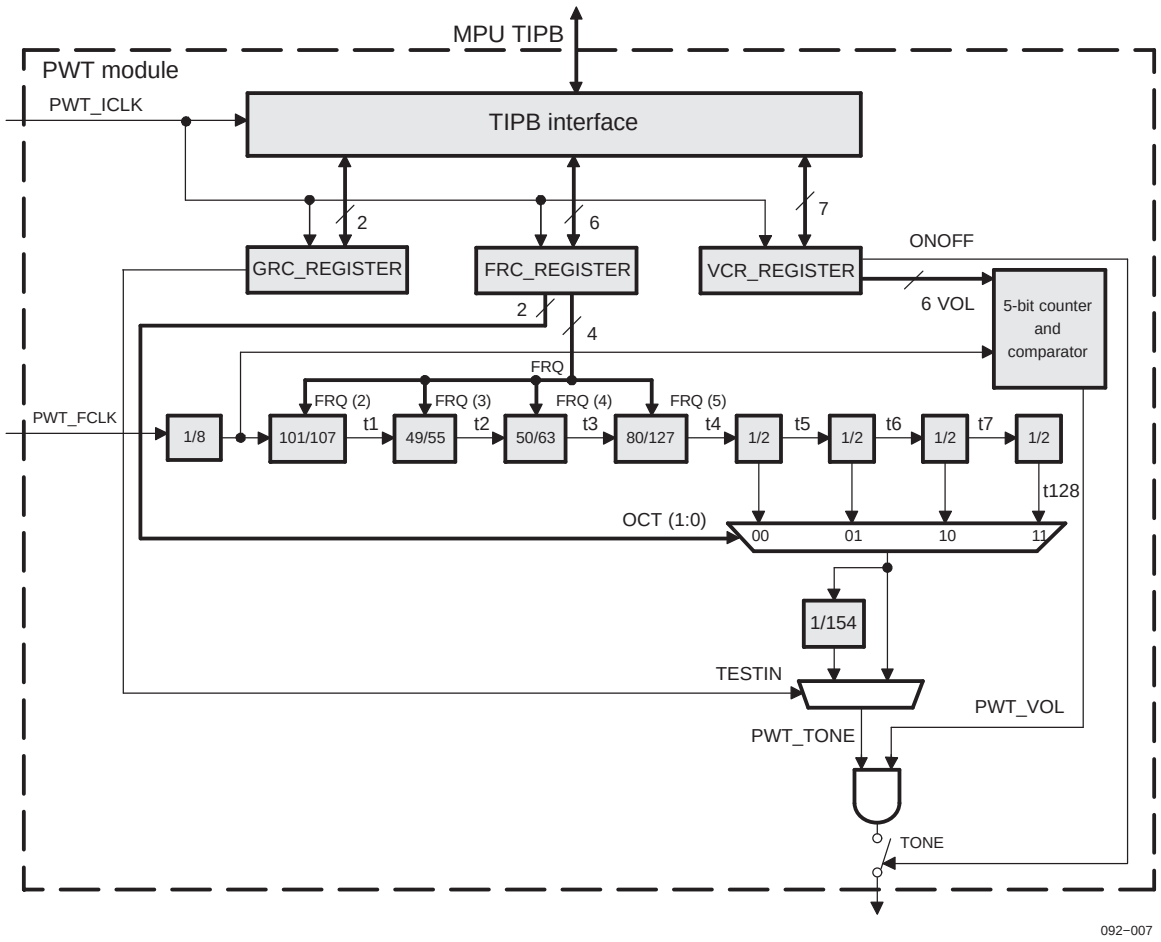
- Setting the CLK_EN bit to 0 disables the LPG functional clock (its default reset value).
- Setting the CLK_EN bit to 1 enables the LPG functional clock.

29.4.2 PWT Module

29.4.2.1 Block Diagram

Figure 29-7 is a block diagram of the PWT module.

Figure 29-7. PWT Block Diagram



29.4.2.2 PWT Description

29.4.2.2.1 Buzzer Frequency

To obtain the required frequencies, the PWT clock is divided in a special way.

Four frequency dividers with the coefficients 101/107, 49/55, 50/63, and 80/127 are connected in a series and can be enabled with the four frequency selection bits in the PWT frequency control register (PWT.FCR_ REGISTER[5:2] FRQ field). If a divider is not enabled, the clock passes through the divider without any change so that different frequencies can be produced. Then, a multiplexer configured by the PWT.FCR_REGISTER[1:0] OCT field selects another clock divider: 2, 4, 8, or 16. The frequency is always halved (this unit is called an octave), thus causing the PWT to have a range of 4 octaves.

The multiplexed output clock is divided by 154 to get the required frequencies on the tone output. The 12 frequencies in an octave, and the octave, are programmable through the FRQ and the OCT bit fields of the PWT frequency control register. Forty-eight frequencies can be programmed, subdivided into 4 octaves with 12 frequencies. The four frequency dividers with the complex coefficients 101/107, 49/55, 50/63, and 80/127 work with the fade-out principle. To get the divider 101/107 from a periodic pulse, 6 pulses every 107 pulses are faded out. Over a long time, the resulting frequency is 101/107. The resulting signal has two different periods, which differ by one period from the original signal. Because of this difference, the resulting signal has jitter. To minimize this jitter, the divider works with high frequencies that result in short periods producing low jitter (see Table 29-7).

Table 29-7. Buzzer Frequencies(1)

PWT.FCR_ REGISTER [5:2] ; [1:0] FREQ; OCT	Buzzer Frequency (Hz)	Tone Note	PWT.FCR_ REGISTER [5:2] ; [1:0] FREQ; OCT	Buzzer Frequency (Hz)	Tone Note
0000 00	5276	e^5	0000 10	1319	e^3
0001 00	4980	dis^5	0001 10	1245	dis^3
0010 00	4700	d^5	0010 10	1175	d^3
0011 00	4437	cis^5	0011 10	1109	cis^3
0100 00	4187	c^5	0100 10	1047	c^3
0101 00	3952	h^4	0101 10	988	h^2
0110 00	3730	ais^4	0110 10	933	ais^2
0111 00	3521	a^4	0111 10	880	a^2
1000 00	3323	gis^4	1000 10	831	gis^2
1001 00	3137	g^4	1001 10	784	g^2
1010 00	2961	fis^4	1010 10	740	fis^2
1011 00	2795	f^4	1011 10	699	f^2
0000 01	2638	e^4	0000 11	659	e^2
0001 01	2490	dis^4	0001 11	623	dis^2
0010 01	2350	d^4	0010 11	588	d^2
0011 01	2218	cis^4	0011 11	555	cis^2
0100 01	2094	c^4	0100 11	523	c^2
0101 01	1976	h^3	0101 11	494	h^1
0110 01	1865	ais^3	0110 11	466	ais^1
0111 01	1761	a^3	0111 11	440	a^1
1000 01	1662	gis^3	1000 11	415	gis^1
1001 01	1569	g^3	1001 11	392	g^1
1010 01	1480	fis^3	1010 11	370	fis^1
1011 01	1397	f^3	1011 11	349	f^1
			11XX XX	Not allowed	-

(1) The frequencies shown in this table are calculated with a 13-MHz input clock.

29.4.2.2.2 Buzzer Volume

Buzzer volume is programmable through the PWT.VCR_REGISTER[6:1] VOL field. The volume of the buzzer/loudspeaker is directly related to the 6-bit value (the higher the 6-bit value, the higher the buzzer/loudspeaker volume). To perform this programming, a 6-bit binary counter is clocked with the PWT clock and rolls over to 0h after reaching its terminal value (0x3F). The counter value is compared to the 6-bit value programmed in the PWT.VCR_REGISTER. If the count value is less than or equal to VOL, the output has a high level; otherwise, the output has a low level, as follows:

- ≤ VOL ≤ 63
- Output signal high level period = (VOL+1).PWT_FCLK
- Output signal low level period = (63_UnicodeEncodeError_VOL).PWT_FCLK

Table 29-8 lists the PWT.VCR_REGISTER register values.

Table 29-8. Buzzer Volume

PWT.VCR_REGISTER [6:1]; [1] VOL; onOFF	Buzzer/Loudspeaker Volume
111111 1	Loud
000000 1	Quiet
XXXXXX 0	Off

29.4.2.2.3 Power Management

To save power, the PWT_FCLK clock can be disabled by using the CLK_EN bit PWT.GCR_REGISTER [0].

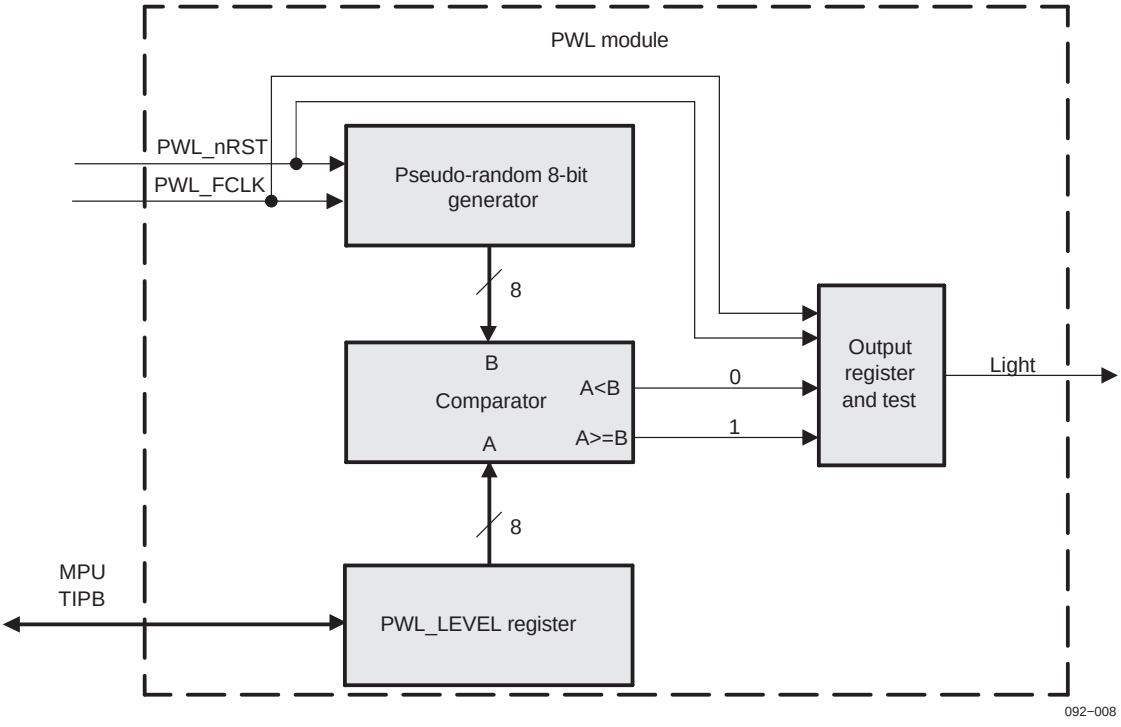
- Setting the CLK_EN bit to 0 (its default reset value) disables the PWT functional clock.
- Setting the CLK_EN bit to 1 enables the PWT functional clock.

29.4.3 PWL Module

29.4.3.1 Block Diagram

Figure 29-8 is a block diagram of the PWL module.

Figure 29-8. PWL Block Diagram



29.4.3.2 PWL Description

29.4.3.2.1 Backlight Level

The blacklight-level module has a pseudo-random 8-bit data generator and a programmable threshold comparator.

- The pseudo-random 8-bit data generator is built using an LFSR. It generates a white normal-law random value between 1 and 255.

  The LFSR polynomial generator is $P(x) = x^7 + x^3 + x^2 + x + 1$.
- The comparator generates the following:
  - 0 if the random value is greater than or equal to the programmable threshold
  - 1 if the random value is less than the programmable threshold

Because the random sequence is normal, the comparator generates a sequence whose mean value is proportional to the comparator threshold.

The backlight level is programmable through the PWL.PWL_LEVEL[7:0] PWL_LEVEL field (the higher the 8-bit value, the higher the backlight level).

29.4.3.2.2 Power Management

To save power, the PWL_FCLK clock can be disabled by using the CLK_ENABLE bit PWL.PWL_CTRL[0].

Setting the CLK_ ENABLE bit to 0 (its default reset value) disables the PWL functional clock.

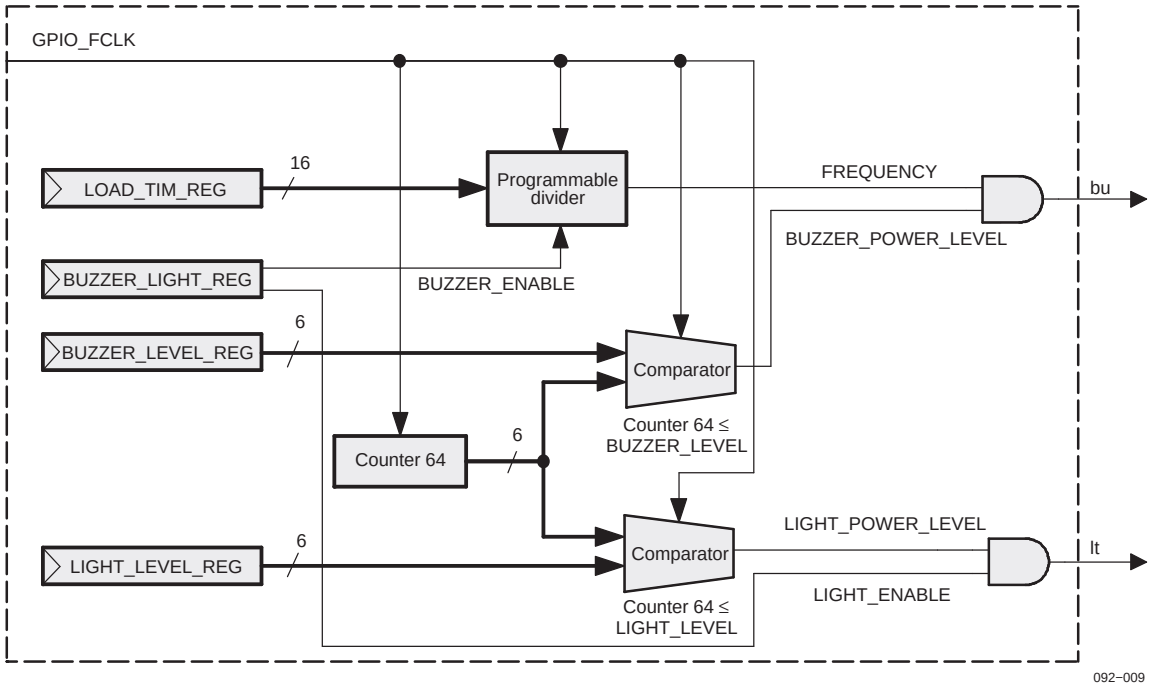
Setting the CLK_ ENABLE bit to 1 enables the PWL functional clock.

29.4.4 Light and Buzzer Part of GPIO Module

29.4.4.1 Block Diagram

Figure 29-9 is a block diagram of the light and buzzer part of the GPIO module.

Figure 29-9. Light and Buzzer Part of GPIO Module Block Diagram



(1) One PWM resource controls the light and buzzer features of each GPIO instance; therefore, only one PWM output (BU or LT) can run at a time.

29.4.4.2 Light and Buzzer Part of GPIO Module Description

29.4.4.2.1 Light and Buzzer Power Level

To control the power level of the light and the buzzer, a basic pulse-width modulation is implemented in the GPIO module.

Two memory-mapped registers (GPIO.LIGHT_LEVEL_REG and GPIO.BUZZER_LEVEL_REG) are used to define power levels. These registers are 6 bits wide and are programmable up to 64 levels.

For human sound perception, use the logarithmic scale to find the correct sound volume steps. The following formula calculates the level with the number of the sound level desired:

Power level = 63 * log [(10:Total_number_of_level_desired) * Sound_level_No]

Table 29-9 lists an example in which the user needs seven sound levels.

Table 29-9. Sound Level

Sound Level No.	Power Level
1	10
2	29
3	40
4	48

Table 29-9. Sound Level (continued)

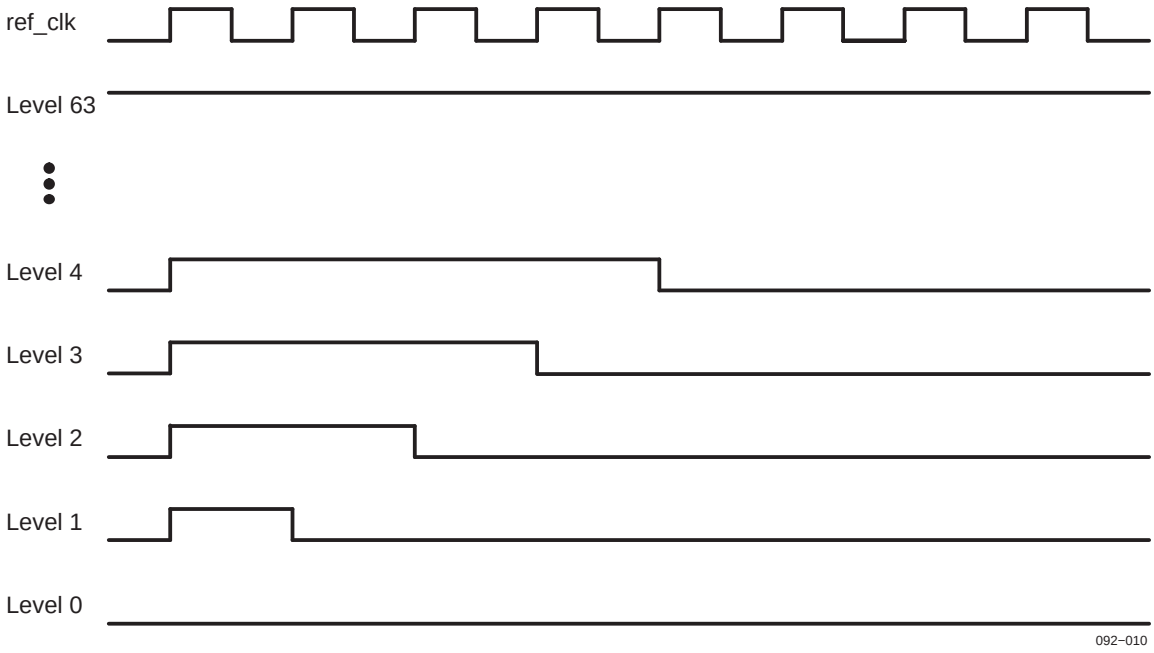
Sound Level No.	Power Level
5	54
6	59
7	63

This is an example of the formula used for sound level number 2:

Power level No2 = 63 * log [(10:7) * 2] = 28.72

Figure 29-10 is the chronogram of the power levels.

Figure 29-10. Power Level Chronogram



29.4.4.2.2 Buzzer Tones

An internal signal (see FREQUENCY in Figure 29-9) can control the frequency of the buzzer; it is functionally ANDed with the buzzer power level.

The value of the load register (LOAD_TIM_REG) can be calculated with the tone frequency as follows:

Fclk =13 MHz

LOAD_TIM_REG = Fclk / (Ftone * 512) - 1

Ftone = Fclk / ((LOAD_TIM_REG+1) * 512)

Table 29-10 lists the tone frequencies.

Table 29-10. Tone Frequencies

LOAD_TIM_REG	Ftone
1	12.70 kHz
2	8.46 kHz
21	1.15 kHz
43	580 Hz
88	290 Hz

Table 29-10. Tone Frequencies (continued)

LOAD_TIM_REG	Ftone
127	200 Hz
255	100 Hz

29.5 Light and Buzzer Programming Model

29.5.1 GPIO

29.5.1.1 Programming Precautions

CAUTION

The BUZZER bit GPIO.BUZZER_LIGHT_REG[0] must be set to 0 to load a new value in the following:

- LOAD_TIM_REG field GPIO.LOAD_TIM_REG[15:0]
- BUZZER_LEVEL_REG field GPIO.BUZZER_LEVEL_REG[5:0]

The LIGHT bit GPIO.BUZZER_LIGHT_REG[1] must be set to 0 to load a new value in the LIGHT_LEVEL_REG field GPIO.LIGHT_ LEVEL_REG[1].

29.5.1.2 Typical Configuration for Buzzer Mode

Before configuring the buzzer registers, the BUZZER bit must be set to 0.

To configure and start a buzzer signal, perform the following steps:

1. Set the GPIO.LOAD_TIM_REG register to the desired value (see Section 29.4.4.2.2, *Buzzer Tones*).
2. Set the GPIO.BUZZER_LEVEL_REG register to the desired value (see Section 29.4.4.2.1, *Light and Buzzer Power Level*).
3. If the clock is not enabled, enable it by setting the GPIO.GPIO_CNTL_REG[5] CLK_ENABLE bit to 1.
4. Start the signal by setting the BUZZER_LIGHT_REG[0] BUZZER bit to 1.

To change the tone of a running buzzer, perform the following steps:

1. Stop the signal by setting the BUZZER bit to 0.
2. Change the GPIO.LOAD_TIM_REG register value to obtain the new tone.
3. Restart the signal by setting the BUZZER bit to 1.

To change the audio level of a running buzzer signal, perform the following steps:

1. Stop the signal by setting the BUZZER bit to 0.
2. Change the GPIO.BUZZER_LEVEL_REG register value to obtain the new audio level.
3. Restart the signal by setting the BUZZER bit to 1.

29.5.1.3 Typical Configuration for Light Mode

Before configuring the light registers, the BUZZER_LIGHT_REG[1] LIGHT bit must be set to 0.

To configure and start a light signal, perform the following steps:

1. Set the GPIO.LIGHT_LEVEL_REG register to the desired value.
2. If the clock is not enabled, enable it by setting the GPIO.GPIO_CNTL_ REG[5] CLK_ENABLE bit to 1.
3. Start the signal by setting the LIGHT bit to 1.

To change the light level of a running light signal, perform the following steps:

1. Stop the signal by setting the LIGHT bit to 0.
2. Change the GPIO.LIGHT_LEVEL_REG register value to obtain the new power level.
3. Restart the signal by setting the LIGHT bit to 1.

For more information about the GPIO programming model, see Chapter 27, *General-Purpose Interface.*

For more information about the GPIO pin mode configuration, see Chapter 18, *Configuration*.

29.6 Light and Buzzer Register Manual

Table 29-11 lists an instance summary for the light and buzzer control modules.

Table 29-11. Instance Summary for MPU

Module Name	Base Address	Size
LPG	0xFFFE 7800	2K bytes
PWT	0xFFFE 8800	2K bytes
PWT	0xFFFE 8000	2K bytes
GPIO	0xFFFE 4800	2K bytes
GPIO1	0xFFFE 5000	2K bytes
GPIO2	0xFFFE 5800	2K bytes

There are three GPIO instances in the LOCOSTO device to obtain 48 I/Os. Table 29-12 lists the five GPIO outputs that can be used in light or buzzer mode.

Table 29-12. Light and Buzzer GPIO Instances

GPIO Output	Instance Used	Mode	Pinout Name
GPIO_INST0_LT/BU(1)	GPIO	Light mode	lt1
GPIO_INST1_LT	GPIO1	Light mode	lt2
GPIO_INST2_LT	GPIO2	Light mode	lt3
GPIO_INST0_LT/BU	GPIO	Buzzer mode	bu

(1) The GPIO_INST0_LT/BU output provides the lt1 and bu pinout outputs, depending on the multiplexing configuration. These outputs are exclusive during the same period.

29.6.1 Light and Buzzer Module Register Mapping Summary

Table 29-13 through Table 29-16 list the light and buzzer modules registers.

Table 29-13. LPG Register Offset Address

Register Name	Register Name	Register Name	MPU Offset
LPG.LCR_REGISTER	R/W	8	0x00
LPG.PMR_REGISTER	R/W	8	0x01

Table 29-14. PWT Register Offset Address

Register Name	Register Name	Register Name	MPU Offset
PWT.FCR_REGISTER	R/W	8	0x00
PWT.VCR_REGISTER	R/W	8	0x01
PWT.GCR_REGISTER	R/W	8	0x02

Table 29-15. PWL Register Offset Address

Register Name	Register Name	Register Name	MPU Offset
PWL.PWL_LEVEL	R/W	8	0x00
PWL.PWL_CTRL	R/W	8	0x01

Table 29-16. Light and Buzzer GPIO Part Register Offset Address

Register Name	Register Name	Register Name	MPU Offset
GPIO.LOAD_TIM_REG	R/W	8	0x08
GPIO.BUZZER_LIGHT_REG	R/W	16	0x0E
GPIO.LIGHT_LEVEL_REG	R/W	16	0x10
GPIO.BUZZER_LEVEL_REG	R/W	16	0x12

Note: Only light and buzzer-related registers are listed here. For more information, see Chapter 27, *General-Purpose Interface.*

29.6.2 Register Descriptions

29.6.2.1 LPG Register Descriptions

Table 29-17 and Table 29-20 describe the individual LPG registers.

Table 29-17. LCR_REGISTER

Address Offset	0x00		
Physical Address	MPU: 0xFFFE 7800	**Instance**	LPG
Description	LPG power-management register		
Type	R/W		

7	6	5	4	3	2	1	0
PERMon	LPGRES	onCTRL			PERCTRL		

Bits	Field Name	Description	Type	Reset
7	PERMon	0x0: No action 0x1: Permanent light on	R/W	0
6	LPGRES(1)	0x0: LPG counter is in reset mode. 0x1: LPG counter is in functional mode.	R/W	0
5;3	onCTRL	LED-on time parameter. See Table 29-19.	R/W	000
2;0	PERCTRL	LED blinking-period parameter. See Table 29-18.	R/W	000

(1) The LPG.LCR_REGISTER register is not reset by a software reset performed by the LPGRES bit.

Table 29-18 lists the PERCTRL field values.

Table 29-18. LED Blinking Period

PERCTRL[2:0]	LED Period	Number of Clk_256Hz Cycles
000	125 ms	32
001	250 ms	64
010	500 ms	128
011	1 s	256
100	1.5 s	384
101	2 s	512
110	2.5 s	640
111	3 s	768

Table 29-19 lists the onCTRL field values.

Table 29-19. LED On Time

onCTRL[2:0]	LED On Time	Number of Clk_256Hz Cycles
000	3.889 ms	1
001	7.789 ms	2
010	15.59 ms	4
011	31.39 ms	8
100	46.59 ms	12
101	62.59 ms	16
110	78.39 ms	20
111	93.59 ms	24

Table 29-20. PMR_REGISTER

Address Offset	0x01		
Physical Address	MPU: 0xFFFE 7801	**Instance**	LPG
Description	LPG power management register		
Type	R/W		

7	6	5	4	3	2	1	0
RESERVED							CLK_EN

Bits	Field Name	Description	Type	Reset
7:1	RESERVED	Read returns 0. Write has no effect on this field.	R	0000000
0	CLK_EN	Functional clock (LPG_FCLK) enable:	R/W	0
		0x0: Read returns 0. Write has no effect on this field.		
		0x1: Clock enabled		

29.6.2.2 PWT Register Descriptions

Table 29-21, Table 29-23, and Table 29-25 describe the individual PWT registers.

Table 29-21. FCR_REGISTER

Address Offset	0x00		
Physical Address	MPU: 0xFFFE 8800	**Instance**	PWT
Description	PWT frequency control register		
	This register controls the frequency of the output tone signal.		
Type	R/W		

7	6	5	4	3	2	1	0
RESERVED			FRQ			OCT	

Bits	Field Name	Description	Type	Reset
7:6	RESERVED	Read returns 0. Write has no effect on this field.	R	00
5:2	FRQ	Frequency selection (12 frequencies)	R/W	0000
		Resynchronized writing, asynchronous reading		
1:0	OCT	Octave selection	R/W	00
		Resynchronized writing, asynchronous reading		

Table 29-22 lists the possible configurations of the PWT.FCR_REGISTER register.

Table 29-22. Buzzer Frequencies[(1)]

PWT.FCR_ REGISTER [5:2] ; [1:0] FREQ; OCT	Buzzer Frequency (Hz)	Tone Note	PWT.FCR_ REGISTER [5:2] ; [1:0] FREQ; OCT	Buzzer Frequency (Hz)	Tone Note
0000 00	5276	e^5	0000 10	1319	e^3
0001 00	4980	dis^5	0001 10	1245	dis^3
0010 00	4700	d^5	0010 10	1175	d^3
0011 00	4437	cis^5	0011 10	1109	cis^3
0100 00	4187	c^5	0100 10	1047	c^3
0101 00	3952	h^4	0101 10	988	h^2
0110 00	3730	ais^4	0110 10	933	ais^2
0111 00	3521	a^4	0111 10	880	a^2
1000 00	3323	gis^4	1000 10	831	gis^2
1001 00	3137	g^4	1001 10	784	g^2
1010 00	2961	fis^4	1010 10	740	fis^2
1011 00	2795	f^4	1011 10	699	f^2
0000 01	2638	e^4	0000 11	659	e^2
0001 01	2490	dis^4	0001 11	623	dis^2
0010 01	2350	d^4	0010 11	588	d^2
0011 01	2218	cis^4	0011 11	555	cis^2
0100 01	2094	c^4	0100 11	523	c^2
0101 01	1976	h^3	0101 11	494	h^1
0110 01	1865	ais^3	0110 11	466	ais^1
0111 01	1761	a^3	0111 11	440	a^1
1000 01	1662	gis^3	1000 11	415	gis^1
1001 01	1569	g^3	1001 11	392	g^1
1010 01	1480	fis^3	1010 11	370	fis^1
1011 01	1397	f^3	1011 11	349	f^1
			11XX XX	Not allowed	-

(1) The frequencies shown in this table are calculated with a 13-MHz input clock.

Table 29-23. VCR_REGISTER

Address Offset	0x01		
Physical Address	MPU: 0xFFFE 8801	**Instance**	PWT
Description	PWT volume-control register		
	This register controls the volume of the output tone signal.		
Type	R/W		

7	6	5	4	3	2	1	0
RESERVED	VOL						onOFF

Bits	Field Name	Description	Type	Reset
7	RESERVED	Read returns 0. Write has no effect on this field.	R	0
6:1	VOL	Volume selection Resynchronized writing, asynchronous reading	R/W	000000
0	onOFF	Switch on/off tone 0x0: Off 0x1: On Resynchronized writing, asynchronous reading	R/W	0

Table 29-24 lists examples of PWT.VCR_REGISTER configurations.

Table 29-24. Buzzer Volume

PWT.VCR_REGISTER [6:1]; [1] VOL; ONOFF	Buzzer/Loudspeaker Volume
111111 1	Loud
000000 1	Quiet
XXXXXX 0	Off

Table 29-25. GCR_REGISTER

Address Offset	0x02		
Physical Address	MPU: 0xFFFE 8802	**Instance**	PWT
Description	PWT general control register This register controls the PWT module.		
Type	R/W		

7	6	5	4	3	2	1	0
RESERVED						TESTIN	CLK_EN

Bits	Field Name	Description	Type	Reset
7:2	RESERVED	Read returns 0. Write has no effect on this field.	R	000000
1	TESTIN	Divider 1/154 switched on/off 0x0: On 0x1: Off Asynchronous writing and reading	R/W	0
0	CLK_EN	PWT_FCLK clock enable 0x0: Clock is disabled. 0x1: Clock is enabled. Asynchronous writing and reading	R/W	0

CAUTION

To access the PWT.FCR_REGISTER and PWT.VCR_REGISTER registers, set the CLK_EN bit in the GCR register to 1. Any attempt to access these registers with the CLK_EN bit set to 0 may lead to unpredictable results.

29.6.2.3 PWL Register Descriptions

Table 29-26 and Table 29-27 describe the individual PWL registers.

Table 29-26. PWL_LEVEL

Address Offset	0x00		
Physical Address	MPU: 0xFFFE 8000	**Instance**	PWL
Description	This register controls the power level of the output light signal.		
Type	R/W		

7	6	5	4	3	2	1	0
PWL_LEVEL							

Bits	Field Name	Description	Type	Reset
7:0	PWL_LEVEL	Defines the mean value of the PWL output signal. 0 leads to a continuous 0 output. 255 leads to an almost continuous 1 output: 255/256 cycles in high level.	R/W	0x00

Table 29-27. PWL_CTRL

Address Offset	0x01		
Physical Address	MPU: 0xFFFE 8001	**Instance**	PWL
Description	This register controls the PWL module.		
Type	R/W		

7	6	5	4	3	2	1	0
RESERVED							CLK_ENABLE

Bits	Field Name	Description	Type	Reset
7:1	RESERVED	Read returns 0. Write has no effect on this field.	R	0000000
0	CLK_ENABLE	PWL_FCLK clock enable 0x0: Clock is disabled. 0x1: Clock is enabled. Asynchronous writing and reading	R/W	0

Initialization

This chapter introduces the LOCOSTO device initialization.

Topic **Page**

30.1 Initialization Overview

30.1.1 Terminology

This chapter provides an overview of the requirements for initializing the LOCOSTO device from power-on to operating system (OS) load and applications running.

- Bootstrap: The initial software launched by the read-only memory (ROM) code during the memory booting phase
- Certificate: The data block plus the trusted signature of the data block; used during the memory and peripheral booting for the high-security (HS) devices
- eFuse: A one-time programmable memory location usually set at the factory
- Flash loader: Downloaded software launched by the ROM code in preflash that programs an image into external NOR flash memories
- GP device: General-purpose (GP) device in which the hash of the manufacturer public key (MPK) is zero; thus, the security features are disabled.
- HS device: High-security device in which the hash of the MPK is non-zero; thus, the security features are enabled.
- Initial software: Software that is executed by any ROM code mechanism (memory booting or peripheral booting); a generic term for bootstrap and downloaded software.
- Memory booting: A ROM code mechanism that allows software residing in external memory (NOR flash memory) to be executed
- MPK, hash(MPK), hash(ManPubKey): Manufacturer public key; the hash of a particular MPK is efused at the manufacturing phase and used to authenticate both memory and peripheral booting.
- Peripheral booting: A ROM code mechanism that consists of polling selected interfaces, downloading, and executing an initial software (in this case, the flash loader) in internal random access memory (RAM)
- Pre-flashing: A specific case of peripheral booting in which the ROM code mechanism is used as part of a flashing process to program a NOR flash memory
- Protected applications: An application signed by the authenticated author and meant to be executed in a secure execution environment
- RAM loader: Part of the ROM code responsible for the flash loader download communication sequence
- ROM code: The on-chip software in secure ROM that implements booting and security features
- Secure mode: The protected execution mode of the device; some peripherals (such as internal secure RAM and secure ROM) are visible and are allowed to be accessed only in this mode.
- Secure RAM: Writable memory inside the chip that can be accessed only when the device is in secure mode
- Secure ROM: Read-only memory inside the chip programmed during the chip manufacturing phase; accessed only when the chip is in secure mode.
- Table of Contents (TOC): Set of data blocks containing basic information (such as code size and entry point of the firmware); required during peripheral booting for a GP device.

30.1.2 Initialization Process

This section describes the overall initialization process, including hardware- and software-related steps, a general overview of the ROM code operational requirements, and behavior expectations.

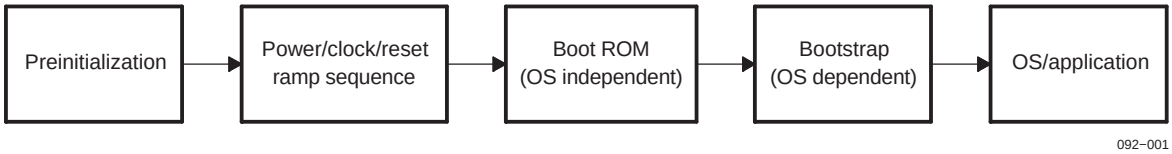
Figure 30-1 shows an overview of the initialization process and its steps. The LOCOSTO device initialization consists of the following steps:

- Pre-initialization
- Ramp sequence for power, clocks, and resets
- ROM code

- Bootstrape
- OS start

The following sections describe all of these steps, up to OS and applications running.

Figure 30-1. Initialization Process for the LOCOSTO Device



Although the first two steps in the initialization process are hardware-oriented, they require an understanding of the process involved to configure system interface pins (balls on the device) that have software-configurable functions.

This configuration, which is an essential part of chip configuration, is application dependent. This chapter refers to those pins and the associated configuration registers that are vital for proper device initialization.

For more information on the pin multiplexing configuration and the pins out of the LOCOSTO device, see Chapter 18, *Configuration*.

30.2 Pre-Initialization

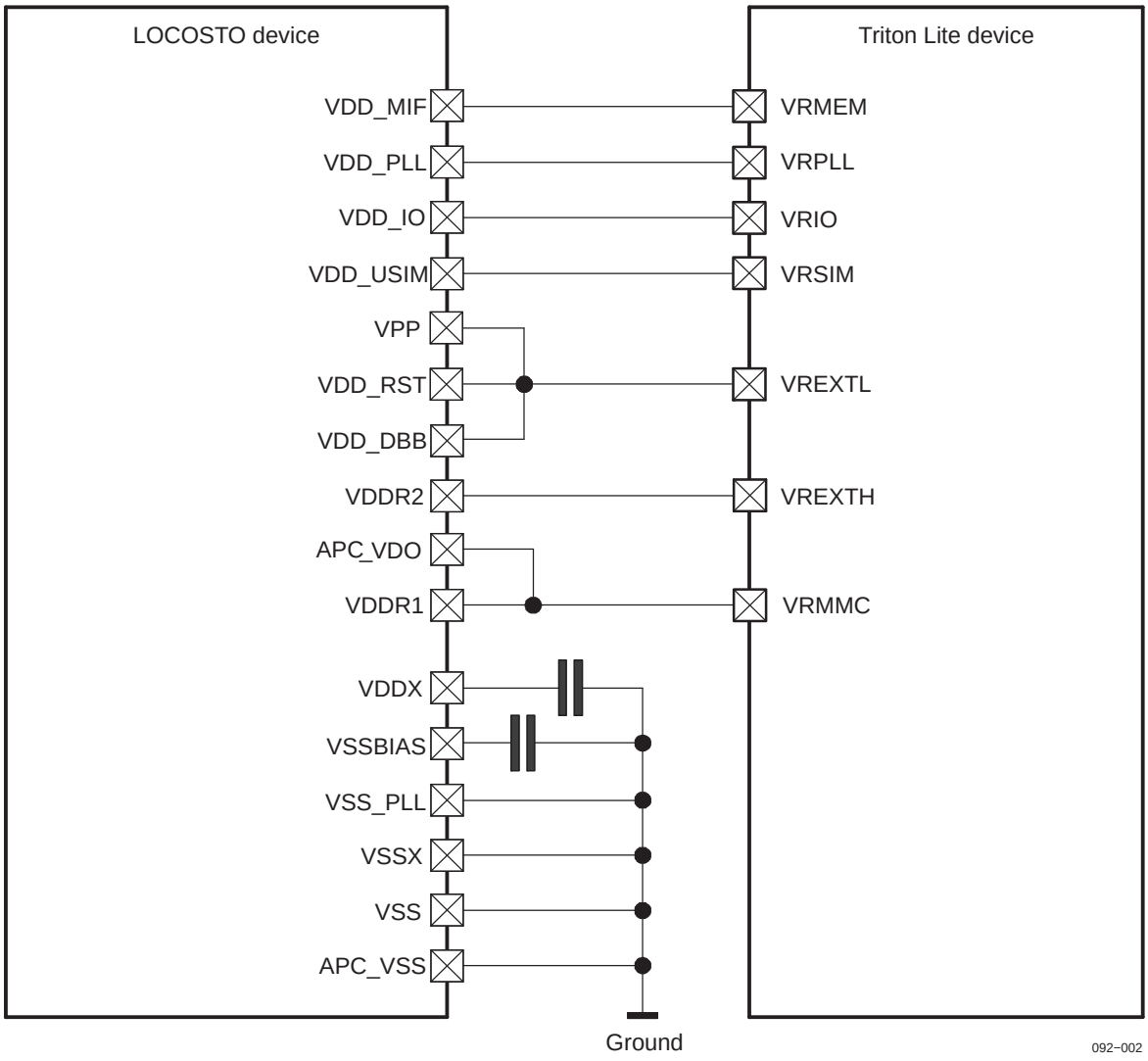
The LOCOSTO device boot-up operation requires certain hardware configuration settings. Clock, reset, and power connections, as well as the pins involved in setting the boot memory space for the microprocessor unit (MPU), must be connected and driven properly for successful device initialization. This section details the specific requirements for the pre-initialization stage.

30.2.1 Power Connections

Power to the LOCOSTO device can be supplied by an external companion chip. Texas Instruments provides a global solution for the LOCOSTO device using an external power device: the TWL3031 power integrated circuit (IC). For more information on the TWL3031 IC, see the *TWL3031 Data Sheet*.

Figure 30-2 presents the power connections between the LOCOSTO device and the TWL3031 device.

Figure 30-2. LOCOSTO Power Connections



For a complete description of power-pin mapping on the LOCOSTO device package, see Chapter 18, *Configuration*.

Table 30-1 details the voltage levels for each LOCOSTO device power pin.

Table 30-1. LOCOSTO Device Voltage Levels

LOCOSTO Pin Voltage Name	TWL3031 Source	Description
VDD_MIF	VRMEM	External memory interface (EMIF) I/O power supply
VDD_PLL	VRPLL	Supply voltage for the digital baseband phase-locked loop (DBB PLL)
VDD_IO	VRIO	Generic I/O power supply
VDD_USIM	VRSIM	Universal subscriber identity module (USIM) I/O power supply
VPP	VREXTL	Digital radio processor (DRP) core digital supply
VDD_RST	VREXTL	Reset I/O power supply
VDD_DBB	VREXTL	DBB application-specific IC (ASIC) core power supply, including the internal static RAM (SRAM) memory and the internal ROM

Table 30-1. LOCOSTO Device Voltage Levels (continued)

LOCOSTO Pin Voltage Name	TWL3031 Source	Description
VDDR2	VREXTH	Pre-regulated input to the low dropout (LDOX) DRP embedded LDO
APC_VDO	VRMMC	Automatic power control (APC) output amplifier power supply
VDDR1	VRMMC	Pre-regulated input to the LDOOS, LDOA and LDORF DRP embedded LDOs

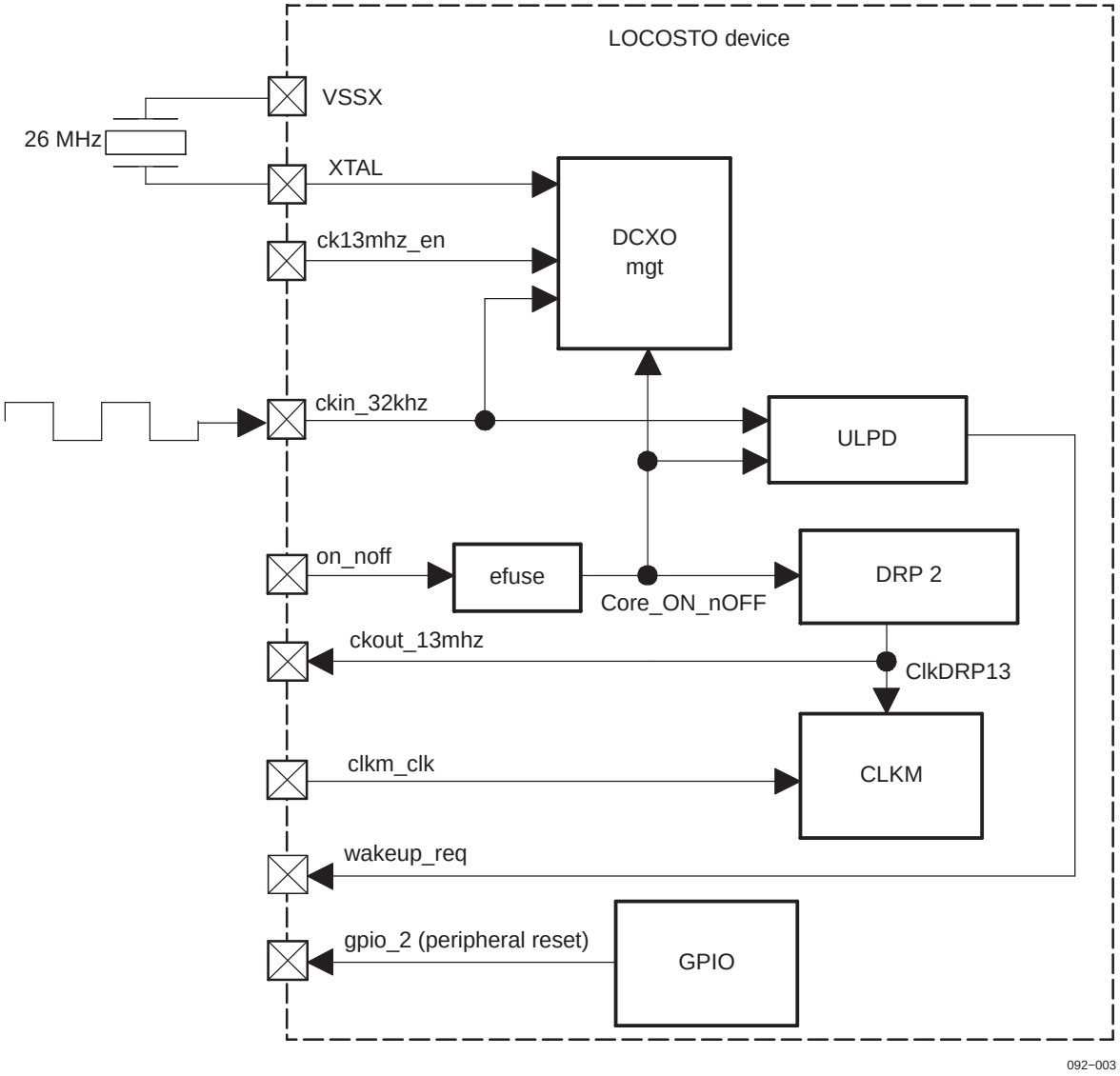
For more information on power management, see Chapter 6, *Power, Reset, and Clock Management*.

30.2.2 Clock and Reset

30.2.2.1 Clock and Reset Overview

Figure 30-3 shows the LOCOSTO clock and reset environment, which gathers all of the signals related at the system level to clocks and resets.

Figure 30-3. LOCOSTO Clock and Reset Environment



The LOCOSTO device requires two external clock inputs: the functional and interface clocks are all generated from the ckin_32khz and the XTAL input clocks.

Table 30-2 describes the clock and reset I/Os of the LOCOSTO device.

Table 30-2. Clock and Reset I/O Description

I/O Name(1)	Type	Description
XTAL and VSSX	Clock input	The 26-MHz system clock is the main source clock of the chip and is connected to a 26-MHz quartz crystal.

(1) Wakeup_req, on_noff, ckin_32khz, ckout_13mhz, XTAL, and VSSX have permanently assigned pin locations with no multiplexing configuration.

Table 30-2. Clock and Reset I/O Description (continued)

I/O Name[1]	Type	Description
ckin_32khz	Clock input	The 32-kHz clock is generated by the TWL3031 power IC, used for low-frequency operations, and supplies the wake-up domain for operation in the lowest power mode.
clkm_clk	Clock input	This external input clock can be an alternate solution for the quartz crystal and supports digital clock frequencies up to 40 MHz.
ck13mhz_en	Logic input	Enables the system clock generation
ckout_13mhz	Clock output	This 13-MHz output clock is available by setting the CONF_LOCOSTO_DEBUG[3] BUT_CLK13M bit to 0.
on_noff	Reset input	Cold reset for the chip.
wakeup_req	wakeup output	When the LOCOSTO device wakes up from the deep sleep state, the ultra-low power down (ULPD) module drives this signal to a high level.
gpio_2	reset output	The LOCOSTO gpio_2 is the peripheral reset signal and is connected to the reset pins of the peripheral devices. During booting, the gpio_2 pin is a pulldown; thus, the resets of all peripheral devices are enabled.

For more information about clock distribution, reset, and wake-up, see Chapter 6, *Power, Reset, and Clock Management.*

30.2.2.2 Clock Configuration

The digital PLL (DPLL) module and the clock manager (CLKM) module control the LOCOSTO clocks. The DPLL module generates the ClkDPLL104 internal clock configured through its control registers. The CLKM module disables/enables the clocks while dispatching them.

The following conditions apply after the LOCOSTO device is reset and all of the clocks are stable:

- The application software can write to the control registers (CLKM.CNTL_ARM_CLK[2:1] CLKIN_SEL and CLKIN_SEL0 bits) to switch to a desired mode of operation. The CLKM.CNTL_ARM_CLK [2:1] bits should be changed before modifying the clock divider ratios in the DPLL_CNTL_REG register.
- For the 104-MHz DPLL clock, the DPLL_CNTL_REG register configuration is 0x2830 (ClkDPLL104 = ClkDRP13 * 16 / 2).
- To determine if the DPLL is locked, poll the DPLL_CNTL_REG[0] LOCK bit.

For a complete description, see Chapter 6, *Power, Reset, and Clock Management.*

For a complete description of all the LOCOSTO configuration registers, see Chapter 18, *Configuration.*

30.2.3 Boot Configuration

Two external LOCOSTO pins, USB_BOOT and nBSCAN, are used to select the boot mode.

The ROM code is configured to download software from the universal serial bus (USB) interface or the universal asynchronous receiver/transmitter (UART) interface selected by the USB_BOOT GPIO pin (see Table 30-3).

Table 30-3. USB_BOOT Configuration

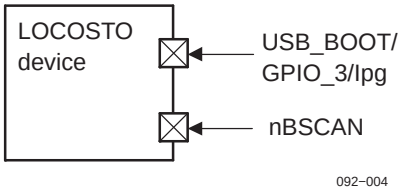
nBSCAN Pin State	USB_BOOT Pin State	UART	USB	Boundary Scan
1 (high)	1 (high)		X	
1 (high)	0 (low)	X		
0 (low)	1 (high)			X
0 (low)	0 (low)			X

This mechanism applies to both initial flashing in production (flash memory is empty) and re-flashing in service (flash memory has content). After booting, the USB_BOOT pin can optionally be used as GPIO or LPG output for other functions.

To set the chip test engineering and the I/O boundary scan, pull down the nBSCAN pin to a low level. To boot the device, the nBSCAN pin should be high (an on-chip pullup is set by default).

Figure 30-4 shows the boot configuration environment used to select the boot mode.

Figure 30-4. Boot Configuration Environment



30.2.4 Pin Multiplexing and Pullup/Pulldown

Each pin with a multiplexing function is assigned a MODE bit field in a configuration register, thus creating up to eight possible multiplexing options per pin (with up to 3 bits in the MODE bit field). At reset (on_noff is low), each pin is forced asynchronously to a default multiplexing configuration mode (mode0). Most pins also can be configured either as a pullup or a pulldown with a 2-bit field.

See Chapter 18, *Configuration*, for a complete description of pin multiplexing and pullup/pulldown configuration.

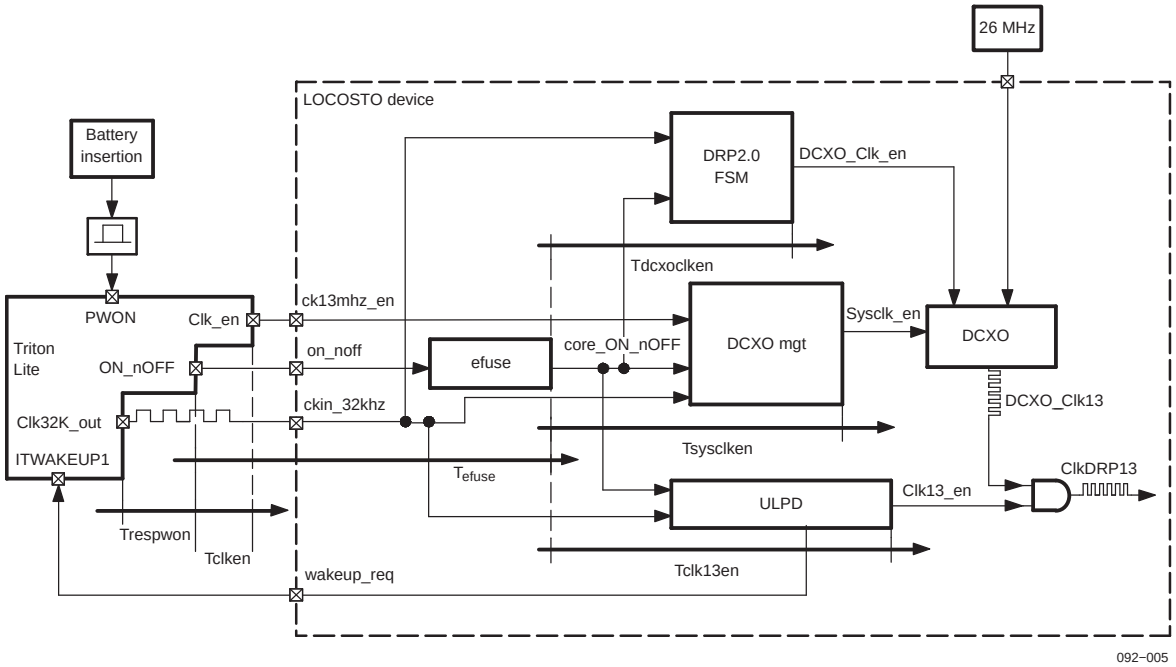
CAUTION

No pin should be left floating. Each floating pin must be configured with a pullup or a pulldown.

30.3 Power, Clock, and Reset Sequence on Power-Up

This section details the ClkDRP13 clock generation from the battery insertion event. Figure 30-5 shows the block and timing diagram of the power-up sequence.

Figure 30-5. Power-Up—13-MHz Clock Generation Sequence



The LOCOSTO device power-up is managed by the TWL3031 device in cooperation with the LOCOSTO device power-management finite state-machine (FSM).

The following steps comprise the power up sequence (see also Figure 30-6):

Step 1: When the main battery is initially plugged in, the TWL3031 device is supplied and clocked after the 32-kHz oscillator start-up phase (T32SU).

Step 2: When the ON/OFF button is pressed, the TWL3031 power-management FSM accepts the start condition as valid only after a stable low-level period of the PWON signal.

Step 3: After a valid switch-on condition is recognized, the TWL3031 FSM provides the correct sequence for power supplies rising up. The VPP, VDD_RST, and VDD_DBB LOCOSTO power domains are supplied first by the VREXTL TWL3031 power domain.

Step 4: After the VREXTL supply rail setup time, the VDD_PLL LOCOSTO power domain is supplied by the VRPLL TWL3031 power output.

Step 5: After the VRPLL supply rail setup time, the VDD_MIF LOCOSTO power domain is supplied by the VRMEM TWL3031 power output.

Step 6: After the VRMEM supply rail setup time, the VDDR2 LOCOSTO power domain is supplied by the VREXTH TWL3031 power output.

Step 7: After the VREXTH, VRPLL, and VRMMC supply rails setup time, the VDD_IO LOCOSTO power domain is supplied by the VRIO TWL3031 power output.

Step 8: When the VRIO power domain is stable, the TWL3031 FSM enables the 32-kHz clock to the external world.

Step 9: When the 32-kHz clock is stable and after the Trespwon time interval, the FSM releases the system reset, driving the on_noff signal to a high level.

Step 10: As a final action, the TWL3031 FSM enables the system clock generation, thus driving the ck13mhz_en signal to a high logical level at the Tclken time after reset release (see Table 30-1).

Step 11: On the LOCOSTO device side after the on_noff signal is driven high by the TWL3031 device, the efuse module releases the core_ON_nOFF signal after a Tefuse delay (see Table 30-1).

Step 12: The following three state-machines start in parallel to activate the 13-MHz clock (ClkDRP13):

- The DCXO management module generates the Sysclk_en signal to the DRP DCXO module following a Tsysclken delay.
- The DRP2.0 FSM activates the DCXO clock with the DCXO_Clk_en following a Tdcxoclken delay.
- The ULPD FSM enables the 13-MHz clock by releasing the Clk13_en signal following a Tclk13en delay.

Note: Because the three FSMs run in parallel, the values of the different delays must satisfy a consistent enable of the system clock; that is, the DCXO_Clk_en signal must be released before the Sysclk_en signal, which must be released before the Clk13_en signal.

Figure 30-6 shows a chronogram of the 10 power-up steps. Table 30-4 describes the delay settings.

Figure 30-6. LOCOSTO Device Power-Up Sequence

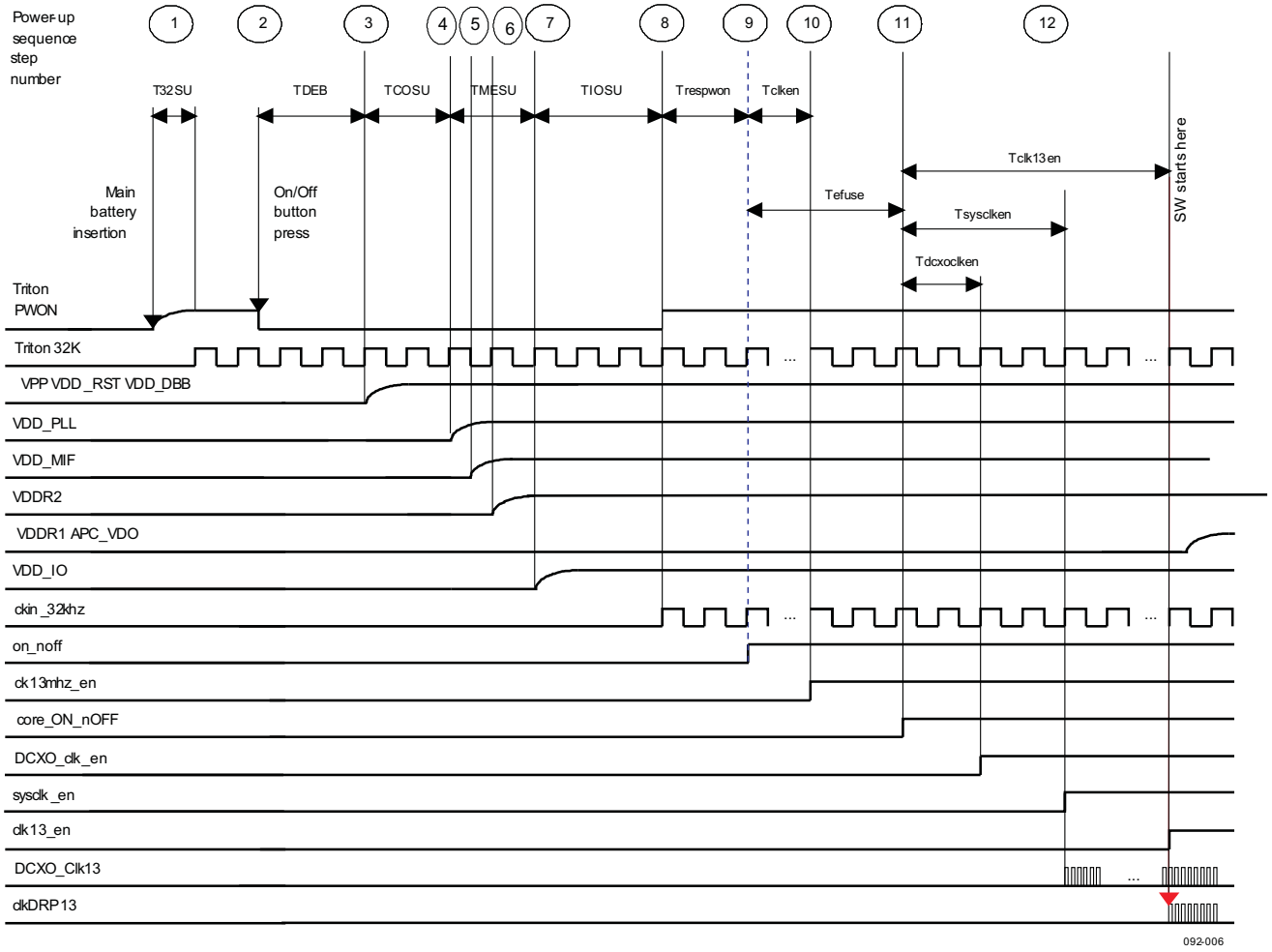


Table 30-4. Power-Up Delay Settings

Delay	Default Value	Configuration
T32SU	See the TWL3031 Data Sheet	
TDEB		
TCOSU		
TIOSU		
TMESU		
Trespwon	7.1411 ms	See the *TWL3031 Data Sheet*
Tclken	19 x 32 khz clock periods (0.5798 ms)	
Tefuse	744 x 32 khz clock periods (22.705 ms)	None
Tdcxoclken	183 x 32 khz clock periods (5.585 ms)	Reset value
Tsysclken	510 x 32 khz clock periods (15.564 ms)	Reset value
Tclk13en	8253 x 32 khz clock periods (251.86 ms)	Reset value

After the voltages and clocks are ramped and stable, the USB_BOOT and nBSCAN pins are sampled and the ROM code is executed.

30.4 Device Booting by ROM Code

This section describes the LOCOSTO device high-level booting concepts and provides basic knowledge of the device booting.

For more detailed information about booting on the LOCOSTO device, see your TI representative.

30.4.1 Booting Overview

30.4.1.1 Device Types

The ROM code supports two different LOCOSTO device types with associated security schemes: the HS device and the GP device.

During booting, the ROM code checks the content of the MPK eFused on the chip. When the MPK is nonzero, the device operates in HS mode; otherwise, the device operates in GP mode.

- The HS device is a production device with all security features enabled. The ROM code requires the authentication of the initial software (firmware or flash loader) before execution. A failed authentication causes the device to enter reset state. The authentication of code in external flash memory or code that is downloaded is handled by routines in the secure ROM.
- The GP device is a production device in which security is disabled. Authentication is not performed before initial software execution. All references made to authentication are ignored. Emulation is always open.

30.4.1.2 Booting Types

Booting is the process of starting a bootstrap from one of the booting memories. The code executed during memory booting is normally used for OS startup.

The ROM code has two functionalities: peripheral booting and memory booting.

- In peripheral booting, the ROM code polls the selected interface and downloads and executes software in internal RAM. The downloaded software from an external host is used to program flash memories connected to the device.

  This special case of peripheral booting is called pre-flashing; the downloaded software for pre-flashing is called the flash loader. The flash loader burns a new client application image into external flash memory. Initial software is a generic term used for bootstrap, downloaded software, and the flash loader. After the image is burned, a software reset can be performed.
- In memory booting, the ROM code finds the bootstrap in external flash memory, authenticates the bootstrap only if it is on an HS device, and then the bootstrap is executed.

30.4.1.3 Main Features

The secure ROM code part of the ROM code includes the following features:

30.4.1.3.1 Boot Loader

- Synchronization on the UART/USB:

  The boot loader supports the USB or UART modem interface for interaction with the host to download the flash loader code. At reset, the boot loader code polls the interface selected through the USB_BOOT pin for a signaling command to be transmitted from the host (that is, a PC).
- Firmware authentication:

  If the signaling command is not detected on the selected interface before the time-out period (memory booting case), the boot loader assumes that a firmware is at the nCS3 location.

  For HS devices, the secure boot loader authenticates and then executes if authentication passes.

  For GP devices, the boot loader jumps directly to the nCS3 location (0x0600 0000).
- Flash loader download:

  If the signaling command is detected on the selected interface (peripheral booting case), the boot loader interacts with the host based on the predefined protocol and then downloads the flash loader

code.

The flash loader code depends on the ROM code only for memory mapping. 131,004 bytes are reserved in internal SRAM to download the flash loader.

- Flash loader authentication:

  For HS devices, the flash loader is certified and downloaded only if the certification passes. After the download, the secure boot loader authenticates the code before handing over control to it. This authentication is similar to the firmware authentication step.

  For GP devices, control is immediately handed over to the flash loader code.

30.4.1.3.2 Secure Services (HS devices only)

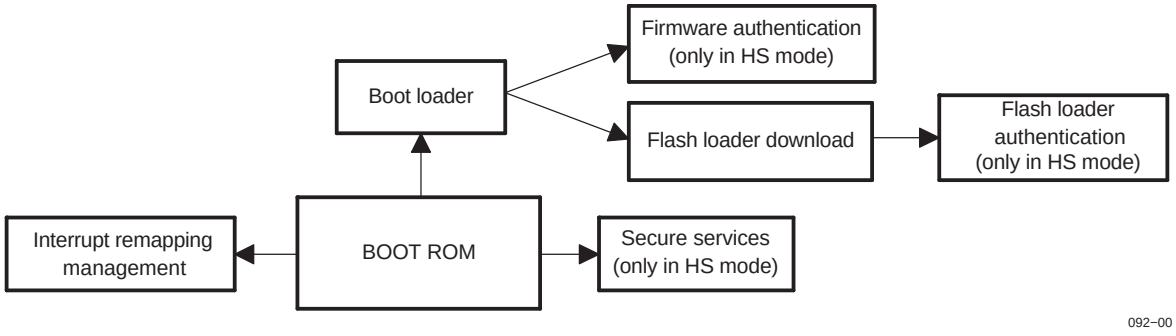
- Secure runtime checker: This service allows a firmware to check the firmware certificate during runtime. This also checks the firmware reliability.
- Secure runtime loader: This service allows creation of a certificate from a piece of code or data to avoid cloning in another mobile application.
- Secure runtime platform data checker: This service provides protection against cloning by providing the application a secure means to verify the unique association of the data/code to the hardware platform.

For more information about secure services, see the Boot ROM Code documentation.

- Remapping of interrupt vectors: This part of the code remaps the interrupt vectors (which maps to boot ROM by default) to a predefined location in the internal SRAM.

Figure 30-7 represents the high-level ROM code features.

Figure 30-7. ROM Code Features



30.4.1.4 ROM Code Hardware Configuration

The hardware is configured during the first step of the ROM code:

- The DPLL is configured to generate the 104-MHz clock.
- The ARM uses the DPLL.
- ARM interrupts are masked at reset in the ARM CPSR register, after which it is not mandatory to configure CPSR or to mask the interrupt in the interrupt handler module.
- TI peripheral bus (TIPB) access factor:
  - 0 on strobe 0
  - 0 on strobe 1
- Application programming interface (API) wait state:
  - 0WS in host-only mode (HOM)
  - 6WS in shared-access mode (SAM)
- EMIF configuration: The ROM code writes the relevant EMIF register (emif_conf_reg, eif_conf_reg_cs3, etc.) to its reset value for safety. After the firmware certificate authentication phase, the registers are programmed based on the certificate values.
- The watchdog timer and secure watchdog timer are disabled.
- The random number generator (RNG) module is initialized.

- The Timer1 module is initialized with the time-out value of the selected interface.

30.4.2 Booting Sequence

The LOCOSTO device uses the following booting sequence (see also Figure 30-8 through Figure 30-10):

Step 1: Regardless of the reset source (power-on reset, on/off, or warm reset), the mobile application always starts the secure ROM code execution. This step cannot be bypassed and forms the basis of the security.

The JTAG port is always disabled independently of the reset source, which implies that it is not possible to plug in an emulator to bypass the boot sequence or to dump sensitive data that is located in the internal SRAM. Disabling the JTAG is managed by hardware.

- HS devices: Enabling emulation of the JTAG port by the ROM code is always the last operation following authentication and the integrity check and before giving the hand to the firmware or the flash loader. In addition, enable is conditioned by the value of the DEBUG parameter in the certificate and die ID field.
- GP devices: Enabling emulation occurs at the start of the boot ROM code.

Step 2: The ROM code first checks the content of the MPK fused on the chip. When the MPK is nonzero, the device operates in the HS mode; otherwise, the device operates in GP mode.

Step 3: In parallel with Step 2, the protected resource reset management (PRRM) module erases the contents of the internal SRAM (first 2M bits) and the API memory. This module directly uses the ARM clock, which is configured at 52 MHz.

Step 4: The ROM code performs basic initialization, such as DPLL and ARM clock configuration. The DPLL provides the ARM clock frequency and is configured at 104 MHz. The DPLL receives a VTCXO input frequency of 13 MHz.

Step 5: Following an optimal hardware configuration, the ROM code checks the reset source. For a watchdog reset (the CNTL_RST[3] bit set to 1), the ROM code bypasses the synchronization on the selected interface to start the memory booting sequence directly (see Step 11). Moreover, when a watchdog reset occurs, the PRRM can be configured to not reset the content of the internal SRAM.

For example, the configuration can set the FULL_RST_EN bit to 0 and the PRRM_INT_START and PRRM_INT_END bits to 0x0000 in the PRRM module. For more information about the PRRM module, see Chapter 6, *Power, Reset, and Clock Management.*

The ROM code does not reset the CNTL_RST[3] bit. This information is kept for firmware and allows you to incorporate a fast recovery mechanism.

30.4.2.1 Peripheral Booting

Step 6: If the reset source is not the watchdog, the USB_BOOT pin is read to see which interface is selected for peripheral booting. The USB or UART is configured based on this setting (see Section 30.4.4.2, *Peripheral Booting on UART*, and Section 30.4.4.3, *Peripheral Booting on USB*).

Step 7: The ROM code starts a hardware timer and awaits host synchronization by polling the selected interface. The synchronization time-out is fixed to 15 ms for the UART interface and 3 s for the USB interface.

Step 8: If synchronization on the selected interface is detected before the time-out, the ROM code starts peripheral booting:

- For HS devices, the ROM code authenticates the MPK, the certificate, and the originator's public key (OPK) before downloading the flash loader.
- For GP devices, authentication is not required before the download.

Step 9: A specific protocol is used to download the flash loader (see Section 30.4.4.4.3, *RAM Loader Communication Procedure*).

Step 10: For GP devices, the flash loader is directly downloaded and executed.

For HS devices, after the full flash loader is received in internal SRAM, the ROM code authenticates the flash loader. If authentication is successful, the downloaded flash loader is executed.

After flash loader identification, the JTAG is enabled only if both of the following conditions occur:

- The debug request field in the certificate is enabled.
- The die-ID field in the certificate is either blank or matches the die-ID of the LOCOSTO device target.

30.4.2.2 Memory Booting

Step 11: If a synchronization time-out is reached or a watchdog reset occurs, the ROM code starts memory booting (for more information about the memory booting sequence, see Section 30.4.5, *Memory Booting*).

- For GP devices, the ROM code jumps and starts execution from the nCS3 location.
- For HS devices, the certificate of the firmware in external flash memory is authenticated and certificated (for information about the manufacturer certificate, see the ROM code documentation).

Step 12: Following a successful certificate authentication and integrity check, the ROM code configures the EMIF based on the parameters in the firmware certificate (for more information about the certificate format, see Section 30.4.3, *Certificate Description*).

Step 13: After hardware configuration, the DIE_ID field of the certificate and the DIE_ID registers are compared, and the firmware is authenticated.

After the die-ID comparison, the JTAG is enabled only if both of the following conditions occur:

- The debug request field in the certificate is enabled.
- The die-ID in the certificate matches the die-ID of the LOCOSTO device target.

Note: A firmware certificate with a blank die-ID does not enable emulation.

Step 14: Following a successful firmware authentication and integrity check, the firmware is executed.

Step 15: If an error occurs in the certification or authentication verifications, the ROM code performs a software reset to restart the boot sequence.

Figure 30-8. HS and GP Device Booting Sequence Start

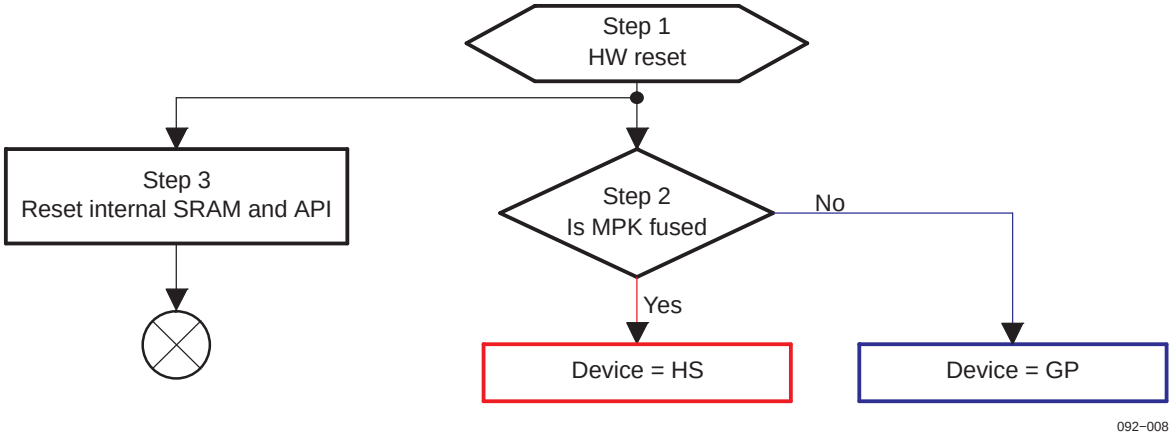


Figure 30-9. GP Device Booting Sequence

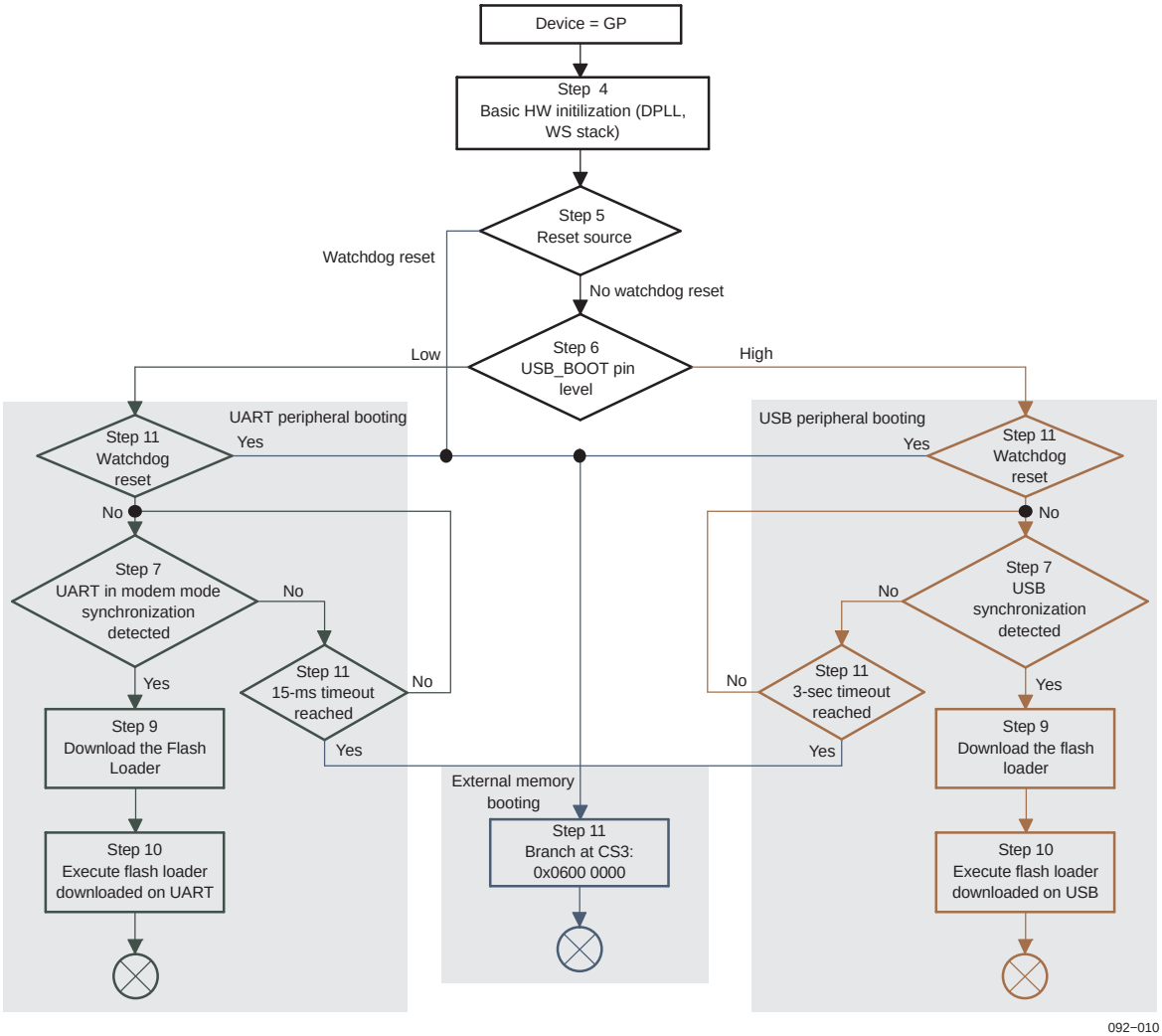
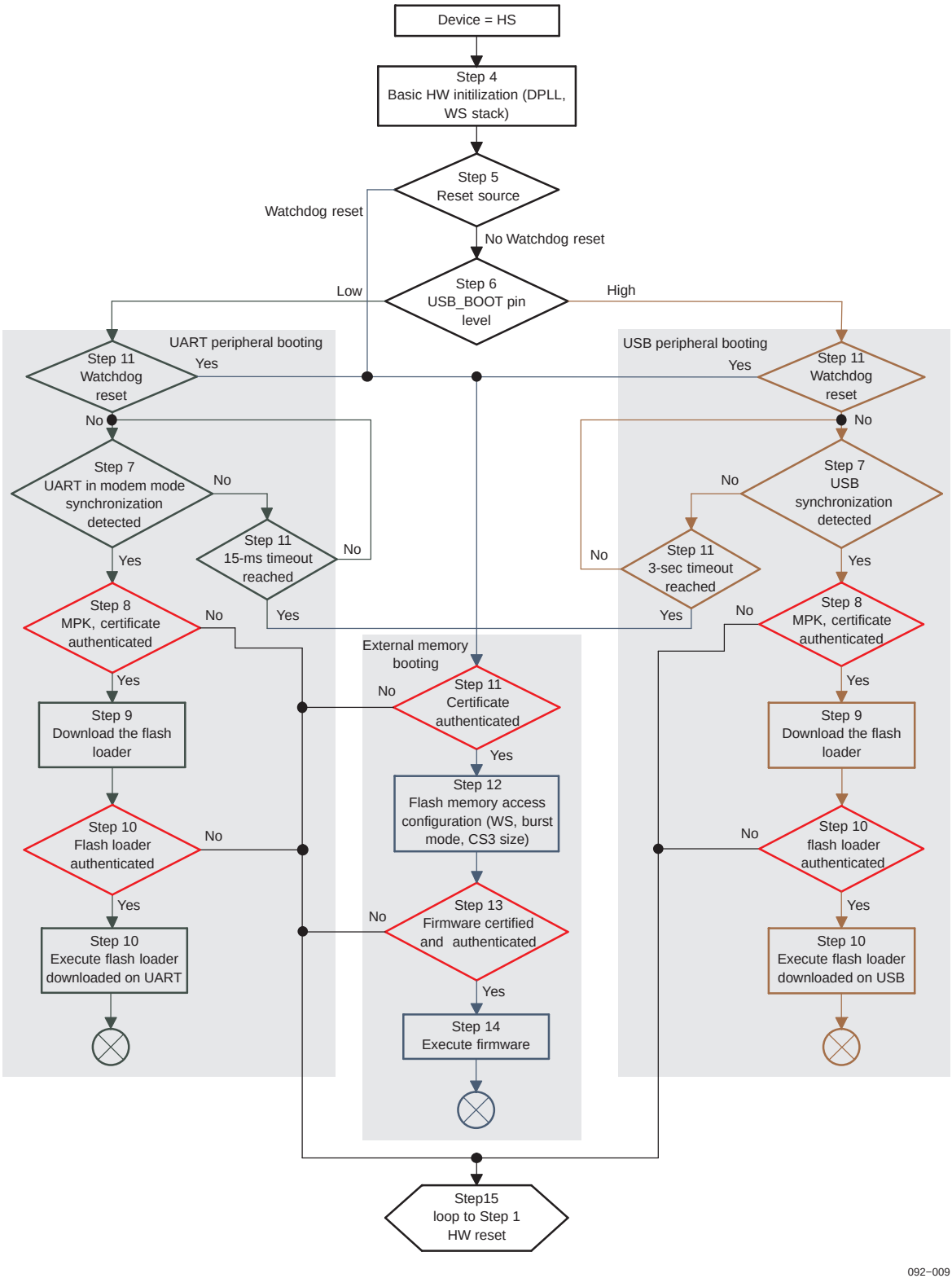


Figure 30-10. HS Device Booting Sequence



30.4.3 Certificate Description

30.4.3.1 Certificate Description Overview

The foundation of the secure computing platform is the authentication of any software installed before execution. Authentication of a software module is based on the certificate associated with the module. The certificate allows authentication of the software module rather than referencing and checking code integrity.

- For the HS device:
  - When the certificate is based on a trusted originator, such as flash loader or firmware, it is called the manufacturer certificate.
  - When the certificate is based on the hardware platform, it is called a platform certificate. A platform certificate binds and unbinds services.

  For more information about the certificates, see the Boot ROM Code documentation or contact your TI representative.
- For the GP device, the ROM code checks the TOC on the flash loader.

30.4.3.2 TOC for the GP Device

A set of data blocks called the TOC is used during peripheral booting for GP devices. The set of data blocks contain basic information such as code size, entry point of firmware, etc.

Table 30-5 describes the TOC structure.

Table 30-5. TOC Description[(1)]

Name	Size in Bytes	Description
CODE_ADDR	4	Firmware start address: The address of the first block write command.
CODE_SIZE	4	Code size in bytes: This block indicates the code size of the flash loader to be downloaded; must be aligned on a 32-bit boundary.
CODE_START_ADDR	4	Address of firmware entry point, which must be a 32-bit function; the code start address must be in the following memory range: 0800:0044≤BlockAddress<0802:0000. This is the address to which the ROM code jumps and from which it starts execution after the full code is downloaded.
SW_HASH	20	Hash on the entire firmware. The hash algorithm used is SHA1, which produces a digest of width160 bits.

(1) All fields are defined in little-endian mode.

30.4.4 Peripheral Booting

30.4.4.1 Peripheral Booting Overview

The ROM code implements drivers for the UART and USB interfaces.

After the interface is selected, the ROM code polls the selected interface until synchronization is reached on the selected interface or before it times out.

A detailed procedure between the host and the target to download the flash loader follows synchronization. The protocol between the PC and the target is independent of the interface chosen to download the flash loader. However, there are differences in the procedures for the GP device and the HS device.

The following sections describe these steps.

30.4.4.2 Peripheral Booting on UART

30.4.4.2.1 UART Configuration

If the UART interface is chosen for the peripheral booting, the ROM code initializes the UART peripheral with the following configuration:

- Modem mode
- 19,200 bps
- 8 bits
- 1 stop bit
- No parity

The ROM code then initializes a time-out of 15 ms to allow 15 signaling commands to be received at 19,200 bps. The code then polls the UART interface for a synchronization event before time-out occurs.

Because the UART module is shared between the MPU and the DSP, the static switch is configured for MPU access. The CLKM module is configured to give the UART module a 48-MHZ clock.

30.4.4.2.2 Synchronization on UART

In the global boot sequence detailed in Section 30.4.2, Booting Sequence, the ROM code first initializes the UART peripheral and then polls the UART module until the < character is received. Receipt of this character signals that the peripheral booting procedure can start.

The synchronization step is the same for both the GP device and the HS device.

30.4.4.3 Peripheral Booting on USB

30.4.4.3.1 USB Configuration

If the USB interface is selected for peripheral booting, the ROM code initializes the USB peripheral to the full-speed mode.

The ROM code initializes a time-out of 3 s and waits for USB enumeration (which is initiated by the host). The ROM code configures the USB to have one bulk-in endpoint and one bulk-out endpoint.

30.4.4.3.2 Synchronization on USB

- After power-on reset, the ROM code expects the following initial setup:
  - The USB cable connected to a host for peripheral booting
  - The TWL3031 device connected and powered so that control registers are accessible by the I^2C
- In the global booting sequence (see Section 30.4.2, Booting Sequence), the ROM code first initializes the USB peripheral and the I^2C. The I^2C is used to configure the TWL3031 control registers. For more information about the TWL3031 configuration, see the Boot ROM Code documentation.
- The ROM code initializes a 3-s timer.
- The timer stops as soon as the host moves the USB to the configured state, which indicates the presence of an active host.
- The host must send the signaling command and start additional communication.
- Once the bus is moved to the configured state, the ROM code waits until the signaling command is received.

30.4.4.4 Flash Loader Download Sequence

30.4.4.4.1 Flash Loader Download Overview

After synchronization on the UART or the USB is detected before time-out occurs, the RAM loader is called. The download process is identical for both interfaces.

After synchronization is detected, it is not possible go back to certify and authenticate the firmware located in external flash. The only way to certify and authenticate the firmware is to reset the device or perform a watchdog reset.

The RAM loader is part of the ROM code responsible for the flash loader download communication sequence. The RAM loader is considered a slave application (that is, it can only receive commands and send associated responses to inform the remote application).

On the host (that is, PC) side, another RAM loader is in charge of the flash loader upload communication sequence.

30.4.4.4.2 Flash Loader Authentication

30.4.4.4.2.1 GP Devices

Authentication steps are valid only for HS devices. GP devices do not include authentication. At the end of the flash loader download, the ROM code directly starts execution. GP devices include the following verifications:

- Basic check on the TOC
- Firmware hashing performed with each write command. When the final write command is received (detected when the number of bytes received so far matches the code size in the TOC), the hash value computed so far is verified against the hash value in the TOC. This step is included to increase the robustness of the download protocol.

30.4.4.4.2.2 HS Devices

The certificate and firmware authentications are performed. The verifications are separated in the RAM loader state-machine to optimize the data transfer on the UART. The authentication sequence follows (see also Figure 30-11 for a flowchart of the sequence):

Step 1: The first authentication steps shown in Figure 30-11 are done before to download the flash loader in internal SRAM. The host sends a parameters command to transmit the flash loader certificate to the LOCOSTO device. When the certificate is received, the ROM code checks the certificate. The certificate verification is composed of the following elements:

- MPK authentication
- Firmware certificate authentication
- OPK authentication

Step 2: Based on the result of the certificate verification, the ROM code either acknowledges or does not acknowledge the parameters command to the remote host.

Step 3: Following successful verification, the protocol allows the code to be downloaded. During downloading, the hashing of the code occurs in parallel on each byte received.

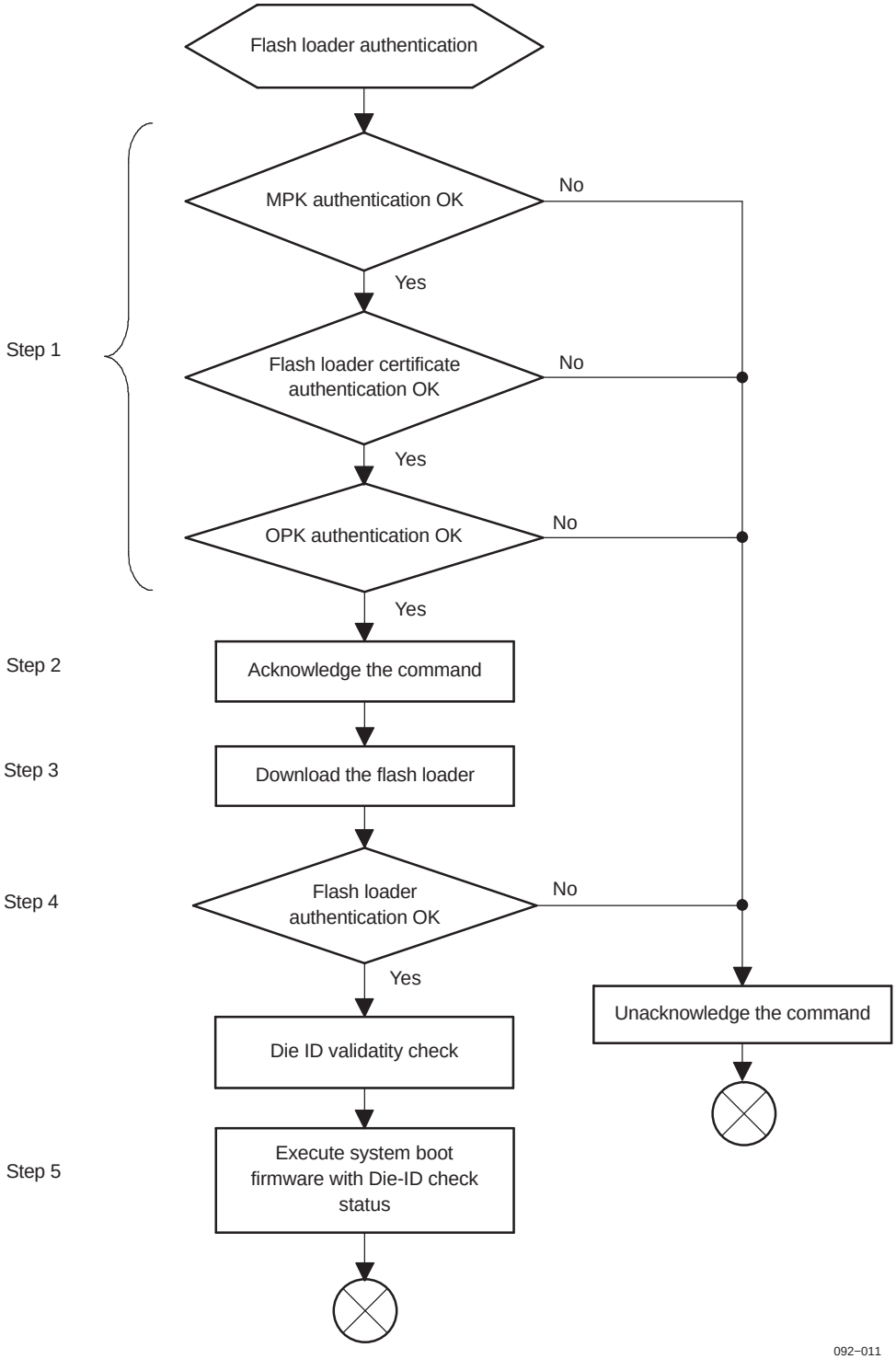
Step 4: After the full flash loader is received, the computed signature is compared to the signature embedded in the certificate after decryption.

If authentication succeeds, emulation is enabled only if the debug request field in the certificate is enabled and the DIE_ID field in the certificate is either blank or matches the die-ID of the device target.

If authentication fails, the RAM loader state-machine sends a Write NACK response to the remote device, returns to its default state, and waits for a new signaling request.

Step 5: The flash loader is executed.

Figure 30-11. Flash Loader Authentication Flowchart



30.4.4.4.3 RAM Loader Communication Procedure

Table 30-6 lists the possible commands exchanged between the host and the device.

Table 30-6. Command List

Host Side		Mobile Side	
Description	**Command**	**Description**	**Response**
Signaling request	<i	Signaling response	>i[X][Y][Z]
Parameter request	<p[X][Y][Z]	Parameter ACK response	>p
		Parameter NACK response	>P[X]
Write request	<w[X][Y][Z]	Write ACK response	>w
		Write NACK response	>W[X]
Abort request(1)	<a		
Certificate request(2)(3)	<c	Certificate ACK response	>c[X]

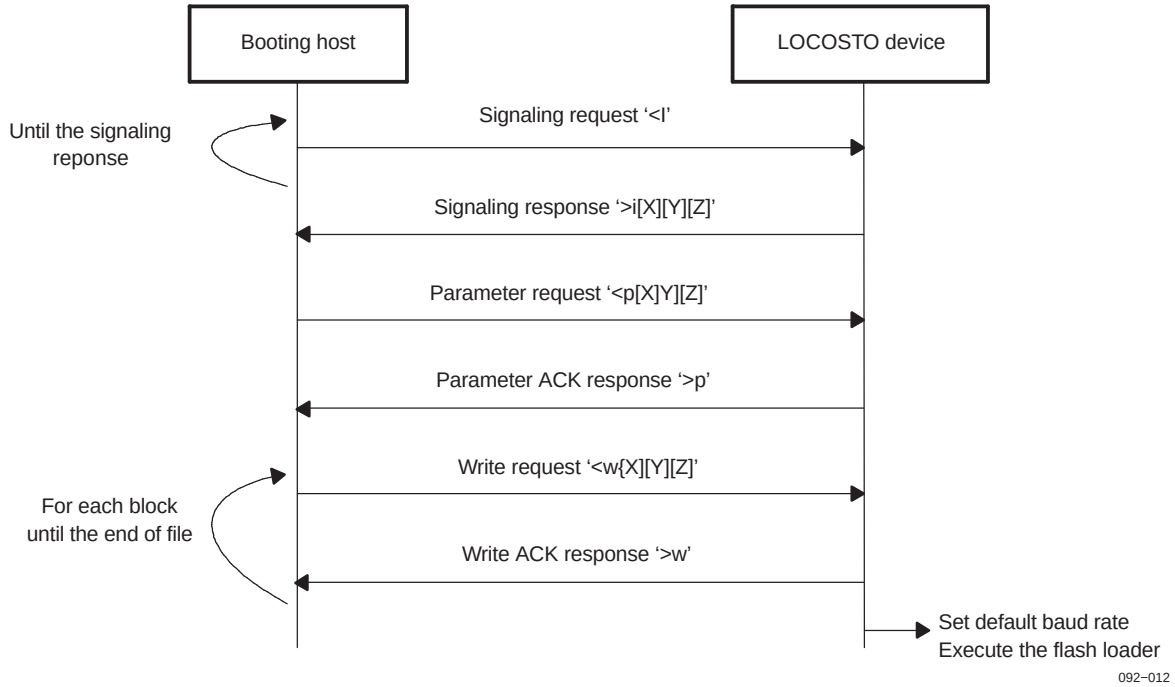
(1) When an abort request is received, the RAM loader changes the UART baud rate to the default configuration and waits for a new signaling request. Acknowledgment is not sent to the remote application.

(2) The certificate request is not applicable for GP devices. The certificate request is a new feature to provide the user with the firmware version for HS devices. The certificate lets the user modify the emulation bit (and die-ID, if required), generate a new certificate, and then reprogram only this certificate in the external flash memory. This helps to permanently re-enable the debug features on the device.

(3) It is not mandatory to reprogram the full flash memory content. However, any modification in the firmware associated with the certificate requires the certificate to be updated. If the certificate is not updated, the firmware authentication fails during the next RAM loader procedure.

Figure 30-12 shows the communication protocol during the flash loader download.

Figure 30-12. Nominal Download Communication Sequence



If an error occurs on the device side during the download communication sequence, the device RAM loader does the following:

- Sends a NACK response
- Sets the default UART baud rate (for the UART mode)
- Resets its state-machine

If a NACK response is received on the host side during the download communication sequence, the host RAM loader does the following:

- Sets the default UART baud rate
- Resets its state-machine
- Informs the user

Table 30-7 lists each command parameter.

Table 30-7. Command Parameters[1]

Format[2]	Parameter	Size / First Byte	Definition
<i	None		Signaling request
>i[X][Y][Z]	X	16 bits LSB first	ROM code version
	Y	128 bits LSB first	Hash(MAN_PUB_KEY) used only in HS devices: Embedded value corresponds to the hashing (SHA-1) of the MAN_PUB_KEY.
	Z	128 bits LSB first	DIE_ID: Device identification code
<p[X][Y][Z]	X	8 bits	UART baud rate to use during download For the USB link, the baud rate has no meaning and is neglected. MODEM baud rates: • 0x00: 92,1600 bps • 0x01: 460,800 bps • 0x02: 230,400 bps • 0x03: 115,200 bps • 0x04: 57,600 bps • 0x05: 38,400 bps • 0x06: 28,800 bps • 0x07: 19,200 bps
	Y	32 bits MSB first	UART time-out configuration updated based on the ARM clock configuration The time-out avoids putting the RAM loader in a dead lock state if a command is not received in its totality. Value 0x00000000 disables the time-out. The command decoder waits until the entire command is received.
	Z	LSB first	TOC or certificate: • For HS devices: Send the manufacturer certificate. • For GP devices: Send the TOC structure. • For the TOC description, see Section 30.4.3, *Certificate Description*.
>p	None		Parameter ACK response
>P[X]	X	8 bits	Parameter NACK error code: • 0x01: Incorrect baud rate • 0x02: Incorrect certificate (HS devices only) • 0x03: Incorrect code address
<w[X][Y][Z]	X	32 bits MSB first	Number of bytes in the block: Must be a modulus of 64 bytes, except for the last command.
	Y	32 bits MSB first	Address where the block must be written. Block address must be in the following memory range: 0800:0044≤BlockAddress<0808:0000. Block address of first write request command must match with the CODE_ADDR field of the manufacturer certificate of the flash loader.

(1) Commands not applicable to GP devices are ignored by the device.
(2) For all commands, each byte is formatted in little-endian mode.

Table 30-7. Command Parameters (continued)

Format[2]	Parameter	Size / First Byte	Definition
			The block address of successive write request commands must not have discontinuity (that is, $BlockAddress_{n+1}$ = Block- $Address_n$ + $BlockSize_n$ and $BlockAddress_0$=CODE_ADDR). For more information about memory mapping, see Section 30.4.7, *Memory Mapping.*
	Z	8 bits for each data	Block data
>w	None		Write ACK response
>W[X]	X	8 bits	Write NACK error code: • 0x01: Address error • 0x02: Bad block size • 0x03: The first block address does not correspond to the code address specified in the certificate. • 0x04: For HS devices: Firmware signature error For GP devices: Firmware hash error • 0x05: The received code size does not correspond to the code size specified in certificate/TOC. • 0x06: Error during hashing of the firmware
<a	None		Abort request
<c	None		Certificate request (HS devices only)
>c[X]	X	LSB first	Firmware certificate (HS devices only) For more information about the certificate, see Section 30.4.3, *Certificate Description.*

30.4.5 Memory Booting

30.4.5.1 Memory Booting Overview

Memory booting is the process of authenticating and executing code from external memory. The ROM code performs memory booting when the following occurs:

- Peripheral booting time-out expires (15 ms for UART or 3 s for USB).
- The ROM code is executed after a watchdog reset.

The firmware must be at the first address of the nCS3 (0x0600 0000), which is the chip-select pin always enabled in the LOCOSTO device. The NOR flash memory type is the only supported memory type by the EMIF at the nCS3 location.

30.4.5.2 Firmware Authentication

Authentication steps are valid only for HS devices. No authentication is involved for GP devices. The ROM code directly starts the firmware execution from the external memory space located at the nCS3 location.

The firmware authentication is similar to the flash loader authentication except for EMIF configuration and image check.

The ROM code configures EMIF with its reset values (worst-case configuration) before firmware authentication.

If hackers replace the manufacturer flash memory with their own, the firmware authentication fails because the hacker cannot sign the firmware certificate with the MPK. The manufacturer is responsible for preventing the MPK from being divulged.

Moreover, the mobile application always boots on the internal boot ROM, which cannot be bypassed because the JTAG and interrupts are disabled at reset.

The modification of external buses (address, data, RnW) cannot influence the behavior of the ROM code.

The following sequence is performed during firmware authentication (Figure 30-14 shows a flowchart of the sequence):

Step 1: The ROM code reads and authenticates the firmware.

Step 2: Only if the certificate authentication passes, the ROM code configures the EMIF registers with values in the configuration parameters of the firmware manufacturer certificate. To speed up the authentication phase and the hashing of the firmware, customize the following configuration parameters:

- Burst mode capability
- Number of wait states
- Dummy cycles needed to access the flash memory at 52 MHz

Step 3: After hardware configuration, the firmware signature is computed and compared with the signature embedded in the certificate after RSA decryption.

Step 4: After a successful firmware authentication, the DIE_ID field of the certificate and the DIE_ID registers are compared:

- If the DIE_ID field value is equal to 0, the manufacturer certificate is not uniquely associated to the given hardware platform.
- If the DIE_ID field value is equal to the DIE_ID registers value, the subsequent usage of the PLATFORM_DATA field value is valid. The firmware is informed that it can securely use the PLATFORM_ DATA field value.
- If the DIE_ID field value is not equal to the DIE_ID register value, using the PLATFORM_DATA field value is not recommended because it is not associated with the given hardware platform. Nevertheless, the firmware is informed.

Step 5: After a successful firmware authentication and integrity check, the firmware is executed with a parameter representing the status of the DIE_ID field check. The parameter is located in R0 register of the ARM processor with the following signification:

- DIE_ID field value equal to the DIE_ID registers: R0 = 1
- DIE_ID field value not equal to the DIE_ID registers: R0 = 4
- DIE_ID field value equal to 0: R0 = 5

The address of the firmware entry point is in the certificate.

Step 6: Image check

Hypothesis

The hacker externally copies the flash memory on a larger one and has inserted a virus in the upper area.

For example: the original flash memory size is 2M bytes, but the chip-select size is 4M bytes. The hacker can then force the ADD [21] bit to 1 to access the second 2M bytes instead of the first 2M bytes when the authentication phase is complete. This modification is possible because of the physical memory duplication when the chip-select is greater than the memory size.

Solution

The solution consists of authenticating the firmware and the firmware image located in the flash memory image in the chip-select (see Figure 30-13). The secure boot loader performs the authentication according to the parameters in the certificate (flash memory size, CS size configuration, code size to check, and image check enable). This method prevents the execution of hacker software.

Figure 30-13. Image Check Representation

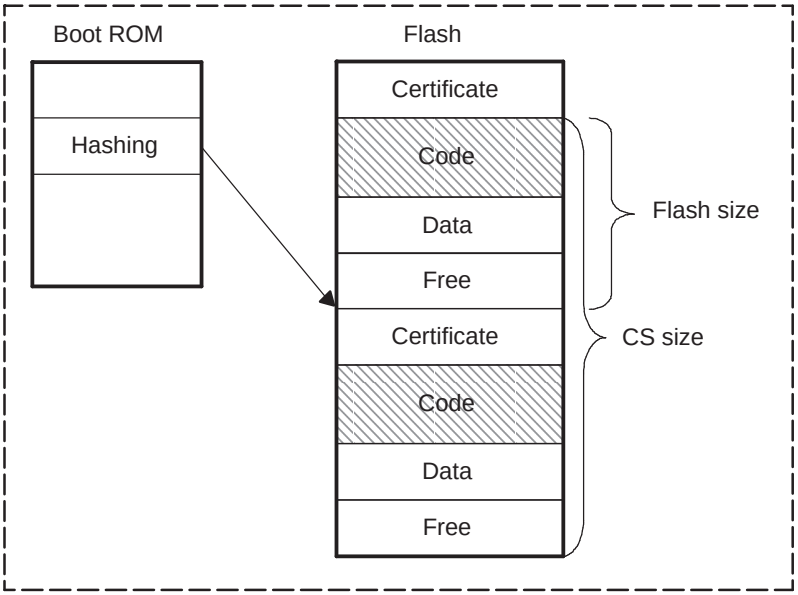
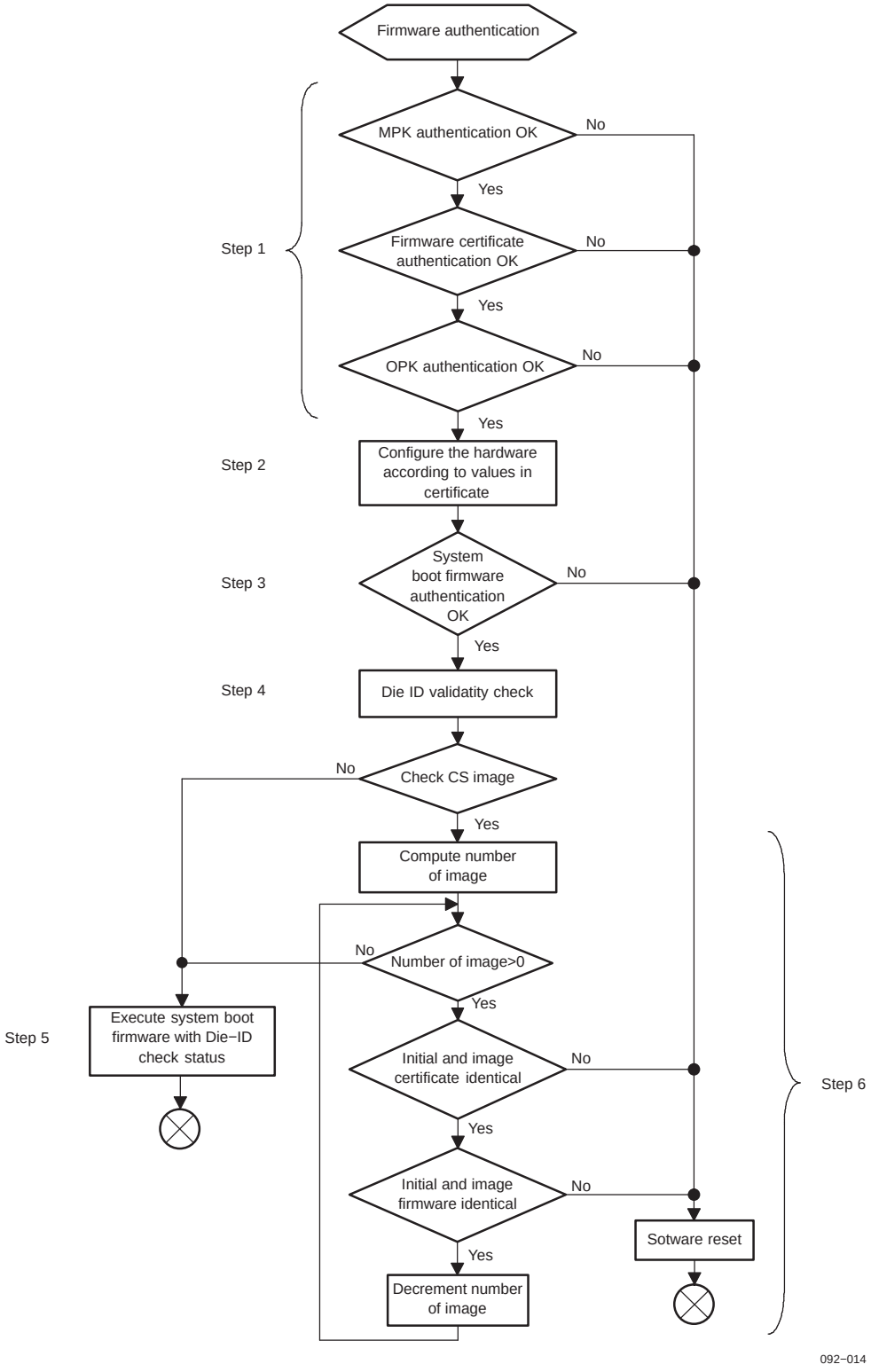


Figure 30-14. Firmware Authentication Flowchart



30.4.6 Interrupt Vectors Remapping

The ROM code is always at the address 0000:0000, which implies that the interrupt vectors must be managed in the ROM code. Nevertheless, the firmware needs these interrupts and thus the ROM code must provide access. To minimize the latency from remapping, only the address of the interrupt subroutine and the remapping instruction must be in the internal SRAM.

The start address of the internal SRAM is 0800:0000, which gives an offset of 128 MB to the boot ROM. This constraint implies that any instructions using the relative offset (B, BL) cannot be used to perform the remapping.

Moreover, to reduce the latency, the BX instruction cannot be used because it requires managing a register, which implies that this register must be previously saved in an interrupt stack. The only method (and the more flexible method) is to use the ADD instruction as follows:

ADD PC, PC, #0x08000000

The advantage of this solution is that only the PC register is updated. Nevertheless, the ARM pipeline must be considered. The ADD instruction executed at address 0000:0004 (undefined interrupt) forces the PC to 0800:000C. (The ARM7 pipeline is two instructions in 32 bits, which gives a difference of 8 bytes. During interrupt handling, the ARM is always in 32-bit mode.)

Table 30-8 details the memory mapping, including interrupt vector remapping.

In the Internal SRAM, the user must correctly set the addresses 0800:000C to 0800:0024 with two possibilities:

- Subroutines are in internal SRAM: Use a B instruction.
- Subroutines are not in internal SRAM: Use an LDR PC, [PC, #0x14] instruction.

  In this case, the addresses 0800:0028 to 0800:0040 must contain the address of the subroutine. This mechanism takes eight cycles per interrupt to perform the indirect call.

If an interrupt occurs during the ROM code execution (that is, an abort interrupt from hardware issues), the ARM executes the content of the address 0800:000C-0800:0024:

- On power-on reset, the 4M bits of SRAM are erased and then the ARM executes the instruction 0000:0000. As a consequence, the system dies and must be reset.
- On a watchdog reset, the 4M bits are not erased, depending on the PRRM configuration done by the firmware and if address 0800:0000 is configured. In this case, if an abort occurs during code fetch in the boot ROM, the abort is managed by the firmware in flash memory.

30.4.7 Memory Mapping

The first 1M bits of the internal SRAM and the full API memory are not used. This implies that the boot application data are in the upper part of internal SRAM and in the secure RAM.

Table 30-8. Memory Mapping

Address	Function	Code	Location
0000:0000	IT vectors	B _SecureBoot	Boot ROM
		ADD PC, PC, #0x08000000	
		ADD PC, PC, #0x08000000	
		ADD PC, PC, #0x08000000	
		ADD PC, PC, #0x08000000	
		ADD PC, PC, #0x08000000	
		ADD PC, PC, #0x08000000	
		ADD PC, PC, #0x08000000	
0000:0024	Service vector	MOV PC, #0x88	
0000:0028	Reserved	…	
0000:0080	ROM code identifier	…	
0000:0088	Service routine management	…	

Table 30-8. Memory Mapping (continued)

Address	Function	Code	Location
0000:00XX	ROM code	…	
0600:0000	Reserved	…	nCS3
0600:0XXX	Code	Firmware	
0600:0XXX	Data		
0600:0XXX	Free	…	
0800:0000	Free		Internal SRAM (1M bit)
0800:000C to 0800:0027	Indirect IT vectors (Not initialized by the ROM code)	LDR PC, [PC, #0x14] LDR PC, [PC, #0x14] LDR PC, [PC, #0x14] LDR PC, [PC, #0x14] LDR PC, [PC, #0x14] LDR PC, [PC, #0x14] LDR PC, [PC, #0x14] Note: The offset 0x14 considers the ARM 3-stage pipeline: 0x14=7*4–8	
0800:0028 to 0800:0043	Address of interrupt subroutine (not initialized by the ROM code)	Undefined subroutine address SWI subroutine address Abort prefetch subroutine address Abort data subroutine address Reserved subroutine address IRQ subroutine address FIQ subroutine address	

Keyboard Controller

This chapter introduces the keyboard controller of the LOCOSTO device.

Topic **Page**

31.1 Keyboard Controller Overview

The LOCOSTO keyboard controller implements a built-in scanning algorithm for hardware-based key press decoding and allows MPU software overhead reduction.

The keyboard controller can handle up to 5 × 5 keyboards, operates on a 32-kHz clock, and can generate wake-up events when the device is in sleep mode.

31.1.1 Main Features

The keyboard controller includes the following main features:

- Support of multiconfiguration keyboards up to 5 rows x 5 columns
- Integrated programmable timer
- Programmable interrupt (IT) generation on key events
- Event detection on both key press and key release
- Multikey press detection and decoding
- Long key detection on prolonged key press
- Programmable time-out on permanent key press or after keyboard release

31.1.2 Signals and I/O Description

Figure 31-1 shows a typical 5 _UnicodeEncodeError_ 5 keyboard connection to the LOCOSTO keyboard controller.

Figure 31-1. Keyboard Controller Overview

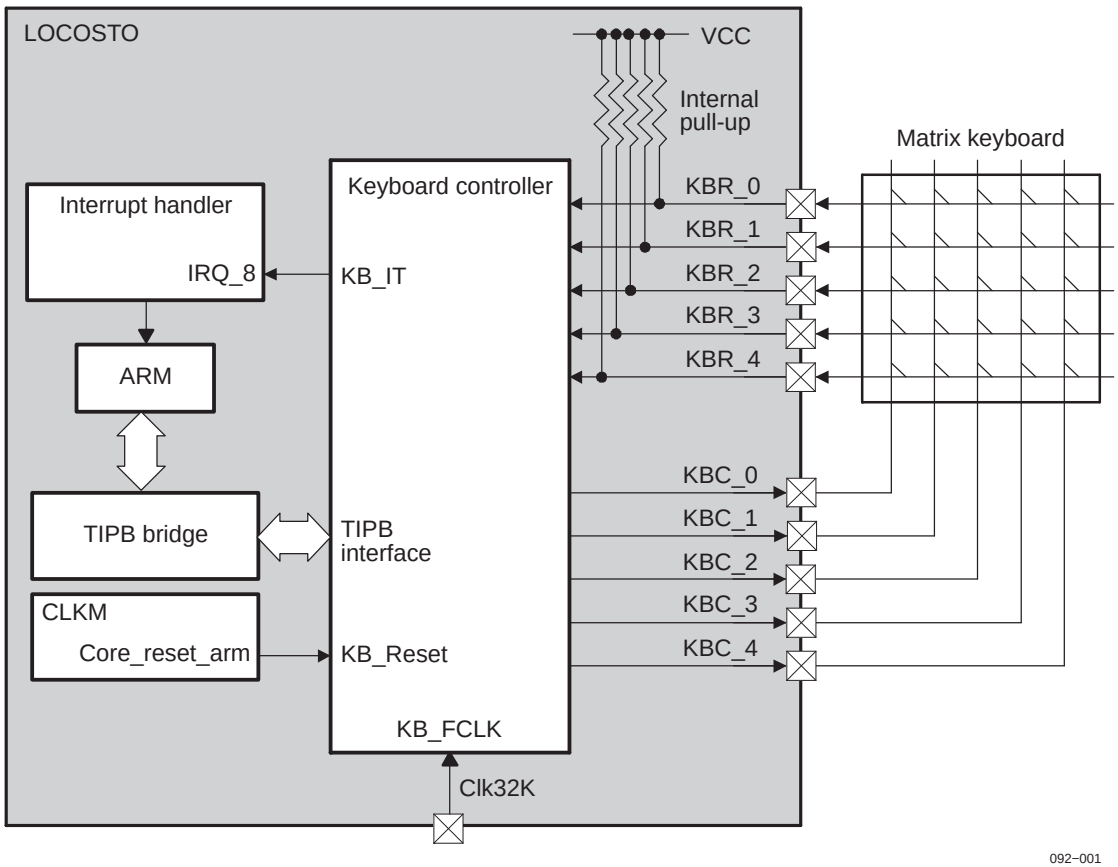


Table 31-1 describes the keyboard controller pins.

Table 31-1. I/O Description

Signal Name	I/O[1]	Description	Value at Reset
KBR_0	I	Keyboard matrix row 0 input	N/A
KBR_1	I	Keyboard matrix row 1 input	N/A
KBR_2	I	Keyboard matrix row 2 input	N/A
KBR_3	I	Keyboard matrix row 3 input	N/A
KBR_4	I	Keyboard matrix row 4 input	N/A
KBC_0	O	Keyboard matrix column 0 output	1
KBC_1	O	Keyboard matrix column 1 output	1
KBC_2	O	Keyboard matrix column 2 output	1
KBC_3	O	Keyboard matrix column 3 output	1
KBC_4	O	Keyboard matrix column 4 output	1

(1) I = Input, O = Output

For the pin configuration, see Chapter 18, *Configuration*.

31.1.3 Clocking, Reset, and Power Management Scheme

31.1.3.1 Clocks

The keyboard controller has two clock domains: the functional clock domain and the interface clock domain.

- KB_FCLK belongs to the functional clock domain and is a fixed 32-kHz clock coming from outside the LOCOSTO device. KB_FCLK clocks the internal module logic (see Chapter 6, *Power, Reset, and Clock Management*, for further details).

Note: The Clk32K clock remains active when the device is in sleep mode, thus allowing the keyboard controller to generate a wake-up event.

- KB_ICLK, the interface clock, triggers access to the keyboard controller registers via the TI peripheral bus (TIPB) (see Chapter 6, *Power, Reset, and Clock Management*, for further details).

Table 31-2 lists the clock domains of the keyboard controller.

Table 31-2. Clock Domains of the Keyboard Controller

Type	Name	Source	Frequency	Description
Functional	KB_FCLK	Clk32K	32 kHz	32-kHz clock
Interface	KB_ICLK	TIPB	52 MHz	Clock fed by the TIPB

31.1.3.2 Power Management and Power Domain

The keyboard controller belongs to the VDD_core power domain (see Chapter 6, *Power, Reset, and Clock Management*, for further details).

31.1.3.3 Hardware and Software Resets

Global reset of the keyboard controller module (KB_Reset) can be performed either by activating the ARM_nRST signal (see Chapter 6, *Power, Reset, and Clock Management*, for further details) or by setting the SOFTWARE_NRESET bit (CNTL_REG[0]) to 1. Setting this bit enables active software reset functionality equivalent to hardware reset.

31.1.3.4 Hardware Requests

The keyboard controller generates one interrupt signal, KB_IT, which is an interrupt to the MPU interrupt handler and is mapped on IRQ_8.

31.2 Keyboard Controller Functional Description

31.2.1 General Description

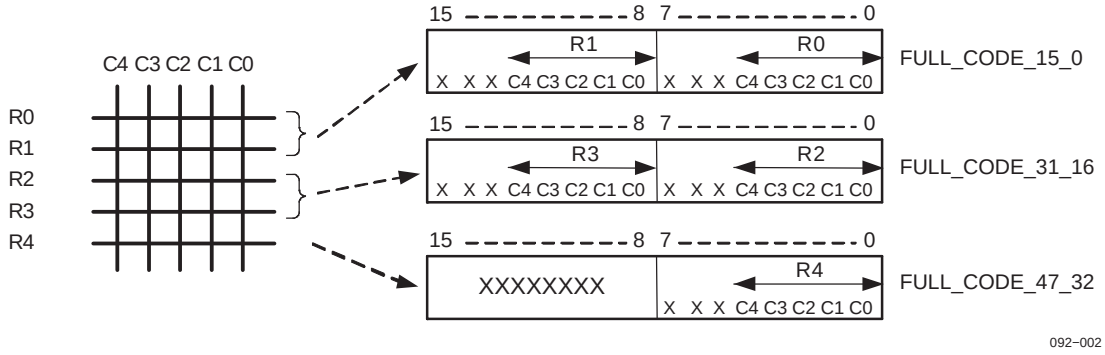
The keyboard controller detects events issued on any key of the connected keyboard and generates an interrupt to alert the host processor.

In addition, the built-in hardware scan algorithm performs decoding of pressed keys, including multikey combinations.

To reduce MPU software overhead, the detecting and decoding processes are hardware-performed within the keyboard controller state machine. Nevertheless, the hardware decoding can be deactivated and the scanning algorithm handled by software.

Figure 31-2 shows the keyboard controller architecture.

Figure 31-2. Keyboard Controller Block Diagram



31.2.2 Software Mode

The SOFTWARE_NRESET bit (CNTL_REG[1]) selects the hardware decoding (activated by default after reset) and software modes.

In the software mode, the keyboard controller internal sequencer, which performs automatic scanning and decoding processes, is disabled. Consequently, the software must manually perform the scanning algorithm.

The KBC_REG and KBR_LATCH registers perform the software scan. KBC_REG sets column output values; KBR_LATCH reflects row input values. After reset, all KBC_REG bits are set to 1, thus enabling the keyboard matrix columns (that is, driving the KBC[4:0] outputs to 0).

Any key press connects a column output (driving a logical 0) to a row input (internally pulled up to a logical 1) and generates an interrupt (KB_IT asserted low) to indicate that the software scan can start.

See Section 31.3, *Programming Model*, for details about the software scan.

31.2.3 Keyboard Controller Hardware Decoding Mode

31.2.3.1 Functional Modes

When running in hardware decoding mode (CNTL_REG[1] NSOFTWARE_ MODE bit set to 1), the keyboard controller offers several functional modes summarized in Table 31-3.

Table 31-3. Keyboard Controller Functional Modes

Functional Mode	Associated Interrupt	Associated Timer Value	Description	Control
Keyboard event	Event IT	Debouncing value	Occurs when a key is pressed or released Always enabled	N/A (always active)
Long key	Long key IT	Long key value	Used to detect a key that is pressed during a long time Must be associated with the long key time-out functionality	CNTL_REG[5] (LONG_KEY_EN)
Repeat key	Long key IT	Long key value	Generates an interrupt every long key delay No time-out can be associated	CNTL_REG[8] (REPEAT_MODE _EN)
Empty time-out	Time-out IT	Empty timeout value	Interrupt generated if no key is pressed during an empty time-out period	CNTL_REG[6] (TIME_OUT_ EMPTY_EN)
Long key time-out	Time-out IT	Long key time-out value	Associated with the long key functionality Generated after a long key IT if no event occurs during a long key time-out period	CNTL_REG[7] (TIME_OUT_ LONG_KEY_EN)

Each mode can be activated/deactivated by setting the corresponding bit in the CNTL_REG register (see Section 31.4, *Register Manual*, for details).

31.2.3.2 Keyboard Controller Timer

As introduced in the previous section, each functional mode is associated with a timer value. Depending on the selected mode, the keyboard controller timer is loaded with the corresponding value as set in the related registers: DEBOUNCING_TIME, LONG_KEY_TIME, and TIME_OUT.

Table 31-4 summarizes the keyboard controller timer values.

Table 31-4. Keyboard Controller Timer Values

Timer Value	Associated Register Field	Description
Debouncing time	DEBOUNCING_TIME[5:0] (DEBOUNCING_REG)	To remove the effects of glitches when an event occurs on the keyboard, the controller waits for a debouncing period before taking a snapshot of the keyboard matrix current state. The timer is loaded with the debouncing time value after each detected event on the keyboard matrix. An event IT is generated after this delay.
Long key time	LONG_KEY_TIME[11:0] (LONG_KEY_REG)	This is the delay before generating a long key IT after an event IT. The timer is loaded with the long key time value after an event IT is generated if the long key mode is selected. In repeat mode, the timer is reloaded with the same value after a long key IT and starts to count down again.
Long key time-out	TIME_OUT[15:0] (TIME_OUT_REG)	The timer is loaded with the time-out value after that a long key IT is generated and starts to count down. When the timer reaches 0, a time-out IT is generated and the keyboard controller comes back to its idle state. This long key time-out does not work in repeat mode.
Empty key time-out	TIME_OUT[15:0] (TIME_OUT_REG)	The time-out IT occurs if no key is pressed during this delay. The keyboard controller then returns to the idle state.

The timer countdown period depends on three factors:

- The loaded value as set in the DEBOUNCING_TIME, LONG_KEY_TIME, and TIME_OUT registers
- The prescale clock timer value as set in the CNTL_REG[4:2] register PTV field. This programmable clock divider allows the clock frequency used by the timer to be reduced.
- The keyboard controller functional clock frequency (32 kHz)

The period is calculated as follows:

$T_{period} = (T_{value} + 1)$ _UnicodeEncodeError_ $2^{PTV + 1}$ _UnicodeEncodeError_ T_{clk}

Where:

T_{value} is the value stored in the DEBOUNCING_TIME, LONG_KEY_TIME, or TIME_OUT register.

PTV is the value of the CNTL_REG[4:2] register PTV bit field.

T_{clk} is the period of the 32-kHz functional clock (that is, 31.25 _UnicodeEncodeError_s).

Note: The timer minimum period is then 62.5 _UnicodeEncodeError_s and its maximum period is 524.288 sec.

CAUTION

To avoid undefined results, the CNTL_REG[4:2] register PTV bit field must not be changed when the timer is running.

The timer value registers (DEBOUNCING_TIME, LONG_KEY_TIME, and TIME_OUT) can be updated at any time. Depending on the updated register, two cases can occur:

- The LONG_KEY_TIME register is updated the new value stored in the LONG_KEY_TIME register is loaded into the timer as soon as the register is written. If the timer is already counting down when LONG_KEY_TIME is updated, the countdown stops and then restarts from the new loaded value.
- The DEBOUNCING_TIME or TIME_OUT register is updated the new value is considered only at the next timer load. If the timer is counting down when the registers are updated, the timer continues counting down from the previous value and is loaded with the new one on the next load.

Whatever the timer state (stopped, counting down any of the values previously described, etc.), when a new event occurs on the keyboard, the timer is stopped and loaded with the debouncing time value. It then starts counting down.

31.2.3.3 Keyboard Controller Interrupt Generation

31.2.3.3.1 Interrupt Generation Scheme

The keyboard controller generates one interrupt signal (KB_IT) to an external interrupt handler.

In addition, each functional mode generates dedicated IT events (logged in the INTERRUPT_STATUS_REG register) that can be masked or unmasked via the INTERRUPT_ENABLE register.

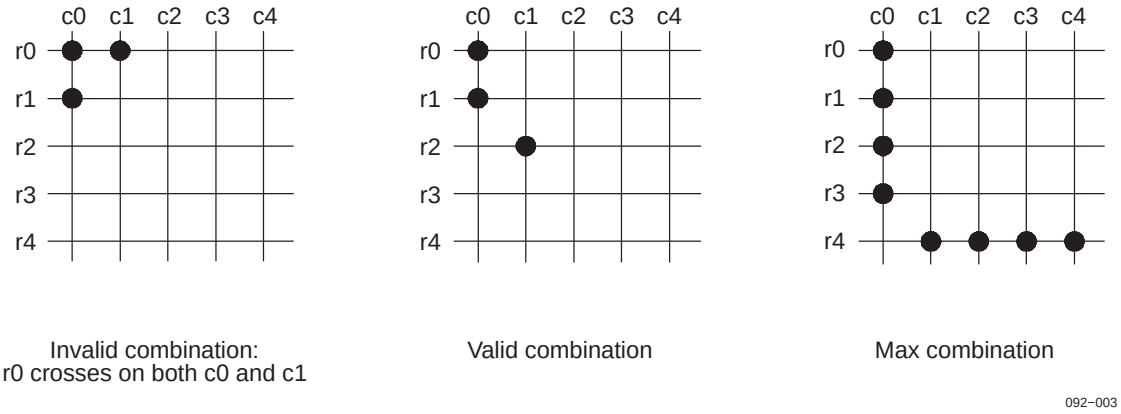
Each time the selected functional mode generates an IT event, the INTERRUPT_STATUS_REG is updated. However, the KB_IT signal is asserted only on the related event when the corresponding bit has been set in the INTERRUPT_ENABLE register.

Note: The INTERRUPT_STATUS_REG read-only register can be cleared only by writing the corresponding bit in the CLEAR_INTERRUPT_STATUS_REG register.

Figure 31-3 shows the different IT events generated in each keyboard controller functional mode and details the relationships between them.

Note: Depending on the selected mode, some IT events cannot be generated.

Figure 31-3. Functional Modes and Related IT Events



While running in hardware decoding mode, the keyboard controller performs automatic scans when not in IDLE state. As soon as a key press event occurs on the keyboard matrix, the keyboard controller leaves the IDLE state and an IT event (INTERRUPT_STATUS_REG[0] register IT_EVENT bit) is set after the timer counts down the debouncing delay. An IT_EVENT is generated as many times as a key is pressed or released.

Note: An IT_EVENT is generated regardless of the selected functional mode. If no time-out is set and no more keys are pressed, the keyboard controller goes back to IDLE state.

In addition, if the long key detection mode is set, once the timer counts down the long key delay, an IT long key (INTERRUPT_STATUS_REG[1] register IT_LONG_KEY bit) is generated. If the repeat mode is set, the IT_LONG_KEY interrupt is generated periodically during every long key delay.

A time-out can also be set in event detection or long key detection mode. In this case, a time-out IT (INTERRUPT_STATUS_REG[2] register IT_TIME_OUT bit) is generated after the time-out delay timer expires. After such an interrupt, the keyboard controller always goes back to the IDLE state.

Note: No time-out can be set in repeat mode. Only a keyboard event can stop the periodic interrupt generation.

31.2.3.3.2 Keyboard Buffer and Missed Events

A dedicated buffer allows the keyboard controller to memorize two successive events. If two successive events occur before a read is performed, the second event is stored and a second interrupt is generated as soon as the first interrupt is cleared, allowing two consecutive key events to be received.

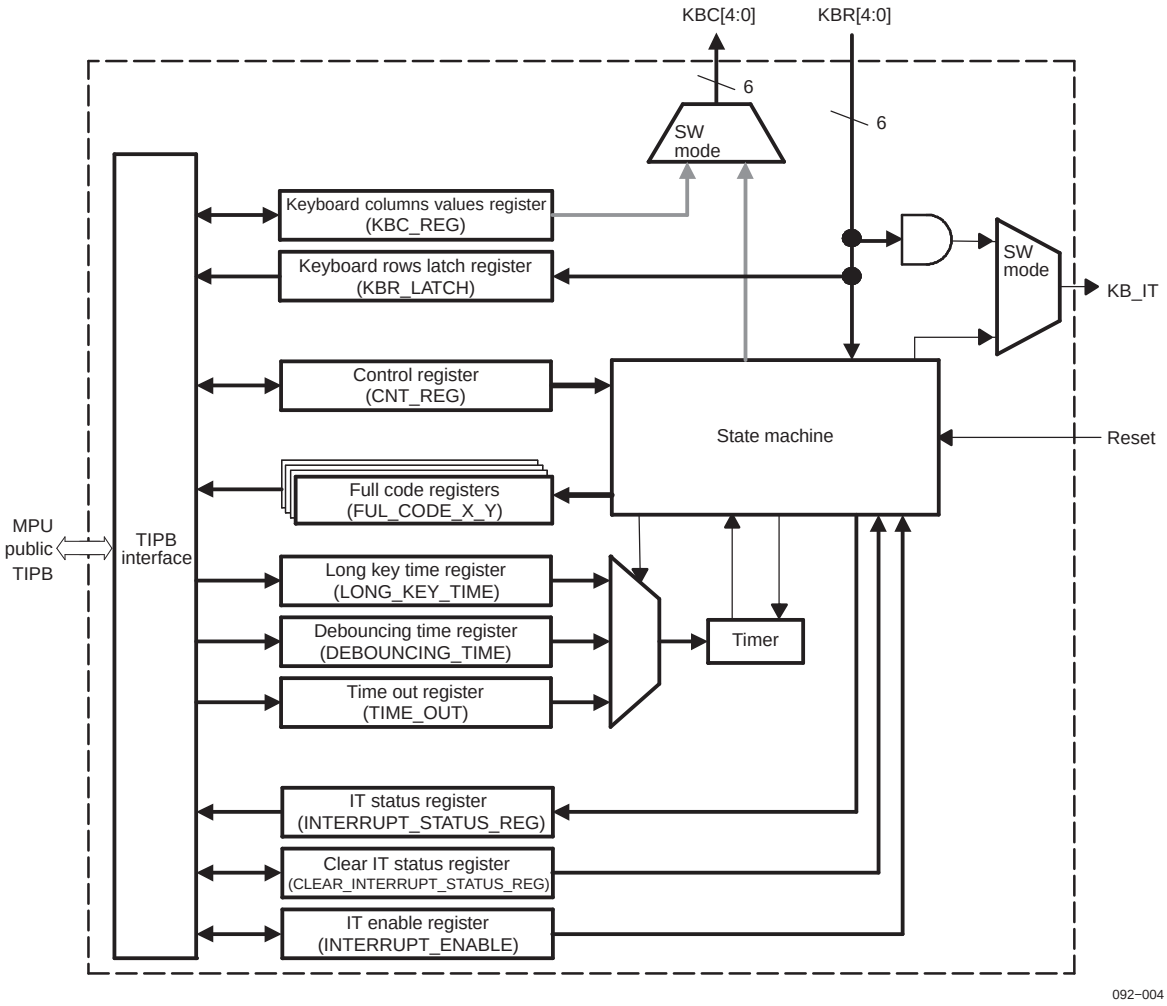
Nevertheless, if a third event occurs before the first event is treated, a missed event IT (INTERRUPT_STATUS_REG[3] register MISS_EVENT bit) is generated to report the lost event.

Note: The MISS_EVENT interrupt is not routed to the KB_IT signal; the software must check the INTERRUPT_STATUS_REG[3] register MISS_EVENT bit to detect any missed events.

31.2.3.4 Keyboard Controller Key Coding Registers

The keyboard controller matrix pressed keys state (that is, which columns/ rows are connected) is reflected in the FULL_CODE_15_0, FULL_CODE_ 31_16, and FULL_CODE_47_32 registers, as shown in Figure 31-4.

Figure 31-4. Key Coding Registers



Each register stores the state of two keyboard rows, the 8 least significant bits (LSBs) giving one row state, and the 8 most significant bits (MSBs) giving the other state. For a given row (that is, 8 MSBs or 8 LSBs in a full code register), each bit corresponds to a keyboard column. A bit set to 1 means that the corresponding key is pressed.

Note: When using a smaller keyboard (for example, 4 x 4), some bits are not used.

31.3 Programming Model

31.3.1 Using the Keyboard Controller in Software Scanning Mode

To deactivate the internal keyboard controller sequencer and use the module in software mode, perform the following steps:

1. Set the CNTL_REG[1] register NSOFTWARE_MODE bit to 0 (not the default mode after reset).
2. Wait for KB_IT.
3. When KB_IT is generated, indicating an event on the keys, begin the software scan:
   a. Disable all the columns to drive a logical 1 on the KBC[4:0] outputs settings KBC_REG[4:0] to 0x0.
   b. Enable the first column to set the corresponding bit to 1 in the KBC_REG register.
   c. Read the KBR_LATCH register.
      A read of 0 in KBR_LATCH means the corresponding row is connected to the column being enabled.
   d. Repeat steps b and c for all columns.

Note: In software mode, during the manual keyboard scan, an interrupt is generated each time a pressed key is detected.

31.3.2 Using the Keyboard Controller in Hardware Decoding Mode (default setting)

The hardware mode is the default mode after reset (the CNTL_REG[1] register NSOFTWARE_MODE bit set to 1). To configure the keyboard controller, perform the following steps:

1. Select the functional mode.
   After reset, all available functional modes are disabled, except the detect event mode, which is always active. To activate a functional mode, set the corresponding bit to 1 in the CNTL_REG register:
   - The LONG_KEY_EN bit (CNTL_REG[5]) activates/deactivates the long key detection mode.
   - The TIME_OUT_EMPTY_EN bit (CNTL_REG[6]) activates/ deactivates the empty time-out mode.
   - The TIME_OUT_LONG_KEY_EN bit (CNTL_REG[7]) activates/ deactivates the long key time-out mode.
   - The REPEAT_MODE_EN bit (CNTL_REG[8]) activates/deactivates the repeat mode.

   Note: Because the long key detection mode and the repeat mode share the same interrupt status bit and are mutually exclusive, they cannot be used simultaneously. The software must ensure that only one of these modes is selected at a time.

2. Set the timer periods (see Section 31.3.3, *Using the Timer*, for timer programming details).
   The LONG_KEY_TIME[11:0] register LONG_KEY_REG bit field is used in both the long key detection and repeat modes. In the long key detection mode, the register field determines the delay required to consider a key press as a long press. In the repeat mode, the register field sets the recursive interrupt period.
   The TIME_OUT[15:0] register TIME_OUT_REG bit field is used for both the empty time-out and long key time-out modes. The same value is applied when the two modes are enabled.
   The DEBOUNCING_TIME[5:0] register DEBOUNCING_REG bit field is used with any selected functional mode. The debouncing countdown is launched after any key event (key pressed or released).
3. Configure the keyboard controller interrupt line.
   As described in *Keyboard Buffer and Missed Events* in Section 31.2.3.3, each functional mode generates a dedicated interrupt bit logged in the INTERRUPT_STATUS_REG register. To activate the interrupt line and trigger KB_IT on one/several of these events, the corresponding bit in the INTERRUPT_ENABLE register must be set to 1.

Note: After reset, all events are enabled by default.

After configuration, the keyboard controller is ready to operate. To decode any event on the keyboard, perform the following steps:

1. Wait for KB_IT to be asserted.

   Whichever mode is chosen, any event on the keyboard matrix generates an IT_EVENT interrupt (INTERRUPT_STATUS_REG[0] register IT_EVENT bit set to 1). The selected mode(s) determine whether or not the other bits are set.
2. Read the INTERRUPT_STATUS_REG register to determine which event caused the interrupt.
3. Read the FULL_CODE_15_0, FULL_CODE_31_16, and FULL_CODE_ 47_32 registers to determine which keys were pressed.
4. Write the CLEAR_INTERRUPT_STATUS_REG to clear the corresponding bit(s) in the INTERRUPT_STATUS_REG register.

Note: When two events occur successively before the first event is read, the second interrupt is generated as soon as the first interrupt is cleared (that is, the corresponding bit is set to 1 in the CLEAR_INTERRUPT_ STATUS_REG). When more than two events occur in a row, the MISS_EVENT bit (INTERRUPT_STATUS_REG[3]) is set. The software must check this bit, which is not reflected on the KB_IT line.

31.3.3 Using the Timer

As described in Section 31.2, *Keyboard Controller Functional Description*, the keyboard controller timer is loaded with the DEBOUNCING_TIME, LONG_KEY_TIME, or TIME_OUT register content, depending on the selected mode and the state of the internal sequencer.

The actual countdown time is calculated as follows:

$T_{period} = (T_{value} + 1)$ _UnicodeEncodeError_ $2^{PTV + 1}$ _UnicodeEncodeError_ T_{clk}

The PTV bit field (CNTL_REG[4:2]) allows the timer clock to be predivided. Table 31-5 lists the divider rates.

Table 31-5. Timer Prescale Values

PTV (CNTL[4:2]) Value	Divisor
0	2
1	4
2	8
3	16
4	32
5	64
6	128
7	256

CAUTION

To avoid undefined results, the CNTL_REG[4:2] register PTV bit field must not be changed when the timer is running.

The DEBOUNCING_TIME, LONG_KEY_TIME, and TIME_OUT registers can be updated at any time, whether or not the timer is running. Nevertheless, the timer is updated only on-the-fly for the long key time value. The new debouncing and time-out values are loaded only on the next load.

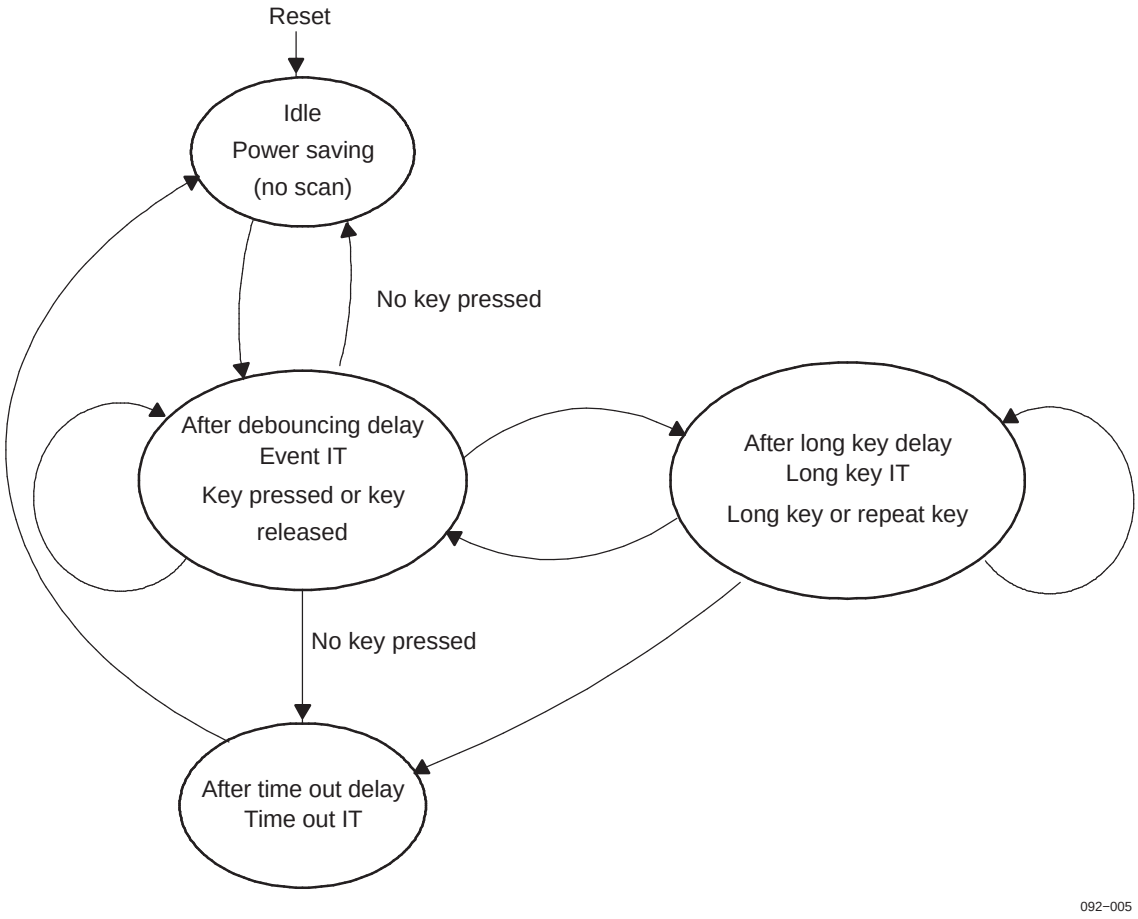
31.3.4 Multikey Limitations

The keyboard controller detects and decodes multikey combinations using the following rules:

- Any 2-key combination is valid and can be decoded.
- Combinations using more than two keys are valid only if the rows and columns used do not cross over on another key to detect. This is due to the equipotential propagation on a row/column.

Figure 31-5 shows an example of a valid 3-key combination, a nonvalid 3-key combination, and the largest combination allowed for a 5 _UnicodeEncodeError_ 5 keyboard.

Figure 31-5. Multikey Limitation Example



31.4 Register Manual

Table 31-6 summarizes the keyboard module instance.

Table 31-6. Instance Summary

Module Name	Base Address	Size
Keyboard	0xFFFE B800	2K byte

31.4.1 Keyboard Controller Register Mapping Summary

Table 31-7 summarizes the keyboard controller register address offset.

Table 31-7. Keyboard Controller Register Address Offset

Register Name	Type	Register Width (Bits)	Offset
CNTL_REG	R/W	16	0x00
DEBOUNCING_TIME	W	16	0x02
LONG_KEY_TIME	W	16	0x04
TIME_OUT	W	16	0x06
INTERRUPT_STATUS_REG	R	16	0x08
CLEAR_INTERRUPT_ STATUS_REG	R/W	16	0x0A
INTERRUPT_ENABLE	R/W	16	0x0C
Reserved	N/A	N/A	0x0E
KBR_LATCH	R	16	0x10
KBC_REG	R/W	16	0x12
FULL_CODE_15_0	R	16	0x14
FULL_CODE_31_16	R	16	0x16
FULL_CODE_47_32	R	16	0x18

31.4.2 Registers Description

Table 31-8 through Table 31-19 describe the keyboard controller register bits.

Table 31-8. CNTL_REG

Address Offset	0x00		
Physical Address	0xFFFE B800	**Instance**	Keyboard
Description	Keyboard controller control register		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							REPEAT_MODE_EN	TIME_OUT_LONG_ KEY_EN	TIME_OUT_EMPTY_ EN	LONG_KEY_EN	PTV			NSOFTWARE_MODE	SOFTWARE_NRESET

Bits	Field Name	Description	Type	Reset
15:9	Reserved	Reserved	-	-
8	REPEAT_MODE_EN			
7	TIME_OUT_LONG_ KEY_EN	Enables time-out long key detection 0x0: Time-out long key detection disabled 0x1: Time-out long key detection enabled	R/W	0x0
6	TIME_OUT_EMPTY_ EN	Enables time-out empty detection 0x0: Time-out empty detection disabled 0x1: Time-out empty detection enabled	R/W	0x0
5	LONG_KEY_EN	Enables long key process detection 0x0: Long key process detection disabled 0x1: Long key process detection enabled	R/W	0x0
4:2	PTV	Prescale clock timer value	R/W	0x7
1	NSOFTWARE_MODE	Enables software mode 0x0: Software mode (using kbr_latch_reg and kbc_reg) enabled 0x1: Hardware decoding (using internal sequencer) enabled	R/W	0x1
0	SOFTWARE_NRESET	Software reset 0x0: Reset 0x1: Normal operation	R/W	0x1

Table 31-9. DEBOUNCING_TIME

Address Offset	0x02		
Physical Address	0xFFFE B802	**Instance**	Keyboard
Description	Debouncing time value		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DEBOUNCING_REG							

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Reserved	-	-
5:0	DEBOUNCING_REG	Debouncing time value	W	0x0

Table 31-10. LONG_KEY_TIME

Address Offset	0x04		
Physical Address	0xFFFE B804	**Instance**	Keyboard
Description	Specifies the long key detection time in long key mode or the interrupt frequency in repeat mode		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				LONG_KEY_REG											

Bits	Field Name	Description	Type	Reset
15:12	Reserved	Reserved	-	-
11:0	LONG_KEY_REG	Long key detection time value (long key mode) or interrupt recursive period value (repeat mode)	W	0x0

Table 31-11. TIME_OUT

Address Offset	0x06		
Physical Address	0xFFFE B806	**Instance**	Keyboard
Description	Specifies the time-out value on permanent key press or after keyboard release		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIME_OUT_REG															

Bits	Field Name	Description	Type	Reset
15:0	TIME_OUT_REG	Time-out value	W	0x0

Table 31-12. INTERRUPT_STATUS_REG

Address Offset	0x08		
Physical Address	0xFFFE B808	**Instance**	Keyboard
Description	Interrupt status register. Each bit is cleared by a write on the corresponding bit of the clear status interrupt register.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												MISS_EVENT	IT_TIME_OUT	IT_LONG_KEY	IT_EVENT

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	-	-
3	MISS_EVENT	Miss event interrupt status 0x0: No miss event 0x1: Miss event has occurred.	R	0x0
2	IT_TIME_OUT	Time-out interrupt status 0x0: Time-out interrupt inactive 0x1: Time-out interrupt active	R	0x0
1	IT_LONG_KEY	Long key interrupt status 0x0: Long key interrupt inactive 0x1: Long key interrupt active	R	0x0
0	IT_EVENT	Event interrupt status 0x0: Event interrupt inactive 0x1: Event interrupt active	R	0x0

Table 31-13. CLEAR_INTERRUPT_STATUS_REG

Address Offset	0x0A		
Physical Address	0xFFFE B80A	**Instance**	Keyboard
Description	Clear interrupt status register. A write on the related bit clears the corresponding interrupt status bit in INTERRUPT_STATUS_REGISTER.		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CLEAR_MISS_EVENT	CLEAR_IT_TIME_OUT	CLEAR_IT_LONG_KEY	CLEAR_IT_EVENT

Bits	Field Name	Description	Type	Reset
15:4	Reserved	Reserved	-	-
3	CLEAR_MISS_EVENT	Clears miss event interrupt status bit 0x0: No effect 0x1: Clears the miss event interrupt status bit	R/W	0x0
2	CLEAR_IT_TIME_OUT	Clears time-out interrupt status bit 0x0: No effect 0x1: Clears the time-out interrupt status bit	R/W	0x0
1	CLEAR_IT_LONG_KEY	Clears long key interrupt status bit 0x0: No effect 0x1: Clears the event interrupt status bit	R/W	0x0
0	CLEAR_IT_EVENT	Clears event interrupt status bit 0x0: No effect 0x1: Clears the event interrupt status bit	R/W	0x0

Note: CLEAR_INTERRUPT_STATUS_REGISTER is auto-cleared when the corresponding bit clear is completed in INTERRUPT_STATUS_REGISTER.

Table 31-14. INTERRUPT_ENABLE

Address Offset	0x0C		
Physical Address	0xFFFE B80C	**Instance**	Keyboard
Description	Enables or disables the interrupt sources		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													IT_TIME_OUT_VALID	IT_TIME_OUT_VALID	IT_EVENT_VALID

Bits	Field Name	Description	Type	Reset
15:3	Reserved	Reserved	-	-
2	IT_TIME_OUT_VALID	Enables the time-out interrupt 0x0: Time-out interrupt disabled 0x1: Time-out interrupt enabled	R	0x1
1	IT_TIME_OUT_VALID	Enables long key interrupt 0x0: Long key interrupt disabled 0x1: Long key interrupt enabled	R	0x1
0	IT_EVENT_VALID	Enables event interrupt 0x0: Event interrupt disabled 0x1: Event interrupt enabled	R	0x1

Table 31-15. KBR_LATCH

Address Offset	0x10		
Physical Address	0xFFFE B810	**Instance**	Keyboard
Description	Keyboard row inputs status		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								KBR_LATCH							

Bits	Field Name	Description	Type	Reset
15:5	Reserved	Reserved	-	-
4:0	KBR_LATCH	Reflects the row inputs. Bit 0 corresponds to row 0, etc.	R	N/A

Table 31-16. KBC_REG

Address Offset	0x12		
Physical Address	0xFFFE B812	**Instance**	Keyboard
Description	Keyboard column outputs values		
Type	R/W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								KBC_REG							

Bits	Field Name	Description	Type	Reset
15:5	Reserved	Reserved	-	-
4:0	KBC_REG	Reflects the row inputs. Bit 0 corresponds to row 0, etc.	R/W	0x1F

Table 31-17. FULL_CODE_15_0

Address Offset	0x14		
Physical Address	0xFFFE B814	**Instance**	Keyboard
Description	Pressed keys status register 1		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			FULL_CODE_12_8					Reserved			FULL_CODE_4_0				

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	-	-
12:8	FULL_CODE_12_8	A bit at 1 indicates that the corresponding key is pressed.	R	0x0
7:5	Reserved	Reserved	-	-
4:0	FULL_CODE_4_0	A bit at 1 indicates that the corresponding key is pressed.	R	0x0

Table 31-18. FULL_CODE_31_16

Address Offset	0x16		
Physical Address	0xFFFE B816	**Instance**	Keyboard
Description	Pressed keys status register 2		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			FULL_CODE_28_24					Reserved			FULL_CODE_20_16				

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	-	-
12:8	FULL_CODE_28_24	A bit at 1 indicates that the corresponding key is pressed.	R	0x0
7:5	Reserved	Reserved	-	-
4:0	FULL_CODE_20_16	A bit at 1 indicates that the corresponding key is pressed.	R	0x0

Table 31-19. FULL_CODE_47_32

Address Offset	0x18		
Physical Address	0xFFFE B818	**Instance**	Keyboard
Description	Pressed keys status register 3		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			FULL_CODE_44_40					Reserved			FULL_CODE_36_32				

Bits	Field Name	Description	Type	Reset
15:13	Reserved	Reserved	-	-
12:8	FULL_CODE_44_40	A bit at 1 indicates that the corresponding key is pressed.	R	0x0
7:5	Reserved	Reserved	-	-
4:0	FULL_CODE_36_32	A bit at 1 indicates that the corresponding key is pressed.	R	0x0

Application Notes Reference

Application Note Title	Deescription	Reference Number
Locosto Debug Features	This document presents the debug functionalities on the LOCOSTO platform (emulation, debug unit, debug module).	APN211
Locosto LCD Driver Customization Guidelines	This document provides information on the changes required in the LCD driver module to support the new LCD hardware modules in the LOCOSTO device.	SWCA008
Imaging Application Note	This document describes a stand-alone camera viewfinder QCIF image and snapshot VGA image system from data capture to image display using the on-chip camera I/F, DMA, and LCD I/F.	APN212
Locosto Memory Interface Software Programming Guidelines	This document provides user guidelines for configuration and use of the LOCOSTO EMIF.	APN213
Deep Sleep Application Note	This document provides an example of a procedure to place the LOCOSTO device in idle mode.	APN210
APC Application Note	This document presents the APC module located in the DRP wrapper. The module controls the PA RAMP and the PA output power level.	NA
DRP Control Using the TPU Application Note	This document presents a programming sequence and associated timings to control the LOCOSTO DRP using TPU drivers.	APN223

Glossary

2D: — Two Dimensional

3D — Three Dimensional

3GPP: — 3rd Generation Partnership Project

AAC: — Advance Audio Coding

ABB: — Analog Baseband

ABU: — Autobuffering Unit

AC97 — Audio Codec 1997.
A standard audio interface that defines a highquality 16-bit audio architecture.

ACB: — AC-Bias Frequency

ACBI: — AC-Bias Line Transitions per Interrupt

ACE: — ASIC Compiler Environment.
The graphical user interface delivery mechanism for submicron gate array memory compiler elements.

ADC: — Analog-to-Digital Converter/Conversion

AES: — Advanced Encryption Standard

AHB: — Advanced High-Performance Bus

AIC: — Analog Interface Chip

ALE: — Address Latch Enable

AMR: — Adaptive Multirate

AMR: — Audio Modem Riser.
An Intel specification that defines a new architecture for the design of motherboards.

ANSI: — American National Standards Institute

APC: — Automatic Power Control

APE: — Application Engine

API: — Application Programming Interface

API: — Arm Port Interface

APLL: — Analog Phase-Locked Loop

APM: — Active Protected Mode

AR: — Automatic Reload

ARGB: — Alpha, Red, Green, Blue

ARM: — Advanced RISC Machine

ARMIO: — Advanced RISC Machine Input/Output. See MPUIO.

ASIC: — Application-Specific Integrated Circuit.

A chip built for a particular application. In the context of this document, this refers to the FPGA that resides on the EVM board.

ASP: — Application-Specific Peripheral

ASSP: — Application-Specific Silicon Product or Application-Specific Standard Product

ATR: — Answer to Reset

AuSPI: — Audio Serial Port Interface

B — Byte, 8 bits

B_MMU: — Measure of the rate at which signals are transmitted over a telecommunications link.

BB: — Busy Bus

BCD — Binary-Coded Decimal.

A representation of decimal digits (0–9) using a nibble that uses a certain number of bits. For example, by using 4 bits, two BCDs can be packed into one byte.

BCM: — BIST Combiner Module

BCPM: — BIST Controller Programmable Module

BE: — Big Endian.

An addressing protocol in which bytes are numbered from left to right within a word. More significant bytes in a word have lower numbered addresses. Endian ordering is specific to hardware and is determined at reset. See also little endian (LE).

BIOS: — Built-In Operating System

BGA: — Ball Grid Array

BGT: — Block Guard Time

BIST: — Built-In Self-Test

Block: — A group of interconnected cells. May contain instances of other blocks.

Bluetooth: — A short-range radio technology aimed at simplifying communications among network devices and between devices and the Internet. It also aims to simplify data synchronization between network devices and other computers.

BNC — British Naval Connector, Bayonet Nut Connector, or Bayonet Neill Concelman.

A type of connector used with coaxial cables.

BOF: — Beginning of Frame

BPP: — Bits Per Pixel

BSP: — Buffered Serial Port.

An on-chip module that consists of a full-duplex, double-buffered serial port interface and an autobuffering unit (ABU). The double-buffered serial port of the BSP is an enhanced version of the standard serial port interface. The double-buffered serial port allows transfer of a continuous communication stream (8-,10-,12- or 16-bit data packets).

BTA: — Bus Turn Around

BWT: — Block Waiting Time

C_MMU: — Cacheable Memory Management Unit.

See also MMU.

C-port: — Codec Port

CamDMA: — Camera Subsystem Direct Memory Access Module

CASP: — Customer Application–Specific Peripheral

CB — Copy Back

CB — Cross Bar

CBC: — Cipher Block Chaining

CCP: — Compact Camera Port

CDMA: — Code Division Multiple Access

CDP: — Coprocessor Data Operation

CE: — Chip Enable

Certificate: — Data block that is digitally signed with private key. Used enabling or disabling certain functionalities.

CFC: — CompactFlash Controller

CGT: — Character Guard Time

CIF: — Common Intermediate Format.

A video format used in video conferencing systems that easily supports both NTSC and PAL signals. CIF is part of the ITU H.261 video conferencing standard. It specifies a data rate of 30 frames per second (fps), with each frame containing 288 lines and 352 pixels per line.

CIO: — Channel Input/Output

CLE: — Command Latch Enable

CLK: — Clock

CLKM: — Clock and Reset Management

CLKSTP: — Clock Stop

Clock Gating: —Removing the ac frequency of the clock, usually through a logic (AND, OR) gate.

Clock Throttling: —Reducing the frequency of the clock

CLUT: — Color Look-Up Table

CMOS: — Complimentary Metal Oxide Semiconductor

CMT: — Cellular Mobile Telephone

CO: — Controlled Oscillator

CODEC: — Coder/Decoder or Compression/Decompression.

A device that codes in one direction of transmission and decodes in another direction of transmission.

COFF: — Common Object File Format

COM: — Communication

Companding: — Compressing and expanding.

A quantization scheme for audio signals in which the input signal is compressed and then, after processing, is reconstructed at the output by expansion. There are two distinct companding schemes: A-law, used in Europe, and ì-law, used in the United States.

ConnID: — Connection Identifier.

An initiator module identifier. A ConnID is transmitted in-band with the request and is used for security and error logging mechanism.

CP15: — Coprocessor 15.

This coprocessor controls the operation and configuration of the TI925T.

CPLD: — Complex Programmable Logic Device.

An integrated circuit that can be programmed to perform complex functions.

CPR: — Clock, Power, Reset

CPU: — Central Processing Unit.

The CPU is the portion of the processor involved in arithmetic, shifting, and Boolean logic operations, as well as the generation of data and program memory addresses. The CPU includes the central arithmetic logic unit (CALU), the multiplier, and the auxiliary register arithmetic unit (ARAU).

CRC — Cyclic Redundancy Check

CS: — Chip-Select

CSMI: — Coprocessor-Shared Memory Interface

CSMM: — Coprocessor-Shared Memory Model

CSRP: — Customer Software Restoration Point

CTRL: — Contro

CTS: — Clear to Send

CWT: — Command Wire Tracer

DABORT: — Data Abort

DAC — Digital to Analog Converter

DAI: — Digital Audio Interface.

A GSM test interface that is used to determine the routing of speech data for the devices being tested.

DARAM: — Dual Access Random Access Memory.

RAM that can be accessed twice in a single CPU clock cycle. For example, your code can read and write to DARAM in the same clock cycle.

DBB: — Digital Baseband

D-CACHE: — Data Cache

DCD: — Data Carrier Detect

DCDL: — Digitally-Controlled Delay Line

DCT: — DCT: Discrete Cosine Transform

A fast Fourier transform used in manipulating compressed still and moving picture data.

DCXO: — Digitally-Controlled Crystal Oscillator

DDMA: — DSP Subsystem Direct Memory Access Module

DDR: — Dual Data Rate

Default Mode: — The default mode is the mode selected at the release of the power on reset.

DES: — Data Encryption Standard

DES3DES: — Data Encryption Standard and Triple DES

DFA: — Differential Frequency Analysis

DFF: — Digital Flip-Flop

DI: — Data In

Die ID: — A 64-bit register composed of eFuse cells programmed during the fabrication process.

DIF: — Display Interface

DIM: — Dynamic Identification Mechanism

DIMM: — Dual Inline Memory Module

DIN: — Data In

DIP: — Dual Inline Package

DISPC: — Display Controller

DLB: — Data Loopback.

A synchronous serial port test mode in which the receive pins are connected internally to the transmit pins on the same device. This mode, enabled or disabled by the DLB bit, allows you to test whether the port is operating correctly.

DLL: — Delay-Locked Loop

DMA: — Direct Memory Access.

A mechanism whereby a device other than the host processor contends for and receives mastery of the memory bus so that data transfers can take place independent of the host.

DMA Controller: — Direct Memory Access Controller.

Controls data block transfers between memories, peripherals, and processors.

DO: — Data Out

DPA: — Differential Power Analysis

DPLL: — Digital Phase-Locked Loop. Digital implementation of PLL.

DRD: — Dual Role Device

DRM: — Digital Rights Management

DRP: — Digital Radio Processor

DSP — Digital Signal Processor.

A semiconductor that manipulates discrete or discontinuous electrical impulses in a manner that implements a desired algorithm.

DSP/BIOS: — Digital Signal Processor/Basic Input/Output System

DSR: — Data Set Ready

D-TLB: — Data Transition Lookaside Buffer.
See also TLB.

DTR: — Data Transmit Ready

DRAM: — Dynamic Random Access Memory.
Single-access data RAM generally used in RAM-based modules to emulate data ROM in future ROM megamodules.

DPRAM: — Dual-Port RAM

DSA: — Digital Signature Algorithm

DU: — Debug Unit

DVS: — Dynamic Voltage Scaling

E^{2}ICE: — Enhanced Embedded ICE Module

EAV: — End of Active Video

ECB: — Electronic Code Book

ECC: — Electronic Code Book

EDC: — Error Detection Code

EDGE: — Electronic Code Book

EEPROM: — Electrically Erasable Programmable Read Only Memory

EFuse: — Electrical Fuse.
A one-time programmable memory location usually set at the factory

EGA: — Enhanced Graphics Adapter

EMI: — Electromagnetic interference

EMIF: — Extended Memory Interface. Consists of the EMIFS and EMIFF.

EMPU: — Enhanced Memory Protection Unit

Emulator Device: —Type of device in which some security rules are relaxed. Intended for development only.

EOF: — End of Frame

EP: — Entry Point

EPROM: — Erasable Programmable Read-Only Memory

ES: — Erase Status

ESC: — Escape

ESS: — Erase Suspend Status

ETK: — Embedded Toolkit

ETLM: — Emulation TAP Linking Module

ETM: — Embedded Trace Macrocell

ETSI: — European Telecommunications Standard Institute

ETU: — Elementary Time Unit

EVM: — Evaluation Module

EXS — Exit Sequence

FAR: — Fault Address Register.

The FAR holds the virtual address of the access, which was attempted when a fault occurred.

FC: — Frame Counter

FCELL: — eFuse Cell

FCS: — Frame Check Sequence

FDD: — FIFO DMA Request Delay

FE: — Framing Error.

An error that occurs when the asynchronous serial port receives a data character that does not have a valid stop bit.

FEC: — Frame End Code

FIFO: — First In First Out.

A queue; a data structure or hardware buffer from which items are taken out in the same order they were put in. A FIFO is useful for buffering a stream of data between a sender and receiver which are not synchronized; that is, the sender and receiver are not sending and receiving at exactly the same rate. If the rates differ by too much in one direction for too long, the FIFO becomes either full (blocking the sender) or empty (blocking the receiver).

FIPS: — Federal Information Processing Standards

FIQ: — Fast Interrupt Request.

See also ISR.

FIR: — Fast Infrared

Flash Loader: —client application that is downloaded during the flashing phase into internal SRAM by the ROM code SW, then executed after successful authentication and integrity checking. This application may download a new image, burn it into flash, then cause a warm reboot.

FPGA: — Field Programmable Gate Array.

A type of logic chip that can be programmed. An FPGA is similar to a PLD; whereas PLDs are generally limited to hundreds of gates, FPGAs support thousands of gates.

Flashing: — How a programming host preloads the internal RAM with a flash loader to program flash memories connected to the device.

FMEA: — Failure Mode and Effects Analysis

FROM: — eFuse ROM

FS: — Frame Synchronization

FSC: — Frame Start Code. Also Frame Start Count.

FSM: — Finite State Machine.

A model of computation consisting of a set of states, a start state, an input alphabet, and a transition function that maps input symbols and current states to a next state.

FSR: — Fault Status Register

FW: — Firmware

GCRM: — Global Clock and Reset Module

GDD: — Generic Distributed DMA

GDI: — Graphics Driver Interface

GEA: — GPRS Encryption Algorithm

GP Device: —General-Purpose Device.
A device in which the security is disabled.

GF: — Galois Field

GND: — Ground

GPE: — General-Purpose Event

GPMC: — General-Purpose Memory Controller

GPE: — General-Purpose Event

GPP: — General-Purpose Processor

GPIO: — General-Purpose Input/Output.
Pins that can be used to accept input signals and/or send output signals but are not linked to specific uses.

GPS: — Global Positioning System

GSM: — Global System for Mobile Communications

GSM-S: — GSM Subsystem

HC: — Host Controller

HCI: — Host Controller Interface

H/W: — Hardware

HBP: — Horizontal Back Porch

HDLC: — High-Level Datalink Control

HFP: — Horizontal Front Porch

HIO: — Host I/O

HIVECT: — High Interrupts Vector

HOM: — Host-Only Mode.
A mode that allows only the host to access host port interface (HPI) memory. The CPU has no access to the HPI memory block during HOM.

HPI: — Host Port Interface

HS Device: —High-Security Device.
Device supporting the full set of security features.

HSPM: — HSPM: Halt/Suspended Protected Mode

HSW: — Horizontal Synchronization Pulse Width

HSYNC: — Horizontal Synchronization.

A bidirectional horizontal timing signal occurring once per line with a pulse width defined as an integral number of frame clock (FCLK) periods. Synchronization signals can be used to enable retrace of the electron beam of a display screen. Also HS.

HW: — Hardware

HWA: — Hardware Accelerators.

In the context of this document, refers to the DCT/IDCT, motion estimation, and half-pixel interpolation accelerators in the OMAP1610 device.

I^2C: — Inter-Integrated Circuit Sound.

A multimaster bus where multiple chips can be connected. Each chip can act as a master by initiating a data transfer

IA: — Initiator Agent

I2S: — Inter-IC Sound.

A digital audio interface standard.

I/O: — Input/Output

IA: — Identifier Address

IABORT: — Instruction Abort

I-Cache: — Instruction Cache

IC: — Integrated Circuit

Integrated Circuit —In-Circuit Emulation

ICR: — Intersystem Communication Registers

ID: — Identification

IDCT: — Inverse Discrete Cosine Transform.

See also DCT.

IDE:: — Integrated Development Environment. A programming environment integrated into an application.

IF: — Interface

IHV: — Independent Hardware Vendor

ILR: — Interrupt Level Register

IM: — Initiator Module.

A module is an initiator whenever it is able to initiate read and write requests to the chip interconnect (typically: processors, DMA, etc.).

IMIF: — Internal Memory Interface

IMX: — Image Extension Coprocessor

In Mux Logic: —Combinational logic implied in input signals functional multiplexing

INT: — Interrupt.

A signal sent by hardware or software to a processor requesting attention. An interrupt tells the processor to suspend its current operation, save the current task status, and perform a particular set of instructions. Interrupts communicate with the operating system and prioritize tasks to be performed.

INTC: — Interrupt Controller

I/O: — Input/Output

IOM-2: — ISDN Oriented Modular Interface Revision 2

IPC: — Interprocessor Communication. (also referred to as “mailbox” on occasion)

IPERIPH: — Memory-Like Peripherals

IrDA: — Infrared Data Association.

Represents the group of device manufacturers that developed a standard for transmitting data via infrared light waves.

IR: — Infrared

IRQ: — Interrupt Request.

IRQs are hardware lines over which devices can send interrupt signals to the microprocessor.

ISO: — Isochronous.

This refers to processes where data must be delivered within certain time constraints. For example, multimedia streams require an isochronous transport mechanism to ensure that data is delivered as fast as it is displayed and to ensure that the audio is synchronized with the video. Also, International Standards Organization.

ISR: — Interrupt Service Routine.

A function or set of functions that are called when an interrupt is encountered.

IST: — Interrupt Service Thread.

A thread that provides additional processing necessary for an interrupt, but which may be on the order of several microseconds.

ISV: — Independent Software Vendor

IT: — Interrupt

I-TLB: — Instruction Translation Lookaside Buffer.

See also TLB.

ITR: — Interrupt Input Register

IV: — Initialization Vector

IVA: — Image and Video Accelerator

JPEG: — Joint Photographics Experts Group

JTAG: — Joint Test Action Group.

The Joint Test Action Group was formed in 1985 to develop economical test methodologies for systems designed around complex integrated circuits and assembled with surface-mount technologies. The group drafted a standard that was subsequently adopted by IEEE as IEEE Standard 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture.

KB: — Kilobyte, 1024 B

KBC: — Keyboard Column

KBD: — Keyboard

Kbps: — Kilobits per second

KBR: — Keyboard Row

L1: — Level 1 Cache/Memory

L2: — Level 2 Cache/Memory

L3: — Level 3 Interconnect

L3 initiator: —L3 port that initiates transfers on the L3 interconnect

—

L3 target: —L3 port that receives transfers on the L3 interconnect

L4: — Level 4 Interconnect

LAN: — Local Area Network

LCD: — Liquid Crystal Display.
A display that uses two sheets of polarizing material with a liquid crystal solution between them.

LCh: — Logical DMA Channel

LDC: — Load (from memory) to Coprocessor

LDM: — Load Multiple

LDO: — Low Dropout

LE: — Little Endian.
An addressing protocol in which bytes are numbered from right to left within a word. More significant bytes in a word have higher numbered addresses. Endian ordering is specific to hardware and is determined at reset. See also big endian (BE).

LEC: — Line End Code

LED: — Light Emitting Diode.
An electronic device that lights up when electricity is passed through it.

LFSR: — Linear-Feedback Shift Register

LH: — Local Host

LLC: — Link Layer Control

LLRC: — Low Latency Response Crossbar.
Subpart of the L3 interconnect optimized for low latency.

LPG: — Light Pulse Generator

LPP: — Lines Per Panel

LRU: — Least Recently Used

LS: — Level Shifter

LSB: — Least Significant Bit

LSC: — Line Start Code

LVDS — Low-Voltage Differential Signaling

MAC: — MAC:

MB: — Megabyte, 1024 KB

McBSP: — Multichannel Buffered Serial Port.

An enhanced buffered serial port that includes the following standard features: buffered data registers, full duplex communication, and independent clocking and framing for receive and transmit. In addition, the McBSP includes the following enhanced features: internal programmable clock and frame generation, multichannel mode, and general-purpose I/O.

MCSI: — Multichannel Serial Interface.

A serial interface with multichannel transmission capability.

MCSI1: — Multichannel Serial Interface 1

MCSI2: — Multichannel Serial Interface 2

MCSPI: — Multichannel Serial Port Interface

MCU: — Microcontroller Unit.

Refers to the MPU.

MD5: — Message Digest Algorithm Revision 5

MDDR: — Mobile Double-Data-

Rate SDRAM, dedicated to mobile applications

MDOC: — Memory Disk On Chip by M-Systems.

A type of flash.

ME: — Motion Estimation

MEMIF: — Memory Interface

MeSSI: — Medium-Speed Screen Interface

MicroWire™: — This module provides a serial synchronous interface that can drive four serial external components.

MIDI: — Musical Instrument Digital Interface Device

MIPS: — Million Instructions Per Second

MIR: — Medium Infrared

MMA: — Multimedia Applications

MMC: — Multimedia Card

MMC/SD: — Multimedia Card/Secure Data

MMIO: — Multimedia Card/Secure Data

MMU: — Memory Management Unit.

The MMU performs virtual-to-physical address translations, performs access permission checks for access to the system memory, and provides the flexibility and security required for the OS to manage a shared physical memory space between the two processors.

MP3: — MPEG Layer 3.

An audio compression format.

MPEG: — Motion Pictures Expert Group
. A compression scheme for full motion video.

MPEG1 — The first MPEG compression scheme specification.

MPEG4 — The most current MPEG compression scheme specification, intended for very narrow bandwidths.

MPU: — Microprocessor Unit

MPU SS: —MPU-Subsystem

MR: — Mode Register. See also MEMSS.

MRC: — Maximal Ratio Combiner

MS: — Memory Stick

MS — Mobile Station

MSB: — Most Significant Bit.
The highest order bit in a word. The plural form (MSBs) refers to a specified number of high-order bits, beginning with the highest order bit and counting to the right. For example, the eight MSBs of a 16-bit value are bits 15 through 8.

MSDR/LPSDR: — Single-Data-Rate SDRAM, Mobile (or low-power) devices, dedicated to mobile applications

MUX: — Multiplex/Multiplexer

Muxed pin: —A pin is muxed when its pin control register field can be reconfigured by software to change the function associated with the PIN.

MuxMode: — 3-bit field of the pin control register field which enables to change the mode. Mode programming is assumed by software and selects a function on the device external interface.

MVIP: — Multivendor Integration Protocol

NAND: — NAND flash memory is a high-capacity, low-cost embedded permanent data storage solution.

NAND CE: —NAND Chip Enable

NAND CE Don't Care: —NAND Chip Enable Don't Care

NC: — Not Connected

NCB: — Non-Cacheable and Buffered

NCNB: — Non-Cacheable and Non-Buffered

NDFLASH: — NAND Flash Controlle

NFMC: — NAND Flash Memory Core

NIRQ: — Negative (Logic) Interrupt Request. See also IRQ.

NIST: — National Institute of Standards and Technology

NMI: — Nonmaskable Interrupt.
An interrupt that can be neither masked nor disabled.

NOR: — A type of flash memory

NR: — Noise Reduction

NRT: — Nonreal-Time

NSC: — National Semiconductor Corporation

NTSC: — National Television System Committee.
Television broadcast system.

OAL: — OEM Adaptation Layer

OCM: — On-Chip Memory

OCP: — Open-Core Protocol

OCPI: — Open-Core Protocol Interface

OE: — Output Enable

OEM: — Original Equipment Manufacturers

OHCI: — Open Host Controller Interface.
This is an industry standard USB Host Controller Interface.

OMAP: — An open software and hardware platform targeted at second/third generation cellular phones with multimedia capabilities.

Out Mux logic: —Combinational logic used in output signal functional multiplexing.

OS: — Operating System

OTG: — On-The-Go

PA: — Program Address

PAL™: — Programmable Array Logic

PB: — Peripheral Bus.
Refers to the TIPB.

PBIAS: — PMOS bias transistor to provide the bias voltage to extended drain I/Os.

PCB: — Printed Circuit Board

PCI — Peripheral Component Interconnect.
A local bus standard that is 64 bits wide, though it is usually implemented as a 32-bit bus. It can run at clock speeds of 33 or 66 MHz. At 32 bits and 33 MHz, it yields a throughput rate of 133M bps.

PCM: — Pulse Code Modulation.
A technique for digitizing speech by sampling the sound waves and converting each sample into a binary number.

PCS: — Personal Communication System.
The U.S. Federal Communications Commission (FCC) term used to describe a set of digital cellular technologies being deployed in the U.S. PCS works over CDMA (also called IS–95), GSM, and North American TDMA (also called IS–136) air interfaces.

PD: — Program Data

PDA: — Personal Digital Assistant

PDRAM: — Program Data Random Access Memory

PDROM: — Program Data Read-Only Memory

PE: — Parity Error

PG: — Protection Group

PGA: — Pin Grid Array

PHY: — Physical Layer Controller

PI: — Pixel Interpolation

PID: — Protocol Identifier.
The PID register is used in Windows CE mode only.

PIM: — Personal Information Management

PISO: — Parallel In/Serial Out

PK: — Public Key

PKA: — Public Key Accelerator

PLD: — Programmable Logic Devices

PLL: — Phase-Locked Loop.
A closed loop frequency control system whose function is based on the phase-sensitive detection of the phase difference between the input signal and the output signal of the controlled oscillator (CO).

PMT: — Parallel Module Test.
One of the test configuration modes of the OMAP1509 processor.

POR: — Power-On Reset

PPA: — Primary Protected Application

PPC: — Palm-Size PC

PPCELL: — Program Protection Cell

PPL: — Pixels per Line

PPS: — Protocol and Parameter Selection

PRAM: — Single-Access Program RAM.
Generally used in RAM-based modules to emulate program ROM in future ROM megamodules.

PRBS: — Pseudorandom Bit Sequence

PRCM: — Power, Reset, Clock Management module

Primary Mode: —Primary mode is mode0.
The function muxed in mode 0 gives its name to the PIN

PRNS: — Pseudo-Random Noise Source

Production ID: —An additional 64-bit register of eFuse cells used to include specific OMAP1610 needs.

PROM: — Programmable Read-Only Memory. A memory chip on which data can be written only once.

Protected Application: —Application that is signed by authenticated author and meant to be executed in secure execution environment

PRRM: — Protected Resource Reset Management

PSA: — Parallel Signature Analyzer

PSC: — Prescaler Counter

PSS: — Program Suspend Status

PTV: — Prescale Clock Timer Value.
Sets the value of the divisor used in scaling the clock.

Public Debug Mode: —A mode of operation of an emulator/test/secure device. In this mode public code debug is enabled and secure code execution is allowed.

Public Debug Mode: —Used to verify the digital signature generated by corresponding private key

Pure Input: —A pure input pin can be muxed on module inputs only.

Pure Output: —A pure output pin can be muxed between module outputs only.

PWL: — Pulse Width Light (modulator).
A 4096-bit random sequence generator that provides control of the LCD backlighting and keypad.

PWM: — Pulse Width Modulation

PWRON: — Power-On Reset

PWT: — Pulse Width Tone.
Creates the output tone signal for a buzzer, programmable both in frequency as well as volume.

QCIF: — Quarter Common Intermediate Format.
A video conferencing format that specifies data rates of 30 frames per second (fps), with each frame containing 144 lines and 176 pixels per line. This is one-fourth the resolution of CIF. QCIF support is required by the ITU H.261 video conferencing standard.

QOS: — Quality of Service

QVGA: — Quarter Video Graphics Array.
One-fourth the resolution of VGA.

RAM: — Random Access Memory.
A memory element that can be written to, as well as read.

R/B: — Read/Busy

RDR: — Receive Data Ready

RDRY: — Receive Data Ready

RE: — Read Enable

RF: — Radio Frequency

RFBI: — Remote Frame Buffer Interface

RGB: — Red, Green, Blue

RI: — Ring Indicator

RIJNDAEL: — Former name of AES

RISC: — Reduced Instruction Set Computer
. A computer whose instruction set and related decode mechanism are much simpler than those of microprogrammed complex instruction set computers. The result is a higher instruction throughput and a faster real-time interrupt service response from a smaller, cost-effective chip.

RNG: — Random Number Generator

ROM: — Read Only Memory.

A semiconductor storage element containing permanent data that cannot be changed.

RSA: — Stands for (Ron) Rivest, (Adi) Shamir and (Leonard) Adleman, its creators.
A public key algorithm.

RST: — Reset

RT: — Real-Time

RTC: — Real-Time Clock.
A clock that keeps track of the time even when the device is turned off.

RTOS: — Real-Time Operating System

RTS: — Request to Send

R/W: — Read/Write

RX: — Receive/Receiver

RXC: — RXC:

RXD: — Receive Data

S/W: — Software

SACK: — Selective Acknowledgement

SAM: — Shared Access Mode.
The mode that allows both the DSP and the host to access host port interface (HPI) memory. In this mode, asynchronous host accesses are synchronized internally, and, in case of conflict, the host has access priority and the DSP waits one cycle.

SARAM: — Single Access Random Access Memory.
Memory space that can only be read from or written to in a single clock cycle; RAM that can be accessed (read from or written to) once in a single CPU cycle.

SAV: — Start of Active Video

SB: — Silicon Backplane™

SB Flash: —Synchronous Burst Flash Memory

SBIM: — Silicon Backplane Initiator Module Agent registers.
All initiator modules interfaced with SB used the same set of registers for their Initiator Agent: Registers of the same names control the same functions and are mapped to the same offset addresses but at different base addresses

SBTM: — Silicon Backplane Target Module Agent registers
. All Target modules interfaced with SB used the same set of registers for their Target Agent: Registers of the same name control the same functions and are mapped to the same offset addresses but at different base addresses.

SBZ: — Should Be Zero

SCL: — Serial Clock. Programmable serial clock used in the I2C interface. Also SCLK.

SCM: — Scan Combiner Module

SCSA: — Signal Computing System Architecture.
An open, modular architecture for computer telephony that leverages applicable standards and evolves solutions to fill the gaps.

SD: — Starting Delimiter or Secure Data

SDA: — Serial Data. Serial data bus in the I^2C interface.

SDB: — Standard Development Board

SDF: — Standard Delay Format

SDI: — Serial Display Interface

SDIO: — Secure Digital Input/Output

SDK: — Software Development Kit

SDMA: — System Direct Memory Access module

SDR: — Single Data Rate

SDRC: — SDRAM Controller

SDRAM: — Synchronous Dynamic Random Access Memory

SDW: — Short Distance Wireless

Secure Debug Mode: — A mode of operation of an emulator/test/secure device. In this mode public and secure code debug are enabled

Secure Device: — A type of device where all security features are enabled. Intended for production.

Secure Execution Environment: — Secure environment enables a protected environment where sensitive code can be executed in a safe place while having access to the secure environment.

Secure Execution Mode: — Protected execution mode of the processor. Some peripherals (like internal secure RAM and secure ROM) are visible and allowed to be accessed only

Secure Initiator: — A system module that initiates secure read or writes transactions through the system interconnect. Typically a CPU, a DMA engine.

Secure Mode: — See Secure Execution Mode.

Secure RAM: — Writable memory inside the device. Protected applications are loaded here for execution and secure data is (temporarily) stored here. Can be accessed only when the device is in secure mode.

Secure ROM: — Read-only memory inside the device. Programmed at device manufacturing phase. Can be accessed only when the device is in secure mode

Secure Target: — A system module that can be the destination of read or write transactions issued by secure initiators. Typically a memory subsystem.

SE: — Secure Environment.

Execution environment inside a device, which is protected against tampering.

SERROR: — Slave Error

SGNS: — Serving GPRS Support Node

SHA-1: — Secure Hash Algorithm Revision 1.

One of the most popular cryptographic one-way hash algorithms, which generates 160 bits digest from any length message.

SIM: — Subscriber Identity Module

SIPO: — Serial In Parallel Out

SIR: — Slow Infrared

SMS: —SDRAM Memory Scheduler

SOF: —Start of Frame

SoSSI: —Specially Optimized Screen Interface

SP: —Serial Port or Small Page

SPC: —Serial Port Control

SPI: —Serial Port Interface. A signaling protocol for exchanging serial data.

SRAM: — Static Random Access Memory.
Fast memory that does not require refreshing, as DRAM does. It is more expensive than DRAM, though, and is not available in as high a density as DRAM.

SRC: —Sample Rate Conversion

SRG: —Sample Rate Generator

SRP: —Session Request Protocol

SS: —Subsystem

SSM: —Security State Machine

SSW: — Static Switch

ST: —Start Timer

ST-BUS: — Serial Telecom Bus

STC: —Store from coprocessor (to memory) or System Time Clock, which is the master clock in an MPEG2 encoder or decoder system.

STI: —Serial Trace Interface

STM: —Synchronous Transfer Mode or Store Multiple

STN: —Super-Twist Nematic. A technique for improving LCD display screens by twisting light rays.

Strict Public Debug Mode: —A mode of operation of an emulator/test/secure device. In this mode public code debug is enabled and secure code execution is forbidden.

SW: —Software

TAP: —Test Access Port

TC: — Traffic Controller.
Allows asynchronous operation among the external memory interface, the MPU, and the DSP.

TCIF: —Traffic Controller Interface

TCK: — Test Clock

TCM: —Trigger Cache Manager

TDDR: — Timer Divide-Down Register

TDI: —Test Data Input

TDMA: —Time Division Multiple Access

TDM: — Time Division Multiplex/Multiplexing.

The process by which a single serial bus is shared by multiple devices with each device taking turns to communicate on the bus. The total number of time slots (channels) depends on the number of devices connected. During a time slot, a given device may talk to any combination of devices on the bus.

TDO: — Test Data Output

TDRY: — Transmit Data Ready

TDT: — Trace Debug Tool

Test Device: — A type of device in which some of the security rules are relaxed. Intended for engineering test only.

TFT: — Thin Film Transistor

. A type of LCD flat panel display screen in which each pixel is controlled by one to four transistors.

TI: — Texas Instruments

TIM: — Timer. Main count register.

TINT: — Timer Interrupt

TIPB: — Texas Instruments Peripheral Bus. Consists of two buses (private and public) that connect the TI925T to the external and internal peripherals.

TLB: — Translation Lookaside Buffer.

A cache that contains entries for virtualto- physical address translation and access permission checking.

TLL: — Transceiverless Link.

This is logic which allows the user to connect two USB transceiver interfaces together directly without the use of differential transceivers.

TMS: — Test Mode Select

TOC: — Table of Contents

TP: — Tiny Page

TPA: — Trace Port Analyzer

TPU: — Time Processing Unit

TPU2OCP: — TPU to OCP Module Interface

TRST: — Test Reset

TRX: — USB Transceiver. The USB analog driver/receiver.

TSS: — Timer Stop Status

TTB: — Translation Table Base.

It points to the base of a table in physical memory that contains section and page table descriptors.

TTL: — Transistor Transistor Logic

TTS/STT: — Text-to-Speech/Speech-to-Text Conversion

TX: — Transmit//Transmitter

TXC: — Bidirectional Serial Transmit Clock

TXD: — Transmit Data

UART: — Universal Asynchronous Receiver/Transmitter
. Another name for the asynchronous serial port.

UE: — Unrecoverable Error

UI: —User Interface

ULPD: — Ultralow-Power Device.
A state machine that can stop the oscillator and restart it on a wake-up signal.

UND: —Undefined

UNP: —Unpredictable

USAR: —Universal Synchronous/Asynchronous Receiver

USART: — Universal Synchronous/Asynchronous Receiver/Transmitter

USB: — Universal Serial Bus.
An external bus standard that supports data transfer rates of 12M bps (12 million bits per second). A single USB port can be used to connect up to 127 peripheral devices.

USIM: —Universal Subscriber Identity Module

VA: — Volt-Amps.
A form of power management. A VA rating is the volts rating multiplied by the amps (current) rating, used to indicate the output capacity of an uninterruptible power supply (UPS) or other power source.

VBP: — Vertical Back Porch

VCO: — Voltage Controlled Oscillator

VENC: — Video Encoder

VFIR: — Very Fast Infrared

VFP: — Vertical Front Porch

VGA: —Video Graphics Array. An industry standard for video cards.

VGP: —Vertex Geometry Processor

VIA: —Versatile Interconnection Architecture

VIVT: —Virtual Index Virtual Tag

VLCD: —Variable Length Coding and Decoding coprocessor

VR: — Voice Recognition

VRFB: —Virtual Rotated Frame Buffer

VSW: — Vertical Synchronization Pulse Width

VSYNC: — Vertical Synchronization.
A bidirectional vertical timing signal occurring once per frame with a pulse-width defined as an integral number of lines (half-lines for interlaced mode). Also VS.

WB: —Write Buffer

WCDMA: — Wideband Code Division Multiple Access.
A third-generation digital cellular technology that uses spread-spectrum techniques.

WCS: — Wireless Computing System

WD: — Watchdog.
A timer that requires the user program or OS periodically write to the count register before the counter underflows.

WDT: — Watchdog Timer

WE: —Write Enable

WMA: —Windows Media Audio

WB: —Write Buffer

Word16: —16-bit word

Word32: — 32-bit word

Word8: — 8-bit word

WP: —Write Protect

WSMS: —Write State Machine Status

WS: — Wait State.
A period of time that the CPU must wait for external program, data, or I/O memory to respond when reading from or writing to that external memory. The CPU waits one extra cycle for every wait state.

WT: — Write Through

WWT: — Work Waiting Time

—

XIO: — External Memory Interface (Input/Output) of the lead processor.

XIP: — eXecution In Place

X-Loader: —A user-defined pre-operating system bootstrap code that resides at the beginning of the external flash.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
Low Power Wireless	www.ti.com/lpw	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265