Technical Documentation - Confidential

GSM PROTOCOL STACK

G23

ACI – APPLICATION CONTROL INTERFACE

Document Number:	8415.012.99.002
Version:	0.3
Status:	Draft
Approval Authority:	
Creation Date:	1999-Jan-29
Last changed:	2015-Mar-08 by XGUTTEFE
File Name:	8415_012.doc

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.

Change History

Date	Changed by	Approved by	Version	Status	Notes
1999-Jan-29	AK et al		0.1		1
1999-Feb-08	MS et al		0.2		2
2003-May-08	XGUTTEFE		0.3	Draft	

Notes:

1. Initial version
2. English check

Table of Contents

List of Figures and Tables

List of References

[ISO 9000:2000] International Organization for Standardization. Quality management systems - Fundamentals and vocabulary. December 2000

1.1 References

[C_8410.001] 8410.001.98.102; September 18, 1998
G23 Product Description; Condat

[C_8410.008] 8410.008.98.002; June 15, 1998
GTI Interface Description; Condat

[C_8410.003] 8410.003.98.103; September 09, 1998
Test Facilities Description; Condat

1.2 Abbreviations

MMI Man Machine Interface

1.3 Terms

---

2 Introduction

G23 is a software package implementing Layers 2 and 3 of the ETSI-defined GSM air interface signaling protocol, and as such represents the part of a GSM mobile station's protocol software which is both, platform and manufacturer independent. Therefore, G23 can be viewed as a building block providing standardized functionality through generic interfaces for easy integration.

The G23 suite of products consists of the following items:

- Layers 2 and 3 for speech & short message services,
- Layers 2 and 3 for fax & data services,
- Application Control Interface,
- Slim MMI [02.30] and
- Test and integration support tools.

For a detailed reference of G23 components, please refer to the Product Description [C_8410.001]. For detailed information regarding integration into the target system, please refer to the Generic Target Interface [C_8410.008]. For detailed information about the compiling and linking procedure, please refer to the User Guide on the delivery CD.

This Technical Documentation document shows how to use the ACI object in target systems. It lists the headers involved and describes how to link ACI with other components. The customer specific functions included are listed and described.

3 Structure

3.1 Headers

The modules include several header files. Header files which are changeable by the user are marked (*). These header files are used to integrate the protocol stack entities into a specific target system.

aci.h

This header contains constants for the application control interface and the prototypes of the component.

aci_cmd.h

This header contains constants for the AT command interpreter and the prototypes of the component.

aci_cmh.h (*)

This header contains constants and prototypes for using the functional interface of the ACI entity. This is a subset of the complete interface regarding the voice stack.

aci_fd.h

This header contains constants and prototypes for using the functional interface of the ACI entity. This is a subset of the complete interface regarding the fax and data stack.

aci_io.h

This header contains constants and prototypes used for the serial I/O communication.

aci_prs.h

This header contains constants and prototypes declared for the AT command parser.

ccdapi.h (*)

This header defines the prototypes and some constants for the Condat Coder Decoder (CCD).

cmh.h

This header contains constants and prototypes which are common for all sub-modules of the command handler.

cmh_cc.h

This header contains constants and prototypes which are specific for the command handler sub-module responsible for handling commands related to the call control protocol stack adapter.

cmh_l2r.h

This header contains constants and prototypes which are specific for the command handler sub-module responsible for handling commands related to the Layer 2 relay protocol stack adapter.

cmh_mm.h

This header contains constants and prototypes which are specific for the command handler sub-module responsible for handling commands related to the mobility management protocol stack adapter.

cmh_mmi.h

This header contains constants and prototypes which are specific for the command handler sub-module responsible for handling commands related to the MMI protocol stack adapter.

cmh_phb.h

This header contains constants and prototypes which are specific for the command handler sub-module responsible for handling commands related to phonebook management.

cmh_ra.h

This header contains constants and prototypes which are specific for the command handler sub-module responsible for handling commands related to the rate adaptation protocol stack adapter.

cmh_sim.h

This header contains constants and prototypes which are specific for the command handler sub-module responsible for handling commands related to the SIM protocol stack adapter.

cmh_ss.h

This header contains constants and prototypes which are specific for the command handler sub-module responsible for handling commands related to the supplementary service protocol stack adapter.

cmh_sms.h

This header contains constants and prototypes which are specific for the command handler sub-module responsible for handling commands related to the short message service protocol stack adapter.

cmh_t30.h

This header contains constants and prototypes which are specific for the command handler sub-module responsible to handle commands related to the T30 protocol stack adapter.

cnf_aci.h (*)

Constants for the dynamic configuration of ACI are defined in this header. It is acceptable to change the commands and the parameter names for the dynamic configurations supported.

custom.h (*)

This header defines global constants for the integration of the protocol stack entity into a specific target system. The user may define the identifier of the communication resource, the supported traces, the communication method (by copying primitives or by exchanging references of primitives) , the custom specific primitive header, etc.

cus_aci.h (*)

Custom specific definitions for the protocol stack entity are located in this header. Timer values and identifiers are changeable. A version identifier is defined. The PLMN description list is located here.

gsm.h

This header contains global definitions for all protocol stack entities. Depending on the definitions in custom.h, many options and traces are defined in this header.

mconst.cdg

This header is generated by the CCD compiler. It includes all message identifiers and some constants needed by the entities.

message.h

Constants for messages are defined. The header includes the header with the RR messages (m_rr.h).

mon_aci.h

Constants for the monitoring of RR are defined in this header.

p_aci.h

This header is generated by the CCD compiler. It includes the C-struct type definitions for primitives of the service access point ACI. The header is included by prim.h.

p_l2r.h

This header is generated by the CCD compiler. It includes the C-struct type definitions for primitives of the service access point L2R. The header is included by prim.h

p_mmi.h

This header is generated by the CCD compiler. It includes the C-struct type definitions for primitives of the service access point MMI. The header is included by prim.h.

p_mncc.h

This header is generated by the CCD compiler. It includes the C-struct type definitions for primitives of the service access point MNCC. The header is included by prim.h.

p_mnreg.h

This header is generated by the CCD compiler. It includes the C-struct type definitions for primitives of the service access point MNREG. The header is included by prim.h.

p_mnsms.h

This header is generated by the CCD compiler. It includes the C-struct type definitions for primitives of the service access point MNSMS. The header is included by prim.h.

p_mnss.h

This header is generated by the CCD compiler. It includes the C-struct type definitions for primitives of the service access point MNSS. The header is included by prim.h.

p_ra.h

This header is generated by the CCD compiler. It includes the C-struct type definitions for primitives of the service access point RA. The header is included by prim.h.

p_sim.h

This header is generated by the CCD compiler. It includes the C-struct type definitions for primitives of the service access point SIM. The header is included by prim.h.

p_t30.h

This header is generated by the CCD compiler. It includes the C-struct type definitions for primitives of the service access point T30. The header is included by prim.h.

pconst.cdg

This header is generated by the CCD compiler. It includes all primitive identifiers and some constants needed by the entities.

pei.h (*)

Prototypes for the protocol stack entity interface are defined in this header. Some parameters and the return type of this function are changeable by the user.

phb.h

This header contains constants and prototypes which are used by the phonebook management.

psa.h

This header contains constants and prototypes which are common for all sub-modules of the protocol stack adapter.

psa_cc.h

This header contains constants and prototypes which are specific for the protocol stack adapter sub-module responsible for handling accesses related to the call control entity.

psa_l2r.h

This header contains constants and prototypes which are specific for the protocol stack adapter sub-module responsible for handling accesses related to the Layer 2 relay entity.

psa_mm.h

This header contains constants and prototypes which are specific for the protocol stack adapter sub-module responsible for handling accesses related to the mobile management entity.

psa_mmi.h

This header contains constants and prototypes which are specific for the protocol stack adapter sub-module responsible for handling accesses related to the MMI entity.

psa_ra.h

This header contains constants and prototypes which are specific for the protocol stack adapter sub-module responsible for handling accesses related to the rate adaptation entity.

psa_sim.h

This header contains constants and prototypes which are specific for the protocol stack adapter sub-module responsible for handling accesses related to the SIM entity.

psa_sms.h

This header contains constants and prototypes which are specific for the protocol stack adapter sub-module responsible for handling accesses related to the short message service entity.

psa_ss.h

This header contains constants and prototypes which are specific for the protocol stack adapter sub-module responsible for handling accesses related to the supplementary service entity.

psa_t30.h

This header contains constants and prototypes which are specific for the protocol stack adapter sub-module responsible for handling accesses related to the T30 entity.

psa_util.h

This header contains constants and prototypes of utility functions which are for common use for all sub-modules of the protocol stack adapter.

prim.h

Constants for primitives are defined and service access point dependent primitive header files are included (p_aci.h, p_l2r.h, p_mmi, p_mncc, p_mnreg, p_mnsms, p_mnss, p_ra.h, p_sim and p_t30).

stddefs.h

This header contains several standard definitions used by the protocol stack entities.

string.h

This header is the standard string header from the target compiler. It defines string and memory functions.

stdlib.h

This header is the standard library header from the target compiler. It defines function declarations for commonly used library functions which either don't fit anywhere else, or, cannot be declared in the normal place for other reasons.

tok.h

The prototypes and some constants for the parse function of the TOK module are defined in this header.

vsi.h (*)

Prototypes for the virtual system interface are defined in this header. Some parameters and the return types of these functions are changeable by the user for integration into a specific target system.

3.2 Dynamic Configuration

Dynamic configuration means to change the behavior of the protocol stack entity at run-time. This is carried out by sending a string with a dedicated format as described in the Test Facilities [C_8410.003]

The dynamic configuration string is a parameter of the pei_config () function which is part of the protocol stack entity interface (PEI).

TIMER_SET =<timer, value>

The timer mode TIMER_SET defines a new timer value instead of the original start value.

TIMER_RESET =timer

The default timer mode is TIMER_RESET which does not manipulate the start value.

TIMER_SPEED_UP =<timer, factor>

TIMER_SPEED_UP is used to speed up a timer by the given factor. The start value is divided by the factor. The minimum time is one unit.

TIMER_SLOW_DOWN = <timer, factor>

The opposite mode is TIMER_SLOW_DOWN. The start value is increased by the given factor.

TIMER_SUPPRESS = <timer>

TIMER_SUPPRESS is used to suppress the timer start.

3.3 Custom Specific Functions

Custom specific functions are implemented in the module rr_csf.c. It is acceptable to replace the functions in this module by functions of the customer. It is not acceptable to change the parameters of the functions.

The idea behind these custom specific functions is to have a mechanism to configure the protocol stack entity at run-time by a source outside the protocol stack entity, for example a non-erasable memory.

GLOBAL BOOL csf_init_timer (void)

The function initializes the timer pool. The timer pool allocates a number of timer resources. This timer resources are allocated to instances on demand.

GLOBAL void csf_close_timer (void)

All timer resources are closed. This function is carried during shutdown or reset.

GLOBAL void csf_alloc_timer (UBYTE id, T_RR_DATA * rr_data, T_VSI_TVALUE value)

The function allocates one timer from the timer pool and starts this timer.

GLOBAL void csf_free_timer (T_VSI_THANDLE handle)

The function frees one timer. The timer is stopped and returned to the timer pool.

GLOBAL BOOL csf_vdb_timeout (T_VSI_THANDLE handle, T_RR_DATA ** rr_data, USHORT * timer)

After time-out, the corresponding instance is searched. The timer is returned to the timer pool.

3.4 Monitoring

The monitor struct includes relevant physical parameters of the protocol stack entity. The parameters are updated continuously. This way the environment has always the possibility of accessing parameters of the protocol stack. These parameters are used to create monitor reports about a display or test system, to create statistical data, etc. outside the functionality of a protocol stack but with access to protocol stack parameters. It is acceptable to read the parameters of the monitor struct, but it is absolutely not acceptable to write to the monitor struct. The first parameter of the monitor struct is the version of the protocol stack entity.

The following monitor struct is defined for the protocol stack entity:

```
typedef struct
{
        T_VERSION    *version;
} T_MONITOR;
```

Appendices

A. Acronyms

DS-WCDMA Direct Sequence/Spread Wideband Code Division Multiple Access

B. Glossary

International Mobile Telecommunication 2000 (IMT-2000/ITU-2000) Formerly referred to as FPLMTS (Future Public Land-Mobile Telephone System), this is the ITU's specification/family of standards for 3G. This initiative provides a global infrastructure through both satellite and terrestrial systems, for fixed and mobile phone users. The family of standards is a framework comprising a mix/blend of systems providing global roaming. <URL: http://www.imt-2000.org/>