**Technical Documentation – Confidential**

# GSM PROTOCOL STACK

# MMI

| | |
|---|---|
| Document Number: | 6147.708.97.100 |
| Version: | 0.2 |
| Status: | Draft |
| Approval Authority: | |
| Creation Date: | 1997-Dec-11 |
| Last changed: | 2015-Mar-08 by XGUTTEFE |
| File Name: | mmi.doc |

## Important Notice



## Change History

| Date | Changed by | Approved by | Version | Status | Notes |
|---|---|---|---|---|---|
| 1997-Dec-11 | Stefan Lemke et al | | 0.1 | | 1 |
| 2003-May-20 | XGUTTEFE | | 0.2 | Draft | |
| | | | | | |

**Notes:**

1. Initial version

## 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

[1] GSM 2.81, Line Identification Supplementary Services - Stage 1
ETS 300 514, ETSI, September 1994

[2] GSM 2.82, Call Forwarding Supplementary Services - Stage 1
ETS 300 515, ETSI, September 1994

[3] GSM 2.83, Call Waiting and Call Hold Supplementary Services - Stage 1
ETS 300 516, ETSI, September 1994

[4] GSM 2.84, Multi Party Supplementary Services - Stage 1
ETS 300 517, ETSI, September 1994

[5] GSM 2.85, Closed User Group Supplementary Services - Stage 1
ETS 300 518, ETSI, September 1994

[6] GSM 2.86, Advice of Charge Supplementary Services - Stage 1
ETS 300 519, ETSI, September 1994

[7] GSM 2.88, Call Barring Supplementary Services - Stage 1
ETS 300 520, ETSI, September 1994

[8] GSM 3.14, Support of Dual Tone Multi Frequency Signalling via the GSM System
ETS 300 532, ETSI, April 1994

[9] GSM 3.40, Technical Realization of the Short Message Service Point-to-Point
ETS 300 536, ETSI, January 1996

[10] GSM 3.41, Technical Realization of Short Message Service Cell Broadcast
ETS 300 537, ETSI, June 1995

[11] GSM 3.81, Line Identification Supplementary Services - Stage 2
ETS 300 542, ETSI, February 1995

[12] GSM 3.82, Call Forwarding Supplementary Services - Stage 2
ETS 300 543, ETSI, February 1995

[13] GSM 3.83, Call Waiting and Call Hold Supplementary Services - Stage 2
ETS 300 544, ETSI, November 1994

[14] GSM 3.84, Multi Party Supplementary Services - Stage 2
ETS 300 545, ETSI, November 1994

[15] GSM 3.85, Closed User Group Supplementary Services - Stage 2
ETS 300 546, ETSI, January 1996

[16] GSM 3.86, Advice of Charge Supplementary Services - Stage 2
ETS 300 547, ETSI, March 1995

[17] GSM 3.88, Call Barring Supplementary Services - Stage 2
ETS 300 548, ETSI, November 1994

[18] GSM 4.01, MS-BSS Interface General Aspects and Principles
ETS 300 550, ETSI, September 1994

[18a] GSM 4.03, MS-BSS Interface Channel Structures and Access Capabilities
ETS 300 552, ETSI, September 1994

[19] GSM 4.05, Data Link Layer General Aspects
ETS 300 554, ETSI, September 1994

[20] GSM 4.06, MS-BSS Interface Data Link Layer Specification
ETS 300 555, ETSI, September 1994

[21] GSM 4.07, Mobile Radio Interface Signalling Layer 3 General Aspects
ETS 300 556, ETSI, February 1995

[22] GSM 4.08, Mobile Radio Interface Layer 3 Specification
ETS 300 557, ETSI, January 1996

[23] GSM 4.10, Mobile Radio Interface Layer 3 Supplementary Services Specification General Aspects
ETS 300 558, ETSI, February 1995

[24] GSM 4.11, Point-to-Point Short Message Service Support on Mobile Radio Interface
ETS 300 559, ETSI, October 1995

[25] GSM 4.12, Short Message Service Cell Broadcast Support on Mobile Radio Interface
ETS 300 560, ETSI, January 1996

[26] GSM 4.80, Mobile Radio Interface Supplementary Services Specification Formats and Coding
ETS 300 564, ETSI, February 1995

[27] GSM 4.81, Line Identification Supplementary Services - Stage 3
ETS 300 565, ETSI, February 1995

[28] GSM 4.82, Call Forwarding Supplementary Services - Stage 3
ETS 300 566, ETSI, February 1995

[29] GSM 4.83, Call Waiting and Call Hold Supplementary Services - Stage 3
ETS 300 567, ETSI, February 1995

[30] GSM 4.84, Multi Party Supplementary Services - Stage 3
ETS 300 568, ETSI, February 1995

[31] GSM 4.85, Closed User Group Supplementary Services - Stage 3
ETS 300 569, ETSI, February 1995

[32] GSM 4.86, Advice of Charge Supplementary Services - Stage 3
ETS 300 570, ETSI, February 1995

[33] GSM 4.88, Call Barring Supplementary Services - Stage 3
ETS 300 571, ETSI, February 1995

[34] GSM 5.01, Physical Layer on the Radio Path General Description
ETS 300 573, ETSI, October 1995

[35] GSM 5.02, Multiplexing and Multiple Access on the Radio Path
ETS 300 574, ETSI, January 1996

[36] GSM 5.08, Radio Sub-system Link Control
ETS 300 578, ETSI, January 1996

[37] GSM 5.10, Radio Sub-system Synchronisation
ETS 300 579, ETSI, October 1995

[38] Service Access Point MMREG
6147.100.96.100; Condat GmbH

[39] Service Access Point MNCC
6147.101.96.100; Condat GmbH

[40] Service Access Point MNSS
6147.102.96.100; Condat GmbH

[41] Service Access Point MNSMS
6147.103.96.100; Condat GmbH

[42] Service Access Point MMCC
6147.104.97.100; Condat GmbH

[43] Service Access Point MMSS
6147.105.97.100; Condat GmbH

[44] Service Access Point MMSMS
6147.106.97.100; Condat GmbH

[45] Service Access Point RR
6147.107.97.100; Condat GmbH

[46] Service Access Point SIM
6147.108.97.100; Condat GmbH

[47] Service Access Point MPH
6147.109.96.100; Condat GmbH

[48] Service Access Point DL
6147.110.96.100; Condat GmbH

[49] Service Access Point MDL
6147.111.96.100; Condat GmbH

[50] Service Access Point PH
6147.112.97.100; Condat GmbH

[51] Service Access Point MMI
6147.113.96.100; Condat GmbH

[52] Message Sequence Charts CC
6147.200.97.100; Condat GmbH

[53] Message Sequence Charts SS
6147.201.97.100; Condat GmbH

[54] Message Sequence Charts SMS
6147.202.97.100; Condat GmbH

[55] Message Sequence Charts MM
6147.203.97.100; Condat GmbH

[56] Message Sequence Charts RR
6147.204.96.100; Condat GmbH

[57] Message Sequence Charts DL
6147.205.96.100; Condat GmbH

[58] Users Guide
6147.300.96.100; Condat GmbH

[59] Test Specification CC
6147.400.97.100; Condat GmbH

[60] Test Specification SS
6147.401.97.100; Condat GmbH

[61] Test Specification SMS
6147.402.97.100; Condat GmbH

[62] Test Specification MM
6147.403.97.100; Condat GmbH

[63] Test Specification RR
6147.404.97.100; Condat GmbH

[64] Test Specification DL
6147.405.97.100; Condat GmbH

[65] Test Specification CCD
6147.406.97.100; Condat GmbH

[66] SDL Specification CC
6147.500.97.100; Condat GmbH

[67] SDL Specification SS
6147.501.97.100; Condat GmbH

[68] SDL Specification SMS
6147.502.97.100; Condat GmbH

[69] SDL Specification MM
6147.503.97.100; Condat GmbH

[70] SDL Specification RR
6147.504.97.100; Condat GmbH

[71] SDL Specification DL
6147.505.97.100; Condat GmbH

[72] Message Specification CC
6147.600.97.100; Condat GmbH

[73] Message Specification SS
6147.601.97.100; Condat GmbH

[74] Message Specification SMS
6147.602.97.100; Condat GmbH

[75] Message Specification MM
6147.603.97.100; Condat GmbH

[76] Message Specification RR
6147.604.97.100; Condat GmbH

[77] Message Specification DL
6147.605.97.100; Condat GmbH

[78] Technical Documentation CC
6147.700.97.100; Condat GmbH

[79] Technical Documentation SS
6147.701.97.100; Condat GmbH

[80] Technical Documentation SMS
6147.702.97.100; Condat GmbH

[81] Technical Documentation MM
6147.703.97.100; Condat GmbH

[82] Technical Documentation RR
6147.704.97.100; Condat GmbH

[83] Technical Documentation DL
6147.705.97.100; Condat GmbH

[84] Technical Documentation CCD
6147.706.97.100; Condat GmbH

## 1.2 Abbreviations

AGCH Access Grant Channel
BCCH Broadcast Control Channel
BS Base Station
BSIC Base Station Identification Code
CBCH Cell Broadcast Channel
CBQ Cell Bar Qualify
CC Call Control
CCCH Common Control Channel
CCD Condat Coder Decoder
CKSN Ciphering Key Sequence Number
C/R Command / Response

C1 Path Loss Criterion
C2 Reselection Criterion
DCCH Dedicated Control Channel
DISC Disconnect Frame
DL Data Link Layer
DM Disconnected Mode Frame
EA Extension Bit Address Field
EL Extension Bit Length Field
EMMI Electrical Man Machine Interface
F Final Bit
FACCH Fast Associated Control Channel
FHO Forced Handover
GP Guard Period
GSM Global System for Mobile Communication
HPLMN Home Public Land Mobile Network
I Information Frame
IMEI International Mobile Equipment Identity
IMSI International Mobile Subscriber Identity
Kc Authentication Key
L Length Indicator
LAI Location Area Information
LPD Link Protocol Discriminator
M More Data Bit
MCC Mobile Country Code
MM Mobility Management
MMI Man Machine Interface
MNC Mobile Network Code
MS Mobile Station
NCC National Colour Code
NECI New Establishment Causes included
N(R) Receive Number
N(S) Send Number
OTD Observed Time Difference
P Poll Bit
PCH Paging Channel
PDU Protocol Description Unit
P/F Poll / Final Bit
PL Physical Layer
PLMN Public Land Mobile Network
RACH Random Access Channel
REJ Reject Frame
RNR Receive Not Ready Frame
RR Radio Resource Management
RR Receive Ready Frame
RTD Real Time Difference
SABM Set Asynchronous Balanced Mode
SACCH Slow Associated Control Channel
SAP Service Access Point
SAPI Service Access Point Identifier
SDCCH Slow Dedicated Control Channel
SIM Subscriber Identity Module
SMS Short Message Service
SMSCB Short Message Service Cell Broadcast
SS Supplementary Services
TCH Traffic Channel
TCH/F Traffic Channel Full Rate
TCH/H Traffic Channel Half Rate
TDMA Time Division Multiple Access
TMSI Temporary Mobile Subscriber Identity

UA Unnumbered Acknowledgement Frame
UI Unnumbered Information Frame
VPLMN Visiting Public Land Mobile Network
V(A) Acknowledgement State Variable
V(R) Receive State Variable
V(S) Send State Variable

## 1.3 Terms

| | |
|---|---|
| Entity: | Program which executes the functions of a layer |
| Message: | A message is a data unit which is transferred between the entities of the same layer (peer-to-peer) of the mobile and infrastructure side. Message is used as a synonym to protocol data unit (PDU). A message may contain several information elements. |
| Primitive: | A primitive is a data unit which is transferred between layers on one component (mobile station or infrastructure). The primitive has an operation code which identifies the primitive and its parameters. |
| Service Access Point: | A Service Access Point is a data interface between two layers on one component (mobile station or infrastructure). |

# 2 Overview

The Protocol Stacks are used to define the functionality of the GSM protocols for interfaces. The GSM specifications are normative when used to describe the functionality of interfaces, but the stacks and the subdivision of protocol layers does not imply or restrict any implementation.

The base of the Protocol Stack rests on the physical layer.

The Data Link Layer (DL) is used to handle an acknowledged connection between mobile and base station. The LAPDm protocol is used.

Radio Resource (RR) manages the resources of the air-interface. That means configuration of physical layer, cell selection and cell reselection, data transfer, RR-Connection handling.

Mobility Management (MM) handles registration aspects for the mobile station. It detects changes of location areas and updates a mobile station in the new location area.

Call Control (CC) provides the call functionality. This includes call establishment, call maintenance procedures like Hold, Retrieve or Modify, and call disconnection.

Supplementary Services (SS) handles all call independent supplementary services like call forwarding or call barring.

Short Message Services (SMS) is used for sending and receiving point-to-point short messages. Additionally the reception of cell broadcast short messages is included.

The man machine interface (MMI) is the interface to the user. Normally it is connected with a keypad as input device and a display as output device.

Between the several entities data interfaces are defined. These data interfaces are called Service Access Points (SAPs), indicating that an upper layer uses the services of a lower layer.

The GSM specification do not set out any implementation of the Protocol Stack. The following diagrams show the implementation described in all these documents for the mobile station. All entities except the Man Machine Interface and Physical Layer are implemented as part of the Protocol Stack.

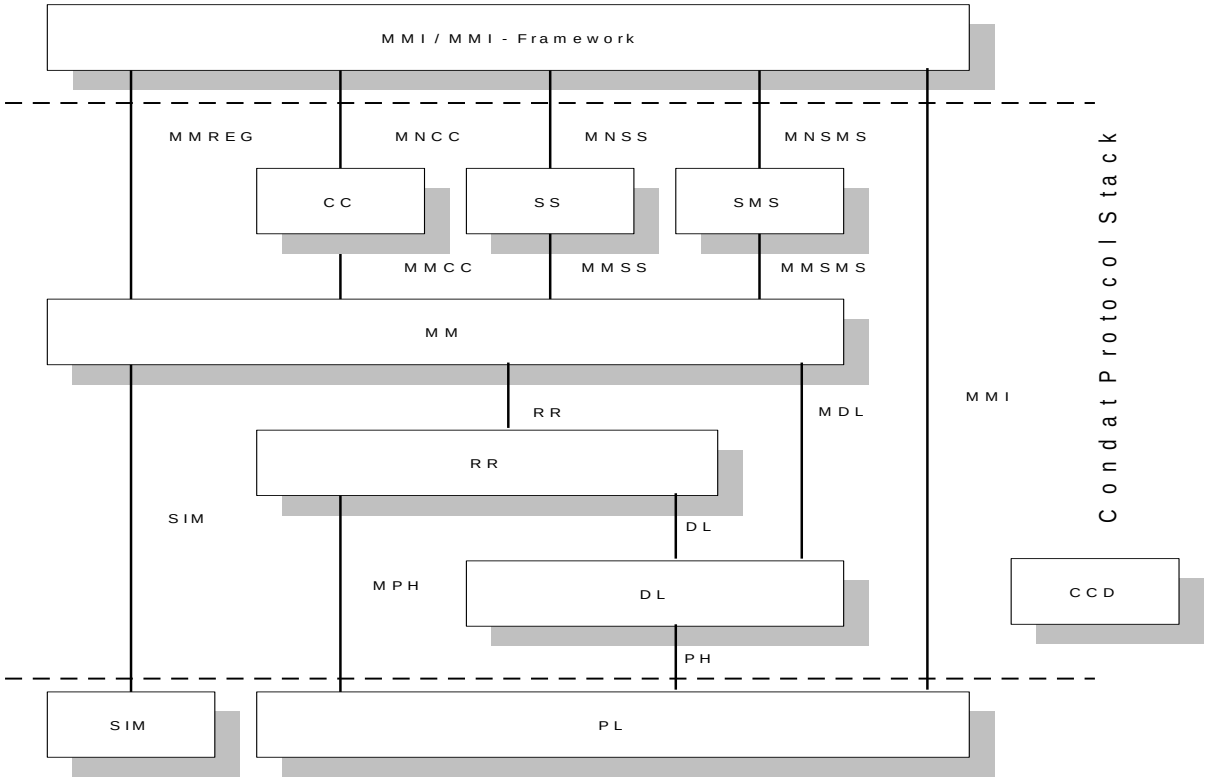


Figure 1: Mobile-station protocol architecture

This document is the technical documentation for the Man Machine Interface (MMI).

# 3 Structure

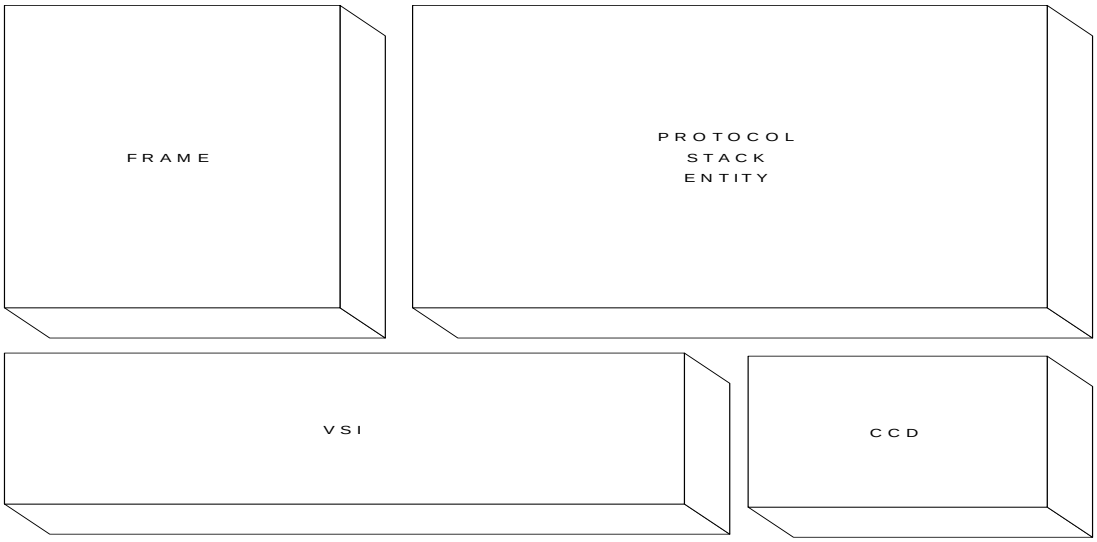


Figure 3: Structure of Protocol Stack Entity

Each protocol stack entity has the same structure and is build by the following modules

- Frame
- Protocol Stack Entity
- Virtual System Interface and
- Condat Coder Decoder

**Frame**
The frame carries out the work depending on the operating system, like startup or shutdown of a entity. This module is implemented by the customer and linked to the protocol stack entity. The interface between both is the protocol stack entity interface (PEI).

**Protocol Stack Entity**
The protocol stack entities are implemented without any dependencies on the underlying operating system. And they contain the protocol stack functionality as described into the ETSI Recommendations. They are sub-divided in the three modules protocol stack entity interface, custom specific functions and finite state machine.
The protocol stack interface module defines the interface to the frame. Custom specific functions are used to configure the protocol stack at run-time with custom specific data from a non-erasable memory for example. The finite state machine modules implement the protocol stack logic.

**Virtual System Interface**
The protocol stack entities requires operation system functionality like communication and timer. The requirements are low and independent from the operating system, so the system interface is virtual from the view of protocol stack entities. The module virtual system interface implements this interface. This implementation is carried out by pre-processor definitions or by function calls.

**Condat Coder Decoder**
GSM message at the air-interface are bit streams. The most efficient method to handle this GSM messages into a protocol stack entity is to convert this bit streams into corresponding C-Structures.

This is done by the module Condat Coder Decoder. It makes sense to integrate this module as a server into the target system because this module is needed by more than one entity.

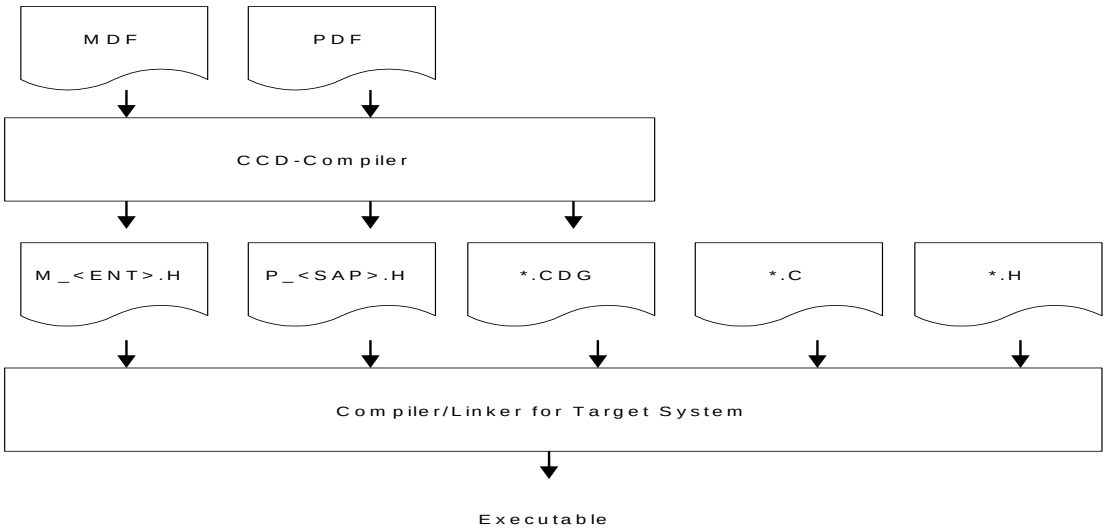


Figure 4: Generating a Protocol Stack Entity

Generating a protocol stack entity is carried out in two steps:

- Generating primitives and message structures
- Compiling / linking for target system

The message definition files (MDF) and primitive definition files (PDF) contain a description for all messages and primitives independent from a target system. The CCD-Compiler generates target dependent header files and tables. Parameters of the CCD-Compiler are Alignment and Byte-ordering of the target system.
The generated header files and the delivered header and source files are compiled and linked for the target system to obtain a executable for the target system.

# 4 Modules

The protocol stack entity DL for the mobile station is divided in the following modules:

- dl_ccch.c
- dl_com.c
- dl_csf.c
- dl_dph.c
- dl_drr.c
- dl_facch.c
- dl_pei.c
- dl_sacch.c
- dl_sdcc3.c
- dl_sdcch.c
- dl_tim.c
- tok.c

**dl_ccch.c**
This module defines the message handling for the common control channels and the cell broadcast control channel. The received message format from PL is converted to the expected format of RR.
**dl_com.c**
This module defines common used functions for building frames etc.
**dl_csf.c**
The customer specific functions are carried out in this module. The module contains the functions needed for the timer pool.
**dl_dph.c**
Incoming frames of the PL are checked and distributed by this module. Errors are detected and MM is informed.
**dl_drr.c**
Incoming primitives from RR are checked and distributed according to the given channel and service access point identifier.

**dl_facch.c**
The protocol for channel FACCH and service access point identifier 0 is located in this module.
**dl_pei.c**
The module implements the interface to the process frame. That means initialising of protocol stack entity data, open of communication and timer resources and routing of incoming primitives. Handling of time-outs, dynamic configuration and monitoring are located in this module.
**dl_sacch.c**
The protocol for channel SACCH and service access point identifier 0 is located in this module. Additionally reception and sending of unacknowledged data is processed by the module.
**dl_sdcc3.c**
The protocol for channel SDCCH and service access point identifier 3 is located in this module.
**dl_sdcch.c**
The protocol for channel SDCCH and service access point identifier 0 is located in this module.
**dl_tim.c**
The module contains functions for the timer handling. That means allocation of a timer from the timer pool, start, stop of configure of a timer and release of an allocated timer.
**tok.c**
This module defines some parse functions for the dynamic configuration strings.

# 5 Header

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 in a specific target system.

- ccdapi.h (*)
- dl.h
- cnf_dl.h (*)
- custom.h (*)
- cus_dl.h (*)
- gsm.h
- mconst.cdg
- message.h
- mon_dl.h
- pconst.cdg
- pei.h (*)
- prim.h
- p_ph.h
- p_dl.h
- p_mdl.h
- stddefs.h
- string.h
- tok.h
- vsi.h (*)

**ccdapi.h**
The header defines the prototypes and some constants for the Condat Coder Decoder (CCD).
**dl.h**
The header contains constants for the data link layer and the prototypes of the component.
**cnf_dl.h**
Constants for dynamic configuration of DL are defined in this header. It is allowed to change the commands and the parameter names for the supported dynamic configurations.
**custom.h**
The 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 and some more.
**cus_dl.h**
Custom specific definitions for the protocol stack entity are located in this header. Timer values and identifier are changeable. A version identifier is defined.
**gsm.h**
The header contains global definitions for all protocol stack entities. Depending on the definitions in custom.h many options and traces are defines in this header.
**mconst.cdg**
This header is generated by the CCD compiler. It includes all message identifier and some constants needed by the entities.
**message.h**
Constants for messages are defined. DL messages are coded and decoded by low level functions of CCD.
**mon_dl.h**
Constants for monitoring of DL are defined in this header.
**pconst.cdg**
This header is generated by the CCD compiler. It includes all primitive identifier and some constants needed by the entities.
**pei.h**
Prototypes for the protocol stack entity interface are defined in this header. Some parameter and the return type of this function are changeable by the user.

**prim.h**
Constants for primitives are defined and service access point dependent primitive header files are included (p_ph.h., p_dl.h and p_mdl.h).
**p_dl.h**
This header is generated by the CCD compiler. It includes the C-struct type definitions for primitives of the service access point DL. The header is included by prim.h
**p_mdl.h**
This header is generated by the CCD compiler. It includes the C-struct type definitions for primitives of the service access points MDL. The header is included by prim.h
**p_ph.h**
This header is generated by the CCD compiler. It includes the C-struct type definitions for primitives of the service access points PH. The header is included by prim.h
**stddefs.h**
The header contains several standard definitions used by the protocol stack entities.
**string.h**
This header is a the standard string header from the target compiler. It defines string and memory functions.
**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 parameter and the return types of these function are changeable by the user for integration in a specific target system.

# 6 Configuration

The protocol stack software is independent from a specific target system. The step from the independent protocol stack software to a target system is carried out by configuration and by custom specific functions. There are three ways to configure the protocol stack entity:

- static configuration
- dynamic configuration
- custom specific functions

## 6.1 Static Configuration

Static configuration is carried out by setting constants or types in some header files before compiling the sources. The following header files are changeable by the user:

- ccdapi.h
- cnf_dl.h
- custom.h
- cus_dl.h
- pei.h
- vsi.h

Some configurations are valid for all protocol stack entities (defined in ccdapi.h, custom.h, pei.h and vsi.h). They are described in the Users guide. The entity specific configurations are defined in cnf_dl.h and cus_dl.h and described in the following:

**TIMER_VALUES (cus_dl.h)**
The constants define the timer values for the DL timer T200 depending on the various channel types and service access point identifiers. If your target system uses other units please change the values.

**TIMER IDENTIFIER (cus_dl.h)**
The constants define the identifier of the timer resources of DL.

**MAX_DL_TIMER (cus_dl.h)**
The constant define the number of timer available in the timer pool.

**PRIMITIVE SIZES (cus_dl.h)**
The primitive size of PH_DATA_REQ depends on the used channel. It is defined separately to achieve an optimum of transmitted bytes to physical layer. The default value which is defined here is calculated without any alignment bits.

E.g. SACCH

| | | |
|---|---|---|
| N201 | = | 18 |
| l2 header | = | 3 |
| sdu offset | = | 2 |
| sdu length | = | 2 |
| ch_type | = | 1 |
| dummy | = | 1 |
| --- | | |
| total | = | 27 |

**FRAME_OFFSET_ZERO (cus_dl.h)**
If the constant is set, PL ensures that all frames are send with offset zero to DL. This enables a faster access on frame header parameter.

**MAX_QUEUED_MESSAGES (cus_dl.h)**
The constant define the maximum number of stored layer 3 messages in data link layer.

**VERSION (cus_dl.h)**
The constants define the type and the value of a version identification. The version is part of the monitor struct.

**CONFIGURATION PARAMETER (cnf_dl.h)**
The constants define the commands for dynamic configuration proposals.

## 6.2 Dynamic Configuration

Dynamic configuration means to change the behaviour of the protocol stack entity at run-time. This is carried out by sending a string with a dedicated format as described in the Users Guide. An additionally feature is to request the old configuration. This feature is may be switched off by an option defined in custom.h.
The dynamic configuration string is a parameter of the pei_config () function which is part of the protocol stack entity interface (PEI). The keywords for the dynamic configurations are changeable by the static configuration mechanism.

**RESET**
The dynamic configuration RESET is used to make a warm start of DL belonging the GSM functionality.

**TIMER_SET =<timer, value>**
The timer mode TIMER_SET defines a new timer value instead of the origin start value.

**TIMER_RESET =timer**
The default timer mode is TIMER_RESET which affects in no manipulation of 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.

**N200 = <chan_sapi, value>**
The configuration N200 is used to redefine the retransmission counter N200. The default value depends on the used channel.

**VS = <chan_sapi, value>** **single instance system**
**VS = <inst, chan_sapi, value>** **multi instance system**
The configuration VS manipulates the send variable V(S) on the defined channel. For multi instance systems this configuration is carried out only for the defined instance. This is used to force error situations on data link layer.

**VR = <chan_sapi, value>** **single instance system**
**VR = <inst, chan_sapi, value>** **multi instance system**
The configuration VR manipulates the receive variable V(R) on the defined channel. For multi instance systems this configuration is carried out only for the defined instance. This is used to force error situations on data link layer.

**VA = <chan_sapi, value>** **single instance system**
**VA = <inst, chan_sapi, value>** **multi instance system**
The configuration VA manipulates the acknowledge variable V(A) on the defined channel. For multi instance systems this configuration is carried out only for the defined instance. This is used to force error situations on data link layer.

**UA_IGNORE = <chan_sapi, value>** **single instance system**
**UA_IGNORE = <inst, chan_sapi, value>** **multi instance system**
To force repetition of sending SABM or DISC frames it is possible to ignore incoming UA frames on the defined channel. The number of ignored frames is defined by the parameter value. For multi instance systems this configuration is carried out only for the defined instance.

**RR_IGNORE = <chan_sapi, value>** **single instance system**
**RR_IGNORE = <inst, chan_sapi, value>** **multi instance system**
To force repetition of sending I frames it is possible to ignore incoming RR frames on the defined channel. The number of ignored frames is defined by the parameter value. For multi instance systems this configuration is carried out only for the defined instance.

**I_IGNORE = <chan_sapi, value>** **single instance system**
**I_IGNORE = <inst, chan_sapi, value>** **multi instance system**
To force repetition of received frames it is possible to ignore incoming I frames on the defined channel. The number of ignored frames is defined by the parameter value. For multi instance systems this configuration is carried out only for the defined instance.

**ADDR_FIELD = value** **single instance system**
**ADDR_FIELD = <inst, value>** **multi instance system**
To force error situation it is possible to create a frame content, which is send with the right timing on the defined channel (see configuration SEND_FRAME). With this configuration it is possible to define the address field of this frame. For multi instance systems this configuration is carried out only for the defined instance.

**CTRL_FIELD = value** **single instance system**
**CTRL_FIELD = <inst, value>** **multi instance system**
To force error situation it is possible to create a frame content, which is send with the right timing on the defined channel (see configuration SEND_FRAME). With this configuration it is possible to define the control field of this frame. For multi instance systems this configuration is carried out only for the defined instance.

**LENGTH_FIELD = value** **single instance system**
**LENGTH_FIELD = <inst, value>** **multi instance system**
To force error situation it is possible to create a frame content, which is send with the right timing on the defined channel (see configuration SEND_FRAME). With this configuration it is possible to define the length field of this frame. For multi instance systems this configuration is carried out only for the defined instance.

**INFO_FIELD = value** **single instance system**
**INFO_FIELD = <inst, value>** **multi instance system**
To force error situation it is possible to create a frame content, which is send with the right timing on the defined channel (see configuration SEND_FRAME). With this configuration it is possible to define the information field of this frame. For multi instance systems this configuration is carried out only for the defined instance.

**SEND_FRAME = chan** **single instance system**
**SEND_FRAME = <inst, chan>** **multi instance system**
To force error situation it is possible to create a frame content, which is send with the right timing on the defined channel. For multi instance systems this configuration is carried out only for the defined instance.

## 6.3 Custom Specific Functions

Custom specific functions are implemented in the module dl_csf.c. It is allowed to replace the functions in this module by functions of the customer. It is not allowed to change parameters of the functions.
The sense of this 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 initialises 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_DL_DATA * dl_data, T_VSI_TVALUE value)**
The function allocates one timer of the timer pool and starts the timer.

**GLOBAL void csf_free_timer (T_VSI_THANDLE handle)**
The function frees one timer. The timer is stopped and given back to the timer pool.

**GLOBAL BOOL csf_vdb_timeout (T_VSI_THANDLE handle, T_DL_DATA ** dl_data, USHORT * timer)**
After time-out the according instance is searched. The timer is given back to the timer pool.

# 7 Monitoring

The monitor struct includes relevant physical parameter of the protocol stack entity. The parameters are updated at every time. By this way the environment has always the possibility of access to parameters of the protocol stack. This parameters are used to create monitor reports to a display or a test system, to create statistical data or other things outside the functionality of a protocol stack but with access to protocol stack parameters. It is allowed to read the parameters of the monitor struct and it is strictly forbidden to write to the monitor struct. The first parameter of the monitor struct is the version of the protocol stack entity.
For the protocol stack entity the following monitor struct is defined:

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

# 8 Resources

## 8.1 Timer

Four timers are defined for each DL instance. In parallel a maximum of two timers per instance is possible (to control service access point 0 and 3).

## 8.2 Memory

The following sizes are measured with a TURBO-C 2.0 Compiler for a INTEL-PC. Options and traces are set as defined in the header at delivery. The size in a target system depends on the controller, the operating system, the compiler and the used options. Consider that a lot of trace output is included at delivery.

| Module | Code | Data | Bss |
|---|---|---|---|
| DL_CCCH.C | 208 | 91 | 0 |
| DL_COM.C | 3019 | 416 | 0 |
| DL_CSF.C | 784 | 88 | 6 |
| DL_DPH.C | 4264 | 401 | 0 |
| DL_DRR.C | 1006 | 197 | 0 |
| DL_FACCH.C | 6768 | 410 | 0 |
| DL_PEI.C | 2373 | 1670 | 1970 |
| DL_SACCH.C | 6086 | 378 | 0 |
| DL_SDCC3.C | 6127 | 334 | 0 |
| DL_SDCCH.C | 6786 | 429 | 0 |
| DL_TIM.C | 836 | 227 | 26 |
| Total | 38257 | 4641 | 2002 |

# 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/>