Technical Document – Confidential

GSM PROTOCOL STACK

GPF

CCD – CONDAT CODER DECODER

APPLICATION INTERFACE SPECIFICATION

Document Number:	8415.024.02.104
Version:	0.7
Status:	Draft
Approval Authority:	
Creation Date:	1999-Jul-07
Last changed:	2015-Mar-08 by XINTEGRA
File Name:	ccd_api.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-Jul-07	IH et al.		0.1	Submitted	1
1999-Nov-01	HJS et al.		0.2	Accepted	
2000-May-16	SIJ		0.3		2
2001-Oct-18	SIJ		0.4		3
2001-Dec-12	SIJ		0.5		4
2002-Oct-22	SIJ		0.6		5
2003-May-20	XINTEGRA		0.7	Draft	

Notes:

1. Initial version
2. Updated function parameters; removed T-FACILITY
3. Added description of ccd_decodeMsgPtr
4. Changed Parameters of ccd_decodeMsgPtr; added description of ccd_codeMsgPtr; added hint on l_buf to ccd_codeMsg
5. Added function and type descriptions related to the extended interface for access to error lists

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 Introduction

In the world of GSM there are messages transmitted over the air-interface. For GSM protocols, these messages are bit strings of variable length, formally a succession of a finite, possibly null, number of bits (i.e., elements of the set {"0", "1"}), with a beginning and an end. Data in GSM protocol stack entities are normally hold in c-structures. To encode bitstream-messages from data in c-structures and vice versa there is the Condat Coder Decoder. Additionally there are some functions included to code and decode simple data types like byte and long.
This document describes the functions which are used for coding and decoding. Also the data types, constants and parameters for the functional specification are introduced.

2 Interface description of the Condat Coder Decoder

2.1 Data types

Name	Description
T_MSGBUF	contains the coded message
T_CCD_ERR_ENTRY	contains general information on the reported error
T_CCD_ERR_PARA	contains detailed information on the reported error
T_CCD_PARA_LIST	contains parameter accompanying the reported error
T_CCD_ERR_TYPE	contains CCD identifier and address of the erroneous element
T_CCD_ERR_INFO	CCD identifier of the erroneous element
T_CCD_ID	CCD identifier of the erroneous element

2.1.1 T_MSGBUF – Coded Message Data Type

Definition:

```
typedef struct
{
  USHORT l_buf;
  USHORT o_buf;
  UBYTE  buf[MAX_BITSTREAM_LEN];
} T_MSGBUF;
```

Description:

T_MSGBUF contains the coded message.

The member o_buf specified the offset (in bits), where the message starts in the buffer.

The member l_buf contains the length (in bits) of the coded message.

The member buf contains the bitcoded message as an array of bytes.

2.1.2 T_CCD_ERR_ENTRY – C-structure for general information on a detected error

Definition:

```
typedef struct
{
  UBYTE error;
  UBYTE kind;
  T_CCD_ERR_PARA para;
} T_CCD_ERR_ENTRY;
```

Description:

In the now extended error access interface a C-structure of type T_CCD_ERR_ENTRY contains general information about the detected errors.

The member error specifies the error code, ERR_NO_MORE_ERROR e.g., as listed in ccdapi.h.

The member kind specifies how to interpret the member para. Using the C-macros CCD_ERR_KIND_PARA_LIST and CCD_ERR_KIND_IE_TYPE, currently two different values for kind are planned to choose between error parameter and error type list. Each error report uses only one of the two formats depending on the source of the error report.

NOTE: For version compatibility CCD still supports the old style of error access which uses calls to ccd_getFirstError() and ccd_getFirstError(). To assure this, kind is set to CCD_ERR_KIND_PARA_LIST for all error codes which have been supported in the previous versions.

The member para contains detailed information on the reported error. See also type description for T_CCD_ERR_PARA.

2.1.3 T_CCD_ERR_PARA – C-structure for detailed information on a detected error

Definition:

```
typedef union
{
  T_CCD_PARA_LIST para_list;
  T_CCD_ERR_TYPE error_type;
} T_CCD_ERR_PARA;
```

Description:

In the now extended error access interface T_CCD_ERR_PARA contains detailed information about the detected errors. For compatibility the union type is used to support both old and new type of error information container. These two types are described in the next sections.

2.1.4 T_CCD_PARA_LIST – C-structure for the list of parameters related to a detected error

Definition:

```
typedef union
{
  UBYTE num_para;
  USHORT error_list[MAX_ERR_PAR];
} T_CCD_PARA_LIST;
```

Description:

T_CCD_PARA_LIST contains room for parameters of a detected error status. If present such parameters help the CCD caller to make a clearer diagnosis. Each error code is accompanied by parameter types specific to and meaningful for it.

The member num_para gives the count of parameters stored to the list.

The member error_list contains the parameters.

2.1.5 T_CCD_ERR_TYPE – C-structure for a direct access to an erroneous element in the message

Definition:

```
typedef struct
{
  T_ERR_INFO err_info;
  ULONG err_IEaddr;
} T_CCD_ERR_TYPE;
```

Description:

T_CCD_ERR_TYPE contains information which help to access (and manipulate) the element reported to be as erroneous.

The member err_info has the value of an enumerated type which uniquely identifies each information element among all the possible IEs. This enumeration is accessible through including of the file ccdid.h.

The member err_IEaddr is the address of that element in the C-structure of the decoded message.

2.1.6 T_CCD_ERR_INFO – C-enum type to identify the erroneous element in the message

Definition:

```
typedef T_CCD_ID T_ERR_INFO;
```

Description:

T_CCD_ERR_INFO is defined to be a C-enum type. This enumeration is accessible through including of the file ccdid.h. See also the description for T_CCD_ID below.

2.1.7 T_CCD_ID – C-enum type to uniquely identify each element in the message

Definition:

```
typedef T_CCD_ID T_ERR_INFO;
```

Description:

T_CCD_ID is defined to be a C-enum type. This enumeration is accessible through including of the file ccdid.h. An example of how ccdid.h looks like is given below:

```
typedef enum
{
/*    0*/ CCD_ID_RR_COM__mob_class_1__spare_0,
/*    1*/ CCD_ID_RR_COM__mob_class_1__rev_lev,
…
/* 3027*/ CCD_ID_SM__SM_STATUS__sm_cause,
/*65535*/ CCD_ID_END = 0x7fffffff
} T_CCD_ID;
```

This enumeration specifies each information elements uniquely among all the possible IEs of all the messages known to CCD.

NOTE: The name of any member of T_CCD_ID is made of the its own name, the name of the IE wrapping it and the name of the message description file, where the IE is defined:

CCD_ID_$(Prefix|Filename)__$(COMP-name)__$(AS-name|VAR-name|COMP-name etc.)

2.2 Constants

Name	Description
UPLINK	Direction parameter for coding / decoding
DOWNLINK	
BOTH	
ccdOK	CCD function return code
ccdWarning	
ccdError	
CCD_REENTRANT	Parameter for ccd_register
CCD_ERR_KIND_PARA_LIST	format kind for error information container
CCD_ERR_KIND_IE_TYPE	
MAX_ERR_PAR	size of parameter list for each reported error
ERR_INVALID_CALC	Error codes to be accessed by ccd_getFirstError() and ccd_getNextError() or
by	
	ccd_getFirstFault() and ccd_getNextFault()
ERR_INVALID_IEI	
ERR_PATTERN_MISMATCH	
ERR_COMPREH_REQUIRED	
ERR_IE_NOT_EXPECTED	
ERR_IE_SEQUENCE	
ERR_MAX_IE_EXCEED	
ERR_MAX_REPEAT	
ERR_MAND_ELEM_MISS	
ERR_INVALID_MID	
ERR_INVALID_TYPE	
ERR_EOC_TAG_MISSING	
ERR_INTERNAL_ERROR	
ERR_NO_MORE_ERROR	
ERR_NO_MEM	
ERR_BUFFER_OF	
ERR_DEFECT_CCDDATA	
ERR_NONCRITICAL_EXT	
ERR_INT_VALUE	Error codes to be accessed by ccd_getFirstFault() and ccd_getNextFault()
ERR_ASN1_ENCODING	
ERR_ASN1_MAND_IE	
ERR_ASN1_OPT_IE	
ERR_CRITICAL_EXT	

2.3 Functions

Name	Description
ccd_init	initializes the Condat Coder Decoder
ccd_register	initializes the Condat Coder Decoder
ccd_exit	withdraws the caller from CCD services
ccd_codeMsg	encodes a C-structure containing the C-representation of an air-interface message to a bit stream.
ccd_decodeMsg	decodes a bit stream, containing a valid message to a corresponding C-structure
ccd_decodeByte	converts a bit stream (1-8 bit) to a numeric value
ccd_codeByte	encodes a value into a CPU-dependent bit string (byte)
ccd_codeLong	encodes a value into a CPU-dependent bit string (long)
ccd_decodeLong	converts a bit stream (1-32 bit) to a numeric value
ccd_bitcopy	allows bit-wise copying with offset and shift operation
ccd_getFirstError	returns the first error code from a list
ccd_getNextError	returns the next error code from a list
ccd_get_numFaults	returns the count of errors/faults of the last coding/decoding action for CCD caller
ccd_getFirstFault	copies information on the first detected error into the function parameter
ccd_getNextError	copies information on the next detected error into the function parameter
ccd_free_faultlist	frees any allocated error/fault information for CCD caller

2.3.1 ccd_init – Condat Coder Decoder Initialization

Definition:

BYTE ccd_init (void)

Parameters:

Name	Description
-	

Return values:

Name	Description
ccdOK	successful
ccdError	an error occurred

Description

This function initializes the Condat Coder Decoder. It must be called once before any coding or decoding takes place. A repeating call to ccd_init will not lead to repeat initialization actions.

See also

ccd_init

2.3.2 ccd_register – Coder/Decoder Initialization

Definition:

int ccd_register (int decmsgbuf_size)

Parameters:

Name	Description
decmsgbuf_size	Reserved for further enhancements. In the current version this parameter must be set to CCD_REENTRANT.

Return values:

Name	Description
ccdOK	successful
ccdError	an error occurred

Description

This function initializes the Coder/Decoder. It must be called with the parameter CCD_REENTRANT instead of ccd_init if the calling entity shall use CCD in a reentrant manner, i.e. without being synchronized with other entities when using CCD. It must be guaranteed that enough memory in the D-partitions is available that CCD can allocate a set of local data for each task that must use CCD in a reentrant manner. The batch file \gpf\CCD\util\globs.bat can be executed to print the size of memory needed for one task.

2.3.3 ccd_exit – withdraw from CCD services

Definition:

int ccd_exit (void)

Parameters:

Name	Description
-	

Return values:

Name	Description
ccdOK	successful
ccdError	an error occurred

Description

This function withdraws the caller from CCD services. It should be called in pei_exit.

2.3.4 ccd_codeMsg - encodes a message (C-structure) to a bit stream

Definition:

BYTE ccd_codeMsg (UBYTE entity, UBYTE direction, T_MSGBUF *mBuf, UBYTE *mStruct, UBYTE mId)

Parameters:

Name	Description
entity	ID of the calling entity. Valid entity identifiers are defined as C-macros in MCONST.CDG.
direction	specifies whether the message goes UPLINK or DOWNLINK. This is necessary because the same PDU-Type can be used for both directions while the message structures can be different.
mBuf	specifies the bit stream buffer of the message. The elements of this structure are l_buf, o_buf and buf. The two first elements specify the length and offset of the bit stream buf. The o_buf component must be specified by the caller. The l_buf component is calculated by CCD in the older versions or is specified by the caller in the new version of CCD.
mStruct	references to the C-structure containing the C-representation of the decoded message. The type should be casted to UBYTE*. The first element must contain the message type (PDU) as a UBYTE value. If this parameter is NULL, CCD uses its internal buffer which must be protected via ccd_begin() in a multithread environment.
mId	specifies the PDU type of the bit stream. CCD reads this value as the message ID if it is not equal to 0xff or 0xfe. Normally this parameter is set to 0xff and CCD reads the PDU type from the first byte of the mStruct.

Return values:

Name	Description
ccdOK	successful
ccdError	error occurred and CCD stopped coding
ccdWarning	error(s) occurred and CCD continued coding

Description

This function encodes a C-structure containing the C-representation of a valid Air-interface message to a bit stream. The parameters entity, direction and mId are used to select the appropriate coding rules. The parameters l_buf and o_buf describe the encoded message in buf. Though they must be first set properly by the caller of CCD. Specially l_buf is used by CCD for ereasing memory in buf. The maximum values for message bit sizes are given in MCONST.CDG. After encoding CCD overwrites l_buf with the actual length of encoded message.
If CCD is used in a preemptive multithreaded system it uses semaphores. Thus the function will block if an entity is currently coding or decoding using CCD. The entity process is blocked untill the semaphore used by CCD is freed and the entity process is the next one in the semaphore queue. There is a deviation from this scenario for a few privileged entities which can always use CCD.

Using the given parameters if CCD is not able to start encoding, the return value of this function will be ccdError. If CCD starts encoding the bit stream and errors occur while encoding, the return value will be ccdError or ccdWarning. In order to classify the occurred errors and react on them the calling entity should check the CCD error list using functions ccd_getFirstError () and ccd_getNextError ().

And finally if CCD is successful on coding, the return value will be ccdOK.

2.3.5 ccd_decodeMsg - decodes a bit stream to the corresponding C-structure

Definition

BYTE ccd_decodeMsg (UBYTE entity, UBYTE direction, T_MSGBUF *mBuf, UBYTE * mStruct, UBYTE mId)

Name	Description
entity	ID of the calling entity. Valid entity identifiers are defined as C-macros in MCONST.CDG
direction	specifies whether the message goes UPLINK or DOWNLINK. This is necessary because the same PDU-Type can be used for both directions while the message structures can be different.
mBuf	specifies the bit stream buffer of the message. The elements of this structure are l_buf, o_buf and buf. The two first elements specify the length and offset of the bit stream buf. The o_buf and l_buf components must be specified by the caller.
mStruct	refers to the C-Structure of the decoded message. The type may differ so the pointer is always typed as UBYTE* and must be casted after decoding. The first element contains the message type (PDU) as a UBYTE after decoding.
mId	specifies the PDU type of the bit stream. CCD reads this value as the message ID if it is not equal to 0xff or 0xfe. Normally this parameter is set to 0xff and CCD reads the PDU type from the first (often 8) bits in the mBuf->buf.

Return values:

Name	Description
ccdOK	successful
ccdError	error occurred and CCD stopped decoding
ccdWarning	error(s) occurred and CCD continued decoding

Description

This Function decodes a bit stream, containing a message of the air-interface to a corresponding C-structure. The parameters entity, direction and mId are used to select the needed coding rules.

If CCD is used in a preemptive multithreaded system it uses semaphores. Thus the function will block if an entity is currently coding or decoding using CCD. The entity process is blocked until the semaphore used by CCD is freed and the entity process is the next one in the semaphore queue. There is a deviation from this scenario for a few privileged entities which can always use CCD.

If CCD is not able to start decoding, the return value of this function will be ccdError. If CCD starts decoding the bit stream and errors occur while decoding, the return value will be ccdError or ccdWarning. In order to classify the occurred errors and react on them the calling entity should check the CCD error list using functions ccd_getFirstError () and ccd_getNextError ().

And finally if CCD is successful in decoding, the return value will be ccdOK.

2.3.6 ccd_codeMsgPtr - encodes a message (C-structure) to a bit stream using dynamic arrays

Definition

S8 ccd_codeMsgPtr (U8 entity, U8 direction, U16 *l_buf, U16 o_buf, U8 *buf, U8* mStruct, U8 mId)

Name	Description
entity	specifies the calling entity of CCD. The Constants for each valid entity is defined in MCONST.CDG.
direction	specifies whether the message goes UPLINK or DOWNLINK. This is necessary because the same PDU-Type can be used for both direction while the message structures can be different.
l_buf	length of encoded message in the bit stream buffer (buf)
o_buf	offset in bits of the bit stream buffer (buf)
buf	specifies the bit stream buffer of the message.
mStruct	refers to the C-structure containing the C-representation of the decoded message. The type should be casted to U8*. The first element must contain the message type (PDU) as a U8 value. If this parameter is NULL CCD uses its internal buffer which must be protected via ccd_begin() in a multithread environment.
mId	specifies the PDU type of the bit stream. CCD reads this value as the message ID if it is not equal to 0xff or 0xfe. Normally this parameter is set to 0xff. Hence CCD reads the PDU type from the first (often 8) bits after o_buf in buf.

Return values:

Name	Description
ccdOK	successful
ccdError	error occurred and CCD stopped coding
ccdWarning	error(s) occurred and CCD continued coding

Description

This function encodes a C-structure containing the C-representation of a valid Air-interface message to a bit stream. It allows the use of pointer types in the C-structure. The parameters entity, direction and mId are used to select the appropriate coding rules. The parameters l_buf and o_buf describe the encoded message in buf. Though they must be first set properly by the caller of CCD. Specially l_buf is used by CCD for erasing memory in buf. The maximum values for message bit sizes are given in MCONST.CDG. After encoding CCD overwrites l_buf with the actual length of encoded message.
If CCD is used in a preemptive multithreaded system it uses semaphores. Thus the function will block if an entity is currently coding or decoding using CCD. The entity process is blocked until the semaphore used by CCD is freed and the entity process is the next one in the semaphore queue. There is a deviation from this scenario for a few privileged entities who can always use CCD.

If CCD is not able to start encoding, the return value of this function will be ccdError. If CCD starts encoding the bit stream and errors occur while encoding, the return value will be ccdError or ccdWarning. In order to classify the occurred errors and react on them the calling entity should check the CCD error list using functions ccd_getFirstError () and ccd_getNextError ().

And finally if CCD is successful on coding, the return value will be ccdOK. In all cases it is the task of the calling entity to free the allocated buffer for dynamic arrays.

2.3.7 ccd_decodeMsgPtr - decodes a bit stream to the C-structure using dynamic arrays

Definition

U8 ccd_decodeMsgPtr (U8 entity, U8 direction, U16 l_buf, U16 o_buf, U8 *buf, U8 **mStructPtr, U8 mId)

Name	Description
entity	specifies the calling entity of CCD. The Constants for each valid entity is defined in MCONST.CDG.
direction	specifies whether the message goes UPLINK or DOWNLINK. This is necessary because the same PDU-Type can be used for both direction while the message structures can be different.
l_buf	length of the bit stream to be decoded
o_buf	offset in bits of the bit stream to be decoded
buf	specifies the bit stream buffer of the message.
mStructPtr	points to the pointer on the C-structure of the decoded message. The buffer containing this message structure will be allocated by CCD. After decoding the first element in the message C-structure contains the message (PDU) type as a UBYTE.
mId	specifies the PDU type of the bit stream. CCD reads this value as the message ID if it is not equal to 0xff or 0xfe. Normally this parameter is set to 0xff. Hence CCD reads the PDU type from the first (often 8) bits after o_buf in buf.

Return values:

Name	Description
ccdOK	successful
ccdError	error occurred and CCD stopped decoding
ccdWarning	error(s) occurred and CCD continued decoding

Description

This Function decodes a bit stream, containing a message of the air-interface to a corresponding C-structure. It allows the use of pointer types in the C-structure. The parameters entity, direction and mId are used to select the needed coding rules. The parameters l_buf and o_buf describe the decoded message in buf and must be set properly by caller of CCD.

If CCD is used in a preemptive multithreaded system it uses semaphores. Thus the function will block if an entity is currently coding or decoding using CCD. The entity process is blocked until the semaphore used by CCD is freed and the entity process is the next one in the semaphore queue. There is a deviation from this scenario for a few privileged entities who can always use CCD.

If CCD is not able to start decoding, the return value of this function will be ccdError and *mStructPtr will be NULL. If CCD starts decoding the bit stream and errors occur while decoding, the return value will be ccdError or ccdWarning. In both cases it is the task of the calling entity to free the allocated buffer referred to by *mStructPtr. In order to classify the occurred errors and react on them the calling entity should check the CCD error list using functions ccd_getFirstError () and ccd_getNextError ().

And finally if CCD is successful in decoding, the return value will be ccdOK. Still the calling entity must free the allocated buffer.

2.3.8 ccd_decodeByte - converts a bit stream (1-8 bit) to a numeric value

Definition

BYTE ccd_decodeByte (UBYTE *bitstream, USHORT startbit, USHORT bitlen, UBYTE *value)

Parameters:

Name	Description
bitstream	points to the buffer containing the bit stream to be decoded.
startbit	bitoffset into the first byte of the bitstream
bitlen	number of the bits to decode (max. 8)
value	pointer to a byte where the result of the decoding is stored to

Return values:

Name	Description
ccdOK	successful
ccdError	an error occurred

Description

This Function converts a bit string beginning from the byte where bit stream is pointing to, at (bit-) position startbit. It concerns the number of bits given in the parameter bitlen and converts them to a numeric value. The bitlen may be between 1 and 8.

2.3.9 ccd_codeByte - encodes a value into a CPU-dependent bit string

Definition

BYTE ccd_codeByte (UBYTE *bitstream, USHORT startbit, USHORT bitlen, UBYTE value)

Name	Description
bitstream	points to a buffer for the result of the coding operation
startbit	bitoffset into the first byte of the bit stream
bitlen	number of the bits to be coded (max. 8)
value	value to be encoded

Return values:

Name	Description
ccdOK	successful
ccdError	an error occurred

Description:

This Function encodes the value of the parameter value into a CPU-dependent bit string. The length of the bit string is specified by the parameter bitlen. The function copies the bit string into the buffer where bit stream is pointing to, at the bit offset given by the parameter startbit. The bitlen may be between 1 and 8.

2.3.10 ccd_codeLong - encodes a value into a CPU-dependent bit string

Definition

BYTE ccd_codeLong (UBYTE *bitstream, USHORT startbit, USHORT bitlen, ULONG value)

Name	Description
bitstream	points to a buffer for the result of the coding operation
startbit	bit offset into the first byte of the bit stream
bitlen	number of the bits to be coded (max. 32)
value	value to be encoded

Return values:

Name	Description
ccdOK	successful
ccdError	an error occurred

Description:

This Function encodes the value of the parameter value into a CPU-dependent bit string. The length of the bit string is specified by the parameter bitlen. The function copies the bit string into the buffer where bit stream is pointing to, at the bit offset given by the parameter startbit. The bitlen may be between 1 and 32.

2.3.11 ccd_decodeLong - converts a bit stream to a numeric value

Definition

BYTE ccd_decodeLong (UBYTE *bitstream, USHORT startbit, USHORT bitlen, ULONG *value)

Name	Description
bitstream	points to the buffer containing the bit stream to be decoded
startbit	bit offset into the first byte of the bit stream
bitlen	number of the bits to decode (max. 32)
value	points to an unsigned long where the result of the decoding is stored to

Return values:

Name	Description
ccdOK	successful
ccdError	an error occurred

Description:

This Function converts a bit string beginning from the byte where bit stream is pointing to, at (bit-) position startbit.
It concerns the number of bits given in the parameter bitlen and converts them to a numeric value. The bitlen may be between 1 and 32.

2.3.12 ccd_bitcopy - allows bit by bit copying with offset and shift operation

Definition

void ccd_bitcopy (UBYTE *dest, UBYTE *source, USHORT bitlen, USHORT offset)

Name	Description
dest	points to the destination of the copy operation
source	points to the source of the copy operation
bitlen	number of bits to copy
offset	position of the first bit in the source bit field

Return values:

Name	Description

-

Description:

This Function copies the number of bits given in the parameter bitlen from the source buffer to the destination buffer.
The parameter offset contains the position of the first bit in the source bit field. This function may perform a left shift to adjust the most significant bit on byte boundaries of the first byte in the destination buffer.

2.3.13 ccd_getFirstError - returns the first error code

Definition

UBYTE ccd_getFirstError (UBYTE entity, USHORT *parlist)

Name	Description
entity	ID of the calling entity
parlist	points to a buffer where the error parameter will be stored if an error has occurred

Return values:

Name	Description
ccdOK	no error occurred
ERR_NO_MORE_ERROR	the end of the error list is reached
ERR_INVALID_CALC	calculation of the element repeat value failed
ERR_PATTERN_MISMATCH	a bit pattern was not expected (error parameter: bit position)
ERR_COMPREH_REQUIRED	check for comprehension required failed (error parameters: TAG and bit position)
ERR_IE_NOT_EXPECTED	an information element was not expected (error parameters: TAG and bit position)
ERR_IE_SEQUENCE	wrong sequence of information elements (error parameters: TAG and bit position)
ERR_MAX_IE_EXCEED	maximum amount of repeatable information elements has exceeded elements (error parameters: TAG and bit position)
ERR_MAX_REPEAT	a repeatable element occurs too often in the message
ERR_MAND_ELEM_MISS	a mandatory information element is missing
ERR_INVALID_MID	unknown message ID (error parameter: Message ID)
ERR_INVALID_TYPE	the information element is not a spare padding (error parameter: bit position)
ERR_EOC_TAG_MISSING	an indefinite length is specified for the ASN.1-BER encoded element but the corresponding end of Component TAG is missing (error parameters: TAG and bit position)
ERR_INTERNAL_ERROR	an internal CCD error occurred
ERR_NO_MORE_ERROR	end of the error list
ERR_NO_MEM	memory allocation failed for dynamic array addition.
ERR_BUFFER_OF	on coding, more bits were written as given by function parameter l_buf
ERR_DEFECT_CCDDATA	unexpected character in a CCD data table; wrong version of CCDDATA loaded
ERR_NONCRITICAL_EXT	on decoding, more bits were to be decoded than CCD expected

Description:

If at least one error is stored in the errorlist, this function copies the additional stored error parameter of the first error in the errorlist into the given parlist and returns the error code. If no error is in the list this function returns 0.

2.3.14 ccd_getNextError - returns the next error code

Definition

UBYTE ccd_getNextError (UBYTE entity, USHORT *parlist)

Name	Description
entity	ID of the calling entity
parlist	points to a buffer where the error parameter will be stored if an error has occurred

Return values:

Name	Description
ccdOK	no error occurred
ERR_NO_MORE_ERROR	the end of the error list is reached
ERR_INVALID_CALC	calculation of the element repeat value failed
ERR_PATTERN_MISMATCH	a bit pattern was not expected (error parameter: bit position)
ERR_COMPREH_REQUIRED	check for comprehension required failed (error parameters: TAG and bit position)
ERR_IE_NOT_EXPECTED	an information element was not expected (error parameters: TAG and bit position)
ERR_IE_SEQUENCE	wrong sequence of information elements (error parameters: TAG and bit position)
ERR_MAX_IE_EXCEED	maximum amount of repeatable information elements has exceeded (error parameters: TAG and bit position)
ERR_MAX_REPEAT	a repeatable element occurs too often in the message
ERR_MAND_ELEM_MISS	a mandatory information element is missing
ERR_INVALID_TYPE	the information element is not a spare padding (error parameter: bit position)
ERR_EOC_TAG_MISSING	an indefinite length is specified for the ASN.1-BER encoded element but the corresponding end of Component TAG is missing (error parameters: TAG and bit position)
ERR_INTERNAL_ERROR	an internal CCD error occurred
ERR_NO_MORE_ERROR	end of the error list
ERR_NO_MEM	memory allocation failed for dynamic array addition.
ERR_BUFFER_OF	on coding, more bits were written as given by function parameter l_buf
ERR_DEFECT_CCDDATA	unexpected character in a CCD data table; wrong version of CCDDATA loaded
ERR_NONCRITICAL_EXT	on decoding, more bits were to be decoded than CCD expected

Description:

If there is still at least one error stored in the errorlist, this function copies the additional stored error parameter into the given parlist and returns the error code. If no error occurred this function returns 0.

2.3.15 ccd_getFirstFault:- copies information on the first error into the function parameter

Definition

ULONG ccd_getFirstFault(T_CCD_ERR_ENTRY **ccd_err_entry)

Name	Description
ccd_err_entry	points to the C-structure which contains information on the errors detected in the last coding/decoding action

Return values:

Name	Description
ccdOK	no error occurred
ERR_NO_MORE_ERROR	the end of the error list is reached
ERR_INVALID_CALC	calculation of the element repeat value failed
ERR_PATTERN_MISMATCH	a bit pattern was not expected (error parameter: bit position)
ERR_COMPREH_REQUIRED	check for comprehension required failed (error parameters: TAG and bit position)
ERR_IE_NOT_EXPECTED	an information element was not expected (error parameters: TAG and bit position)
ERR_IE_SEQUENCE	wrong sequence of information elements (error parameters: TAG and bit position)
ERR_MAX_IE_EXCEED	maximum amount of repeatable information elements has exceeded (error parameters: TAG and bit position)
ERR_MAX_REPEAT	a repeatable element occurs too often in the message
ERR_MAND_ELEM_MISS	a mandatory information element is missing
ERR_INVALID_MID	unknown message ID (error parameter: Message ID)
ERR_INVALID_TYPE	the information element is not a spare padding (error parameter: bit position)
ERR_EOC_TAG_MISSING	an indefinite length is specified for the ASN.1-BER encoded element but the corresponding end of Component TAG is missing (error parameters: TAG and bit position)
ERR_INTERNAL_ERROR	an internal CCD error occurred
ERR_NO_MORE_ERROR	end of the error list
ERR_NO_MEM	memory allocation failed for dynamic array addition.
ERR_BUFFER_OF	on coding, more bits were written as given by function parameter l_buf
ERR_DEFECT_CCDDATA	unexpected character in a CCD data table; wrong version of CCDDATA loaded
ERR_NONCRITICAL_EXT	on decoding, more bits were to be decoded than CCD expected
ERR_INT_VALUE	on coding, integer value was out of its range
ERR_ASN1_ENCODING	ASN.1 (PER) CHOICE index is out of its range
ERR_ASN1_MAND_IE	on decoding, mandatory integer has a value out of its range
ERR_ASN1_OPT_IE	on decoding, optional integer has a value out of its range
ERR_CRITICAL_EXT	met an IE of type CriticalExtensions; CCD can not handle the unknown extension

Description:

If at least one error is stored in the error list, this function copies information on the error into the function parameter and returns the error code. If no error is in the list this function returns 0.

The error information is made of error number, error-union type, CCD id for the faulty element and the address of this element in the C-structure of the decoded message.

2.3.16 ccd_getNextFault: copies information on the next error into the function parameter

Definition

ULONG ccd_getNextFault (T_CCD_ERR_ENTRY **ccd_err_entry)

Name	Description
ccd_err_entry	points to the C-structure which contains information on the errors detected in the last coding/decoding action

Return values:

Name	Description
ccdOK	no error occurred
ERR_NO_MORE_ERROR	the end of the error list is reached
ERR_INVALID_CALC	calculation of the element repeat value failed
ERR_PATTERN_MISMATCH	a bit pattern was not expected (error parameter: bit position)
ERR_COMPREH_REQUIRED	check for comprehension required failed (error parameters: TAG and bit position)
ERR_IE_NOT_EXPECTED	an information element was not expected (error parameters: TAG and bit position)
ERR_IE_SEQUENCE	wrong sequence of information elements (error parameters: TAG and bit position)
ERR_MAX_IE_EXCEED	maximum amount of repeatable information elements has exceeded (error parameters: TAG and bit position)
ERR_MAX_REPEAT	a repeatable element occurs too often in the message
ERR_MAND_ELEM_MISS	a mandatory information element is missing
ERR_INVALID_TYPE	the information element is not a spare padding (error parameter: bit position)
ERR_EOC_TAG_MISSING	an indefinite length is specified for the ASN.1-BER encoded element but the corresponding end of Component TAG is missing (error parameters: TAG and bit position)
ERR_INTERNAL_ERROR	an internal CCD error occurred
ERR_NO_MORE_ERROR	end of the error list
ERR_NO_MEM	memory allocation failed for dynamic array addition.
ERR_BUFFER_OF	on coding, more bits were written as given by function parameter l_buf
ERR_DEFECT_CCDDATA	unexpected character in a CCD data table; wrong version of CCDDATA loaded
ERR_NONCRITICAL_EXT	on decoding, more bits were to be decoded than CCD expected
ERR_INT_VALUE	on coding, integer value was out of its range
ERR_ASN1_ENCODING	ASN.1 (PER) CHOICE index is out of its range
ERR_ASN1_MAND_IE	on decoding, mandatory integer has a value out of its range
ERR_ASN1_OPT_IE	on decoding, optional integer has a value out of its range
ERR_CRITICAL_EXT	met an IE of type CriticalExtensions; CCD can not handle the unknown extension

Description:

If at least one error is stored in the error list, this function copies information on the error into the function parameter and returns the error code. If no error is in the list this function returns 0.

The error information is made of error number, error-union type, CCD id for the faulty element and the address of this element in the C-structure of the decoded message.

2.3.17 ccd_free_faultlist: frees any allocated error/fault information for CCD caller

Definition

void ccd_free_faultlist ()

Name	Description

Return values:

Name	Description

Description:

This function frees any previously allocated memory by CCD for error/fault information of the caller entity. In the run-time CCD frees memory sections which have been allocated in the last encode/decode action. However it is recommended to use this function especially after detection of large amounts of error or when a frequent usage of CCD is not foreseen for an entity.

2.3.18 ccd_get_numFaults: returns the count of errors/faults

Definition

int ccd_get_numFaults ()

Name	Description

Return values:

Name	Description

Description:

This function returns the count of errors/faults detected in the last coding/decoding action for the caller task.

2.4 Parameters

Name	Description
Bitlen	number of bits to decode, copy
bitstream	points to the buffer containing the bit stream.
buf	specifies the bit stream buffer of the encoded message.
dest	points to the destination of a copy operation.
direction	specifies whether the message goes UPLINK or DOWNLINK. This is necessary because the same PDU-Type can be used for both directions while the message structures can be different.
entity	ID of the calling entity. Valid entity identifiers are defined as C-macros in MCONST.CDG.
l_buf	length of the encoded message in the bit stream buffer
mBuf	specifies the bit stream buffer of the message. The elements of this structure are l_buf, o_buf and buf. The two first elements specify the length and offset of the bitstream buf. The member o_buf must be specified by the caller while the member o_buf may be calculated by CCD (in older CCD versions).
mId	specifies the PDU-Type of the bit stream. CCD reads this value as the message ID if it is not equal to 0xff or 0xfe. Normally this parameter is set to 0xff and CCD reads the pdu-type from the appropriate component of *mStruct.
mStruct	reference to the C-structure containing the C-representation of the decoded message. The type should be casted to UBYTE*.
mStructPtr	points to the pointer on the C-structure of the decoded message. The buffer containing this message structure will be allocated by CCD. After decoding the first element in the message C-structure contains the message (PDU) type as a UBYTE.
o_buf	offset in bits of the bit stream buffer
offset	position of the first bit to copy in the source bit field
parlist	points to a buffer where additional error parameters will be stored to.
ccd_err_entry	points to the C-structure which contains information on the errors detected in the last coding/decoding action
source	points to the source of a copy operation.
startbit	Bit offset into the first byte of the bit stream
value	pointer to a value. The result of a code or decode operation is stored here.

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/>