LOCOSTO Program

Locosto Memory Interface Software Programming Guidelines

Application note

Rev:1.0

Document Number:	APN213
Version:	**1.0**
Status:	**Released**
Creation Date:	8 Nov 2005
Last changed:	15 Dec 2005
File Name:	APN213.doc

REVISION HISTORY

Rev#	Date	Author	Reason for change
0.1	8 Nov 2005	GT	Creation
0.2	12 Dec 2005	GT	Added CSST adaptation section
1.0	15 Dec 2005		First release

Important Notice

Texas Instruments Incorporated and/or its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products, software and services at any time and to discontinue any product, software or service without notice. Customers should obtain the latest relevant information during product design and 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 products, software and/or services. To minimize the risks associated with customer products and applications, customers should provide adequate design, testing and operating safeguards.

Any access to and/or use of TI software described in this document is subject to Customers entering into formal license agreements and payment of associated license fees. TI software may solely be used and/or copied subject to and strictly in accordance with all the terms of such license agreements.

Customer acknowledges and agrees that TI products and/or software may be based on or implement industry recognized standards and that certain third parties may claim intellectual property rights therein. The supply of products and/or the licensing of software does not convey a license from TI to any third party intellectual property rights and TI expressly disclaims liability for infringement of third party intellectual property rights.

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, software or services are used.

Information published by TI regarding third–party products, software or services does not constitute a license from TI to use such products, software or services or a warranty, endorsement thereof or statement regarding their availability. Use of such information, products, software or services 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.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

REFERENCE DOCUMENTS

[1] Locosto EMIF BUM chapter

GLOSSARY

GSM	**G**lobal **S**ystem for **M**obile communications
GPRS	**G**eneralized **P**acket **R**adio **S**ervice
ABB	**A**nalog **B**ase **B**and
DBB	**D**igital **B**ase **B**and
EMIF	**E**xternal **M**emory **I**nter**f**ace
CFI	**C**ommon **F**lash **I**nterface
FFS	**F**lash **F**ile **S**ystem

1 Introduction

Scope of this documentation is to give user guidelines for configuration and use of Locosto Memory Interface (EMIF). It will focus on the software programming of this interface within the TCS3.2 software package:

1. MEMIF driver adaptation to external RAM and external NOR flash
2. FFS driver adaptation to external NOR flash
3. CSST tools suite adaptation to external NOR flash

2 Locosto Memory Interface presentation

The external memory interface (EMIF) manages Synchronous/Asynchronous 16-bits data bus read/write accesses between external memories like Flashes, SRAMs and the MPU and DMA.

2.1 Overview - Main Features

- 8/16/32-bit read/write from MPU or DMA. 16-bit word data-bus burst read/write from DMA.
- Chip Selects (CS [0..3]), Up to 32MByte for each chip select (except for CS0 with 28 MByte)
- Each CS supports:
  - Asynchronous Read/Write (default) with programmable wait-states and support for dynamic access time extension (through external nRDY signal)
  - Synchronous burst read/write
    - Continuous, 4-, 8- and 16-bit-word burst
    - 52 MHz clock for burst accesses
    - Support dynamic access time extension (through external nRDY signal)
- Insertion of wait states for read and write accesses
- Multiplexing of address/data
- External memory frequency configuration
- Protect mode support

The synchronous/asynchronous external memory interface supports the most common memory interface protocols through a flexible programming and timing signal control. There are a number of different muxed/non-muxed memory types that can connect with the EMIF interface. These memory types share the same pins on the device, but their functionalities and controlling logic may differ.
The Locosto external memory interface module supports PSRAM, NOR, ADD/DATA multiplexed protocols. The EMIF can control up to 4 memory devices, with 4 chip select signals for a total address range up to 124-MByte (1*28-MByte + 3*32MByte). The following list summarizes the type of memories compatible with EMIF and gives an interconnection example:

Reference: NOR : S29NS016J/S29NS064J/S29NS128J
Type: Read/Write Burst Mode flash memory
ADD/DAT: Muxed
Size: 16 / 64 / 128 Mbits
Data bus: 16 bits
Manufacturer: Spansion
Power supply: 1V8

Reference: NOR : M58WR016FU
Type: Read/Write Burst Mode flash memory
ADD/DAT: Muxed
Size: 16 Mbits
Data bus: 16 bits
Manufacturer: ST Microelectronics
Power supply: 1V8

Reference: NOR : RD38F3050L0YBQ1S
Type: Read/Write Burst Mode flash memory
ADD/DAT: Muxed
Size: 64 Mbits
Data bus: 16 bits
Manufacturer: Intel

Power supply:	1V8
Reference:	PSRAM : MT45W4MW16BFB
Type:	Read/Write Burst Mode flash memory
ADD/DAT:	Non muxed
Size:	64 Mbits
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:	128Mb Spansion NOR / 64Mb Micron PSRAM
Data bus:	16 bits
Manufacturer:	Spansion (Micron PSRAM)
Power supply:	1V8

LOCOSTO						S29NS064J			
Signal	**Ball**	**Mode**	**I/O**	**Power**	**DIR**	**Signal**	**Ball**	**I/O**	**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
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	

2.2 Differences with Calypso EMIF

This section details the main differences between the Locosto and the Calypso's EMIF module.

Feature	Locosto	Calypso
Max addressable space	124 MBytes	124 MBytes
Number of CS	4	6
Page mode read access	No	Yes
Burst mode access	Yes	No
Muxed data/address	Yes	No
Internal Access	MPU, DMA	MPU
Pre-fetch buffer	Yes	No
Power management scheme	Auto-idle, power down acknowledgement	*tbd*(1)
Bank switching management	*tbd*(1)	*tbd*(1)

Table 1. Locosto/Calypso EMIF differences

New registers have been added to the Locosto EMIF memory map to handle the added features:

- EMIF_ERLU_PRIO register (Priority arbitration between ARM and DMA)
- EMIF_CONF register (Pre-fetch buffer, Handshake and power mode management)
- EMIF_ADV_CS0/1/2/3 (OE_SETUP, ADV_HOLD setup, handshaking mode, etc.)
- EMIF_DWS (EMIF full handshaking or not full handshaking protocol with masks during write access)
- EMIF_ATOR (enable timeout feature and Timeout value)

The following registers were dispatched on MPU module register on Calypso:

- -EMIF_PRT_MODE (Enable read only mode)
- -EMIF_BOUND (Address boundary)
- -EMIF_MASK (Enable external protection memory of CS0/1/2/3 ...)
- -EMIF_AADD (Report the address of the abort)
- -EMYF_ATYPER (Status of the abort)

Notes:

(1) *tbd*: Information will be supplied in next document revision.

3 Software Programming Guidelines

This section will focus on the following topics:

- MEMIF driver API and related programming for NOR flash and external RAM supports
- Adaptation of the FFS to the on-board NOR flash
- Adaptation of the CSST tool to the on-board NOR flash

3.1 MEMIF driver

The MEMIF driver allows configuring Locosto's external memory interface.

3.1.1 MEMIF driver API summary

Function	Description
f_emif_set_priority(SYS_UWORD8 d_dma_access, SYS_UWORD8 d_mpu_access);	Configures the access priority between MPU and DMA. **d_dma_access** sets up DMA consecutive access from 1 to 15 for priority. **d_mpu_access** sets up MPU consecutive access from 1 to 8 for priority.
f_emif_set_conf(T_EMIF_CONF* emif_conf);	Configures the memory interface (prefetch mode configuration, write protect pin control, power down configuration….) **emif_conf** structure is defined below.
f_emif_cs_mode(SYS_UWORD8 d_cs, T_EMIF_CS_CONFIG* p_emif_cs_config);	Configure a memory interface indivivual chip select (operating mode, wait states number, speed, etc…). **d_cs** selects the chip select to configure (0 to 3). **p_emif_cs_config** structure is defined below.
f_emif_abort_conf(SYS_UWORD8 d_timeout_enable, SYS_UWORD8 d_timeout);	Configures the abort upon time-out. **d_timeout_enable** enables the time-out counter. **d_timeout** is the time-out counter value in cycles (0 to 255).
f_emif_abort_status(T_ABORT_STATUS* p_abort_status);	Returns information upon EMIF abort. **p_abort_status** structure is defined below.
f_emif_protect_conf(SYS_UWORD32 d_bound_address, SYS_UWORD32 d_protect_cs, SYS_UWORD32 d_protect_mask);	Configures the memory protection parameters. **d_bound_address** defines the upper bound of the protected space from the external memory base address (refer to TRM). (0 to 0x3FFFF) **d_protect_cs** selects the chip select which protection is applied on (0 to 3) **d_protect_mask** defines the external memory masked address (0 to 0xFF)
f_emif_protect_enable(	Enable/disable memory protection

SYS_UWORD8 enable);	

Table 2. MEMIF driver API summary

Note: all functions return 0.

3.1.2 MEMIF driver API details

This section describes the structures introduced in the API summary section.

T_EMIF_CONF structure description

Field name	Description
d_prefetch_mode	Defines the memory interface pre-fetch mode. Possible values are: C_EMIF_PREFETCH_OFF (pre-fetch off) C_EMIF_PREFETCH_INSTRUCTION_DATA (pre-fetch for instruction and data) C_EMIF_PREFETCH_INSTRUCTION (pre-fetch for instruction)
d_pde_enable	Defines the system power-down acknowledgement. Possible values are: C_EMIF_PDE_DISABLE C_EMIF_PDE_ENABLE
d_pwd_enable	Enable/disable the EMIF power down mode. Possible values are: C_EMIF_PWD_DISABLE C_EMIF_PWD_ENABLE
flush_prefetch	Enable/Disable flush of the pre-fetch buffer
write_protect	Write protect output pin control. Possible values are 0: disable write in external memory 1: all sectors are writable

Table 3. MEMIF driver T_EMIF_CONF structure description

T_EMIF_CS_CONFIG structure description

Field name	Description
d_wait_read_write_trans	Defines the number of wait states inserted during read to write transition. Possible values are from 0 to 15
d_memmode	Defines the external device operating mode. Possible values are: C_EMIF_ASYNC_READ_ASYNC_WRITE C_EMIF_SYNC_READ_ASYNC_WRITE C_EMIF_SYNC_READ_SYNC_WRITE
d_we	Defines the length of WE pulse duration for write access. Possible values are from 0 to 15.
d_wait_write	Number of wait states for write operation. Possible values are from 0 to 15.
d_wait_read	Number of wait states for read operation. Possible values are from 0 to 15.
d_retime	Defines whether data are re-timed. C_EMIF_NO_RETIME: data are not re-timed C_EMIF_RETIME: data are re-timed with REF_CLK
d_flash_clk_div	Defines the flash clock divider. C_EMIF_CLK_DIVIDER_1 (divide by 1) C_EMIF_CLK_DIVIDER_2 (divide by 2)

	C_EMIF_CLK_DIVIDER_4 (divide by 4) C_EMIF_CLK_DIVIDER_6 (divide by 6)
d_non_full_handshake_mode	Enable full/non-full hand-shaking mode C_EMIF_DYN_WAIT_DISABLE: Full-hand-shaking C_EMIF_DYN_WAIT_ENABLE: Non-full-hand-shaking
d_oe_setup	Controls the number of cycles inserted from CS low to OE low (0 to 15).
d_oe_hold	Not used
d_adv_hold	Hold cycle for ADV signal in async and sync prot. 1&2. 0: hold cycle, ADV length is 1 flash clock cycle 1: one flash clock cycle, ADV length is 2 flash clock cycles
d_bus_turn_mode	Enables extended BTWST usage. C_EMIF_BT_RD_TRANS: bus turn around cycles are inserted between each read operation to avoid data contention C_EMIF_BT_RD_WR_TRANS: bus turn around cycles are inserted between WR-> RD and WR->WR of the same chip select apart from the RC to any transaction
d_clk_mask	Clock masking for synchronous mode writes C_EMIF_CLK_NO_MASK: EMIF_MEM_CLK is sent to the device during writes in synchronous modes C_EMIF_CLK_MASK: EMIF_MEM_CLK is masked and us held low on the device during writes
d_ready_configuration	nRDY signal control 0: nRDY asserted one EMIF_MEM_CLK cycle before data phase 1: nRDY asserted in the same EMIF_MEM_CLK cycle as the data phase

Table 4. MEMIF driver T_EMIF_CS_CONFIG structure description

T_ABORT_STATUS structure description

Field name	Description
d_abort_state	Defines whether abort occurred. Possible values are: 0: no abort 1: abort occurred
d_abort_address	32-bit word holding the address of the abort
d_abort_host	Defines source of error. Possible values are: C_EMIF_ABORT_HOST_MPU (MPU) C_EMIF_ABORT_HOST_DMA (DMA)
d_abort_protect	Defines whether abort is a protected mode error. Possible values are: 0: no protected mode error 1: protected mode error occurred
d_abort_timeout	Defines whether abort was generated by a time-out. 0: abort was not generated by a time-out 1: abort was generated by a time-out

Table 5. MEMIF driver T_ABORT_STATUS structure description

3.1.3 MEMIF driver usage

TCS3.2.x software invokes the MEMIF driver early in the software boot-up process in order to allow early usage of the on-board PSRAM and NOR flash. Configuration of the MEMIF and the chip selects which external devices are connected is done during the dynamic clock configuration process. This process configures the main DPLL, the ARM and DSP clock frequencies among other system parameters settings. User should refer to this code and replace the corresponding parts with his specific MEMIF settings.
TCS3.2.x reference software is designed to run on I-Sample reference design boards. I-Sample board v1.x to v2.0 are fitted with Spansion MCP memories including NOR flash and pSRAM.

- MCP NOR flash die is connected to external memory interface CS3
- MCP pSRAM die is connected to external memory interface CS0

Thus, at init, TCS3.2.x software configures the memory interface parameters then its chip selects CS0 and CS3.
Definition of the MEMIF CS0 and CS3 parameters is part of a global system clock configuration structure named "**`T_DYNAMIC_CLOCK`**" and is located in:
\chipsetsw\system\init_common\dynamic_clock_15.h.

The code that sets the MEMIF and MEMIF CS0&CS3 parameters is located in:
\chipsetsw\system\init_common\dynamic_clock.c.

TCS3.2.x configures the memory interface module as follows:
In **\chipsetsw\system\init_common\dynamic_clock.c.**

```
T_EMIF_CONF emif_conf = {
      C_EMIF_PDE_ENABLE,
      C_EMIF_PREFETCH_INSTRUCTION_DATA,
      C_EMIF_PWD_ENABLE,
      0,
      0};

f_emif_set_conf(&emif_conf);
```

This configuration enables the pre-fetch buffer for both instruction and data, also enables global and dynamic power down modes. Write protect output pin control is set to 0, but this signal is not used on I-Sample.

Additionally, this code sets the MEMIF access priority parameters between MPU and DMA as follows:
In **\chipsetsw\drivers\drv_core\memif\sys_memif.h.**

```
#define C_EMIF_DEFAULT_MCU_ACCESS    3
#define C_EMIF_DEFAULT_DMA_ACCESS    4
```

In **\chipsetsw\system\init_common\dynamic_clock.c.**

```
f_emif_set_priority(C_EMIF_DEFAULT_DMA_ACCESS, C_EMIF_DEFAULT_MCU_ACCESS);
```

Chip select 0 is configured as follows:
In **\chipsetsw\system\init_common\dynamic_clock_15.h.**

```
//CS0
{
      /* CONF_CS0 register configuration */
      0,C_EMIF_ASYNC_READ_ASYNC_WRITE,4,3,5,C_EMIF_NO_RETIME,C_EMIF_CLK_DIV
IDER_1,
      /* FULL HANDSHAKE register configuration */
```

```
      C_EMIF_DYN_WAIT_ENABLE,
      /* ADV_CONF_CS0 register configuration */
      3,0,0,C_EMIF_BT_RD_TRANS,C_EMIF_CLK_NO_MASK,0
},
```

In **\chipsetsw\system\init_common\dynamic_clock.c**

```
f_emif_cs_mode(0,(T_EMIF_CS_CONFIG *)& p_dynamic_clock_cfg->d_cs0);
```

in summary:

I-Sample CS0 configuration (Spansion MCP pSRAM)	
d_wait_read_write_trans	0
d_memmode	Asynchronous read and asynchronous write
d_we	4
d_wait_write	3
d_wait_read	5
d_retime	No re-time of data
d_flash_clk_div	Flash clock divided bv 1
d_non_full_handshake_mode	Non-full-hand-shaking mode
d_oe_setup	3
d_adv_hold	hold cycle, ADV length is 1 flash clock cycle
d_bus_turn_mode	bus turn around cycles are inserted between each read operation
d_clk_mask	EMIF_MEM_CLK is sent to the device during writes in synchronous modes
d_ready_configuration	nRDY asserted one EMIF_MEM_CLK cycle before data phase

Table 6. I-Sample CS0 configuration for Spansion RAM

Chip select 3 is configured as follows:
In **\chipsetsw\system\init_common\dynamic_clock_15.h.**

```
// CS3
{
      /* CONF_CS3 register configuration */
      0,C_EMIF_ASYNC_READ_ASYNC_WRITE,3,6,6,C_EMIF_NO_RETIME,C_EMIF_CLK_DIV
IDER_1,
      /* FULL HANDSHAKE register configuration */
      C_EMIF_DYN_WAIT_ENABLE,
      /* ADV_CONF_CS2 register configuration */
      4,0,0,C_EMIF_BT_RD_TRANS,C_EMIF_CLK_NO_MASK,0
},
```

In **\chipsetsw\system\init_common\dynamic_clock.c**

```
f_emif_cs_mode(3,(T_EMIF_CS_CONFIG *)& p_dynamic_clock_cfg->d_cs3);
```

in summary:

I-Sample CS3 configuration (Spansion MCP NOR flash)	
d_wait_read_write_trans	0
d_memmode	Asynchronous read and asynchronous write
d_we	3
d_wait_write	6
d_wait_read	6

d_retime	No re-time of data
d_flash_clk_div	Flash clock divided bv 1
d_non_full_handshake_mode	Non-full-hand-shaking mode
d_oe_setup	4
d_adv_hold	hold cycle, ADV length is 1 flash clock cycle
d_bus_turn_mode	bus turn around cycles are inserted between each read operation
d_clk_mask	EMIF_MEM_CLK is sent to the device during writes in synchronous modes
d_ready_configuration	nRDY asserted one EMIF_MEM_CLK cycle before data phase

Table 7. I-Sample CS3 configuration for Spansion NOR flash

MEMIF driver programmer is recommended to replace the corresponding parts in these files with the specific platform's devices parameters (chip select, type of access, wait-states management) as detailed above..

3.2 Adaptation of FFS to on-board NOR Flash device

3.2.1 FFS driver customization

This section details the customization steps to follow in order to have the system FFS working on the customer's selected NOR flash.

TCS3.2.x software supports by default a number of NOR flashes available on the market. If customer's selected devices is covered by the list of supported NOR flashes, adapting the FFS requires very few effort.
Adapting the FFS to a device which is not supported y TCS3.2.x reference software requires a bit more effort.

The FFS driver resides at the following location: `\chipsetsw\drivers\drv-app\ffs\.`
Specific implementation of the driver is in: `\chipsetsw\drivers\drv-app\ffs\board`

The code implementing the read/write/erase/query state machines supports AMD, Intel, ST and Fujitsu devices. User only needs to provide information on the on-board device to have the FFS running.

FFS needs only very limited configuration. Its entire configuration is contained in the `cfgffs.c` and `dev.c` files.

In the `cfgffs.c` file the user should specify two things:

1. The manufacturer (`uint16 ffs_flash_manufact`) and device ID (`uint16 ffs_flash_device`) of the flash device used.
2. The implementation of the `ffs_is_modifiable(name)` function.

Actually, if the user specifies the manufacturer & device IDs as zero, FFS will auto-detect the device.

The `dev.c` file lists all the flash devices known to FFS. Another definition is readily added to this file in order to add support for another flash device.
The `dev.c` defines two structures that shall contain information on the flash device:

1. `block_info_s`
   The flash device memory map (also known as sector address table) which defines the sectors blocks organization of the flash device. Several memory maps are supported (256x64kB, 128x64KB, 16x64KB, 8x128KB, etc…). Definition of non-supported memory maps can be added using the examples provided in this file.

2. `flash_info_s`
   This array contains the flash device information FFS driver needs to be able to use the flash device:
   - *Device memory map* (from the `block_info_s` structure defined above)
   - *Absolute address of the first sector to be used by/for FFS*.
   - *Manufacturer Id*. Values are from FFS_MANUFACTURER enumeration in `\chipsetsw\drivers\drv-app\ffs\board\drv.h`. Several manufacturers are supported (AMD, ATMEL, Intel, Fujitsu, etc…).

```
/* Manufacturer identifiers. These should never have to be
changed. They are ordered in alphabetically ascending order. */
enum FFS_MANUFACTURER {
```

```
    MANUFACT_AMD      = 0x01,
    MANUFACT_ATMEL    = 0x1F,
    MANUFACT_FUJITSU  = 0x04,
    MANUFACT_INTEL    = 0x89,
    MANUFACT_MXIC     = 0xC2,
    MANUFACT_SAMSUNG  = 0xEC,
    MANUFACT_SHARP    = 0xB0,
    MANUFACT_ST       = 0x20,
    MANUFACT_SST      = 0xBF,
    MANUFACT_TOSHIBA  = 0x98,
    MANUFACT_RAM      = 0xFE, // Ram
    MANUFACT_TEST     = 0x54  // 'T'est manufacturer
};
```

- *Flash device Id*. Available from device datasheet.
- *FFS device driver to use*. Values are from FFS_DRIVER enumeration in `\chipsetsw\drivers\drv-app\ffs\board\drv.h`. Supported: AMD multi-bank, AMD single-bank, Spansion multi-bank, Intel multi-bank, Intel single-bank, etc…

```
// Flash driver identifiers.
enum FFS_DRIVER {
    FFS_DRIVER_NULL = 0, // Null driver

    FFS_DRIVER_AMD = 2, // AMD dual/multi-bank driver
    FFS_DRIVER_AMD_SB = 3, // AMD single-bank driver
    FFS_DRIVER_AMD_NOR = 4, /* for locosto NOR flash driver,
Spansion multibank driver */
    FFS_DRIVER_AMD_MIRROR_BIT = 5, /* for locosto Mirror bit
NOR flash driver, Spansion multibank driver */
    FFS_DRIVER_SST = 8, // SST dual/multi-bank driver
    FFS_DRIVER_SST_SB = 9, // SST single-bank driver
    FFS_DRIVER_INTEL = 16, // Intel dual/multi-bank driver
    FFS_DRIVER_INTEL_SB = 17, // Intel single-bank driver
    FFS_DRIVER_INTEL_BW = 18, // Intel buffer write driver
    FFS_DRIVER_AMD_PSEUDO_SB = 32, // Test driver
    FFS_DRIVER_TEST = 34, // Test driver
    FFS_DRIVER_TEST_BW = 35, // Test driver (use buffer write)
    FFS_DRIVER_RAM           = 64  // Ram driver
};
```

- *Number of sectors to use*, starting from the address given in column 2.

Manufacturer Id and Flash device Id pair is used by FFS driver as an index to locate the correct flash device information in the `flash_info_s` table. Driver uses the FFS device driver parameter to pick up the correct procedure for erasing/writing/suspending the flash device from the list of procedures it supports (AMD, INTEL, …).

3.2.2 System Memory Map adaptation

TCS3.2 system build linker command file contains a description of the system memory map (internal RAM base address and size, external RAM base address and size, external flash base address, size and segments) so that it knows where to place the different build sections in memory.
The external flash information includes the FFS base address and size. This needs to be updated to reflect the parameters defined in the FFS driver.
TCS3.2 system build linker command file path is the following:
`\chipsetsw\system\template\gsm_is.template.`

The external flash information is located in the "system memory map" section of the file that looks like this:

```
/*********************************/
/* SPECIFY THE SYSTEM MEMORY MAP */
/*********************************/

MEMORY
{
  /* Secure Boot ROM */
  BOOT_MEM (RXI)   : org = 0x00000000   len = 0x00100000

  /* Will Keep External RAM for I-Sample to start With */
  /* CS0: External SRAM 2 Mbytes */
  D_MEM0    (RW)   : org = 0x00400000   len = 0x00800000

  /* CS3: Flash 6 Mbytes */
  /* Breaking into Multiple Segments , Thanks to V1.22E the Pre
   * historic Tool chain */
  P_MEM0    (RXI) : org = 0x06000000   len = 0x00410000
  P_MEM1    (RXI) : org = 0x06410000   len = 0x002f0000

  /* FFS Area */
  FFS_MEM   (RI)   : org = 0x06700000   len = 0x00100000

/* CS3: Flash 8 Mbytes */
/* Breaking into Multiple Segments , Thanks to V1.22E the Prehistoric Tool
chain */
P_MEM2    (RXI) : org = 0x06800000   len = 0x00400000
P_MEM3    (RXI) : org = 0x06C00000   len = 0x00400000

/* CS6: Locosto Internal SRAM 320 kbytes */
/* Code & Variables Memory */
S_MEM     (RXW) : org = 0x08000000   len = 0x00050000
}
```

3.2.3 I-Sample implementation

The I-Sample board v1.x to v2.0 is fitted with a Spansion MCP including a S29NS128J NOR Flash device. Details of this flash device are the following:

- Size: 128MBit (memory map: 256 sectors x 64kbytes)
- Manufacturer ID: 0x0001
- Device code: 0x007E

Flash info array is defined as follows on I-Sample board in `dev.c`:

- Device memory map: 256x64KB (128 Mbit)
- Absolute address of the first sector to be used by/for FFS: 0x06700000 (NOR flash is connected to EMIF CS3. EMIF CS3 base address is 0x06000000).
- Manufacturer code: MANUFACT_AMD (0x0001)
- Device code: 0x007E
- FFS device driver to use: FFS_DRIVER_AMD_NOR (Spansion multi-bank driver)
- Number of sectors to use for FFS: 15 (~ 1MB).

```
/* spansion S29NS128J  Excluding 8kb sectors of bank A */
{   &flash_256x64[0],   (char   *)   0x06700000,   MANUFACT_AMD,   0x007E,
FFS_DRIVER_AMD_NOR, 15 },
```

In `cfgffs.c`, manufacturer & device IDs are defined as zero, thus device is auto-detected.

```
uint16 ffs_flash_manufact = 0x00; // autodetect device
uint16 ffs_flash_device   = 0x0000; // autodetect device
```

The `gsm_is.template` file is updated accordingly:

```
/* FFS Area */
FFS_MEM  (RI)  : org = 0x06700000   len = 0x00100000
```

In summary, supporting a flash device for TCS3.2 FFS driver requires to alter `cfgffs.c`, `dev.c` and `gsm_is.template` files. User should be careful when defining the absolute address of the first sector to be used by/for FFS in order to avoid overlap between code and FFS sections.

3.3 Adaptation of CSST to on-board NOR flash device

CSST (Cellular Systems Software Tools) is a suite of tools for Board diagnostics, Download, Image formatting and signing.
The Primary feature of the software is that, it provides a unique solution for the user to perform various functions on the Software Development Platforms and Reference Designs.
The CSST Package contains two components, a **Host application**, which runs on a Windows 2000/XP Machine and a **Target Component,** which resides in (or downloaded to) the target.

The features that this document relates to are the following:

- Image download
- NOR Flash operations (read/write/erase/verify)
- IMEI, MEPD files binding

If customer platform supports a NOR flash different from the ones supported by CSST, then the tools suite needs adaptation.

The NOR flash families supported by CSST are the following:

- AMD flashes family (including mirror-bit)
- Intel flashes family

The CSST tool uses the same NOR flash information/driver to perform all the operations listed above (image download, NOR flash write/erase/verify, IMEI, MEPD files binding).

CSST tool NOR flash procedure is depicted below:
In `<CSST install dir>\CSST I-Sample 1.x.0.1\host\config\driver_config.xml`, the following information is provided:

- Flash device vendor Id code
- Flash device driver

Formatted as follows:

```
<nor_drivers>
<driver1>
       <vendor_id>0x0001</vendor_id>
       <drv_file>..\..\..\..\target\flashdrv\nor_intel_drv.out</drv_file>
</driver1>
<driver2>
       <vendor_id>0x0002</vendor_id>
       <drv_file>..\..\..\..\target\flashdrv\nor_amd_drv.out</drv_file>
       <wb_drv_file>..\..\..\..\target\flashdrv\nor_mirrorbit_drv.out</wb_drv_file>
</driver2>
</nor_drivers>
```

This pair of parameter is used by CSST to know which driver it shall use to operate on the flash device. The CSST tool when asked to perform any write/erase operation on the NOR flash will proceed that way:

1. Request Flash device's vendor code through its Common Flash Interface (CFI) protocol.
2. Parse the `driver_config.xml` file to locate the driver corresponding to the vendor code it has read from the flash device.
3. When driver is found, use it for write/erase operations (image download, security files binding).

Note that information like flash device Id, flash memory map, flash size are retrieved automatically by the CSST tool through the flash CFI protocol.

Customization of the CSST tool for NOR flash adaptation is pretty much straightforward then:

1. Get the flash device vendor code (**NB: flash device VENDOR CODE is NOT the flash device MANUFACTURER ID. Please contact your flash device retailer to obtain the VENDOR CODE**).
2. Add a new NOR flash driver block definition in `driver_config.xml`. Pretty much all the flash devices follow either the AMD commands protocol or the INTEL commands protocol. The "`drv_file`" field should be filled up accordingly.
3. If the flash device supports none of the protocols provided by default by CSST, then a new driver needs to be produced. Section 3.3.1 details this procedure.

Example:
Customer board is fitted with a Fujitsu flash device. Fujitsu flash devices support the *AMD commands* protocol. Let's say that Fujitsu flash devices vendor code is *0x0048* (**THIS IS AN EXAMPLE!!!**). User would then add the following block (in bold characters) in "`driver_config.xml`":

```
<nor_drivers>
<driver1>
       <vendor_id>0x0001</vendor_id>
       <drv_file>..\..\..\..\target\flashdrv\nor_intel_drv.out</drv_file>
</driver1>
```

```
<driver2>
      <vendor_id>0x0002</vendor_id>
      <drv_file>..\..\..\..\target\flashdrv\nor_amd_drv.out</drv_file>
      <wb_drv_file>..\..\..\..\target\flashdrv\nor_mirrorbit_drv.out</wb_drv_file>
</driver2>
<driver3>
      <vendor_id>0x0048</vendor_id>
      <drv_file>..\..\..\..\target\flashdrv\nor_amd_drv.out</drv_file>
</driver3>
</nor_drivers>
```

3.3.1 Writing a new CSST flash driver

If customer's on-board flash device does not follow the AMD or the INTEL commands protocol, a new flash driver needs to produced. CSST tools suite source code is needed.

Drivers source code reside in the following directory:
`<CSST install directory>\csst-1.x.0.1\csst\target\flashdrv\src`

A flash driver needs to implement the following functions:

Function name	Description
`flash_init`	This function initializes the flash device by querying its buffer size (this information is queried through the CFI protocol).
`flash_erase`	Implements the flash erase procedure
`flash_write`	Implements the flash program procedure
`flash_read`	Reads from the flash at specified address

Table 8. CSST flash driver functions

Driver's implementer can follow the design of the AMD and INTEL drivers provided in this directory.

Drivers are by default built up with Code Composer Studio. Code Composer Studio projects used to build the AMD or INTEL drivers are located in the following directory:
`<CSST install directory>\csst-1.x.0.1\csst\target\flashdrv\build`

Contact your local TI support for more information about Code Composer Studio.