GSM Fax & Data Services

Test Specification
UART

Author: Texas Instruments Berlin AG
Alt-Moabit 90a
D-10559 Berlin
Germany

Date: 08-Mar-2015
Document No.: 8441.412.00.008
File: uart.doc

Table of Contents

0 Document Control

Texas Instruments AG
Alt Moabit 90a
10559 Berlin
Germany

Telephone: +49.30.3983-0
Fax: +49.30.3983-1300
Internet: http://www.ti.com
E-mail: gsm@ti.de

0.1 Document History

Document Id.	Date	Author	Remarks
8441.412.00.001	20-Sep-2000	STW	Initial
8441.412.00.002	27-Jul-2001	IH	add testcases for escape sequence detection
8441.412.00.003	11-Sep-2001	STW	extended internal buffer sizes
8441.412.00.004	08-Oct-2001	TVO	adapted for use with DTI2 SAP/DTILIB
8441.412.00.005	17-Jan-2002	STW	mark testcase 110-118 as "not yet implemented"
8441.412.00.006	12-Mar-2002	TVO	Rename dti and parameter primitives, add UART_PARAMETERS_IND
8441.412.00.007	16-Jan-2003	STW	Error corrections
8441.412.00.008	24-Jan-2003	STW	add TCs for HDLC Escape characters

0.2 References

[1] GSM 05.02 version 8.0.0 Release 1999
Digital cellular telecommunications system (Phase 2+);
Multiplexing and multiple access on the radio path

[2] GSM 04.60 version 6.3.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
General Packet Radio Service (GPRS);
Mobile Station (MS) - Base Station System (BSS) interface;
Radio Link Control/ Medium Access Control (RLC/MAC) protocol

[3] GSM 04.08 version 6.3.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
Mobile radio interface layer 3 specification

[4] GSM 03.64 version 6.1.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
General Packet Radio Service (GPRS);
Overall description of the GPRS radio interface; Stage 2

[5] GSM 03.60 version 6.3.1 Release 1997
Digital cellular telecommunications system (Phase 2+);
General Packet Radio Service (GPRS);
Service description; Stage 2

[6] GSM 04.07 version 6.3.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
Mobile radio interface signalling layer 3; General aspects

[7] GSM 04.64 version 6.3.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
General Packet Radio Service (GPRS);
Mobile Station - Serving GPRS Support Node (MS-SGSN)
Logical Link Control (LLC) layer specification

[8] GSM 05.08 version 6.4.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
Radio subsystem link control

[9] GSM 05.10 version 6.3.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
Radio subsystem synchronization

[10] GSM 03.20 TS 100 929: July 1998 (GSM 03.20 version 6.0.1)
Security related network functions, ETSI

[11] Draft GSM 03.22: August 1998 (GSM 03.22 version 6.1.0)
Functions related to Mobile Station (MS) in idle mode and group receive mode, ETSI

[12] GSM 04.65 V6.3.0: Subnetwork Dependant Convergence Protocol
ETSI, March 1999

[13] ITU-T V42bis ITU-T, Recommendation V.42 bis 1990

[14] GSM 09.60 GPRS Tunneling Protocol (GTP) across the Gn and Gp Interface

[15] RFC 1661 IETF STD 51 July 1994
The Point-to-Point Protocol (PPP)

[16] RFC 1662 IETF STD 51 July 1994
PPP in HDLC-like Framing

[17] RFC 1570 January 1994
PPP LCP Extensions

[18] RFC 1989 August 1996
PPP Link Quality Monitoring

[19] RFC 1332 May 1992
The PPP Internet Protocol Control Protocol (IPCP)

[20] RFC 1877 December 1995
PPP IPCP Extensions for Name Server Addresses

[21] RFC 2153 May 1997
PPP Vendor Extensions

[22] RFC 1334 October 1992
PPP Authentication Protocols (for Password Authentication Protocol only)

[23] RFC 1994 August 1996
PPP Challenge Handshake Authentication Protocol (CHAP)

[24] TIA/EIA-136-370
Packet-Data Services – Enhanced General Packet Radio for TIA/EIA-136 (EGPRS-136) - Overview, Telecommunications Industry Association

[25] TIA/EIA-136-376
Packet-Data Services – EGPRS-136 Mobility Management, Telecommunications Industry Association

[26] TIA/EIA-136-972
Packet-Data Services – Stage 2 Description, Telecommunications Industry Association

0.3 Abbreviations

ACI	Application Control Interface
AGCH	Access Grant Channel
AT	Attention sequence "AT" to indicate valid commands of the ACI
BCCH	Broadcast Control Channel
BS	Base Station
BSIC	Base Station Identification Code
C/R	Command/Response
C1	Path Loss Criterion
C2	Reselection Criterion
CBCH	Cell Broadcast Channel
CBQ	Cell Bar Qualify
CC	Call Control
CCCH	Common Control Channel
CCD	Condat Coder Decoder
CCI	Compression and Ciphering Interface
CHAP	Challenge Handshake Authentication Protocol
CKSN	Ciphering Key Sequence Number
CRC	Cyclic Redundancy Check
DCCH	Dedicated Control Channel
DCOMP	Identifier of the user data compression algorithm used for the N-DPU
DISC	Disconnect Frame
DL	Data Link Layer
DM	Disconnected Mode Frame
DTX	Discontinuous Transmission
E	Extension bit
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
GACI	GPRS Application Control Interface
GMM	GPRS Mobility Management
GP	Guard Period
GRR	GPRS RR
GSM	Global System for Mobile Communication
HDLC	High-level Data Link Control
HISR	High level Interrupt Service Routine
HPLMN	Home Public Land Mobile Network
I	Information Frame
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
IPCP	Internet Protocol Control Protocol
ITU	International Telecommunication Union
IWF	Interworking Function
Kc	Ciphering Key

L	Length Indicator
LAI	Location Area Information
LCP	Link Control Protocol
LISR	Low level Interrupt Service Routine
LLC	Logical Link Control
LPD	Link Protocol Discriminator
LQM	Link Quality Monitoring
M	More bit used to indicate the last segment of N-DPU
MAC	Medium Access Control
MCC	Mobile Country Code
MM	Mobility Management
MMI	Man Machine Interface
MNC	Mobile Network Code
MS	Mobile Station
MT	Mobile Termination
N(R)	Receive Number
N(S)	Send Number
NC	Network Control
NCC	National Colour Code
NCP	Network Control Protocol
NECI	New Establishment Causes included
N-PDU	Network Protocol Data Unit
NSAPI	Network Layer Service Access Point Identifier
OTD	Observed Time Difference
P	Poll Bit
P/F	Poll/Final Bit
PACCH	Packet Associated Control Channel
PAP	Password Authentication Protocol
PBCCH	Packet BCCH
PCCCH	Packet CCCH
PCOMP	Identifier of the protocol control information compression algorithm used for the N-DPU
PDCH	Packet Data Channel
PDP	Packet Data Protocol e.g. IP or X.25
PDTCH	Packet Data Traffic Channel
PRACH	Packet RACH
PSI	Packet System Information
PCH	Paging Channel
PCO	Point of Control and Observation
PDU	Protocol Data Unit
PL	Physical Layer
PLMN	Public Land Mobile Network
PPC	Packet Physical Convergence
PPP	Point-to-Point Protocol
PTP	Point to Point
QoS	Quality of Service
RACH	Random Access Channel
REJ	Reject Frame
RLC	Radio Link Control
RNR	Receive Not Ready Frame
RR	Radio Resource Management
RR	Receive Ready Frame
RTD	Real Time Difference
RTOS	Real Time Operating System

SABM	Set Asynchronous Balanced Mode
SACCH	Slow Associated Control Channel
SAP	Service Access Point
SAPI	Service Access Point Identifier
SDCCH	Stand alone Dedicated Control Channel
SDU	Service Data Unit
SGSN	Serving GPRS Support Node
SIM	Subscriber Identity Module
SM	Session Management
SMS	Short Message Service
SMSCB	Short Message Service Cell Broadcast
SNDCP	Subnetwork Dependant Convergence Protocol
SNSM	SNDCP-SM
SS	Supplementary Services
TAP	Test Application Program
TBF	Temporary Block Flow
TCH	Traffic Channel
TCH/F	Traffic Channel Full Rate
TCH/H	Traffic Channel Half Rate
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TE	Terminal Equipment - e. g. a PC
TFI	Temporary Flow Identifier
TLLI	Temporary Logical Link Identifier
TMSI	Temporary Mobile Subscriber Identity
TOM	Tunnelling of Messages
TQI	Temporary Queuing Identifier
UA	Unnumbered Acknowledgement Frame
UART	Universal Asynchronous Receiver Transmitter
UI	Unnumbered Information Frame
USF	Uplink State Flag
V(A)	Acknowledgement State Variable
V(R)	Receive State Variable
V(S)	Send State Variable
VPLMN	Visited Public Land Mobile Network

0.4 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).

1 Parameters

/*

1.1 Primitive elements

*/

```
BYTE UART_DEVICE                0          /* number of UART device */
BYTE UART_INSTANCES             1          /* total number of UART instances */
BYTE DLCI_A                     42         /* example data link connection identifier */
BYTE DLCI_B                     1          /* second data connection identifier */
BYTE DLCI_C                     3          /* third data connection identifier */
BYTE DLCI_D                     0x04       /* special HDLC escape forcing DLCI */
BYTE DLCI_E                     0x1f       /* special HDLC escape forcing DLCI */
BYTE DLCI_ANY                   0          /* a dummy dlci which is used when dlci is unnecessary */
BYTE DLCI_CONTROL               0          /* data link connection identifier for control channel */
LONG UART_PEER_NAME_DUMMY       0          /* dummy for pointer to peer name string */
BYTE P_ID_PEER_A                DTI_PID_UOS /* Procotol ID of protocol towards peer A */
BYTE P_ID_PEER_B                DTI_PID_UOS /* Protocol ID of protocol towards peer B */
BYTE UART_BREAK_LEN             0x02       /* length of break signal */
BYTE LINK_ID_1                  0x22       /* an example channel id */
BYTE LINK_ID_2                  0x45       /* another example channel id */
BYTE LINK_ID_3                  0x33       /* another example channel id */
BYTE LINK_ID_4                  0x44       /* another example channel id */
BYTE LINK_ID_5                  0x55       /* another example channel id */

/* these are __in no way__ real link_ids. Instead, the variable is used for communication between */
/* the entity and the test environment, here. In previous versions, the tui was used for this, but since */
/* DTI2 primitives do not contain such a parameter any more, the dirty work has been passed on */
/* to the link_id ….. */

SHORT LINK_READDATA_PORT_1      0          /* misused link_id for read_data call */
SHORT LINK_DISABLE_PORT_1       1          /* misused link_id for disable call */
SHORT LINK_ENABLE_PORT_1        2          /* misused link_id for enable call */
SHORT LINK_WRITEDATA_PORT_1     3          /* misused link_id for write_data call */

SHORT LINK_READDATA_PORT_2      10         /* misused link_id for read_data call */
SHORT LINK_DISABLE_PORT_2       11         /* misused link_id for disable call */
SHORT LINK_ENABLE_PORT_2        12         /* misused link_id for enable call */
SHORT LINK_WRITEDATA_PORT_2     13         /* misused link_id for write_data call */

SHORT LINK_UART_OUT_PORT_1      0          /* misused link_id of UART_OUT test interface */
SHORT LINK_UART_OUT_PORT_2      10         /* misused link_id of UART_OUT test interface */

SHORT LINK_PORT_THRESHOLD       9          /* to be able to distinguish between port 1 and port2 */

BYTE DTI_PID_UART_OUT           0          /* protocol identifier of UART_OUT test interface */

BYTE    ESC_SEQ_CHAR_PLUS       0x2B       /* Escape Sequence Char '+' */
SHORT   GUARD_PERIOD_1000       1000       /* Escape Sequence Guard Period */

/*
Set Default Timer Values for Simulation Testing - Original Values too Fast for TAP
*/
```

```
SHORT UART_MUX_T1_DEFAULT_TEST      30
SHORT UART_MUX_T2_DEFAULT_TEST      90
SHORT UART_MUX_T3_DEFAULT_TEST      10

/*
```

1.2 Struct elements

```
*/
BYTE COMPAR_SPEED_57600         UART_IO_SPEED_57600             /* baudrate of 57600 */
BYTE COMPAR_SPEED_19200         UART_IO_SPEED_19200             /* baudrate of 19200 */
BYTE COMPAR_SPEED_9600          UART_IO_SPEED_9600              /* baudrate of 9600 */
BYTE COMPAR_BPC_8               UART_IO_BPC_8                   /* 8 bit per character */
BYTE COMPAR_NSB_1               UART_IO_SB_1                    /* 1 stop bit */
BYTE COMPAR_PARITY_EVEN         UART_IO_PA_EVEN                 /* even parity */
BYTE COMPAR_PARITY_NO               UART_IO_PA_NONE             /* no parity no space */
BYTE COMPAR_XON_CHAR                UART_IO_XON_DEFAULT         /* default value for XON character */
BYTE COMPAR_XOFF_CHAR               UART_IO_XOFF_DEFAULT        /* default value for XOFF character */

BYTE FCS_DUMMY_OK               0xcf                            /* a valid FCS: UART_GOOD_FCS */
BYTE FCS_DUMMY_INVALID          0xdf                            /* an invalid FCS */
BYTE UART_UNKNOWN_COMMAND       0xd7                            /* an unsupported command type*/

SHORT OFFSET_0                      0x00                                /* SDU offset of 0 */
SHORT LEN_HELLO                 0x40                            /* length of hello strings */

/*
```

1.3 Fields

1.3.1 Strings

```
*/
/* String Hello1
```

0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a	String "Hello1\n"

```
*/
FIELD(STRING_HELLO1)
                        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a
ENDFIELD(STRING_HELLO1, 8)

/* String Hello2
```

0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x32, 0x0d, 0x0a	String "Hello2\n"

```
*/
FIELD(STRING_HELLO2)
                        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x32, 0x0d, 0x0a
ENDFIELD(STRING_HELLO2, 8)

/* String Hello3
```

0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x33, 0x0d, 0x0a	String "Hello3\n"

```
*/
FIELD(STRING_HELLO3)
```

```
                0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x33, 0x0d, 0x0a
ENDFIELD(STRING_HELLO3, 8)

/* String h
```

0x68	String "h"

```
*/
FIELD(STRING_H)
                0x68
ENDFIELD(STRING_H, 1)

/* String e
```

0x65	String "e"

```
*/
FIELD(STRING_E)
                0x65
ENDFIELD(STRING_E, 1)

/* String l
```

0x6c	String "l"

```
*/
FIELD(STRING_L)
                0x6c
ENDFIELD(STRING_L, 1)

/* String o
```

0x6f	String "o"

```
*/
FIELD(STRING_O)
        0x6f
ENDFIELD(STRING_O, 1)

/* Field String Hello1 SDU
```

0x40, 0x00	SDU length in bit (8 Byte = 64 Bit)
0x00, 0x00	SDU offset in bit
0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a	String "Hello1\n\r"

```
*/
FIELD(STRING_HELLO1_SDU)
        0x40, 0x00,
        0x00, 0x00,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a
ENDFIELD(STRING_HELLO1_SDU, 12)

/* Field String 15x Hello1 SDU
```

0xc0, 0x03	SDU length in bit (120 Byte = 960 Bit)
0x00, 0x00	SDU offset in bit
0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c,	15 times String "Hello1\n\r"

0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a	

```
*/
FIELD(STRING_15XHELLO1_SDU)
	0xc0, 0x03,
	0x00, 0x00,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
	0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a
ENDFIELD(STRING_15XHELLO1_SDU, 124)

/* Field String He SDU
```

0x10, 0x00	SDU length in bit (2 Byte = 16 Bit)
0x00, 0x00	SDU offset in bit
0x48, 0x65	String "He"

```
*/
FIELD(STRING_HE_SDU)
	0x10, 0x00,
	0x00, 0x00,
	0x48, 0x65
ENDFIELD(STRING_HE_SDU, 6)

/* Field String 30 octets of Hello1\n\r SDU
```

0xf0, 0x00	SDU length in bit (30 Byte = 240 Bit)
0x00, 0x00	SDU offset in bit
0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31	first 30 octets of 15 times String "Hello1\n\r"

```
*/
FIELD(STRING_1ST30HELLO1_SDU)
	0xf0, 0x00,
	0x00, 0x00,
```

```
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31
ENDFIELD(STRING_1ST30HELLO1_SDU, 34)
```

/* Field String ll SDU

0x10, 0x00	SDU length in bit (2 Byte = 16 Bit)
0x00, 0x00	SDU offset in bit
0x6c, 0x6c	String "ll"

*/

```
FIELD(STRING_LL_SDU)
        0x10, 0x00,
        0x00, 0x00,
        0x6c, 0x6c
ENDFIELD(STRING_LL_SDU, 6)
```

/* Field String second 30 octets of Hello1\n\r SDU

0xf0, 0x00	SDU length in bit (30 Byte = 240 Bit)
0x00, 0x00	SDU offset in bit
0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c	second 30 octets of 15 times String "Hello1\n\r"

*/

```
FIELD(STRING_2ND30HELLO1_SDU)
        0xf0, 0x00,
        0x00, 0x00,
        0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c
ENDFIELD(STRING_2ND30HELLO1_SDU, 34)
```

/* Field String o1 slash SDU

0x18, 0x00	SDU length in bit (3 Byte = 24 Bit)
0x00, 0x00	SDU offset in bit
0x6f, 0x31, 0x0d	String "o1\"

*/

```
FIELD(STRING_O1_SLASH_SDU)
        0x18, 0x00,
        0x00, 0x00,
        0x6f, 0x31, 0x0d
ENDFIELD(STRING_O1_SLASH_SDU, 7)
```

/* Field String third 40 octets of Hello1\n\r SDU

0x40, 0x01	SDU length in bit (40 Byte = 320 Bit)
0x00, 0x00	SDU offset in bit
0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,	third 40 octets of 15 times String "Hello1\n\r"

0x48, 0x65, 0x6c, 0x6c	

*/

```
FIELD(STRING_3RD40HELLO1_SDU)
        0x40, 0x01,
        0x00, 0x00,
        0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c
ENDFIELD(STRING_3RD40HELLO1_SDU, 44)
```

/* Field String n SDU

0x08, 0x00	SDU length in bit (1 Byte = 8 Bit)
0x00, 0x00	SDU offset in bit
0x0a	String "n"

*/

```
FIELD(STRING_N_SDU)
                0x08, 0x00,
                0x00, 0x00,
                0x0a
ENDFIELD(STRING_N_SDU, 5)
```

/* Field String reset of Hallo1\n\r SDU

0xa0, 0x00	SDU length in bit (20 Byte = 160 Bit)
0x00, 0x00	SDU offset in bit
0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a	rest octets of 15 times String "Hello1\n\r"

*/

```
FIELD(STRING_RESTHALLO1_SDU)
        0xa0, 0x00,
        0x00, 0x00,
        0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a
ENDFIELD(STRING_RESTHALLO1_SDU, 24)
```

/* Field String hHe SDU

0x18, 0x00	SDU length in bit (3 Byte = 24 Bit)
0x00, 0x00	SDU offset in bit
0x68, 0x48,0x65	String "hHe"

*/

```
FIELD(STRING_HHE_SDU)
        0x18, 0x00,
        0x00, 0x00,
        0x68, 0x48, 0x65
ENDFIELD(STRING_HHE_SDU, 7)
```

/* Field String 10x h and 20 octets of Hello1 SDU

0xf0, 0x00	SDU length in bit (30 Byte = 240 Bit)

0x00, 0x00	SDU offset in bit
0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c	10 times String "h" and 20 octets of "Hello1\n\r" String

*/

```
FIELD(STRING_1ST30HHELLO1_SDU)
        0xf0, 0x00,
        0x00, 0x00,
        0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c
ENDFIELD(STRING_1ST30HHELLO1_SDU, 34)
```

/* Field String llo1 SDU

0x20, 0x00	SDU length in bit (4 Byte = 32 Bit)
0x00, 0x00	SDU offset in bit
0x6c, 0x6c, 0x6f, 0x31	String "llo1"

*/

```
FIELD(STRING_LLO1_SDU)
        0x20, 0x00,
        0x00, 0x00,
        0x6c, 0x6c, 0x6f, 0x31
ENDFIELD(STRING_LLO1_SDU, 8)
```

/* Field String second 40 octets of Hallo1\n\r SDU

0x40, 0x01	SDU length in bit (40 Byte = 320 Bit)
0x00, 0x00	SDU offset in bit
0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c	second part: 40 octets of "Hallo1\n\r" String

*/

```
FIELD(STRING_2ND40HHALLO1_SDU)
        0x40, 0x01,
        0x00, 0x00,
        0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c
ENDFIELD(STRING_2ND40HHALLO1_SDU, 44)
```

/* Field String \n SDU

0x10, 0x00	SDU length in bit (2 Byte = 16 Bit)
0x00, 0x00	SDU offset in bit
0x0d, 0x0a	String "\n"

*/

```
FIELD(STRING_SLASH_N_SDU)
```

```
        0x10, 0x00,
        0x00, 0x00,
        0x0d, 0x0a
ENDFIELD(STRING_SLASH_N_SDU, 6)
```

/* Field String third 50 octets of Hallo1\n\r SDU

0x90, 0x01	SDU length in bit (50 Byte = 400 Bit)
0x00, 0x00	SDU offset in bit
0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31	third part: 50 octets of "Hallo1\n\r" String

*/

```
FIELD(STRING_3RD50HHALLO1_SDU)
        0x90, 0x01,
        0x00, 0x00,
        0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31
ENDFIELD(STRING_3RD50HHALLO1_SDU, 54)
```

/* Field String reset of Hallo1\n\r SDU

0x50, 0x00	SDU length in bit (10 Byte = 80 Bit)
0x00, 0x00	SDU offset in bit
0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a	rest of "Hallo1\n\r" String

*/

```
FIELD(STRING_RESTHHALLO1_SDU)
        0x50, 0x00,
        0x00, 0x00,
        0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a
ENDFIELD(STRING_RESTHHALLO1_SDU, 14)
```

/* Field String Hello2 SDU

0x40, 0x00	SDU length in bit (8 Byte = 64 Bit)
0x00, 0x00	SDU offset in bit
0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x32, 0x0d, 0x0a	String "Hello2\n"

*/

```
FIELD(STRING_HELLO2_SDU)
        0x40, 0x00,
        0x00, 0x00,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x32, 0x0d, 0x0a
ENDFIELD(STRING_HELLO2_SDU, 12)
```

/* Field String 0x10 0x11 0x12 SDU

0x18, 0x00	SDU length in bit (3 Byte = 24 Bit)

0x00, 0x00	SDU offset in bit
0x10, 0x11, 0x12	String "0x10 0x11 0x12"

*/

```
FIELD(STRING_101112_SDU)
        0x18, 0x00,
        0x00, 0x00,
        0x10, 0x11, 0x12
ENDFIELD(STRING_101112_SDU, 7)
```

/* Field String 0x13 0x14 0x15 SDU

0x18, 0x00	SDU length in bit (3 Byte = 24 Bit)
0x00, 0x00	SDU offset in bit
0x13, 0x14, 0x15	String "0x13 0x14 0x15"

*/

```
FIELD(STRING_131415_SDU)
        0x18, 0x00,
        0x00, 0x00,
        0x13, 0x14, 0x15
ENDFIELD(STRING_131415_SDU, 7)
```

/* Field String 0x7c 0x7d 0x7e SDU

0x18, 0x00	SDU length in bit (3 Byte = 24 Bit)
0x00, 0x00	SDU offset in bit
0x7c, 0x7d, 0x7e	String "0x7c 0x7d 0x7e"

*/

```
FIELD(STRING_7c7d7e_SDU)
        0x18, 0x00,
        0x00, 0x00,
        0x7c, 0x7d, 0x7e
ENDFIELD(STRING_7c7d7e_SDU, 7)
```

/* Field String 0x7e 0x7f 0x80 SDU

0x18, 0x00	SDU length in bit (3 Byte = 24 Bit)
0x00, 0x00	SDU offset in bit
0x7e, 0x7f, 0x80	String "0x7e 0x7f 0x80"

*/

```
FIELD(STRING_7e7f80_SDU)
        0x18, 0x00,
        0x00, 0x00,
        0x7e, 0x7f, 0x80
ENDFIELD(STRING_7e7f80_SDU, 7)
```

/* Field String Hello3 SDU

0x40, 0x00	SDU length in bit (8 Byte = 64 Bit)
0x00, 0x00	SDU offset in bit
0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x33, 0x0d, 0x0a	String "Hello3\n"

*/

```
FIELD(STRING_HELLO3_SDU)
        0x40, 0x00,
        0x00, 0x00,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x33, 0x0d, 0x0a
```

```
ENDFIELD(STRING_HELLO3_SDU, 12)

/* Field String h SDU
```

0x08, 0x00	SDU length in bit (1 Byte = 8 Bit)
0x00, 0x00	SDU offset in bit
0x68	String "h"

```
*/
FIELD(STRING_H_SDU)
        0x08, 0x00,
        0x00, 0x00,
        0x68
ENDFIELD(STRING_H_SDU, 5)

/* Field String h++h SDU
```

0x20, 0x00	SDU length in bit (4 Byte = 32 Bit)
0x00, 0x00	SDU offset in bit
0x68, 0x2b, 0x2b, 0x68	String "h++h"

```
*/
FIELD(STRING_HPLUSPLUSH_SDU)
        0x20, 0x00,
        0x00, 0x00,
        0x68, 0x2b, 0x2b, 0x68
ENDFIELD(STRING_HPLUSPLUSH_SDU, 8)

/* Field String 10 times h SDU
```

0x50, 0x00	SDU length in bit (10 Byte = 80 Bit)
0x00, 0x00	SDU offset in bit
0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68	String "hhhhhhhhhh"

```
*/
FIELD(STRING_10H_SDU)
        0x50, 0x00,
        0x00, 0x00,
        0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68, 0x68
ENDFIELD(STRING_10H_SDU, 14)

/* Field String A SDU
```

0x08, 0x00	SDU length in bit (1 Byte = 8 Bit)
0x00, 0x00	SDU offset in bit
0x41	String "A"

```
*/
FIELD(STRING_A_SDU)
        0x08, 0x00,
        0x00, 0x00,
        0x41
ENDFIELD(STRING_A_SDU, 5)

/* Field String 3 time "+" and "A" SDU
```

0x20, 0x00	SDU length in bit (4 Byte = 32 Bit)
0x00, 0x00	SDU offset in bit
0x2b, 0x2b, 0x2b, 0x41	String "+++A"

```
*/
FIELD(STRING_3PLUSA_SDU)
        0x20, 0x00,
        0x00, 0x00,
        0x2b, 0x2b, 0x2b, 0x41
ENDFIELD(STRING_3PLUSA_SDU, 8)

/* Field String AT SDU
```

0x20, 0x00	SDU length in bit (1 Byte = 8 Bit)
0x00, 0x00	SDU offset in bit
0x41, 0x54, 0x0D, 0x0A	String "AT\n"

```
*/
FIELD(STRING_AT_SDU)
        0x20, 0x00,
        0x00, 0x00,
        0x41, 0x54, 0x0D, 0x0A,
ENDFIELD(STRING_AT_SDU, 8)

/* Field String e SDU
```

0x08, 0x00	SDU length in bit (1 Byte = 8 Bit)
0x00, 0x00	SDU offset in bit
0x65	String "e"

```
*/
FIELD(STRING_E_SDU)
                0x08, 0x00,
                0x00, 0x00,
                0x65
ENDFIELD(STRING_E_SDU, 5)

/* Field String l SDU
```

0x08, 0x00	SDU length in bit (1 Byte = 8 Bit)
0x00, 0x00	SDU offset in bit
0x6c	String "l"

```
*/
FIELD(STRING_L_SDU)
                0x08, 0x00,
                0x00, 0x00,
                0x6c
ENDFIELD(STRING_L_SDU, 5)

/* Field String o SDU
```

0x08, 0x00	SDU length in bit (1 Byte = 8 Bit)
0x00, 0x00	SDU offset in bit
0x6f	String "o"

```
*/
FIELD(STRING_O_SDU)
                0x08, 0x00,
                0x00, 0x00,
                0x6f
ENDFIELD(STRING_O_SDU, 5)

/* Empty String
```

0x00, 0x00	SDU length in bit (0 Byte = 0 Bit)
0x00, 0x00	SDU offset in bit

*/

```
FIELD(EMPTY_STRING_SDU)
        0x00, 0x00,
        0x00, 0x00
ENDFIELD(EMPTY_STRING_SDU, 4)
```

/* Field String Hello1Hello2 SDU

0x80, 0x00	SDU length in bit (16 Byte = 128 Bit)
0x00, 0x00	SDU offset in bit
0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a, 0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x32, 0x0d, 0x0a	String "Hello1\nHello2\n"

*/

```
FIELD(STRING_HELLO1_HELLO2_SDU)
        0x80, 0x00,
        0x00, 0x00,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x32, 0x0d, 0x0a
ENDFIELD(STRING_HELLO1_HELLO2_SDU, 20)
```

/* Field String 8x"ABCDEFGH" and 1x"h+cops"

0x30, 0x02	SDU length in bit (70 Byte = 560 Bit)
0x00, 0x00	SDU offset in bit
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x68, 0x2b, 0x63, 0x6f, 0x70, 0x73	8x String "ABCDEFGH" and 1x String "h+cops"

*/

```
FIELD(STRING_PLUSCOPS_SDU)
        0x30, 0x02,
        0x00, 0x00,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x68, 0x2b, 0x63, 0x6f, 0x70, 0x73
ENDFIELD(STRING_PLUSCOPS_SDU, 74)
```

/* Field String + SDU

0x08, 0x00	SDU length in bit (1 Byte = 8 Bit)
0x00, 0x00	SDU offset in bit
0x2B	String "+"

*/

```
FIELD(STRING_PLUS_SDU)
        0x08, 0x00,
        0x00, 0x00,
        0x2B
ENDFIELD(STRING_PLUS_SDU, 5)
```

/* Field String ++ SDU

0x10, 0x00	SDU length in bit (2 Byte = 16 Bit)
0x00, 0x00	SDU offset in bit
0x2B, 0x2B	String "++"

*/

```
FIELD(STRING_PLUSPLUS_SDU)
                0x10, 0x00,
                0x00, 0x00,
                0x2B, 0x2B
ENDFIELD(STRING_PLUSPLUS_SDU, 6)
```

/* Field String +++ SDU

0x18, 0x00	SDU length in bit (3 Byte = 24 Bit)
0x00, 0x00	SDU offset in bit
0x2B, 0x2B, 0x2B	String "+++"

*/

```
FIELD(STRING_ESC_SEQ_SDU)
                0x18, 0x00,
                0x00, 0x00,
                0x2B, 0x2B, 0x2B
ENDFIELD(STRING_ESC_SEQ_SDU, 7)
```

/* Field String of 64 octets SDU

0x00, 0x02	SDU length in bit (64 Byte = 200 Bit)
0x00, 0x00	SDU offset in bit
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48	String "ABCDEFGHABCDEFGHABCDEFGHABC DEFGHABCDEFGHABCDEFGHABCDEF GHABCDEFGH"

*/

```
FIELD(STRING_64_SDU)
        0x00, 0x02,
        0x00, 0x00,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48
ENDFIELD(STRING_64_SDU, 68)
```

/*

1.3.2 Frames

1.3.2.1 Multiplexer Start Up / Close Down

*/

/* SDU: MUX Startup (TE initiated) SABM Frame

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0x3f	Control Field: SABM command
FCS_DUMMY_OK_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_MUX_START0)
                    0x28, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0x03,
                    0x3f,
                    (0XCF),
                    0x7e
ENDFIELD(SDU_MUX_START0, 9)
```

/* SDU: Command to close down multiplexer TE->UE

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xc3	Type: Multiplexer Close Down Cmd
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CMD_TE2UE_CLOSE)
                    0x38, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0x03,
                    0xff,
                    0xc3,
                    0x01,
                    (0XCF),
                    0x7e
ENDFIELD(SDU_CMD_TE2UE_CLOSE, 11)
```

/* SDU: Command to close down multiplexer UE->TE

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110

0x01	Address Field: DLCI 0, C/R 0, EA 1
0xff	Control Field: UIH command
0xc3	Type: Multiplexer Close Down Cmd
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CMD_UE2TE_CLOSE)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0xc3,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_CMD_UE2TE_CLOSE, 11)
```

/* SDU: Confirm Multiplexer Close Down, MMI initiated

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 0, EA 1
0xff	Control Field: UIH command
0xc1	Type: Multiplexer Close Down Res
0x01	Length is 0, the EA bit is set to 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_RES_TE2UE_CLOSE)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0xc1,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_RES_TE2UE_CLOSE, 11)
```

/* SDU: Indicate Multiplexer Close Down, TE initiated, Variant A, Multiplexer Close Cmd

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xc3	Type: Multiplexer Close Down Cmd
0x01	Length
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_CMD_TE2UE_CLOSE0)
                    0x38, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0x03,
                    0xff,
                    0xc3,
                    0x01,
                    (0XCF),
                    0x7e
ENDFIELD(SDU_CMD_TE2UE_CLOSE0, 11)
```

/* SDU: Respond to Multiplexer Close Down, TE initiated, Variant A, Multiplexer Close Cmd

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xc1	Type: Multiplexer Close Down Cmd
0x01	Length
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_RES_UE2TE_CLOSE0)
                    0x38, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0x03,
                    0xff,
                    0xc1,
                    0x01,
                    (0XCF),
                    0x7e
ENDFIELD(SDU_RES_UE2TE_CLOSE0, 11)
```

/* SDU: Indicate Multiplexer Close Down, TE initiated, Variant B, DISC command on DLCI 0

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0x53	Control Field: DISC command
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_CMD_TE2UE_CLOSE1)
                    0x28, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0x03,
                    0x53,
                    (0XCF),
                    0x7e
```

ENDFIELD(SDU_CMD_TE2UE_CLOSE1, 9)

/*

1.3.2.2 Unnumbered Acknowledgements (UA)

*/

/* SDU: UA Response from UE to TE

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0x73	Control Field: UA response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_UA_UE2TE)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0x73,
                (0XCF),
                0x7e
ENDFIELD(SDU_UA_UE2TE, 9)
```

/* SDU: Negative UA Response from UE to TE

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0x63	Control Field: neg. UA response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_NEG_UA_UE2TE)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0x63,
                (0XCF),
                0x7e
ENDFIELD(SDU_NEG_UA_UE2TE, 9)
```

/* SDU: UA Response from TE to UE

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 0, EA 1
0x73	Control Field: UA response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)

0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_UA_TE2UE)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0x73,
                (0XCF),
                0x7e
ENDFIELD(SDU_UA_TE2UE, 9)
```

/* SDU: UA Response from UE to TE for DLCI A

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI_A, C/R 1, EA 1
0x73	Control Field: UA response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_UA_UE2TE_DLCI_A)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x03,
                0x73,
                (0XCF),
                0x7e
ENDFIELD(SDU_UA_UE2TE_DLCI_A, 9)
```

/* SDU: UA Response from TE to UE for DLCI A

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x01	Address Field: DLCI_A, C/R 0, EA 1
0x73	Control Field: UA response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_UA_TE2UE_DLCI_A)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x01,
                0x73,
                (0XCF),
                0x7e
ENDFIELD(SDU_UA_TE2UE_DLCI_A, 9)
```

/* SDU: Negative UA Response from UE to TE for DLCI A

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)

0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI_A, C/R 1, EA 1
0x63	Control Field: Negative UA response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_NEG_UA_UE2TE_DLCI_A)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x03,
                0x63,
                (0XCF),
                0x7e
ENDFIELD(SDU_NEG_UA_UE2TE_DLCI_A, 9)
```

/* SDU: Negative UA Response from TE to UE for DLCI A

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x01	Address Field: DLCI_A, C/R 0, EA 1
0x63	Control Field: Negative UA response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_NEG_UA_TE2UE_DLCI_A)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x01,
                0x63,
                (0XCF),
                0x7e
ENDFIELD(SDU_NEG_UA_TE2UE_DLCI_A, 9)
```

/* SDU: UA Response from UE to TE for DLCI B

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_B << 2) | 0x03	Address Field: DLCI_B, C/R 1, EA 1
0x73	Control Field: UA response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_UA_UE2TE_DLCI_B)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_B << 2) | 0x03,
                0x73,
                (0XCF),
```

```
                0x7e
ENDFIELD(SDU_UA_UE2TE_DLCI_B, 9)
```

/* SDU: UA Response from TE to UE for DLCI B

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_B << 2) | 0x01	Address Field: DLCI_B, C/R 0, EA 1
0x73	Control Field: UA response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_UA_TE2UE_DLCI_B)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_B << 2) | 0x01,
                0x73,
                (0XCF),
                0x7e
ENDFIELD(SDU_UA_TE2UE_DLCI_B, 9)
```

/* SDU: UA Response from UE to TE for DLCI C

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_C << 2) | 0x03	Address Field: DLCI_C, C/R 1, EA 1
0x73	Control Field: UA response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_UA_UE2TE_DLCI_C)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_C << 2) | 0x03,
                0x73,
                (0XCF),
                0x7e
ENDFIELD(SDU_UA_UE2TE_DLCI_C, 9)
```

/* SDU: UA Response from UE to TE for DLCI D

0x30, 0x00	SDU length in bit (6 Bytes = 48 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x7d, ((DLCI_D << 2) | 0x03) ^ 0x20	Address Field: DLCI_D, C/R 1, EA 1
0x73	Control Field: UA response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_UA_UE2TE_DLCI_D)
                0x30, 0x00,
```

```
                0x00, 0x00,
                0x7e,
                0x7d, ((DLCI_D << 2) | 0x03) ^ 0x20,
                0x73,
                (0XCF),
                0x7e
ENDFIELD(SDU_UA_UE2TE_DLCI_D, 10)

/* SDU: UA Response from UE to TE for DLCI E
```

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_E << 2) | 0x03	Address Field: DLCI_E, C/R 1, EA 1
0x73	Control Field: UA response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_UA_UE2TE_DLCI_E)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_E << 2) | 0x03,
                0x73,
                (0XCF),
                0x7e
ENDFIELD(SDU_UA_UE2TE_DLCI_E, 9)

/*
```

1.3.2.3 Flow Control

```
*/
/* SDU: Flow Control On Command, TE -> UE
```

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xa3	Type: Flow On command
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_FLOWONCMD_TE2UE)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0xa3,
                0x01,
                (0XCF),
                0x7e
```

ENDFIELD(SDU_FLOWONCMD_TE2UE, 11)

/* SDU: Flow Control On Command, UE -> TE

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 0, EA 1
0xff	Control Field: UIH command
0xa3	Type: Flow On command
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FLOWONCMD_UE2TE)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0xa3,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_FLOWONCMD_UE2TE, 11)
```

/* SDU: Flow Control On Response, UE -> TE

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xa1	Type: Flow On response
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FLOWONRES_UE2TE)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0xa1,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_FLOWONRES_UE2TE, 11)
```

/* SDU: Flow Control On Response, TE -> UE

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 1, EA 1

0xff	Control Field: UIH command
0xa1	Type: Flow On response
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FLOWONRES_TE2UE)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0xa1,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_FLOWONRES_TE2UE, 11)
```

/* SDU: Flow Control Off Command, TE -> UE

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0x63	Type: Flow Off command
0x01	Length is 0, EA bit is 0
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FLOWOFFCMD_TE2UE)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0x63,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_FLOWOFFCMD_TE2UE, 11)
```

/* SDU: Flow Control Off Command, UE -> TE

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 0, EA 1
0xff	Control Field: UIH command
0x61	Type: Flow Off command
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FLOWOFFCMD_UE2TE)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0x61,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_FLOWOFFCMD_UE2TE, 11)
```

/* SDU: Flow Control Off Response, UE -> TE

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0x61	Type: Flow Off response
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FLOWOFFRES_UE2TE)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0x61,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_FLOWOFFRES_UE2TE, 11)
```

/* SDU: Flow Control Off Response, TE -> UE

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0x61	Type: Flow Off response
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FLOWOFFRES_TE2UE)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0x61,
```

```
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_FLOWOFFRES_TE2UE, 11)


/*
```

1.3.2.4 Power Saving Control (PSC)

```
*/
/* SDU: Request Power Saving Mode, MMI initiated
```

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 0, EA 1
0xff	Control Field: UIH command
0x43	Type
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_CMD_UE2TE_PSM)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0x43,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_CMD_UE2TE_PSM, 11)

/* SDU: Positive Response to Power Saving Mode, MMI initiated
```

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 0, EA 1
0xff	Control Field: UIH command
0x41	Type
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_RES_TE2UE_PSM)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0x41,
                0x01,
                (0XCF),
```

```
                0x7e
ENDFIELD(SDU_RES_TE2UE_PSM, 11)
```

/* SDU: Indicate Power Saving Mode, Peer initiated

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0x43	Type
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CMD_TE2UE_PSM)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0x43,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_CMD_TE2UE_PSM, 11)
```

/* SDU: Accept Power Saving Mode, Peer initiated

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0x41	Type
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_RES_UE2TE_PSM)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0x43,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_RES_UE2TE_PSM, 11)
```

/* SDU: Wake Up from Power Saving Mode, Peer initiated

0xe0, 0x00	SDU length in bit (28 Bytes =224 Bit)
0x00, 0x00	SDU offset in bit
0x7e...0x7e	Enough Flags 0111 1110 in order to wake

	up the multiplexer

*/

```
FIELD(SDU_FLAGS_TE2UE)
                0xe0, 0x00,
                0x00, 0x00,
                0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
                0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
                0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e
ENDFIELD(SDU_FLAGS_TE2UE, 32)
```

/* SDU: Accept Wake Up, Peer initiated

0x30, 0x00	SDU length in bit (6 Bytes = 48 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 3, EA 1
0xff	Control Field: UIH cmd/res frame
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_RES_UE2TE_WAKEUP)
                0x30, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_RES_UE2TE_WAKEUP, 10)
```

/* SDU: Wake Up from Power Saving Mode, MMI initiated

0xe0, 0x00	SDU length in bit (28 Bytes =224 Bit)
0x00, 0x00	SDU offset in bit
0x7e...0x7e	Enough Flags 0111 1110 in order to wake up the peer's multiplexer (the number of flags depends on the time the peer needs to wakeup, max. T3 seconds.) Has to be adapted to specific implementation for this test case.

*/

```
FIELD(SDU_FLAGS_UE2TE)
                0xe0, 0x00,
                0x00, 0x00,
                0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
                0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
                0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e
ENDFIELD(SDU_FLAGS_UE2TE, 32)
```

/* SDU: Accept Wake Up, MMI initiated

0x30, 0x00	SDU length in bit (6 Bytes = 48 Bit)
0x00, 0x00	SDU offset in bit

0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 0, EA 1
0xff	Control Field: UIH cmd/res frame
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_RES_TE2UE_WAKEUP)
                0x30, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_RES_TE2UE_WAKEUP, 10)


/*
```

1.3.2.5 DLC Establishment / Release

*/

/* SDU: Establish DLC A, TE initiated, SABM Frame

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI_A, C/R 1, EA 1
0x3f	Control Field: SABM command
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_ESTABLISH_DLCI_A_TE2UE)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x03,
                0x3f,
                (0XCF),
                0x7e
ENDFIELD(SDU_ESTABLISH_DLCI_A_TE2UE, 9)
```

/* SDU: Establish DLC B, TE initiated, SABM Frame

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_B << 2) | 0x03	Address Field: DLCI_B, C/R 1, EA 1
0x3f	Control Field: SABM command
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_ESTABLISH_DLCI_B_TE2UE)
```

```
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_B << 2) | 0x03,
                0x3f,
                (0XCF),
                0x7e
ENDFIELD(SDU_ESTABLISH_DLCI_B_TE2UE, 9)
```

/* SDU: Establish DLC B, UE initiated, SABM Frame

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_B << 2) | 0x01	Address Field: DLCI_B, C/R 1, EA 1
0x3f	Control Field: SABM command
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_ESTABLISH_DLCI_B_UE2TE)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_B << 2) | 0x01,
                0x3f,
                (0XCF),
                0x7e
ENDFIELD(SDU_ESTABLISH_DLCI_B_UE2TE, 9)
```

/* SDU: Establish DLC, UE initiated, SABM Frame

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x01	Address Field: DLCI_A, C/R 1, EA 1
0x3f	Control Field: SABM command
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_ESTABLISH_DLCI_A_UE2TE)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x01,
                0x3f,
                (0XCF),
                0x7e
ENDFIELD(SDU_ESTABLISH_DLCI_A_UE2TE, 9)
```

/* SDU: Establish DLC C, TE initiated, SABM Frame

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_C << 2) | 0x03	Address Field: DLCI_C, C/R 1, EA 1
0x3f	Control Field: SABM command

FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_ESTABLISH_DLCI_C_TE2UE)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_C << 2) | 0x03,
                0x3f,
                (0XCF),
                0x7e
ENDFIELD(SDU_ESTABLISH_DLCI_C_TE2UE, 9)
```

/* SDU: Establish DLC D, TE initiated, SABM Frame

0x30, 0x00	SDU length in bit (6 Bytes = 48 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x7d, ((DLCI_D << 2) | 0x03) ^ 0x20	Address Field: DLCI_D, C/R 1, EA 1
0x3f	Control Field: SABM command
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_ESTABLISH_DLCI_D_TE2UE)
                0x30, 0x00,
                0x00, 0x00,
                0x7e,
                0x7d, ((DLCI_D << 2) | 0x03) ^ 0x20,
                0x3f,
                (0XCF),
                0x7e
ENDFIELD(SDU_ESTABLISH_DLCI_D_TE2UE, 10)
```

/* SDU: Establish DLC E, TE initiated, SABM Frame

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_E << 2) | 0x03	Address Field: DLCI_E, C/R 1, EA 1
0x3f	Control Field: SABM command
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_ESTABLISH_DLCI_E_TE2UE)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_E << 2) | 0x03,
                0x3f,
                (0XCF),
                0x7e
ENDFIELD(SDU_ESTABLISH_DLCI_E_TE2UE, 9)
```

/* SDU: Disconnect DLC A, TE initiated, DISC Frame

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI_A, C/R 1, EA 1
0x53	Control Field: DISC command
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_DISC_DLCI_A_TE2UE)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x03,
                0x53,
                (0XCF),
                0x7e
ENDFIELD(SDU_DISC_DLCI_A_TE2UE, 9)
```

/* SDU: Disconnect DLC A, UE initiated, DISC Frame

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x01	Address Field: DLCI_A, C/R 0, EA 1
0x53	Control Field: DISC command
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_DISC_DLCI_A_UE2TE)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x01,
                0x53,
                (0XCF),
                0x7e
ENDFIELD(SDU_DISC_DLCI_A_UE2TE, 9)
```

/* SDU: Disconnect DLC B, TE initiated, DISC Frame

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_B<< 2) | 0x03	Address Field: DLCI_B, C/R 1, EA 1
0x53	Control Field: DISC command
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_DISC_DLCI_B_TE2UE)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_B << 2) | 0x03,
                0x53,
```

```
                (0XCF),
                0x7e
ENDFIELD(SDU_DISC_DLCI_B_TE2UE, 9)

/* SDU: Disconnect DLC B, UE initiated, DISC Frame
```

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_B<< 2) | 0x01	Address Field: DLCI_B, C/R 0, EA 1
0x53	Control Field: DISC command
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_DISC_DLCI_B_UE2TE)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_B << 2) | 0x01,
                0x53,
                (0XCF),
                0x7e
ENDFIELD(SDU_DISC_DLCI_B_UE2TE, 9)

/*
```

1.3.2.6 Data Frames

```
*/
/* SDU: Send h String in Information Frame on DLCI A, TE -> UE
```

0x30, 0x00	SDU length in bit (6 Bytes = 48Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI A, C/R 1, EA 1
0xef	Control Field: UIH information frame
0x68,	String "h"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_FRAME_H_TE2UE_A)
        0x30, 0x00,
        0x00, 0x00,
        0x7e,
        (DLCI_A << 2) | 0x03,
        0xef,
        0x68,
        (0XCF),
        0x7e
ENDFIELD(SDU_FRAME_H_TE2UE_A, 10)

/* SDU: Send Hello1 String in Information Frame on DLCI A, TE -> UE
```

0x68, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI A, C/R 1, EA 1

0xef	Control Field: UIH information frame
0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,	String "Hello1\n"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_HELLO1_TE2UE_A)
        0x60, 0x00,
        0x00, 0x00,
        0x7e,
        (DLCI_A << 2) | 0x03,
        0xef,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        (0XCF),
        0x7e
ENDFIELD(SDU_FRAME_HELLO1_TE2UE_A, 17)
```

/* SDU: Send Hello1 String in Information Frame on DLCI B, UE -> TE

0x68, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_B<< 2) | 0x01	Address Field: DLCI B, C/R 0, EA 1
0xef	Control Field: UIH information frame
0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,	String "Hello1\n"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_HELLO1_UE2TE_B)
        0x68, 0x00,
        0x00, 0x00,
        0x7e,
        (DLCI_B << 2) | 0x01,
        0xef,
        0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
        (0XCF),
        0x7e
ENDFIELD(SDU_FRAME_HELLO1_UE2TE_B, 17)
```

/* SDU: Send 0x10 0x11 0x12 in Information Frame on DLCI A, UE -> TE

0x48, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x01	Address Field: DLCI A, C/R 0, EA 1
0xef	Control Field: UIH information frame
0x10, 0x7d, 0x11 ^ 0x20, 0x12,	String "0x10 0x11 0x12"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_101112_UE2TE_A)
        0x48, 0x00,
        0x00, 0x00,
        0x7e,
        (DLCI_A << 2) | 0x01,
```

```
        0xef,
        0x10, 0x7d, 0x11 ^ 0x20, 0x12,
        (0XCF),
        0x7e
ENDFIELD(SDU_FRAME_101112_UE2TE_A, 13)
```

/* SDU: Send 0x13 0x14 0x15 in Information Frame on DLCI B, UE -> TE

0x48, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_B << 2) | 0x01	Address Field: DLCI B, C/R 0, EA 1
0xef	Control Field: UIH information frame
0x7d, 0x13 ^ 0x20, 0x14, 0x15,	String "0x13 0x14 0x15"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_131415_UE2TE_B)
        0x48, 0x00,
        0x00, 0x00,
        0x7e,
        (DLCI_B << 2) | 0x01,
        0xef,
        0x7d, 0x13 ^ 0x20, 0x14, 0x15,
        (0XCF),
        0x7e
ENDFIELD(SDU_FRAME_131415_UE2TE_B, 13)
```

/* SDU: Send 0x7c 0x7d 0x7e in Information Frame on DLCI D, UE -> TE

0x58, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x7d, ((DLCI_D << 2) | 0x01) ^ 0x20	Address Field: DLCI D, C/R 0, EA 1
0xef	Control Field: UIH information frame
0x7c, 0x7d, 0x7d ^ 0x20, 0x7d, 0x7e ^ 0x20,	String "0x7c 0x7d 0x7e"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_7c7d7e_UE2TE_D)
        0x58, 0x00,
        0x00, 0x00,
        0x7e,
        0x7d, ((DLCI_D << 2) | 0x01) ^ 0x20,
        0xef,
        0x7c, 0x7d, 0x7d ^ 0x20, 0x7d, 0x7e ^ 0x20,
        (0XCF),
        0x7e
ENDFIELD(SDU_FRAME_7c7d7e_UE2TE_D, 15)
```

/* SDU: Send 0x7e 0x7f 0x80 in Information Frame on DLCI D, UE -> TE, special XON and XOFF

0x58, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_D << 2) | 0x01	Address Field: DLCI D, C/R 0, EA 1

0x7d, 0xef ^ 0x20	Control Field: UIH information frame
0x7d, 0x7e ^ 0x20, 0x7f, 0x80,	String "0x7e 0x7f 0x80"
0x7d, FCS_DUMMY_OK ^ 0x20	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_7e7f80_UE2TE_D_X)
        0x58, 0x00,
        0x00, 0x00,
        0x7e,
        (DLCI_D << 2) | 0x01,
        0x7d, 0xef ^ 0x20,
        0x7d, 0x7e ^ 0x20, 0x7f, 0x80,
        0x7d, (0XCF) ^ 0x20,
        0x7e
ENDFIELD(SDU_FRAME_7e7f80_UE2TE_D_X, 15)
```

/* SDU: Send 0x7c 0x7d 0x7e in Information Frame on DLCI E, UE -> TE, special XON and XOFF

0x68, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x7d, ((DLCI_E << 2) | 0x01) ^ 0x20	Address Field: DLCI E, C/R 0, EA 1
0x7d, 0xef ^ 0x20	Control Field: UIH information frame
0x7c, 0x7d, 0x7d ^ 0x20, 0x7d, 0x7e ^ 0x20,	String "0x7c 0x7d 0x7e"
0x7d, FCS_DUMMY_OK ^ 0x20	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_7c7d7e_UE2TE_E_X)
        0x68, 0x00,
        0x00, 0x00,
        0x7e,
        0x7d, ((DLCI_E << 2) | 0x01) ^ 0x20,
        0x7d, 0xef ^ 0x20,
        0x7c, 0x7d, 0x7d ^ 0x20, 0x7d, 0x7e ^ 0x20,
        0x7d, (0XCF) ^ 0x20,
        0x7e
ENDFIELD(SDU_FRAME_7c7d7e_UE2TE_E_X, 17)
```

/* SDU: Send 0x7e 0x7f 0x80 in Information Frame on DLCI E, UE -> TE

0x50, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x7d, ((DLCI_E << 2) | 0x01) ^ 0x20	Address Field: DLCI E, C/R 0, EA 1
0xef	Control Field: UIH information frame
0x7d, 0x7e ^ 0x20, 0x7f, 0x80,	String "0x7e 0x7f 0x80"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_7e7f80_UE2TE_E)
        0x50, 0x00,
        0x00, 0x00,
        0x7e,
        0x7d, ((DLCI_E << 2) | 0x01) ^ 0x20,
```

```
        0xef,
        0x7d, 0x7e ^ 0x20, 0x7f, 0x80,
        (0XCF),
        0x7e
ENDFIELD(SDU_FRAME_7e7f80_UE2TE_E, 14)
```

/* SDU: Send 'h' in Information Frame on DLCI B, TE -> UE

0x30, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_B<< 2) | 0x01	Address Field: DLCI B, C/R 0, EA 1
0xef	Control Field: UIH information frame
0x68	String "h"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_H_TE2UE_B)
                0x30, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_B << 2) | 0x01,
                0xef,
                0x68,
                (0XCF),
                0x7e
ENDFIELD(SDU_FRAME_H_TE2UE_B, 10)
```

/* SDU: Send '+++' in Information Frame on DLCI B, TE -> UE

0x40, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_B<< 2) | 0x03	Address Field: DLCI B, C/R 1, EA 1
0xef	Control Field: UIH information frame
0x2B, 0x2B, 0x2B	String "+++"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_ESC_SEQ_B)
                0x40, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_B << 2) | 0x03,
                0xef,
                0x2B, 0x2B, 0x2B,
                (0XCF),
                0x7e
ENDFIELD(SDU_FRAME_ESC_SEQ_B, 12)
```

/* SDU: Send '+++' in Information Frame on DLCI C, TE -> UE

0x40, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_C<< 2) | 0x03	Address Field: DLCI C, C/R 1, EA 1

0xef	Control Field: UIH information frame
0x2B, 0x2B, 0x2B	String "+++"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_ESC_SEQ_C)
                0x40, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_C << 2) | 0x03,
                0xef,
                0x2B, 0x2B, 0x2B,
                (0XCF),
                0x7e
ENDFIELD(SDU_FRAME_ESC_SEQ_C, 12)
```

/* SDU: Send '+++' in Information Frame on DLCI A TE -> UE

0x40, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A< 2) | 0x03	Address Field: DLCI A C/R 1, EA 1
0xef	Control Field: UIH information frame
0x2B, 0x2B, 0x2B	String "+++"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_ESC_SEQ_A)
                0x40, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A<< 2) | 0x03,
                0xef,
                0x2B, 0x2B, 0x2B,
                (0XCF),
                0x7e
ENDFIELD(SDU_FRAME_ESC_SEQ_A, 12)
```

/* SDU: Send 'I' in Information Frame on DLCI C, TE -> UE

0x30, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_C<< 2) | 0x03	Address Field: DLCI C, C/R 1, EA 1
0xef	Control Field: UIH information frame
0x6C	String "I"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_L_TE2UE_C)
                0x30, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_C << 2) | 0x03,
```

```
                0xef,
                0x6C,
                (0XCF),
                0x7e
ENDFIELD(SDU_FRAME_L_TE2UE_C, 10)

/* SDU: Send 'e' in Information Frame on DLCI A, TE -> UE
```

0x30, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A<< 2) | 0x03	Address Field: DLCI A, C/R 1, EA 1
0xef	Control Field: UIH information frame
0x65	String "e"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_FRAME_E_TE2UE_A)
                0x30, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x03,
                0xef,
                0x65,
                (0XCF),
                0x7e
ENDFIELD(SDU_FRAME_E_TE2UE_A, 10)

/* SDU: Send Hello2 String in Information Frame on DLCI A, UE -> TE
```

0x68, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x01	Address Field: DLCI A, C/R 0, EA 1
0xef	Control Field: UIH information frame
0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x32, 0x0d, 0x0a,	String "Hello2\n"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_FRAME_HELLO2_UE2TE_A)
                0x68, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x01,
                0xef,
                0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x32, 0x0d, 0x0a,
                (0XCF),
                0x7e
ENDFIELD(SDU_FRAME_HELLO2_UE2TE_A, 17)

/* SDU: Send Hello1 String in Information Frame on DLCI A, UE -> TE
```

0x68, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A<< 2) | 0x01	Address Field: DLCI A, C/R 0, EA 1

0xef	Control Field: UIH information frame
0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,	String "Hello1\n"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_HELLO1_UE2TE_A)
                0x68, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x01,
                0xef,
                0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
                (0XCF),
                0x7e
ENDFIELD(SDU_FRAME_HELLO1_UE2TE_A, 17)
```

/* SDU: Send Hello2 String in Information Frame on DLCI A, TE -> UE

0x68, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI A, C/R 1, EA 1
0xef	Control Field: UIH information frame
0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x32, 0x0d, 0x0a,	String "Hello2\n"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_HELLO2_TE2UE_A)
                0x68, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x03,
                0xef,
                0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x32, 0x0d, 0x0a,
                (0XCF),
                0x7e
ENDFIELD(SDU_FRAME_HELLO2_TE2UE_A, 17)
```

/* SDU: Send 0x10 0x11 0x12 in Information Frame on DLCI A, TE -> UE

0x48, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI_A, C/R 1, EA 1
0xef	Control Field: UIH information frame
0x10, 0x7d, 0x11 ^ 0x20, 0x12,	Data "0x10 0x11 0x12"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_101112_TE2UE_A)
        0x48, 0x00,
        0x00, 0x00,
        0x7e,
        (DLCI_A << 2) | 0x03,
```

```
        0xef,
        0x10, 0x7d, 0x11 ^ 0x20, 0x12,
        (0XCF),
        0x7e
ENDFIELD(SDU_FRAME_101112_TE2UE_A, 13)
```

/* SDU: Send 0x7e 0x7f 0x80 in Information Frame on DLCI B, TE -> UE

0x48, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_B << 2) | 0x03	Address Field: DLCI B, C/R 1, EA 1
0xef	Control Field: UIH information frame
0x7d, 0x7e ^ 0x20, 0x7f, 0x80,	Data "0x7e 0x7f 0x80"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_7e7f80_TE2UE_B)
        0x48, 0x00,
        0x00, 0x00,
        0x7e,
        (DLCI_B << 2) | 0x03,
        0xef,
        0x7d, 0x7e ^ 0x20, 0x7f, 0x80,
        (0XCF),
        0x7e
ENDFIELD(SDU_FRAME_7e7f80_TE2UE_B, 13)
```

/* SDU: Send 0x13 0x14 0x15 in Information Frame on DLCI D, TE -> UE

0x50, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x7d, ((DLCI_D << 2) | 0x03) ^ 0x20	Address Field: DLCI D, C/R 1, EA 1
0xef	Control Field: UIH information frame
0x7d, 0x13 ^ 0x20, 0x14, 0x15	Data "0x13 0x14 0x15"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_131415_TE2UE_D)
        0x50, 0x00,
        0x00, 0x00,
        0x7e,
        0x7d, ((DLCI_D << 2) | 0x03) ^ 0x20,
        0xef,
        0x7d, 0x13 ^ 0x20, 0x14, 0x15,
        (0XCF),
        0x7e
ENDFIELD(SDU_FRAME_131415_TE2UE_D, 14)
```

/* SDU: Send 0x7c 0x7d 0x7e in Information Frame on DLCI D, TE -> UE, special XON and XOFF

0x60, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_D << 2) | 0x03	Address Field: DLCI D, C/R 1, EA 1

0x7d, 0xef ^ 0x20	Control Field: UIH information frame
0x7c, 0x7d, 0x7d ^ 0x20, 0x7d, 0x7e ^ 0x20	Data "0x7c 0x7d 0x7e"
0x7d, FCS_DUMMY_OK ^ 0x20	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_7c7d7e_TE2UE_D_X)
        0x60, 0x00,
        0x00, 0x00,
        0x7e,
        (DLCI_D << 2) | 0x03,
        0x7d, 0xef ^ 0x20,
        0x7c, 0x7d, 0x7d ^ 0x20, 0x7d, 0x7e ^ 0x20,
        0x7d, (0XCF) ^ 0x20,
        0x7e
ENDFIELD(SDU_FRAME_7c7d7e_TE2UE_D_X, 16)
```

/* SDU: Send 0x13 0x14 0x15 in Information Frame on DLCI E, TE -> UE, special XON and XOFF

0x50, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_E << 2) | 0x03	Address Field: DLCI E, C/R 1, EA 1
0x7d, 0xef ^ 0x20	Control Field: UIH information frame
0x13, 0x14, 0x15	Data "0x13 0x14 0x15"
0x7d, FCS_DUMMY_OK ^ 0x20	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_131415_TE2UE_E_X)
        0x50, 0x00,
        0x00, 0x00,
        0x7e,
        (DLCI_E << 2) | 0x03,
        0x7d, 0xef ^ 0x20,
        0x13, 0x14, 0x15,
        0x7d, (0XCF) ^ 0x20,
        0x7e
ENDFIELD(SDU_FRAME_131415_TE2UE_E_X, 14)
```

/* SDU: Send 0x7c 0x7d 0x7e in Information Frame on DLCI E, TE -> UE

0x50, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_E << 2) | 0x03	Address Field: DLCI E, C/R 1, EA 1
0xef	Control Field: UIH information frame
0x7c, 0x7d, 0x7d ^ 0x20, 0x7d, 0x7e ^ 0x20	Data "0x7c 0x7d 0x7e"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_7c7d7e_TE2UE_E)
        0x50, 0x00,
        0x00, 0x00,
        0x7e,
        (DLCI_E << 2) | 0x03,
```

```
        0xef,
        0x7c, 0x7d, 0x7d ^ 0x20, 0x7d, 0x7e ^ 0x20,
        (0XCF),
        0x7e
ENDFIELD(SDU_FRAME_7c7d7e_TE2UE_E, 14)
```

/* SDU: Send AT command string in Information Frame on DLCI A, TE -> UE

0x48, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI A, C/R 1, EA 1
0xef	Control Field: UIH information frame
0x41, 0x54, 0x0d, 0x0a,	String "AT\n"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_AT_TE2UE_A)
        0x48, 0x00,
        0x00, 0x00,
        0x7e,
        (DLCI_A << 2) | 0x03,
        0xef,
        0x41, 0x54, 0x0d, 0x0a,
        (0XCF),
        0x7e
ENDFIELD(SDU_FRAME_AT_TE2UE_A, 13)
```

/* SDU: Send AT command string in Information Frame on DLCI B, TE -> UE

0x48, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_B << 2) | 0x03	Address Field: DLCI B, C/R 1, EA 1
0xef	Control Field: UIH information frame
0x41, 0x54, 0x0d, 0x0a,	String "AT\n"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_FRAME_AT_TE2UE_B)
                0x48, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_B << 2) | 0x03,
                0xef,
                0x41, 0x54, 0x0d, 0x0a,
                (0XCF),
                0x7e
ENDFIELD(SDU_FRAME_AT_TE2UE_B, 13)
```

/* SDU: Send AT command string in Information Frame on DLCI C, TE -> UE

0x48, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_C << 2) | 0x03	Address Field: DLCI C, C/R 1, EA 1

0xef	Control Field: UIH information frame
0x41, 0x54, 0x0d, 0x0a,	String "AT\n"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_FRAME_AT_TE2UE_C)
                0x48, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_C << 2) | 0x03,
                0xef,
                0x41, 0x54, 0x0d, 0x0a,
                (0XCF),
                0x7e
ENDFIELD(SDU_FRAME_AT_TE2UE_C, 13)

/* SDU: Send 64 octets in Information Frame on DLCI A, TE -> UE
```

0x28, 0x02	SDU length in bit (69 octets)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI A, C/R 1, EA 1
0xef	Control Field: UIH information frame
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,	String "ABCDEFGH"
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,	String "ABCDEFGH"
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,	String "ABCDEFGH"
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,	String "ABCDEFGH"
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,	String "ABCDEFGH"
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,	String "ABCDEFGH"
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,	String "ABCDEFGH"
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,	String "ABCDEFGH"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_FRAME_64_TE2UE_A)
                0x28, 0x02,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x03,
                0xef,
                0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
                0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
                0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
                0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
                0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
                0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
                0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
                0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
                (0XCF),
                0x7e
ENDFIELD(SDU_FRAME_64_TE2UE_A, 73)

/*
```

1.3.2.7 Disconnected Mode (DM) Frames

*/

/* SDU: DM Response from UE to TE for DLCI A

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI_A, C/R 1, EA 1
0x1f	Control Field: DM Response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_DM_UE2TE_A)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x03,
                0x1f,
                (0XCF),
                0x7e
ENDFIELD(SDU_DM_UE2TE_A, 9)
```

/* SDU: DM Response from UE to TE for DLCI A, Type 0F

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI_A, C/R 1, EA 1
0x0f	Control Field: DM Response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_DM_UE2TE_A_0F)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x03,
                0x0f,
                (0XCF),
                0x7e
ENDFIELD(SDU_DM_UE2TE_A_0F, 9)
```

/* SDU: DM Response from TE to UE for DLCI A

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x01	Address Field: DLCI_A, C/R 0, EA 1
0x1f	Control Field: DM Response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_DM_TE2UE_A)
                0x28, 0x00,
                0x00, 0x00,
```

```
                0x7e,
                (DLCI_A << 2) | 0x01,
                0x1f,
                (0XCF),
                0x7e
ENDFIELD(SDU_DM_TE2UE_A, 9)

/* SDU: DM Response from TE to UE for DLCI B
```

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_B << 2) | 0x01	Address Field: DLCI_B, C/R 0, EA 1
0x1f	Control Field: DM Response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_DM_TE2UE_B)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_B << 2) | 0x01,
                0x1f,
                (0XCF),
                0x7e
ENDFIELD(SDU_DM_TE2UE_B, 9)

/*
/* SDU: DM Response from UE to TE for SDU_CORRUPTED_DATA8
```

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(0x43 << 2) | 0x03	Address Field: DLCI_A, C/R 1, EA 1
0x0f	Control Field: DM Response for UIH
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_DM_CORRUPTED_DATA8)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (0x43 << 2) | 0x03,
                0x0f,
                (0XCF),
                0x7e
ENDFIELD(SDU_DM_CORRUPTED_DATA8, 9)

/*
```

1.3.2.8 Modem Status Commands (MSC)

```
*/
/* SDU: Normal Modem Status Cmd, TE->UE
```

0x48, 0x00	SDU length in bit (9 Bytes)
0x00, 0x00	SDU offset in bit

0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xe3	Type: MSC command
0x05	Length is 2, EA bit is 1
(DLCI_A << 2) | 0x01	status for DLCI A
0x8d	V.24 signals: 1000 1101
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_MSC_CMD_TE2UE_A)
                0x48, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0xe3,
                0x05,
                (DLCI_A << 2) | 0x01,
                0x8d,
                (0XCF),
                0x7e
ENDFIELD(SDU_MSC_CMD_TE2UE_A, 13)
```

/* SDU: Normal Modem Status Res, UE->TE

0x48, 0x00	SDU length in bit (9 Bytes)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xe1	Type: MSC response
0x05	Length is 2, EA bit is 1
(DLCI_A << 2) | 0x01	status for DLCI A
0x8d	V.24 signals: 1000 1101
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_MSC_RES_UE2TE_A)
                0x48, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0xe1,
                0x05,
                (DLCI_A << 2) | 0x01,
                0x8d,
                (0XCF),
                0x7e
ENDFIELD(SDU_MSC_RES_UE2TE_A, 13)
```

/* SDU: Break Modem Status Cmd, TE->UE

0x50, 0x00	SDU length in bit (10 Bytes = 80 Bit)

0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xe3	Type: MSC command
0x07	Length is 3, EA bit is 1
(DLCI_A << 2) | 0x01	status for DLCI A
0x8d	V.24 signals: 1000 1101
(UART_BREAK_LEN << 4) | 0x03	high byte: break length, low byte: 3
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_MSCBREAK_CMD_TE2UE_A)
                0x50, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0xe3,
                0x07,
                (DLCI_A << 2) | 0x01,
                0x8d,
                (UART_BREAK_LEN << 4) | 0x03,
                (0XCF),
                0x7e
ENDFIELD(SDU_MSCBREAK_CMD_TE2UE_A, 14)
```

/* SDU: Break Modem Status Res, UE->TE

0x50, 0x00	SDU length in bit (10 Bytes = 80 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xe1	Type: MSC response
0x07	Length is 3, EA bit is 1
(DLCI_A << 2) | 0x01	status for DLCI A
0x8d	V.24 signals: 1000 1101
(UART_BREAK_LEN << 4) | 0x03	high byte: break length, low byte: 3
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_MSCBREAK_RES_UE2TE_A)
                0x50, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0xe1,
                0x07,
                (DLCI_A << 2) | 0x01,
                0x8d,
                (UART_BREAK_LEN << 4) | 0x03,
                (0XCF),
```

```
                0x7e
ENDFIELD(SDU_MSCBREAK_RES_UE2TE_A, 14)

/* SDU: Break Modem Status Cmd, UE->TE
```

0x50, 0x00	SDU length in bit (10 Bytes = 80 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 0, EA 1
0xff	Control Field: UIH command
0xe3	Type: MSC command
0x07	Length is 3, EA bit is 1
(DLCI_A << 2) | 0x01	status for DLCI A
0x8d	V.24 signals: 10001101
(UART_BREAK_LEN << 4) | 0x03	high byte: break length, low byte: 3
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_MSCBREAK_CMD_UE2TE_A)
                0x50, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0xe3,
                0x07,
                (DLCI_A << 2) | 0x03,
                0x8d,
                (UART_BREAK_LEN << 4) | 0x03,
                (0XCF),
                0x7e
ENDFIELD(SDU_MSCBREAK_CMD_UE2TE_A, 14)

/* SDU: Break Modem Status Res, TE->UE
```

0x50, 0x00	SDU length in bit (10 Bytes = 80 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xe1	Type: MSC response
0x07	Length is 3, EA bit is 1
(DLCI_A << 2) | 0x03	status for DLCI A
0x8d	V.24 signals: 10001101
(UART_BREAK_LEN << 4) | 0x03	high byte: break length, low byte: 3
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_MSCBREAK_RES_TE2UE_A)
                0x50, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0xe1,
```

```
                0x07,
                (DLCI_A << 2) | 0x03,
                0x8d,
                (UART_BREAK_LEN << 4) | 0x03,
                (0XCF),
                0x7e
ENDFIELD(SDU_MSCBREAK_RES_TE2UE_A, 14)
```

/* SDU: Invalid Break Modem Status Res, TE->UE

0x50, 0x00	SDU length in bit (10 Bytes = 80 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xe1	Type: MSC response
0x07	Length is 3, EA bit is 1
(DLCI_A << 2) | 0x01	invalid: bit 1 is 0
0x01	V.24 signals: invalid, only EA bit set
(UART_BREAK_LEN << 4) | 0x03	high byte: break length, low byte: 3
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_MSCBREAK_RES_TE2UE_A_INVALID)
                0x50, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0xe1,
                0x07,
                (DLCI_A << 2) | 0x01,
                0x01,
                (UART_BREAK_LEN << 4) | 0x03,
                (0XCF),
                0x7e
ENDFIELD(SDU_MSCBREAK_RES_TE2UE_A_INVALID, 14)
```

/* SDU: Ring Modem Status Cmd, UE->TE

0x48, 0x00	SDU length in bit (9 Bytes)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 0, EA 1
0xff	Control Field: UIH command
0xe3	Type: MSC command
0x05	Length is 2, EA bit is 1
(DLCI_A << 2) | 0x03	status for DLCI A
0xcd	V.24 signals: 1100 1101
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_MSCRING_CMD_UE2TE_A)
                0x48, 0x00,
                0x00, 0x00,
```

```
                0x7e,
                0x01,
                0xff,
                0xe3,
                0x05,
                (DLCI_A << 2) | 0x03,
                0xcd,
                (0XCF),
                0x7e
ENDFIELD(SDU_MSCRING_CMD_UE2TE_A, 13)

/* SDU: Ring Modem Status Res, TE->UE
```

0x48, 0x00	SDU length in bit (9 Bytes)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xe1	Type: MSC response
0x05	Length is 2, EA bit is 1
(DLCI_A << 2) | 0x03	status for DLCI A
0xcd	V.24 signals: 1100 1101
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_MSCRING_RES_TE2UE_A)
                0x48, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0xe1,
                0x05,
                (DLCI_A << 2) | 0x03,
                0xcd,
                (0XCF),
                0x7e
ENDFIELD(SDU_MSCRING_RES_TE2UE_A, 13)

/* SDU: Flow Control Modem Status Cmd, UE->TE
```

0x48, 0x00	SDU length in bit (9 Bytes = 72 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 0, EA 1
0xff	Control Field: UIH command
0xe3	Type: MSC command
0x05	Length is 2, EA bit is 1
(DLCI_A << 2) | 0x01	status for DLCI A
0x8f	V.24 signals: 10001101
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_MSCFCOFF_CMD_UE2TE_A)
                0x48, 0x00,
```

```
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0xe3,
                0x05,
                (DLCI_A << 2) | 0x03,
                0x8f,
                (0XCF),
                0x7e
ENDFIELD(SDU_MSCFCOFF_CMD_UE2TE_A, 13)

/* SDU: Normal Modem Status Cmd, UE -> TE
```

0x48, 0x00	SDU length in bit (9 Bytes)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 0, EA 1
0xff	Control Field: UIH command
0xe3	Type: MSC command
0x05	Length is 2, EA bit is 1
(DLCI_A << 2) | 0x01	status for DLCI A
0x8d	V.24 signals: 1000 1101
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_MSC_CMD_UE2TE_A)
                0x48, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0xff,
                0xe3,
                0x05,
                (DLCI_A << 2) | 0x03,
                0x8d,
                (0XCF),
                0x7e
ENDFIELD(SDU_MSC_CMD_UE2TE_A, 13)

/*
```

1.3.2.9 Invalid Frames

```
*/
/* SDU: An SDU with completely corrupted data
```

0x00, 0x03	SDU length in bit
0x00, 0x00	SDU offset in bit
0x55, 0x67, 0x54, 0x73, 0xff, 0x10, 0x22, 0x14, 0x22, 0x65, 0x34, 0xfe, 0xff, 0x20, 0x30, 0x66, 0x55, 0x67, 0x54, 0x73, 0xff, 0x10, 0x22, 0x14, 0x22, 0x65, 0x34, 0xfe, 0xff, 0x20, 0x30, 0x660x55, 0x67, 0x54, 0x73, 0xff, 0x10, 0x22, 0x14, 0x22, 0x65, 0x34, 0xfe, 0xff, 0x20, 0x30, 0x66, 0x55, 0x67, 0x54, 0x73, 0xff, 0x10, 0x22, 0x14, 0x22, 0x65, 0x34, 0xfe, 0xff, 0x20, 0x30, 0x66	some random data

0x55, 0x67, 0x54, 0x73, 0xff, 0x10, 0x22, 0x14, 0x22, 0x65, 0x34, 0xfe, 0xff, 0x20, 0x30, 0x66, 0x55, 0x67, 0x54, 0x73, 0xff, 0x10, 0x22, 0x14, 0x22, 0x65, 0x34, 0xfe, 0xff, 0x20, 0x30, 0x66	

*/

```
FIELD(SDU_CORRUPTED_DATA1)
            0x00, 0x03,
            0x00,0x00,
            0x55, 0x67, 0x54, 0x73, 0xff, 0x10, 0x22, 0x14,
            0x22, 0x65, 0x34, 0xfe, 0xff, 0x20, 0x30, 0x66,
            0x55, 0x67, 0x54, 0x73, 0xff, 0x10, 0x22, 0x14,
            0x22, 0x65, 0x34, 0xfe, 0xff, 0x20, 0x30, 0x66,
            0x55, 0x67, 0x54, 0x73, 0xff, 0x10, 0x22, 0x14,
            0x22, 0x65, 0x34, 0xfe, 0xff, 0x20, 0x30, 0x66,
            0x55, 0x67, 0x54, 0x73, 0xff, 0x10, 0x22, 0x14,
            0x22, 0x65, 0x34, 0xfe, 0xff, 0x20, 0x30, 0x66,
            0x55, 0x67, 0x54, 0x73, 0xff, 0x10, 0x22, 0x14,
            0x22, 0x65, 0x34, 0xfe, 0xff, 0x20, 0x30, 0x66,
            0x55, 0x67, 0x54, 0x73, 0xff, 0x10, 0x22, 0x14,
            0x22, 0x65, 0x34, 0xfe, 0xff, 0x20, 0x30, 0x66
ENDFIELD(SDU_CORRUPTED_DATA1, 100)
```

/* SDU: MUX Startup (TE initiated) SABM Frame - upper Flag corrupted

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7f	Flag Sequence (corrupt) 0111 1111
0x03	Address Field: DLCI 0, C/R 1, EA 1
0x3f	Control Field: SABM command
FCS_DUMMY_OK_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA2)
                        0x28, 0x00,
                        0x00, 0x00,
                        0x7f,
                        0x03,
                        0x3f,
                        (0XCF),
                        0x7e
ENDFIELD(SDU_CORRUPTED_DATA2, 9)
```

/* SDU: DM Response from TE to UE for DLCI A - lower Flag corrupted

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x01	Address Field: DLCI_A, C/R 0, EA 1
0x1f	Control Field: DM Response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x0e	corrupted Flag Sequence 0000 1110

*/

```
FIELD(SDU_CORRUPTED_DATA3)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x01,
                0x1f,
                (0XCF),
                0x0e
ENDFIELD(SDU_CORRUPTED_DATA3, 9)
```

/* SDU: An SDU with flags only

0x00, 0x03	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e, 0x7e	some flags

*/

```
FIELD(SDU_CORRUPTED_DATA4)
        0x00, 0x03,
        0x00,0x00,
        0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
        0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
        0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
        0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
        0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
        0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
        0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
        0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
        0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
        0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
        0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e,
        0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e
ENDFIELD(SDU_CORRUPTED_DATA4, 100)
```

/* SDU: MUX Startup (TE initiated) SABM Frame - wrong FCS

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0x3f	Control Field: SABM command
FCS_DUMMY_INVALID	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA6)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0x3f,
                FCS_DUMMY_INVALID,
                0x7e
ENDFIELD(SDU_CORRUPTED_DATA6, 9)
```

/* SDU: MUX Startup (TE initiated) SABM Frame - only two bytes between flags

0x20, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x3f	Control Field: SABM command
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA7)
                0x20, 0x00,
                0x00, 0x00,
                0x7e,
                0x3f,
                (0XCF),
                0x7e
ENDFIELD(SDU_CORRUPTED_DATA7, 8)
```

/* SDU: Send h String in Information Frame on invalid DLCI, TE -> UE

0x30, 0x00	SDU length in bit (6 Bytes = 48Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(0x43 << 2) | 0x03	Address Field: DLCI , C/R 1, EA 1
0xef	Control Field: UIH information frame
0x68,	String "h"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA8)
                0x30, 0x00,
                0x00, 0x00,
                0x7e,
                (0x43 << 2) | 0x03,
                0xef,
                0x68,
                (0XCF),
                0x7e
ENDFIELD(SDU_CORRUPTED_DATA8, 10)
```

/* SDU: Send Hello1 String in Information Frame on DLCI A - no EA bit

0x68, 0x00	SDU length in bit
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x02	Address Field: DLCI A, C/R 1, EA 0

0xef	Control Field: UIH information frame
0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,	String "Hello1\n"
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA9)
                0x60, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x02,
                0xef,
                0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x31, 0x0d, 0x0a,
                (0XCF),
                0x7e
ENDFIELD(SDU_CORRUPTED_DATA9, 17)
```

/* SDU: invalid control octet

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI_A, C/R 1, EA 1
0x3d	Control Field: invalid
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA11)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x03,
                0x3d,
                (0XCF),
                0x7e
ENDFIELD(SDU_CORRUPTED_DATA11, 9)
```

/* SDU: SABM mit P bit set to 0

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
(DLCI_A << 2) | 0x03	Address Field: DLCI_A, C/R 1, EA 1
0x2f	Control Field: SABM with P set 0
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA12)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                (DLCI_A << 2) | 0x03,
                0x2f,
                (0XCF),
                0x7e
```

ENDFIELD(SDU_CORRUPTED_DATA12, 9)

/* SDU: DISC command on DLCI 0, P bit set to 0

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0x43	Control Field: DISC command, P set 0
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA13)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0x43,
                (0XCF),
                0x7e
ENDFIELD(SDU_CORRUPTED_DATA13, 9)
```

/* SDU: UA Response from TE to UE, F bit set to 0

0x28, 0x00	SDU length in bit (5 Bytes = 40 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x01	Address Field: DLCI 0, C/R 0, EA 1
0x63	Control Field: UA response
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA14)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                0x01,
                0x63,
                (0XCF),
                0x7e
ENDFIELD(SDU_CORRUPTED_DATA14, 9)
```

/* SDU: Message Type has EA bit 0

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xe2	Type: EA bit is 0
0x05	Length is 2, EA bit is 1
DLCI_A	status for DLCI A
0x8d	V.24 signals: 1000 1101
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA16)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0xe2,
                0x05,
                DLCI_A,
                0x8d,
                (0XCF),
                0x7e
ENDFIELD(SDU_CORRUPTED_DATA16, 16)
```

/* SDU: Message Length has EA bit 0

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xe3	Type: MSC
0x04	Length is 2, EA bit is 0
DLCI_A	status for DLCI A
0x8d	V.24 signals: 1000 1101
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA17)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0xe3,
                0x04,
                DLCI_A,
                0x8d,
                (0XCF),
                0x7e
ENDFIELD(SDU_CORRUPTED_DATA17, 16)
```

/* SDU: MSC, Peer initiated, corrupted length field

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xe3	Type
0xf1	Length is corrupt, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA18)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0xe3,
                0xf1,
                (0XCF),
                0x7e
ENDFIELD(SDU_CORRUPTED_DATA18, 11)
```

/* SDU: Break Modem Status Cmd, TE->UE, info shorter than length

0x50, 0x00	SDU length in bit (10 Bytes)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xe3	Type: MSC command
0x09	Length is 4, EA bit is 1
DLCI_A	status for DLCI A
0x8d	V.24 signals: 1000 1101
(UART_BREAK_LEN << 4) | 0x03	high byte: break length, low byte: 3
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA19)
                0x50, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0xe3,
                0x09,
                DLCI_A,
                0x8d,
                (UART_BREAK_LEN << 4) | 0x03,
                (0XCF),
                0x7e
ENDFIELD(SDU_CORRUPTED_DATA19, 14)
```

/* SDU: Break Modem Status Cmd, TE->UE, info longer than length

0x50, 0x00	SDU length in bit (10 Bytes)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xe3	Type: MSC command
0x05	Length is 2, EA bit is 1
DLCI_A	status for DLCI A
0x8d	V.24 signals: 1000 1101
(UART_BREAK_LEN << 4) | 0x03	high byte: break length, low byte: 3
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)

0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA20)
                0x50, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                0xe3,
                0x05,
                DLCI_A,
                0x8d,
                (UART_BREAK_LEN << 4) | 0x03,
                (0XCF),
                0x7e
ENDFIELD(SDU_CORRUPTED_DATA20, 14)
```

/* SDU: Break Modem Status Cmd, TE->UE, frame longer than 64 octets

0x30, 0x02	SDU length in bit (70 Bytes)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0xe3, 0x05, 0x07, 0x8d, 0x13	MSC command for DLCI 1
0xe3, 0x05, 0x0b, 0x8d, 0x13	MSC command for DLCI 2
0xe3, 0x05, 0x0f, 0x8d, 0x13	MSC command for DLCI 3
0xe3, 0x05, 0x13, 0x8d, 0x13	MSC command for DLCI 4
0xe3, 0x05, 0x17, 0x8d, 0x13	MSC command for DLCI 5
0xe3, 0x05, 0x1b, 0x8d, 0x13	MSC command for DLCI 6
0xe3, 0x05, 0x1f, 0x8d, 0x13	MSC command for DLCI 7
0xe3, 0x05, 0x23, 0x8d, 0x13	MSC command for DLCI 8
0xe3, 0x05, 0x27, 0x8d, 0x13	MSC command for DLCI 9
0xe3, 0x05, 0x2b, 0x8d, 0x13	MSC command for DLCI 10
0xe3, 0x05, 0x2f, 0x8d, 0x13	MSC command for DLCI 11
0xe3, 0x05, 0x33, 0x8d, 0x13	MSC command for DLCI 12
0xe3, 0x05, 0x37, 0x8d, 0x13	MSC command for DLCI 13
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

*/

```
FIELD(SDU_CORRUPTED_DATA21)
        0x30, 0x02,
        0x00, 0x00,
        0x7e,
        0x03,
        0xff,
        0xe3, 0x05, 0x07, 0x8d, 0x13,
        0xe3, 0x05, 0x0b, 0x8d, 0x13,
        0xe3, 0x05, 0x0f, 0x8d, 0x13,
        0xe3, 0x05, 0x13, 0x8d, 0x13,
        0xe3, 0x05, 0x17, 0x8d, 0x13,
        0xe3, 0x05, 0x1b, 0x8d, 0x13,
        0xe3, 0x05, 0x1f, 0x8d, 0x13,
        0xe3, 0x05, 0x23, 0x8d, 0x13,
```

```
        0xe3, 0x05, 0x27, 0x8d, 0x13,
        0xe3, 0x05, 0x2b, 0x8d, 0x13,
        0xe3, 0x05, 0x2f, 0x8d, 0x13,
        0xe3, 0x05, 0x33, 0x8d, 0x13,
        0xe3, 0x05, 0x37, 0x8d, 0x13,
        (0XCF),
        0x7e
ENDFIELD(SDU_CORRUPTED_DATA21, 74)

/*
```

1.3.2.10 Other Frames

```
*/
/* SDU: Unknown Command, Peer initiated
```

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
UART_UNKNOWN_COMMAND	Type
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_UNKNOWN_CMD_TE2UE_A)
                0x38, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                UART_UNKNOWN_COMMAND,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_UNKNOWN_CMD_TE2UE_A, 11)

/* SDU: Unsupported Command Response, UE->TE
```

0x48, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7e	Flag Sequence 0111 1110
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
0x7d, 0x31 = 0x11 (transparency)	Type: unsupported command
0x03	Length is 1, EA bit is 1
UART_UNKNOWN_COMMAND	Type of the Unsupported Cmd
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_UNSUPP_CMD_UE2TE_A)
                0x48, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
```

```
                0xff,
                0x7d,
                0x31,
                0x03,
                UART_UNKNOWN_COMMAND,
                (0XCF),
                0x7e
ENDFIELD(SDU_UNSUPP_CMD_UE2TE_A, 12)

/* SDU: Corrupted/Invalid Frame I
```

0x38, 0x00	SDU length in bit (7 Bytes = 56 Bit)
0x00, 0x00	SDU offset in bit
0x7f	Flag Sequence 0111 1111 (corrupted!)
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
UART_UNKNOWN_COMMAND	Type
0x01	Length is 0, EA bit is 1
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_CORRUPTED_1)
                0x38, 0x00,
                0x00, 0x00,
                0x7f,
                0x03,
                0xff,
                0x43,
                0x01,
                (0XCF),
                0x7e
ENDFIELD(SDU_CORRUPTED_1, 11)

/* SDU: Empty Response UE->TE
```

0x28, 0x00	SDU length in bit (5 Bytes)
0x00, 0x00	SDU offset in bit
0x7f	Flag Sequence 0111 1111 (corrupted!)
0x03	Address Field: DLCI 0, C/R 1, EA 1
0xff	Control Field: UIH command
FCS_DUMMY_OK	Frame Check Sequence (FCS, dummy)
0x7e	Flag Sequence 0111 1110

```
*/
FIELD(SDU_CORRUPTED_RES_EMPTY)
                0x28, 0x00,
                0x00, 0x00,
                0x7e,
                0x03,
                0xff,
                (0XCF),
                0x7e
ENDFIELD(SDU_CORRUPTED_RES_EMPTY, 9)

/* SDU: Multiframe TE->UE (Data, wrong MSC res, 2. wrong MSC res, 4 x Data)
```

0x80, 0x01	SDU length in bit (48 Bytes)

0x00, 0x00	SDU offset in bit
0x7e, (DLCI_A << 2) | 0x03, 0xef, 0x2b, FCS_DUMMY_OK, 0x7e	Data frame "+"
0x7e, 0x03, 0xff, 0xe1, 0x05, (DLCI_A << 2) | 0x03, 0x8f, (0XCF), 0x7e	1. wrong (C/R is set in address field) MSC response
0x7e, 0x03, 0xff, 0xe1, 0x05, (DLCI_A << 2) | 0x03, 0x8f, FCS_DUMMY_OK, 0x7e,	2. wrong (C/R is set in address field) MSC response
0x7e, (DLCI_A << 2) | 0x03, 0xef, 0x63, (0XCF), 0x7e	Data frame "c"
0x7e, (DLCI_A << 2) | 0x03, 0xef, 0x6f, FCS_DUMMY_OK, 0x7e	Data frame "o"
0x7e, (DLCI_A << 2) | 0x03, 0xef, 0x70, FCS_DUMMY_OK, 0x7e	Data frame "p"
0x7e, (DLCI_A << 2) | 0x03, 0xef, 0x73, FCS_DUMMY_OK, 0x7e	Data frame "s"

*/

```
FIELD(SDU_DMSCMSCDDDD_RES_A)
        0x80, 0x01,
        0x00, 0x00,
        0x7e, (DLCI_A << 2) | 0x03, 0xef, 0x2b, (0XCF), 0x7e,
        0x7e, 0x03, 0xff, 0xe1, 0x05, (DLCI_A << 2) | 0x03, 0x8f, (0XCF), 0x7e,
        0x7e, 0x03, 0xff, 0xe1, 0x05, (DLCI_A << 2) | 0x03, 0x8f, (0XCF), 0x7e,
        0x7e, (DLCI_A << 2) | 0x03, 0xef, 0x63, (0XCF), 0x7e,
        0x7e, (DLCI_A << 2) | 0x03, 0xef, 0x6f, (0XCF), 0x7e,
        0x7e, (DLCI_A << 2) | 0x03, 0xef, 0x70, (0XCF), 0x7e,
        0x7e, (DLCI_A << 2) | 0x03, 0xef, 0x73, (0XCF), 0x7e
ENDFIELD(SDU_DMSCMSCDDDD_RES_A, 52)
```

/* SDU: Multiframe TE->UE (wrong MSC res, 2. wrong MSC res)

0x90, 0x00	SDU length in bit (18 Bytes)
0x00, 0x00	SDU offset in bit
0x7e, 0x03, 0xff, 0xe1, 0x05, (DLCI_A << 2) | 0x03, 0x8f, FCS_DUMMY_OK, 0x7e	1. wrong (C/R is set in address field) MSC response
0x7e, 0x03, 0xff, 0xe1, 0x05, (DLCI_A << 2) | 0x03, 0x8d, FCS_DUMMY_OK, 0x7e,	2. wrong (C/R is set in address field) MSC response

*/

```
FIELD(SDU_MSCMSC_RES_A)
        0x90, 0x00,
        0x00, 0x00,
        0x7e, 0x03, 0xff, 0xe1, 0x05, (DLCI_A << 2) | 0x03, 0x8f, (0XCF), 0x7e,
        0x7e, 0x03, 0xff, 0xe1, 0x05, (DLCI_A << 2) | 0x03, 0x8d, (0XCF), 0x7e
ENDFIELD(SDU_MSCMSC_RES_A, 22)
```

/* SDU: wrong MSC response frame TE->UE

0x48, 0x00	SDU length in bit (9 Bytes)
0x00, 0x00	SDU offset in bit
0x7e, 0x03, 0xff, 0xe1, 0x05, (DLCI_A << 2) | 0x03, 0x8f, FCS_DUMMY_OK, 0x7e	1. wrong (C/R is set in address field) MSC response

*/

```
FIELD(SDU_CORRUPT_MSC_RES_A)
        0x48, 0x00,
        0x00, 0x00,
        0x7e,
        0x03,
```

```
        0xff,
        0xe1,
        0x05,
        (DLCI_A << 2) | 0x03,
        0x8d,
        (0XCF),
        0x7e
ENDFIELD(SDU_CORRUPT_MSC_RES_A, 13)

/*
```

1.4 Declarations

```
*/
DECLARATION (DTI_CHANNELNAME_PPP)
DECLARATION (DTI_CHANNELNAME_ACI)
DECLARATION(COMPAR_57600_8_E_1_HW)
DECLARATION(COMPAR_57600_8_N_1_HW)
DECLARATION(COMPAR_19200_8_N_1_HW)
DECLARATION(COMPAR_9600_8_N_1_HW)
DECLARATION(COMPAR_XON_CTRL_XOFF_FCS)

/* Array declaration */
DECLARATION ( DTI_PARAMETER_FRAME )
DECLARATION ( DTI_PARAMETER_FRAME_BREAK )
DECLARATION ( DTI_PARAMETER_UART_OUT )
DECLARATION ( DTI_PARAMETER_PEER_A )
DECLARATION ( DTI_PARAMETER_PEER_B )

DECLARATION ( DTI_PARA_ST_LINES_BREAK_OFF)
DECLARATION ( DTI_PARA_ST_LINES_BREAK_ON)


/*
Service Data Units and Content
*/
DECLARATION(STRUCT_HELLO1_SDU)
DECLARATION(STRUCT_HELLO2_SDU)
DECLARATION(STRUCT_HELLO3_SDU)
DECLARATION(STRUCT_H_SDU)
DECLARATION(STRUCT_E_SDU)
DECLARATION(STRUCT_L_SDU)
DECLARATION(STRUCT_O_SDU)
DECLARATION(STRUCT_EMPTY_SDU)

/*
```

1.5 Arrays

```
*/
/* DTI Communication Channel Name PPP
```

0x50, 0x50, 0x50, 0x00, 0x00, 0x0	name value (“PPP\0”)

```
*/
BEGINARRAY(DTI_CHANNELNAME_PPP, 6)
                0x50, 0x50, 0x50, 0x00, 0x00, 0x0
```

```
ENDARRAY

/* DTI Communication Channel Name ACI
```

0x41, 0x43, 0x44, 0x00, 0x00, 0x0	name value (“ACI\0”)

```
*/
BEGINARRAY(DTI_CHANNELNAME_ACI, 6)
                0x41, 0x43, 0x44, 0x00, 0x00, 0x0
ENDARRAY

/*
```

1.6 Structs

```
*/
/*
```

1.6.1 Communication Parameters

```
*/
BEGIN_PSTRUCT("comPar", COMPAR_57600_8_E_1_HW)
        SET_COMP("speed", COMPAR_SPEED_57600)
        SET_COMP("bpc", COMPAR_BPC_8)
        SET_COMP("nsb", COMPAR_NSB_1)
        SET_COMP("parity", COMPAR_PARITY_EVEN)
        SET_COMP("flow_rx", UART_IO_FC_RX_RTS)
        SET_COMP("flow_tx", UART_IO_FC_TX_RTS)
        SET_COMP("xon_valid", UART_IO_XON_VALID)
        SET_COMP("xon", COMPAR_XON_CHAR)
        SET_COMP("xoff_valid", UART_IO_XOFF_VALID)
        SET_COMP("xoff", COMPAR_XOFF_CHAR)
        SET_COMP("esc_valid", UART_IO_ESC_VALID)
        SET_COMP("esc_char", ESC_SEQ_CHAR_PLUS)
        SET_COMP("esc_gp", GUARD_PERIOD_1000)
ENDSTRUCT

/* set XON and XOFF to important values of Control field and FCS field */

BEGIN_PSTRUCT("comPar", COMPAR_XON_CTRL_XOFF_FCS)
        SET_COMP("speed", UART_IO_SPEED_UNDEF)
        SET_COMP("bpc", UART_IO_BPC_UNDEF)
        SET_COMP("nsb", UART_IO_SB_UNDEF)
        SET_COMP("parity", UART_IO_PA_UNDEF)
        SET_COMP("flow_rx", UART_IO_FC_RX_UNDEF)
        SET_COMP("flow_tx", UART_IO_FC_TX_UNDEF)
        SET_COMP("xon_valid", UART_IO_XON_VALID)
        SET_COMP("xon", 0xef)
        SET_COMP("xoff_valid", UART_IO_XOFF_VALID)
        SET_COMP("xoff", (0XCF))
        SET_COMP("esc_valid", UART_IO_ESC_UNDEF)
        SET_COMP("esc_char", UART_IO_ESC_CHAR_DEFAULT)
        SET_COMP("esc_gp", UART_IO_ESC_GP_DEFAULT)
ENDSTRUCT

BEGIN_PSTRUCT("comPar", COMPAR_57600_8_N_1_HW)
        SET_COMP("speed", COMPAR_SPEED_57600)
        SET_COMP("bpc", COMPAR_BPC_8)
```

```
            SET_COMP("nsb", COMPAR_NSB_1)
            SET_COMP("parity", COMPAR_PARITY_NO)
            SET_COMP("flow_rx", UART_IO_FC_RX_RTS)
            SET_COMP("flow_tx", UART_IO_FC_TX_RTS)
            SET_COMP("xon_valid", UART_IO_XON_VALID)
            SET_COMP("xon", COMPAR_XON_CHAR)
            SET_COMP("xoff_valid", UART_IO_XOFF_VALID)
            SET_COMP("xoff", COMPAR_XOFF_CHAR)
            SET_COMP("esc_valid", UART_IO_ESC_VALID)
            SET_COMP("esc_char", ESC_SEQ_CHAR_PLUS)
            SET_COMP("esc_gp", GUARD_PERIOD_1000)
ENDSTRUCT

BEGIN_PSTRUCT("comPar", COMPAR_9600_8_N_1_HW)
            SET_COMP("speed", COMPAR_SPEED_9600)
            SET_COMP("bpc", COMPAR_BPC_8)
            SET_COMP("nsb", COMPAR_NSB_1)
            SET_COMP("parity", COMPAR_PARITY_NO)
            SET_COMP("flow_rx", UART_IO_FC_RX_RTS)
            SET_COMP("flow_tx", UART_IO_FC_TX_RTS)
            SET_COMP("xon_valid", UART_IO_XON_VALID)
            SET_COMP("xon", COMPAR_XON_CHAR)
            SET_COMP("xoff_valid", UART_IO_XOFF_VALID)
            SET_COMP("xoff", COMPAR_XOFF_CHAR)
            SET_COMP("esc_valid", UART_IO_ESC_VALID)
            SET_COMP("esc_char", ESC_SEQ_CHAR_PLUS)
            SET_COMP("esc_gp", GUARD_PERIOD_1000)
ENDSTRUCT

BEGIN_PSTRUCT("comPar", COMPAR_19200_8_N_1_HW)
            SET_COMP("speed", COMPAR_SPEED_19200)
            SET_COMP("bpc", COMPAR_BPC_8)
            SET_COMP("nsb", COMPAR_NSB_1)
            SET_COMP("parity", COMPAR_PARITY_NO)
            SET_COMP("flow_rx", UART_IO_FC_RX_RTS)
            SET_COMP("flow_tx", UART_IO_FC_TX_RTS)
            SET_COMP("xon_valid", UART_IO_XON_VALID)
            SET_COMP("xon", COMPAR_XON_CHAR)
            SET_COMP("xoff_valid", UART_IO_XOFF_VALID)
            SET_COMP("xoff", COMPAR_XOFF_CHAR)
            SET_COMP("esc_valid", UART_IO_ESC_VALID)
            SET_COMP("esc_char", ESC_SEQ_CHAR_PLUS)
            SET_COMP("esc_gp", GUARD_PERIOD_1000)
ENDSTRUCT

/* Parameters for sending DTI primitives */

BEGIN_PSTRUCT ("st_lines", DTI_PARA_ST_LINES_BREAK_OFF)
        SET_COMP ("st_flow",                DTI_FLOW_ON)
            SET_COMP ("st_line_sa",         DTI_SA_ON)
            SET_COMP ("st_line_sb",         DTI_SB_ON)
            SET_COMP ("st_break_len", DTI_BREAK_OFF)
ENDSTRUCT

BEGIN_PSTRUCT ("st_lines", DTI_PARA_ST_LINES_BREAK_ON)
        SET_COMP ("st_flow",                DTI_FLOW_ON)
```

```
        SET_COMP ("st_line_sa",         DTI_SA_ON)
        SET_COMP ("st_line_sb",         DTI_SB_ON)
        SET_COMP ("st_break_len", UART_BREAK_LEN)
ENDSTRUCT

BEGIN_PSTRUCT ("parameters", DTI_PARAMETER_FRAME)
        SET_COMP ("p_id",       DTI_PID_UOS)
        SET_COMP ("st_lines",   DTI_PARA_ST_LINES_BREAK_OFF)
ENDSTRUCT
BEGIN_PSTRUCT ("parameters", DTI_PARAMETER_FRAME_BREAK)
        SET_COMP ("p_id",       DTI_PID_UOS)
        SET_COMP ("st_lines",   DTI_PARA_ST_LINES_BREAK_ON)
ENDSTRUCT
BEGIN_PSTRUCT ("parameters", DTI_PARAMETER_UART_OUT)
        SET_COMP ("p_id",       DTI_PID_UART_OUT)
        SET_COMP ("st_lines",   DTI_PARA_ST_LINES_BREAK_OFF)
ENDSTRUCT
BEGIN_PSTRUCT ("parameters", DTI_PARAMETER_PEER_A)
        SET_COMP ("p_id",       P_ID_PEER_A)
        SET_COMP ("st_lines",   DTI_PARA_ST_LINES_BREAK_OFF)
ENDSTRUCT
BEGIN_PSTRUCT ("parameters", DTI_PARAMETER_PEER_B)
        SET_COMP ("p_id",       P_ID_PEER_B)
        SET_COMP ("st_lines",   DTI_PARA_ST_LINES_BREAK_OFF)
ENDSTRUCT

/*
```

1.6.2 Service Data Units

```
*/
/* String Hello1 SDU
```

0x40	SDU length in bit (8 Byte = 64 Bit)
0x00	SDU offset in bit
STRING_HELLO1	String "Hello1\n"

```
*/
BEGIN_PSTRUCT("sdu", STRUCT_HELLO1_SDU)
        SET_COMP("l_buf", LEN_HELLO)
        SET_COMP("o_buf", OFFSET_0)
        SET_COMP("buf", STRING_HELLO1)
ENDSTRUCT

/* String Hello2 SDU
```

0x40	SDU length in bit (8 Byte = 64 Bit)
0x00	SDU offset in bit
STRING_HELLO2	String "Hello2\n"

```
*/
BEGIN_PSTRUCT("sdu", STRUCT_HELLO2_SDU)
        SET_COMP("l_buf", 0x40)
        SET_COMP("o_buf", 0x00)
        SET_COMP("buf", STRING_HELLO2)
ENDSTRUCT

/* String Hello3 SDU
```

0x40	SDU length in bit (8 Byte = 64 Bit)
0x00	SDU offset in bit
STRING_HELLO3	String "Hello3\n"

*/

```
BEGIN_PSTRUCT("sdu", STRUCT_HELLO3_SDU)
        SET_COMP("l_buf", 0x40)
        SET_COMP("o_buf", 0x00)
        SET_COMP("buf", STRING_HELLO3)
ENDSTRUCT
```

/* String h SDU

0x08	SDU length in bit (1 Byte = 8 Bit)
0x00	SDU offset in bit
0x68	String "h"

*/

```
BEGIN_PSTRUCT("sdu", STRUCT_H_SDU)
        SET_COMP("l_buf", 0x08)
        SET_COMP("o_buf", 0x00)
        SET_COMP("buf", STRING_H)
ENDSTRUCT
```

/* String e SDU

0x08, 0x00	SDU length in bit (1 Byte = 8 Bit)
0x00, 0x00	SDU offset in bit
0x65	String "e"

*/

```
BEGIN_PSTRUCT("sdu", STRUCT_E_SDU)
        SET_COMP("l_buf", 0x08)
        SET_COMP("o_buf", 0x00)
        SET_COMP("buf", STRING_E)
ENDSTRUCT
```

/* String l SDU

0x08, 0x00	SDU length in bit (1 Byte = 8 Bit)
0x00, 0x00	SDU offset in bit
0x6c	String "l"

*/

```
BEGIN_PSTRUCT("sdu", STRUCT_L_SDU)
        SET_COMP("l_buf", 0x08)
        SET_COMP("o_buf", 0x00)
        SET_COMP("buf", STRING_L)
ENDSTRUCT
```

/* String o SDU

0x08, 0x00	SDU length in bit (1 Byte = 8 Bit)
0x00, 0x00	SDU offset in bit
0x6f	String "o"

*/

```
BEGIN_PSTRUCT("sdu", STRUCT_O_SDU)
        SET_COMP("l_buf", 0x08)
        SET_COMP("o_buf", 0x00)
```

```
        SET_COMP("buf", STRING_O)
ENDSTRUCT

/* Empty String SDU
```

0x00, 0x00	SDU length in bit (0 Byte = 0 Bit)
0x00, 0x00	SDU offset in bit

```
*/
BEGIN_PSTRUCT("sdu", STRUCT_EMPTY_SDU)
        SET_COMP("l_buf", 0x00)
        SET_COMP("o_buf", 0x00)
        SKIP_COMP("buf")
ENDSTRUCT
```

2 TEST CASES

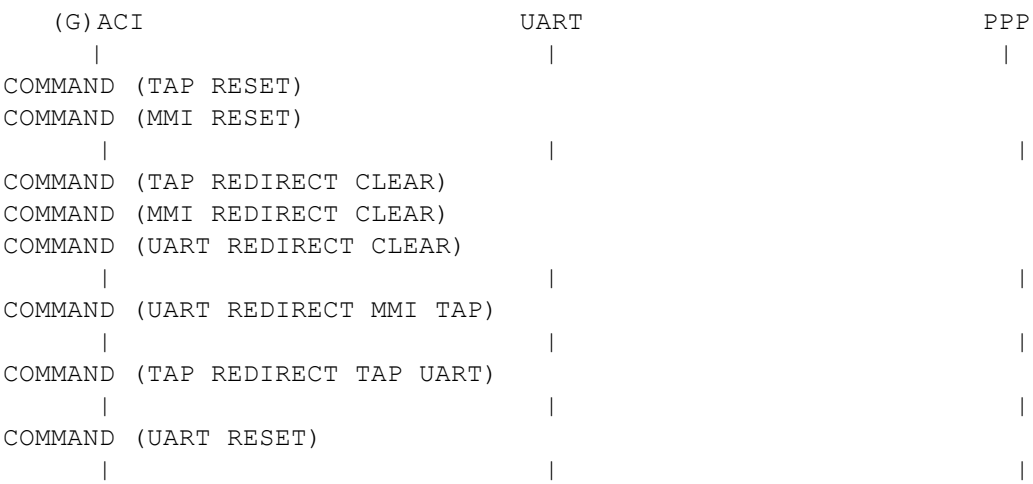
2.1 Setup (UART001 - UART010)

2.1.1 UART001: TAP Setup

Description: TAP commands in order to do the setup for all following test cases.

Preamble:

None



Parametrization:

Primitive	Parameter	Value

History:

20-Sep-2000	STW	Initial

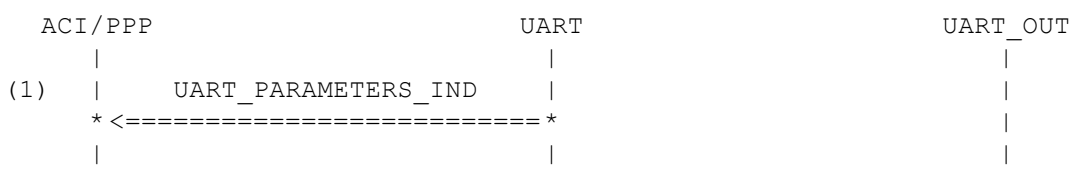
2.1.2 UART002: Entity setup

Description:

In non-simulation mode, UART sends a UART_PARAMETERS_IND to ACI in pei_init(). Since this cannot be monitored in simulation mode, just do nothing here. Another possibility would be to send the UART_PARAMETERS_IND in simulation mode after reception of an according config primitive.

Preamble:

UART001



Parametrization:

	Primitive	Parameter	Value
(1)	UART_PARAMETERS_IND		
		uart_instances	UART_INSTANCES

History:

12-Mar-2002	TVO	Initial

2.2 Configuration (UART011 – UART019)

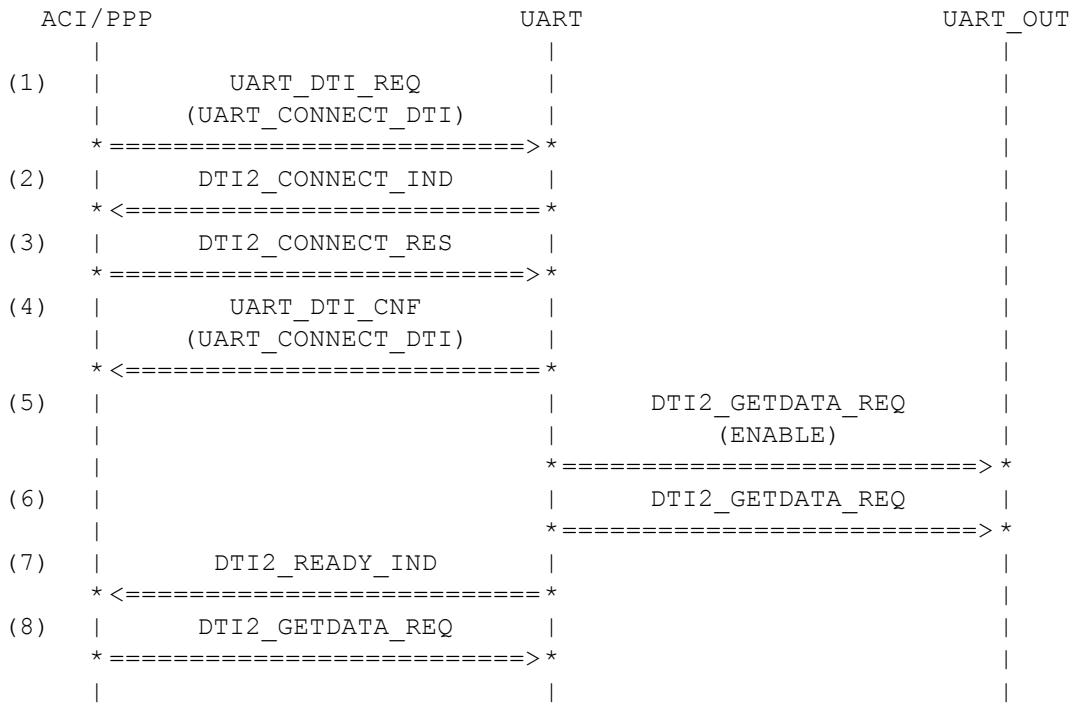
2.2.1 UART011: Setup Connection to PPP (default parameters)

Description:

DTI connection to PPP without any changes of parameters and escape sequence. The RX service issues a GETDATA request in order to signal it is ready to receive to the UART_OUT interface. The DTI connection is confirmed towards the ACI. Then the DTX service initialises the DTI interface flow control by sending an empty primitive to the PPP entity and sending a DTI2_READY_IND.

Preamble:

UART002



Parametrization:

	Primitive	Parameter	Value
(1)	UART_DTI_REQ		
		dti_conn	UART_CONNECT_DTI
		device	UART_DEVICE
		dlci	DLCI_ANY
		direction	FALSE

	link_id	LINK_ID_1
	entity_name	DTI_CHANNELNAME_PPP
(2) DTI2_CONNECT_IND		
	link_id	LINK_ID_1
	version	DTI_VERSION_10
(3) DTI2_CONNECT_RES		
	link_id	LINK_ID_1
	version	DTI_VERSION_10
(4) UART_DTI_CNF		
	dti_conn	UART_CONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_ANY
(5) DTI2_GETDATA_REQ		
	link_id	LINK_ENABLE_PORT_1
(6) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(7) DTI2_READY_IND		
	link_id	LINK_ID_1
(8) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1

History:

20-Sep-2000	STW	Initial
1-Mar-2001	LW	UART_OUT as lower layer
25-Apr-2001	STW	add Enable primitive
11-Jun-2001	STW	add DTI_GETDATA_REQ primitive
08-Oct-2001	TVO	adapted for use with DTI2

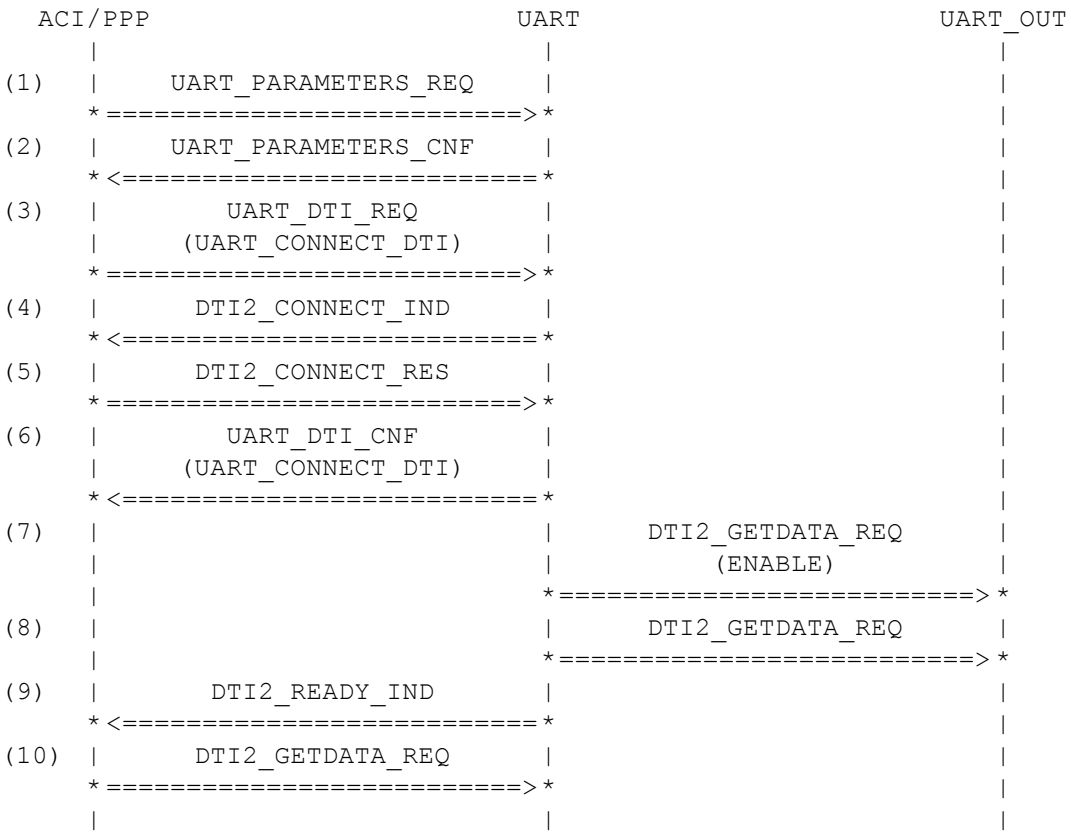
2.2.2 UART012: Setup Connection to PPP (explicit parameter setting)

Description:

DTI connection to PPP with 57600 baud, 8 bpc, even parity, 1 stopbit and RTS/CTS flow control. The rest is analogous to test case UART012.

Preamble:

UART002



Parametrization:

Primitive	Parameter	Value
(1) UART_PARAMETERS_REQ		
	device	UART_DEVICE
	comPar	COMPAR_57600_8_E_1_HW
(2) UART_PARAMETERS_CNF		
	device	UART_DEVICE
(3) UART_DTI_REQ		
	dti_conn	UART_CONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_ANY
	direction	FALSE
	link_id	LINK_ID_1
	entity_name	DTI_CHANNELNAME_PPP
(4) DTI2_CONNECT_IND		
	link_id	LINK_ID_1
	version	DTI_VERSION_10
(5) DTI2_CONNECT_RES		
	link_id	LINK_ID_1
	version	DTI_VERSION_10
(6) UART_DTI_CNF		
	dti_conn	UART_CONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_ANY

(7) DTI2_GETDATA_REQ		
	link_id	LINK_ENABLE_PORT_1
(8) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(9) DTI2_READY_IND		
	link_id	LINK_ID_1
(10) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1

History:

25-Sep-2000	STW	Initial
01-Mar-2001	LW	Adapted to UART_OUT
11-Jun-2001	STW	add DTI_GETDATA_REQ primitive
08-Oct-2001	TVO	adapted for use with DTI2

2.3 Data Transmission - Sending (UART020 - UART029)

2.3.1 UART020: Send Hello Strings

Description:

Send Strings Hello1, Hello2 and Hello3 in non-multiplexer mode: The UART entity receives the strings from PPP. Each string is then sent on the UART_OUT test interface. A DTI2_READY_IND on the test interface is forwarded to PPP.

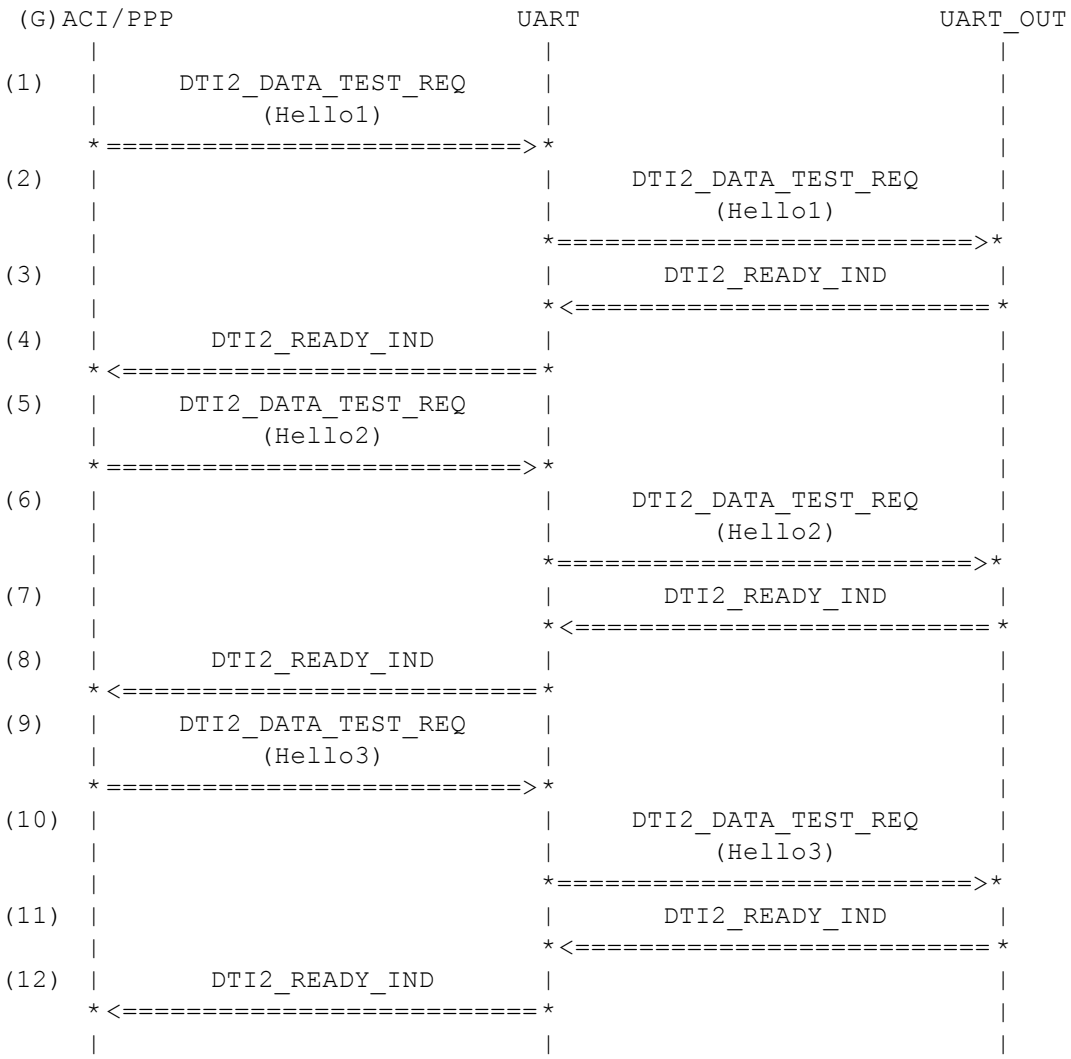
The Variant A does a setup with default parameters, variant B sets explicit parameters and variant C has an multiplexer mode establishment failure in the preamble.

Variants:

<A>....<C>

Preamble:

<A>UART011
<B>UART012
<C>UART102



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_HELLO1_SDU
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_HELLO1_SDU
(3) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(4) DTI2_READY_IND		
	link_id	LINK_ID_1
(5) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_1

	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_HELLO2_SDU
(6) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_HELLO2_SDU
(7) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(8) DTI2_READY_IND		
	link_id	LINK_ID_1
(9) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_HELLO3_SDU
(10) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_HELLO3_SDU
(11) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(12) DTI2_READY_IND		
	link_id	LINK_ID_1

History:

28-Mar-2001	LW	Initial
08-Oct-2001	TVO	adapted for use with DTI2

2.3.2 UART021: Send Hello Strings - Ignore String with Invalid Channel Id

Description:

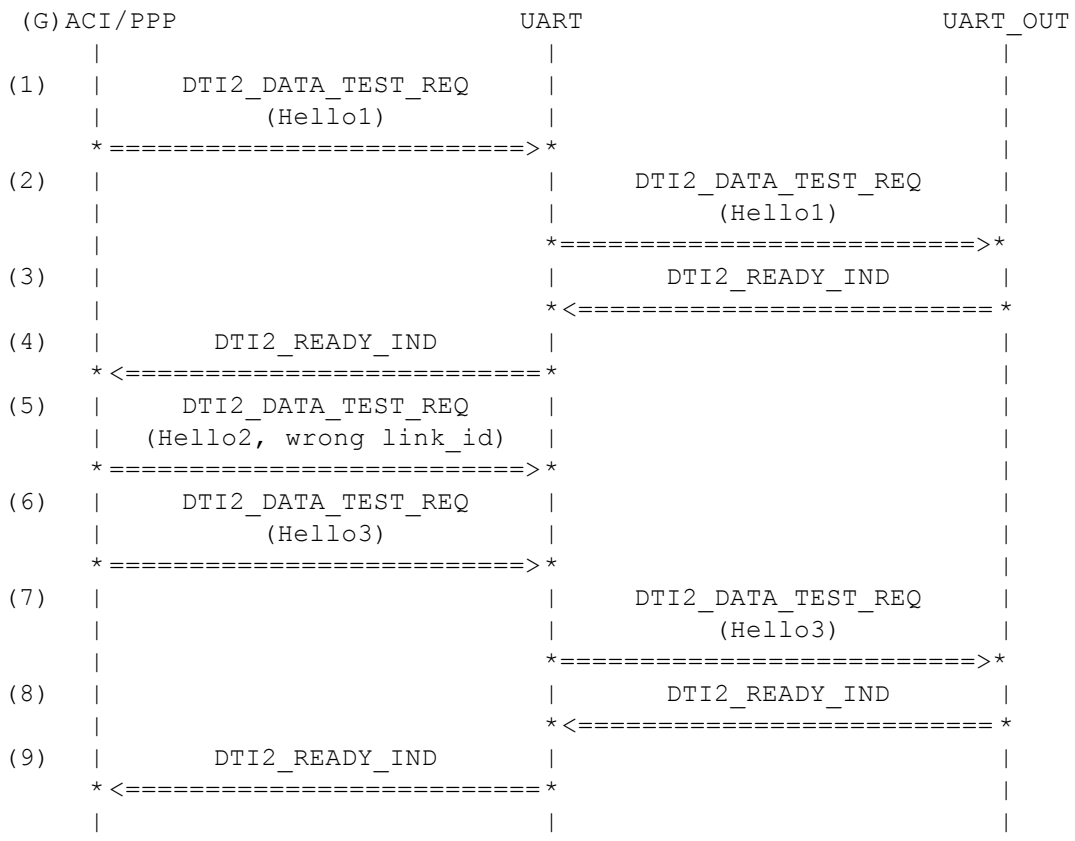
Send Strings Hello1, Hello2 and Hello3 in non-multiplexer mode: The UART entity receives the strings from PPP. The PPP entity has sent the Hello2 string on the DTI interface with an errorneous channel id. Therefore the Hello2 string is ignored since in non-multiplex mode there is only one valid channel id. The two correcty received strings Hello1 and Hello3 are then sent on the UART_OUT test interface. A DTI2_READY_IND on the test interface is forwarded to PPP.

Variants:

<A>....<B>

Preamble:

<A>UART011
<B>UART012



Parametrization:

	Primitive	Parameter	Value
(1)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_HELLO1_SDU
(2)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_HELLO1_SDU
(3)	DTI2_READY_IND		
		link_id	LINK_UART_OUT_PORT_1
(4)	DTI2_READY_IND		
		link_id	LINK_ID_1
(5)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_2
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_HELLO2_SDU
(6)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_HELLO3_SDU

(7)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_HELLO3_SDU
(8)	DTI2_READY_IND		
		link_id	LINK_UART_OUT_PORT_1
(9)	DTI2_READY_IND		
		link_id	LINK_ID_1

History:

6-Mar-2001	LW	Initial
08-Oct-2001	TVO	adapted for use with DTI2

2.4 Data Transmission - Receiving (UART030 - UART039)

2.4.1 UART030: Receive One Letter String

Description:

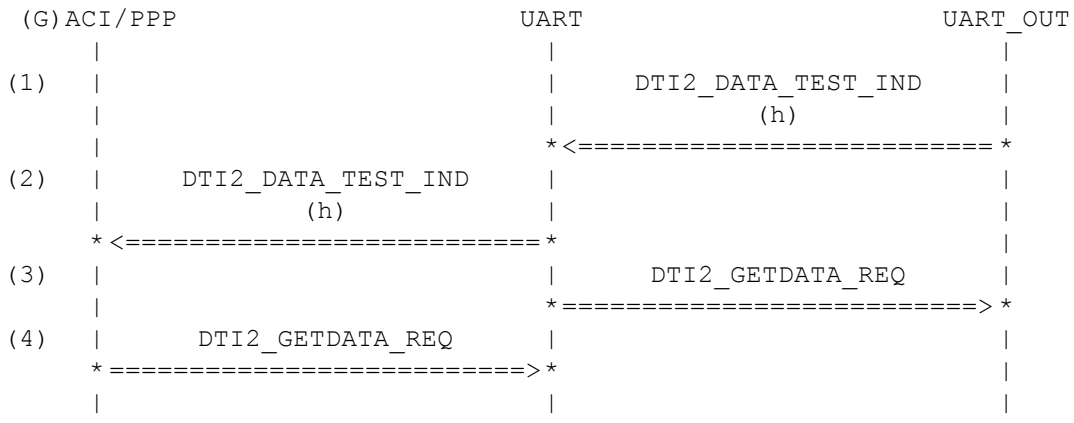
The UART entity receives String "h" via the UART_OUT test interface and forwards them to PPP. In variant A the DTI connection was started normally in variant B it was started after a multiplexer setup failure.

Variants:

<A>....<C>

Preamble:

<A>	UART011
<B>	UART020B
<C>	UART020C



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU

(2)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_H_SDU
(3)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(4)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1

History:

28-Feb-2001	LW	Initial
11-Jun-2001	STW	remove DTI_GETDATA_REQ primitive
08-Oct-2001	TVO	adapted for use with DTI2

2.4.2 UART031: Receive String (Size > N1)

Description:

The UART entity receives 15 times the String "Hello1" via the UART_OUT test interface. Since the parameter N1 of the DTX service is initially configured to 10 in non multiplexer simulation mode, this string is forwarded in pieces of 30/40 bytes to the PPP entity.

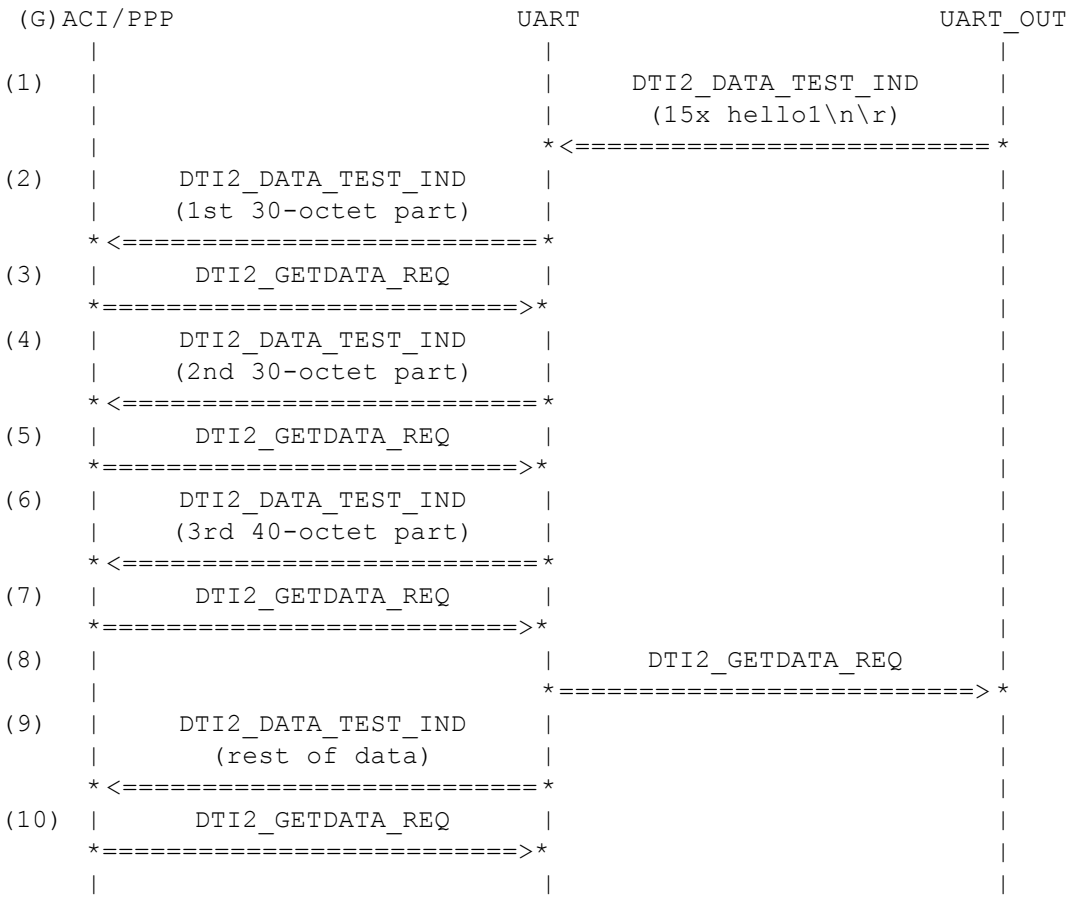
Note: In variant A the DTI connection was started normally in variant B it was started normally and some data was transmitted before receiving and in variant C it was started after a multiplexer setup failure.

Variants:

<A>....<C>

Preamble:

<A>	UART011
<B>	UART020B
<C>	UART020C



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_15XHELLO1_SDU
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_1ST30HELLO1_SDU
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_2ND30HELLO1_SDU
(5) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_3RD40HELLO1_SDU

(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(8) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(9) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_RESTHALLO1_SDU
(10) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1

History:

28-Feb-2001	LW	Initial
11-Jun-2001	STW	remove DTI_GETDATA_REQ primitive
28-Aug-2001	STW	extend to N1=10 octets
11-Sep-2001	STW	extend to minimum size_multiplier=3
11-Oct-2001	TVO	adapted for use with DTI2

2.5 Data Transmission - Mixed Sending/Receiving (UART040 - UART049)

2.5.1 UART040: Mixed Receiving/Sending One Letter Strings

Description:

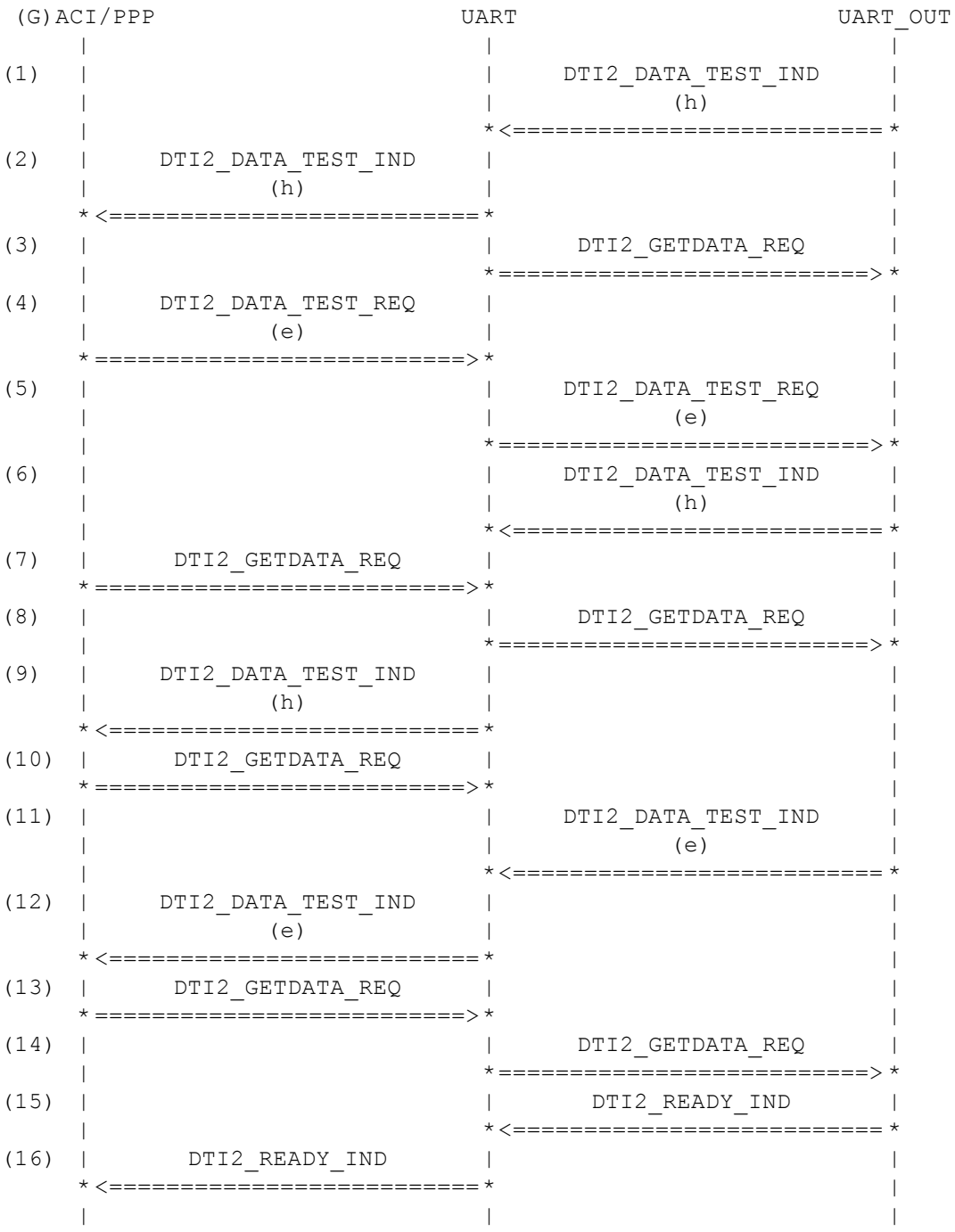
The UART entity receives and sends one letter strings. (In variant A it is a completely new DTI connection in variant B some receiving/sending of data was done before on the connection.)

Variants:

<A>....<B>

Preamble:

<A>	UART012
<B>	UART031C



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU

(3)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(4)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_E_SDU
(5)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_E_SDU
(6)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_H_SDU
(7)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1
(8)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(9)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_H_SDU
(10)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1
(11)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_E_SDU
(12)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_E_SDU
(13)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1
(14)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(15)	DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(16)	DTI2_READY_IND		
		link_id	LINK_ID_1

History:

06-Mar-2001	LW	Initial
11-Jun-2001	STW	move DTI_GETDATA_REQ to testcase 012
11-Oct-2001	TVO	adapted for use with DTI2

2.5.2 UART041: Mixed Receiving/Sending Multi Letter Strings

Description:

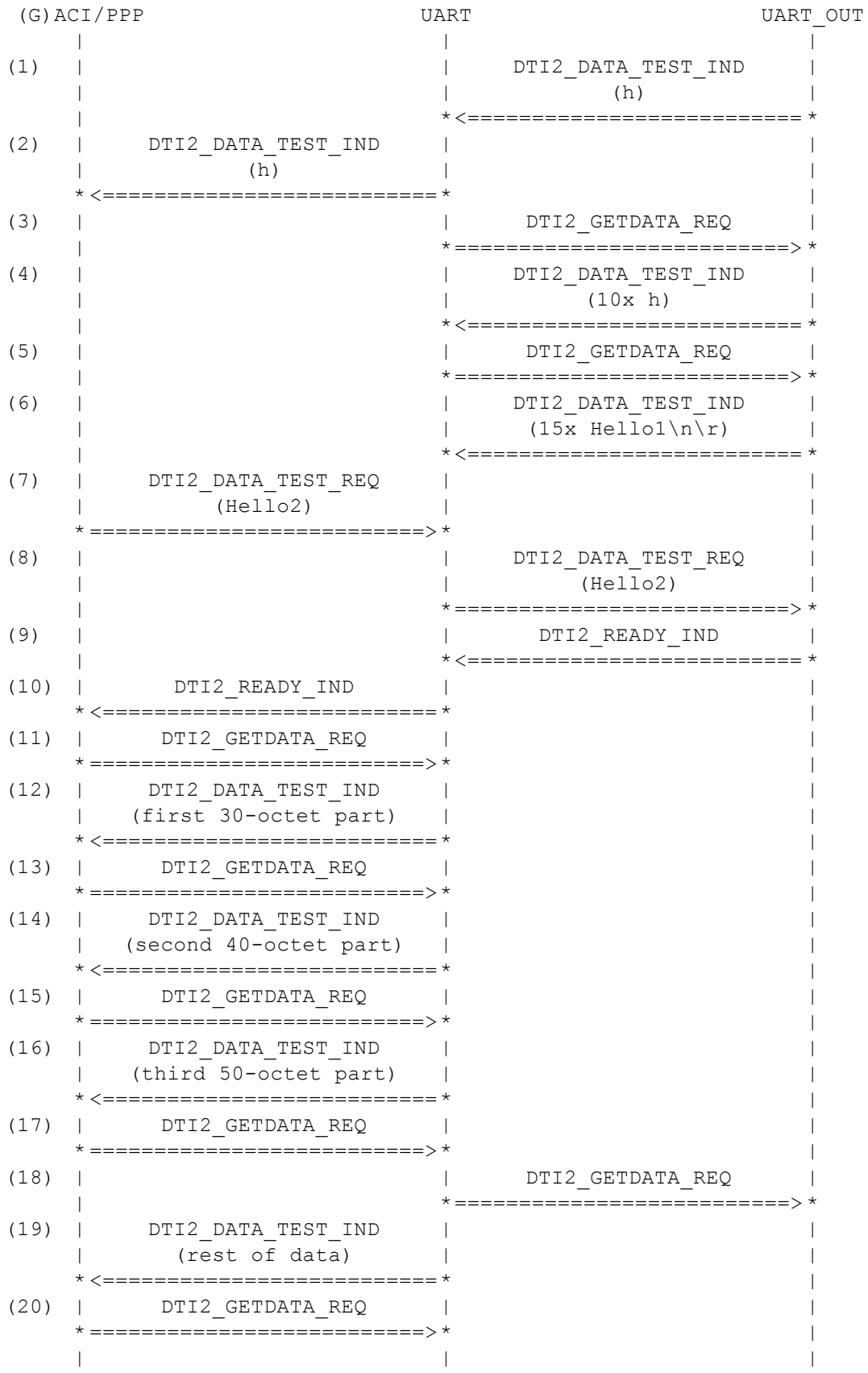
The UART entity receives and sends strings with a larger number of bytes than in UART040. Since the test case UART040 is in the preamble, the buffer size of the receive descriptor already has been increased once. Therefore the data flow is not disabled after receiving a one letter string.

Variants:

<A>....<B>

Preamble:

<A> UART040A
<B> UART040B



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_10H_SDU
(5) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_15XHELLO1_SDU
(7) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_HELLO2_SDU
(8) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_HELLO2_SDU
(9) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(10) DTI2_READY_IND		
	link_id	LINK_ID_1
(11) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(12) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_1ST30HHELLO1_SDU
(13) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(14) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_2ND40HHALLO1_SDU
(15) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1

(16)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_3RD50HHALLO1_SDU
(17)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1
(18)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(19)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_RESTHHALLO1_SDU
(20)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1

History:

6-Mar-2001	LW	Initial
11-Sep-2001	STW	extende to N1=10 octets and minimum size_multiplier=3
11-Oct-2001	TVO	adapted for use with DTI2

3 Multiplexer Tests (UART100 - UART200)

3.1 Setup/End Multiplexer Mode

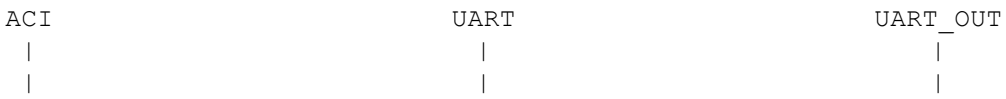
3.1.1 UART100: Additional Setup for Multiplexer Tests

Description:

This test case does additional parameter setup which is only required in multiplexer tests. Currently there is nothing to do - it only provides free space where parameter setups/primitive can be put later which are to be used in every multiplexer test.

Preamble:

UART002



Parametrization:

Primitive	Parameter	Value

History:

16-Jan-2001		LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2	

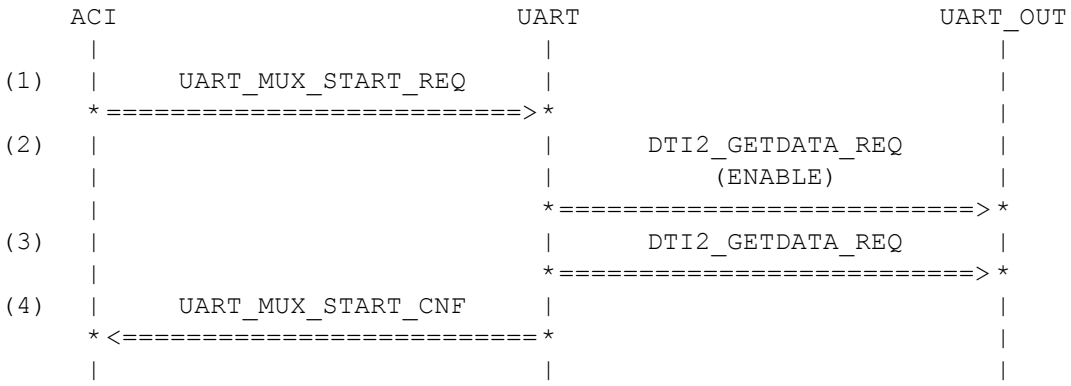
3.1.2 UART101: ACI Prepares UART to Start Multiplexer

Description:

The (G)ACI starts the TS 27.010 multiplexer operation over a serial link. Only advanced mode with UIH frames ares supported. The UART starts the timer T3 to time waiting for the SABM frame.

Preamble:

UART002



Parametrization:

Primitive	Parameter	Value

(1)	UART_MUX_START_REQ		
		device	UART_DEVICE
		mode	UART_MUX_MODE_ADVANCED
		frame_type	UART_MUX_FRAME_UIH
		n1	UART_MUX_N1_ADVANCED_DEFAULT
		t1	UART_MUX_T1_DEFAULT_TEST
		n2	UART_MUX_N2_DEFAULT
		t2	UART_MUX_T2_DEFAULT_TEST
		t3	UART_MUX_T3_DEFAULT_TEST
(2)	DTI2_GETDATA_REQ		
		link_id	LINK_ENABLE_PORT_1
(3)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(4)	UART_MUX_START_CNF		
		device	UART_DEVICE

History:

16-Jan-2001	LW	Initial
11-Jun-2001	STW	add UART_PARAMETERS_CNF primitive
11-Oct-2001	TVO	adapted for use with DTI2

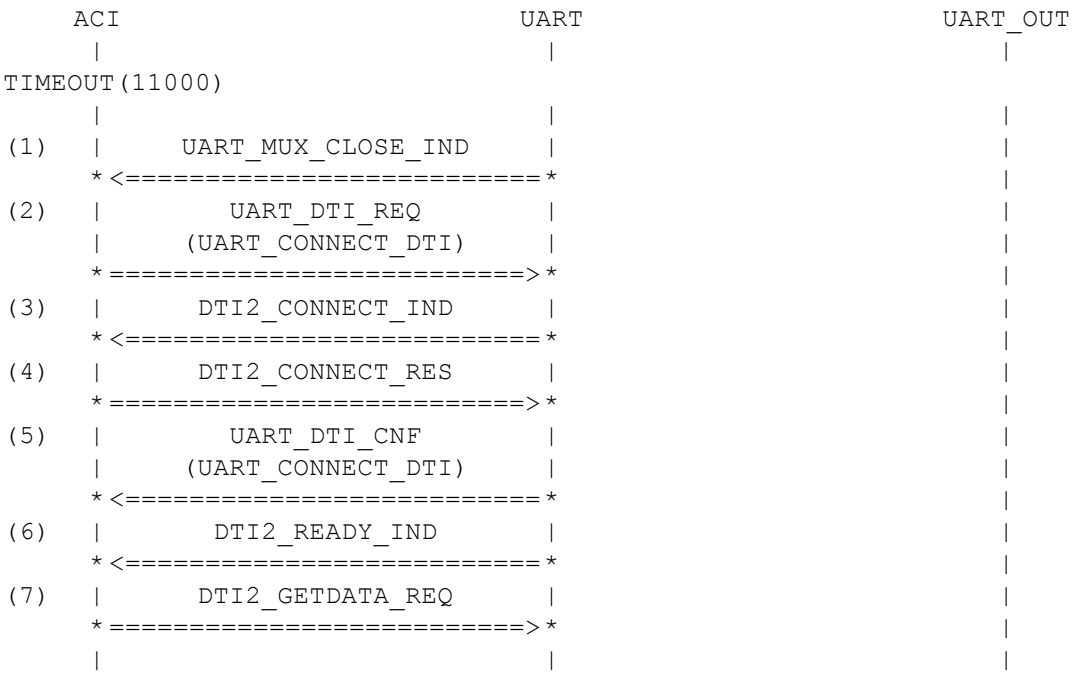
3.1.3 UART102: Link Establishment Fails (Timeout T3)

Description:

The timer T3 expires. UART informs ACI that multiplexer operation establishment has failed and the UART returns to normal AT mode.

Preamble:

UART101



Parametrization:

Primitive	Parameter	Value
(1) UART_MUX_CLOSE_IND		
	device	UART_DEVICE
(2) UART_DTI_REQ		
	dti_conn	UART_CONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_ANY
	direction	FALSE
	link_id	LINK_ID_1
	entity_name	UART_PEER_NAME_DUMMY
(3) DTI2_CONNECT_IND		
	link_id	LINK_ID_1
	version	DTI_VERSION_10
(4) DTI2_CONNECT_RES		
	link_id	LINK_ID_1
	version	DTI_VERSION_10
(5) UART_DTI_CNF		
	dti_conn	UART_CONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_ANY
(6) DTI2_READY_IND		
	link_id	LINK_ID_1
(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1

History:

18-Jan-2001	LW	Initial
11-Jun-2001	STW	add DTI_GETDATA_REQ primitive
11-Oct-2001	TVO	adapted for use with DTI2

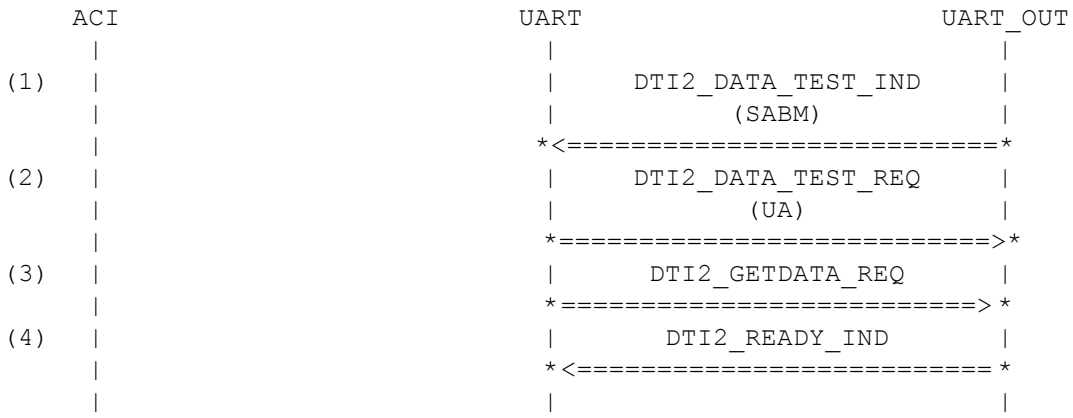
3.1.4 UART103: Reception of SABM for Multiplexer Control Channel

Description:

The terminal equipment sends a SABM frame for the multiplexer control channel. The UART stops its timer T3 and answers with a positive UA frame for the control channel (DLCI 0).

Preamble:

UART101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MUX_START0
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_UA_UE2TE
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1

History:

18-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

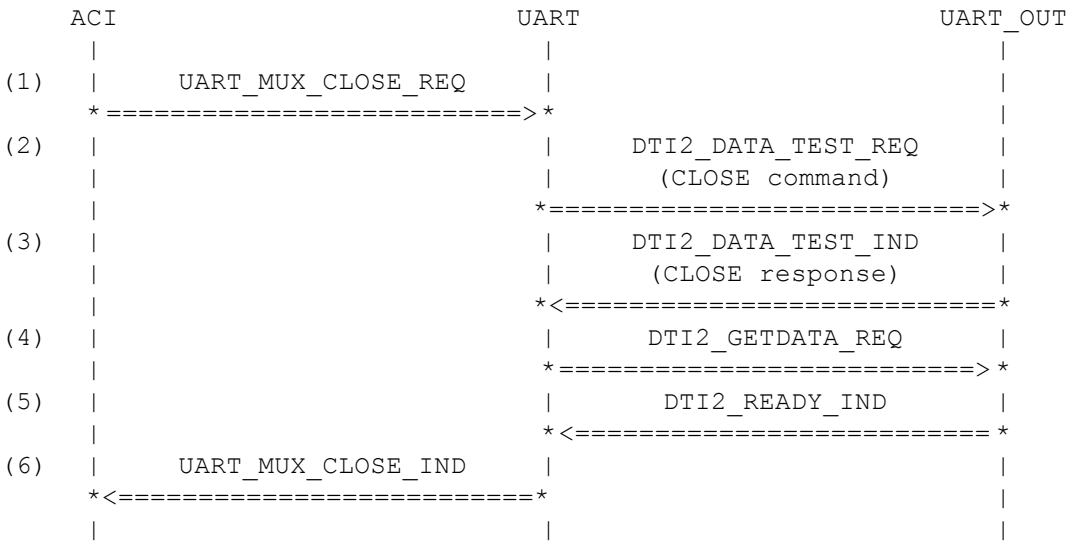
3.1.5 UART105: Multiplexer Close-down (UE initiated)

Description:

The ACI requests termination of the multiplexer operation on the link. The peer is informed by sending a multiplexer close down command. A multiplexer close down response is received and confirms the successful reset of the link into normal AT command mode.

Preamble:

UART103



Parametrization:

	Primitive	Parameter	Value
(1)	UART_MUX_CLOSE_REQ		
		device	UART_DEVICE
(2)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CMD_UE2TE_CLOSE
(3)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_RES_TE2UE_CLOSE
(4)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(5)	DTI2_READY_IND		
		link_id	LINK_UART_OUT_PORT_1
(6)	UART_MUX_CLOSE_IND		
		device	UART_DEVICE

History:

17-Jan-2001		LW	Initial
8-Mar-2001		LW	Testcase now expects DTI_GETDATA_REQ
11-Oct-2001	TVO	adapted for use with DTI2	

3.1.6 UART106: Multiplexer Close-down (TE initiated)

Description:

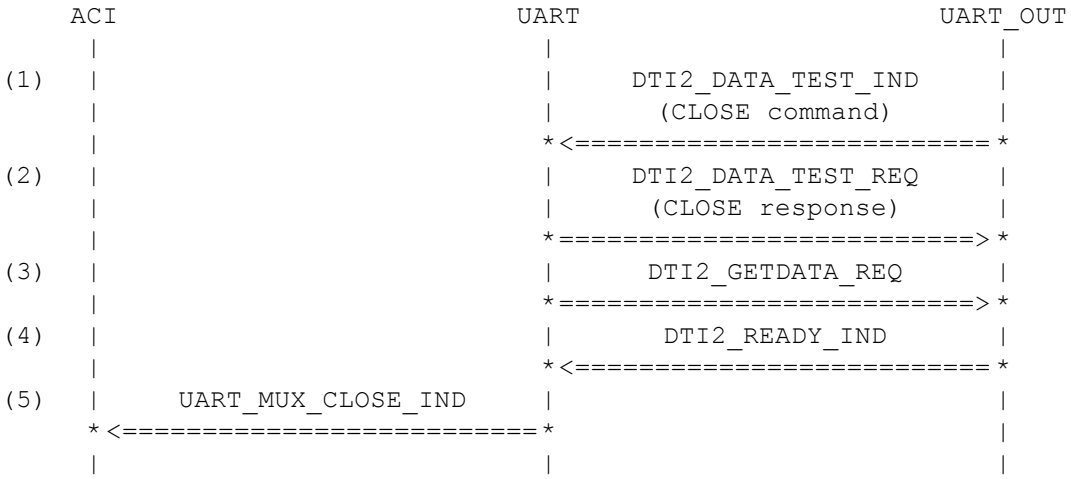
The TE requests termination of the multiplexer operation on the link. The MMI is informed and a response is sent to the TE. There a two possibilities in which the TE can request close down of the multiplexer: The first one (variant A) is by using a multiplexer close down command/response and the second one (variant B) is using a DISC command/response with DLCI 0.

Variants:

<A>....<B>

Preamble:

UART103



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
<A>	sdu	SDU_CMD_TE2UE_CLOSE0
<B>	sdu	SDU_CMD_TE2UE_CLOSE1
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
<A>	sdu	SDU_RES_UE2TE_CLOSE0
<B>	sdu	SDU_UA_UE2TE
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_READY_IND	link_id	LINK_WRITEDATA_PORT_1
(5) UART_MUX_CLOSE_IND		
	device	UART_DEVICE

History:

17-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

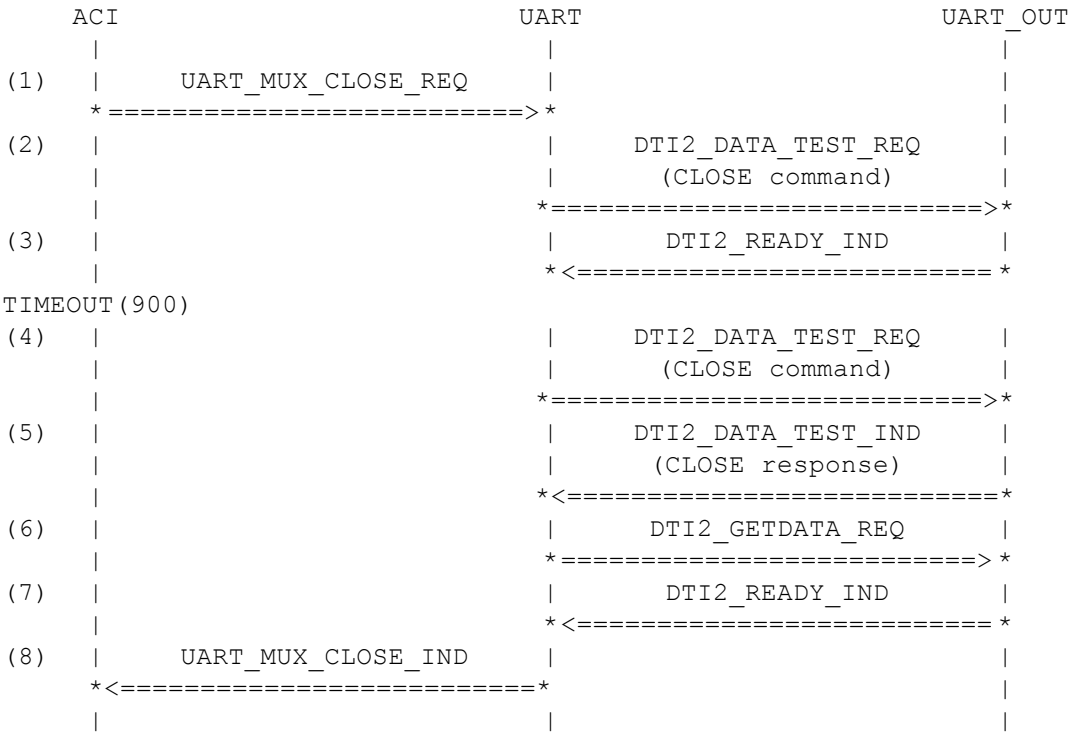
3.1.7 UART107: Retransmission in Multiplexer Close-down (UE initiated, Timeout T2)

Description:

The ACI requests termination of the multiplexer operation on the link. The peer is informed by sending a multiplexer close down command. The Timer T2 expires and the first retransmission is done since the timer T3 is not expired yet. A multiplexer close down response is received and confirms the successful reset of the link into normal AT command mode.

Preamble:

UART103



Parametrization:

Primitive	Parameter	Value
(1) UART_MUX_CLOSE_REQ		
	device	UART_DEVICE
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_CMD_UE2TE_CLOSE
(3) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1

(4)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CMD_UE2TE_CLOSE
(5)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_RES_TE2UE_CLOSE
(6)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(7)	DTI2_READY_IND		
		link_id	LINK_WRITEDATA_PORT_1
(8)	UART_MUX_CLOSE_IND		
		device	UART_DEVICE

History:

24-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

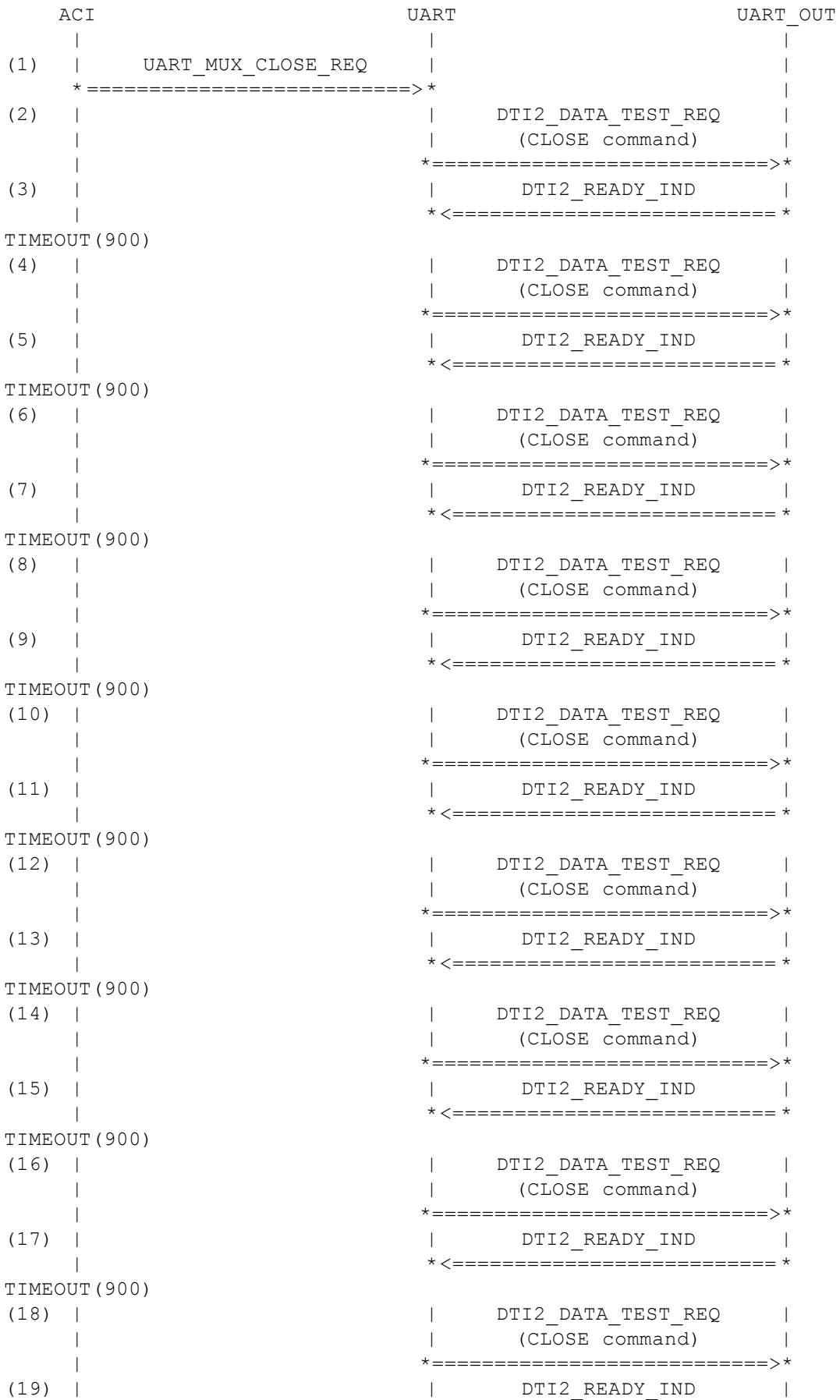
3.1.8 UART108: Multiplexer Close-down (UE initiated) - No Response

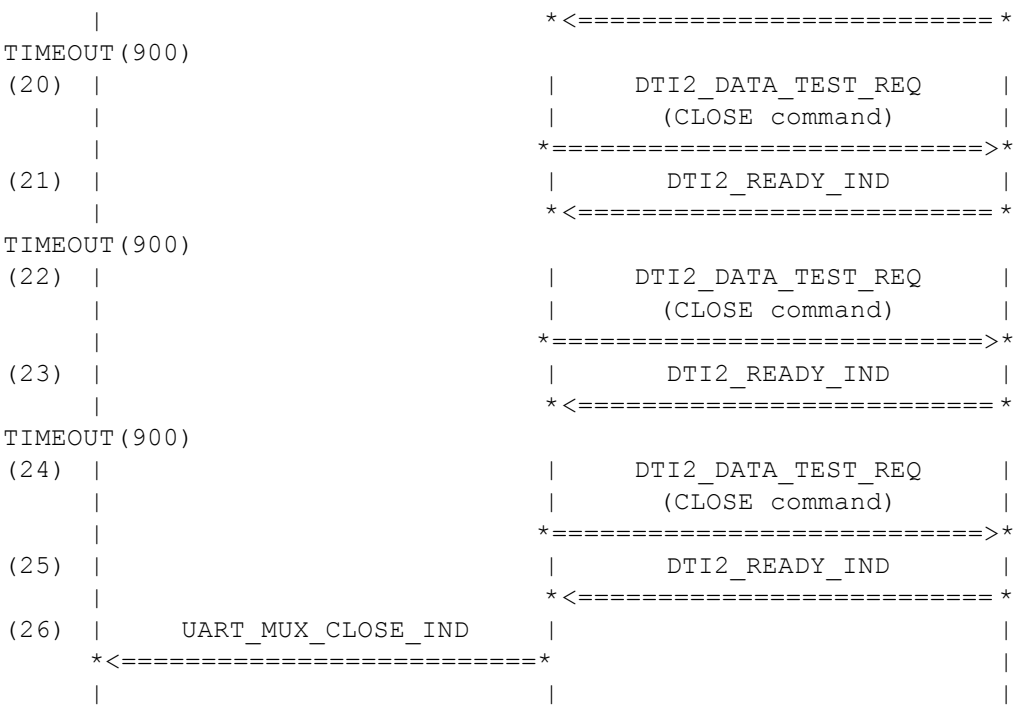
Description:

The ACI requests termination of the multiplexer operation on the link. The peer is informed by sending a multiplexer close down command. No close down command response is received - the timer T2 expires and 33 retransmissions are done until T3 expires. Then the UART switches to normal mode.

Preamble:

UART103


ACI
UART
UART_OUT
(1) UART_MUX_CLOSE_REQ
(2) DTI2_DATA_TEST_REQ (CLOSE command)
(3) DTI2_READY_IND
TIMEOUT(900)
(4) DTI2_DATA_TEST_REQ (CLOSE command)
(5) DTI2_READY_IND
TIMEOUT(900)
(6) DTI2_DATA_TEST_REQ (CLOSE command)
(7) DTI2_READY_IND
TIMEOUT(900)
(8) DTI2_DATA_TEST_REQ (CLOSE command)
(9) DTI2_READY_IND
TIMEOUT(900)
(10) DTI2_DATA_TEST_REQ (CLOSE command)
(11) DTI2_READY_IND
TIMEOUT(900)
(12) DTI2_DATA_TEST_REQ (CLOSE command)
(13) DTI2_READY_IND
TIMEOUT(900)
(14) DTI2_DATA_TEST_REQ (CLOSE command)
(15) DTI2_READY_IND
TIMEOUT(900)
(16) DTI2_DATA_TEST_REQ (CLOSE command)
(17) DTI2_READY_IND
TIMEOUT(900)
(18) DTI2_DATA_TEST_REQ (CLOSE command)
(19) DTI2_READY_IND



Parametrization:

Primitive	Parameter	Value
(1) UART_MUX_CLOSE_REQ		
	device	UART_DEVICE
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_CMD_UE2TE_CLOSE
(3) DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(4) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_CMD_UE2TE_CLOSE
(5) DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(6) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_CMD_UE2TE_CLOSE
(7) DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(8) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_CMD_UE2TE_CLOSE
(9) DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(10) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1

		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CMD_UE2TE_CLOSE
(11)	DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(12)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CMD_UE2TE_CLOSE
(13)	DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(14)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CMD_UE2TE_CLOSE
(15)	DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(16)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CMD_UE2TE_CLOSE
(17)	DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(18)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CMD_UE2TE_CLOSE
(19)	DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(20)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CMD_UE2TE_CLOSE
(21)	DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(22)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CMD_UE2TE_CLOSE
(23)	DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(24)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CMD_UE2TE_CLOSE
(25)	DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(26)	UART_MUX_CLOSE_IND		
		device	UART_DEVICE

History:

13-Mar-2001	LW	Initial
21-Mar-2001	LW	retransmissions until T3 expires
11-Oct-2001	TVO	adapted for use with DTI2

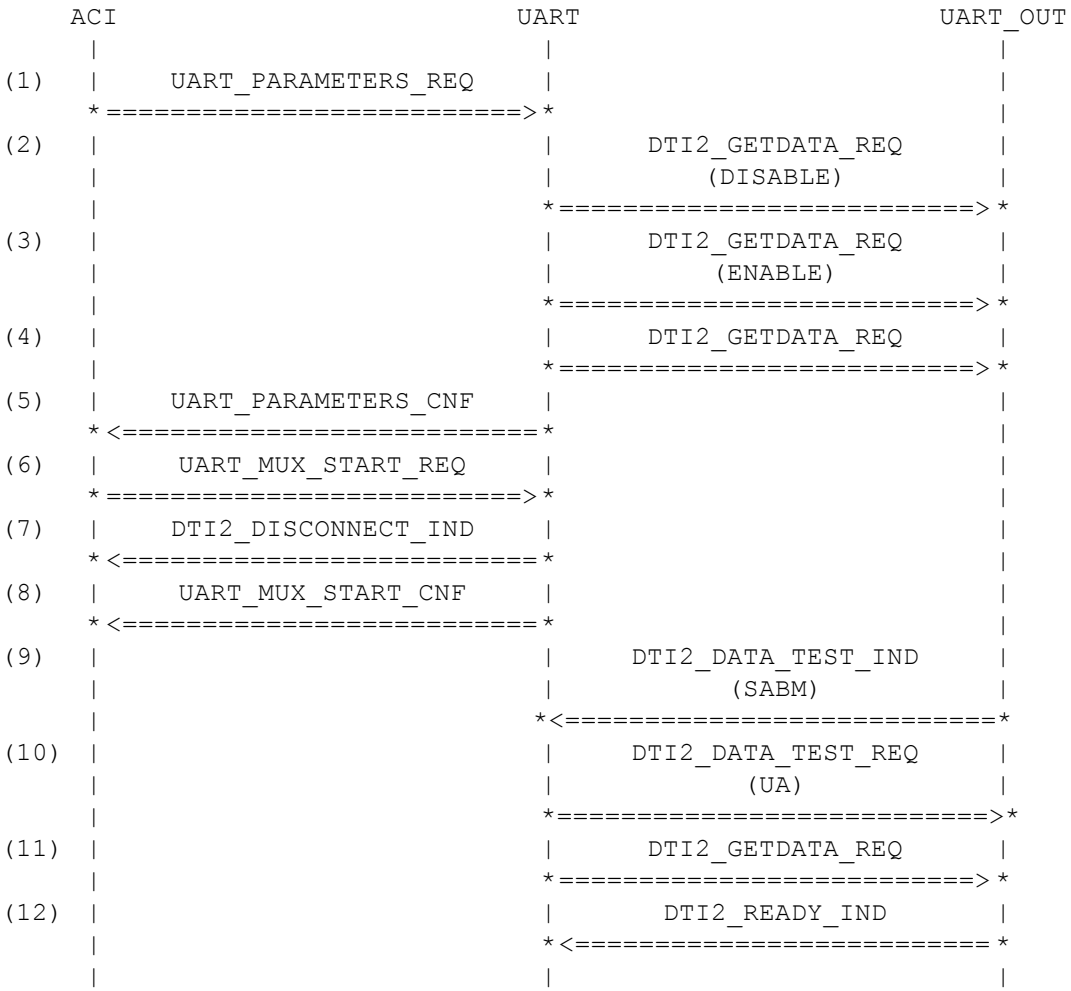
3.1.9 UART109: Start Multiplexer with previous data transfer

Description:

ACI send first new parameters for serial link. After that ACI starts the TS 27.010 multiplexer operation over a serial link. Only advanced mode with UIH frames ares supported. The UART starts the timer T3 to time waiting for the SABM frame.

Preamble:

UART041A



Parametrization:

Primitive	Parameter	Value
(1) UART_PARAMETERS_REQ		
	device	UART_DEVICE
	comPar	COMPAR_57600_8_N_1_HW
(2) DTI2_GETDATA_REQ		
	link_id	LINK_DISABLE_PORT_1
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ENABLE_PORT_1
(4) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1

(5) UART_PARAMETERS_CNF		
	device	UART_DEVICE
(6) UART_MUX_START_REQ		
	device	UART_DEVICE
	mode	UART_MUX_MODE_ADVANCED
	frame_type	UART_MUX_FRAME_UIH
	n1	UART_MUX_N1_ADVANCED_DEFAULT
	t1	UART_MUX_T1_DEFAULT_TEST
	n2	UART_MUX_N2_DEFAULT
	t2	UART_MUX_T2_DEFAULT_TEST
	t3	UART_MUX_T3_DEFAULT_TEST
(7) DTI2_DISCONNECT_IND		
	link_id	LINK_ID_1
	cause	DTI_CAUSE_NORMAL_CLOSE
(8) UART_MUX_START_CNF		
	device	UART_DEVICE
(9) DTI2_DATA_TEST_IND		
	link_id	LINK_READDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MUX_START0
(10) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_UA_UE2TE
(11) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(12) DTI2_READY_IND		
	link_id	LINK_WRITEDATA_PORT_1

History:

26-Apr-2001	STW	Initial
11-Oct-2001	TVO	adapted for use with DTI2
16-Jan-2003	STW	error correction

3.2 DLC Establishment

3.2.1 UART123: DLC Establishment Requested by TE

Description:

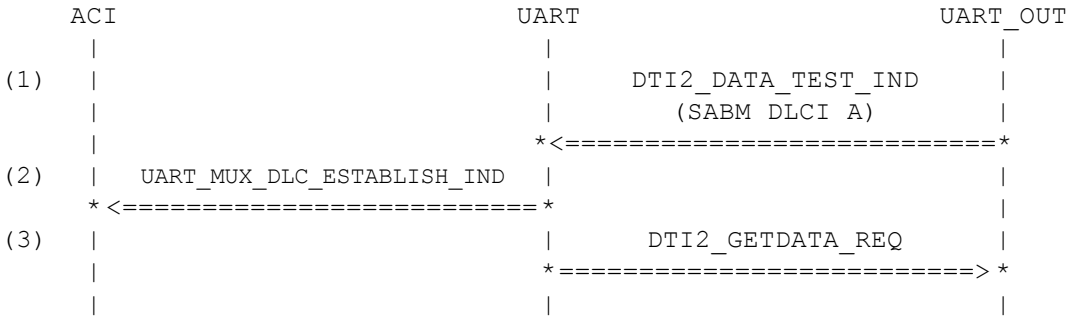
The TE wants to establish a data link connection (DLC) with a not yet allocated (variant A) or already allocated (variant B) data link connection identifier (DLCI). The MMI is notified of the new data link connection.

Variants:

<A>....<B>

Preamble:

<A>UART103
<B>UART109



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_ESTABLISH_DLCI_A_TE2UE
(2) UART_MUX_DLC_ESTABLISH_IND		
	device	UART_DEVICE
	dlci	DLCI_A
	convergence	UART_MUX_CONVERGENCE_UOS
	n1	UART_MUX_N1_ADVANCED_DEFAULT
	service	UART_MUX_SERVICE_DEFAULT
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1

History:

17-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.2.2 UART124: MMI Confirmation of TE Initiated DLC

Description:

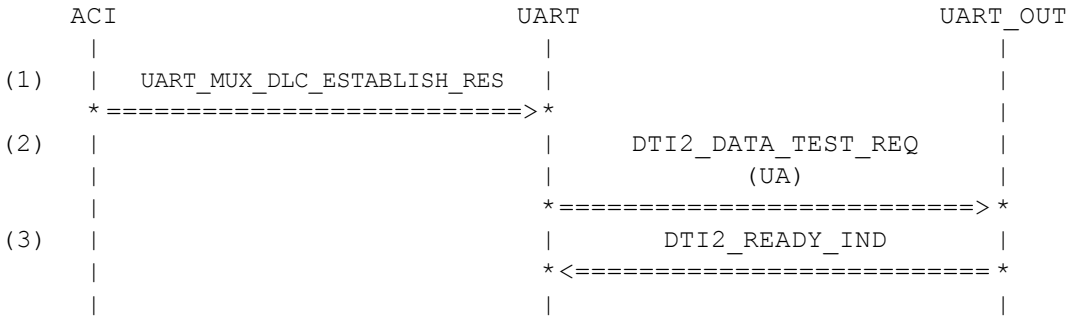
The MMI accepts the TE initiated DLC, a confirmation is sent to the TE. In variant B there existed a valid connection with this DLCI before, in variant A it did not.

Variants:

<A>....<B>

Preamble:

<A>UART123A
<B>UART123B



Parametrization:

	Primitive	Parameter	Value
(1)	UART_MUX_DLC_ESTABLISH_RES		
		device	UART_DEVICE
		dlci	DLCI_A
		n1	UART_MUX_N1_ADVANCED_DEFAULT
(2)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_UA_UE2TE_DLCI_A
(3)	DTI2_READY_IND		
		link_id	LINK_WRITEDATA_PORT_1

History:

17-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.2.3 UART125: MMI Cancellation of TE Initiated DLC

Description:

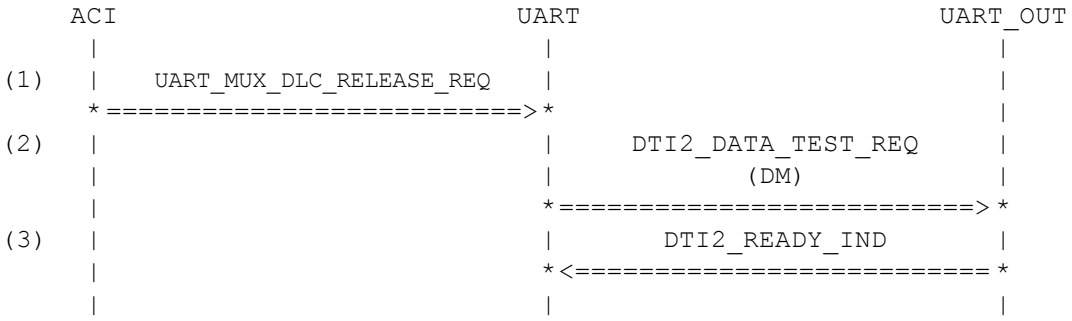
The MMI rejects the UE initiated DLC, a negative confirmation is sent to the UE. In variant B there existed a valid connection with this DLCI before, in variant A it did not.

Variants:

<A>....<B>

Preamble:

<A>UART123A
<B>UART123B



Parametrization:

	Primitive	Parameter	Value
(1)	UART_MUX_DLC_RELEASE_REQ		
		device	UART_DEVICE
		dlci	DLCI_A
(2)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_DM_UE2TE_A
(3)	DTI2_READY_IND		
		link_id	LINK_WRITEDATA_PORT_1

History:

17-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.2.4 UART126: Second DLC Successfully Requested by TE

Description:

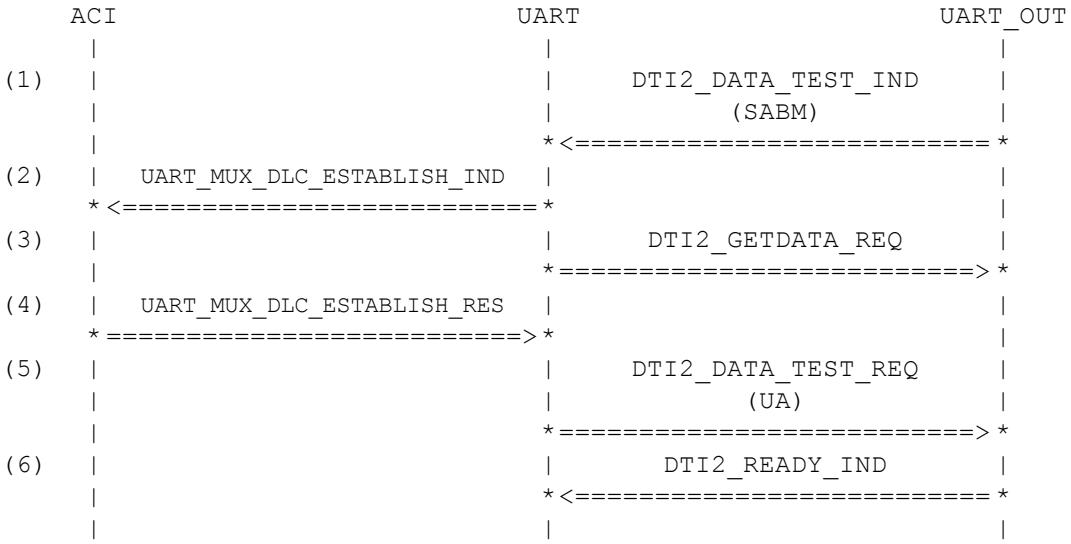
The TE wants to establish a second data link connection (DLC) with a not yet allocated data link connection identifier (DLCI). A SABM frame is received in order to set up the connection. In variant A the first DLC request has been completed before this second request in variant B the first request is still pending.

Variants:

<A>....<C>

Preamble:

<A>UART124A
<B>UART124B
<C>UART131G



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_ESTABLISH_DLCI_B_TE2UE
(2) UART_MUX_DLC_ESTABLISH_IND		
	device	UART_DEVICE
	dlci	DLCI_B
	convergence	UART_MUX_CONVERGENCE_UOS
	n1	UART_MUX_N1_ADVANCED_DEFAULT
	service	UART_MUX_SERVICE_DEFAULT
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) UART_MUX_DLC_ESTABLISH_RES		
	device	UART_DEVICE
	dlci	DLCI_B
	n1	UART_MUX_N1_ADVANCED_DEFAULT
(5) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_UA_UE2TE_DLCI_B
(6) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1

History:

17-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.2.5 UART127: DTI Assignment on first DLC

Description:

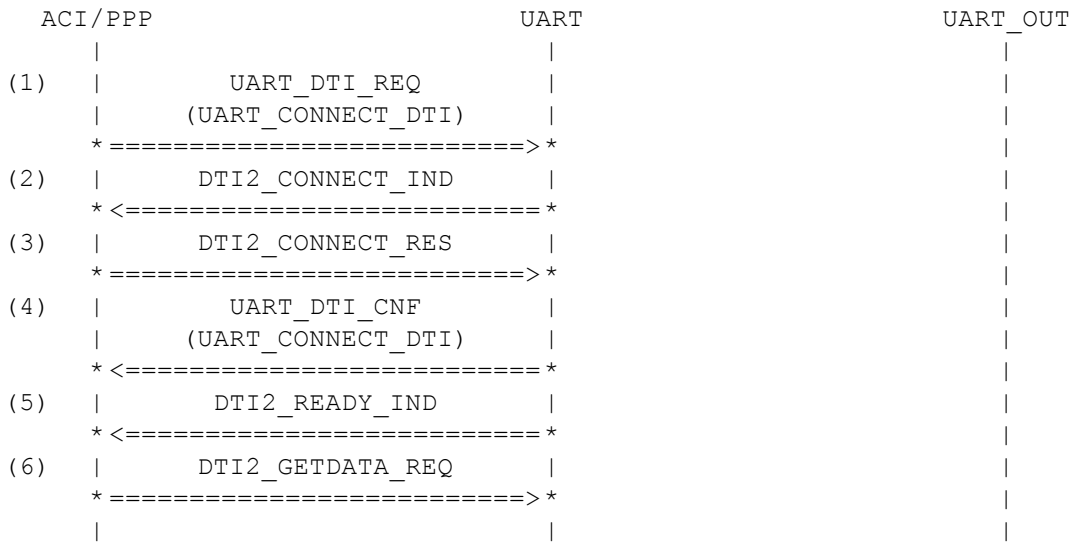
After successfully establishing a DLC the ACI assigns a DTI connection to PPP.

Variants:

<A>....<D>

Preamble:

<A>UART124A
<B>UART124B
<C>UART126A
<D>UART126B



Parametrization:

Primitive	Parameter	Value
(1) UART_DTI_REQ		
	dti_conn	UART_CONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_A
	direction	FALSE
	link_id	LINK_ID_1
	entity_name	DTI_CHANNELNAME_PPP
(2) DTI2_CONNECT_IND		
	link_id	LINK_ID_1
	version	DTI_VERSION_10
(3) DTI2_CONNECT_RES		
	link_id	LINK_ID_1
	version	DTI_VERSION_10
(4) UART_DTI_CNF		
	dti_conn	UART_CONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_A
(5) DTI2_READY_IND		
	link_id	LINK_ID_1

(6) DTI2_GETDATA_REQ

link_id	LINK_ID_1

History:

13-Mar-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.2.6 UART128: DTI Assignment on second DLC

Description:

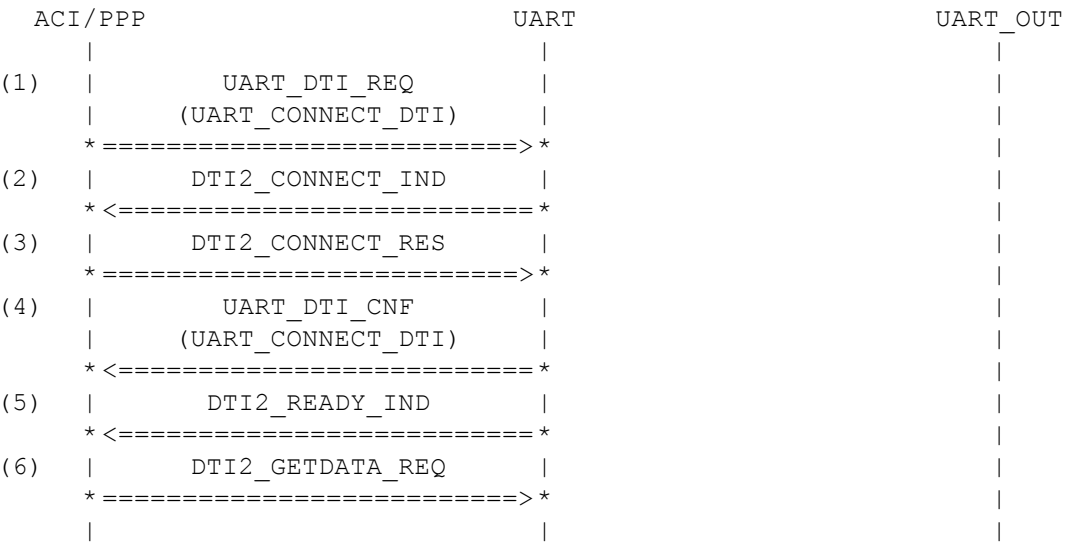
A second DTI connection is setup by ACI.

Variants:

<A>....<B>

Preamble:

<A> UART127C
<B> UART127D



Parametrization:

Primitive	Parameter	Value
(1) UART_DTI_REQ		
	dti_conn	UART_CONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_B
	direction	FALSE
	link_id	LINK_ID_2
	entity_name	DTI_CHANNELNAME_PPP
(2) DTI2_CONNECT_IND		
	link_id	LINK_ID_2
	version	DTI_VERSION_10
(3) DTI2_CONNECT_RES		
	link_id	LINK_ID_2
	version	DTI_VERSION_10

(4) UART_DTI_CNF

dti_conn	UART_CONNECT_DTI
device	UART_DEVICE
dlci	DLCI_B

(5) DTI2_READY_IND

link_id	LINK_ID_2

(6) DTI2_GETDATA_REQ

link_id	LINK_ID_2

History:

13-Mar-2001		LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2	

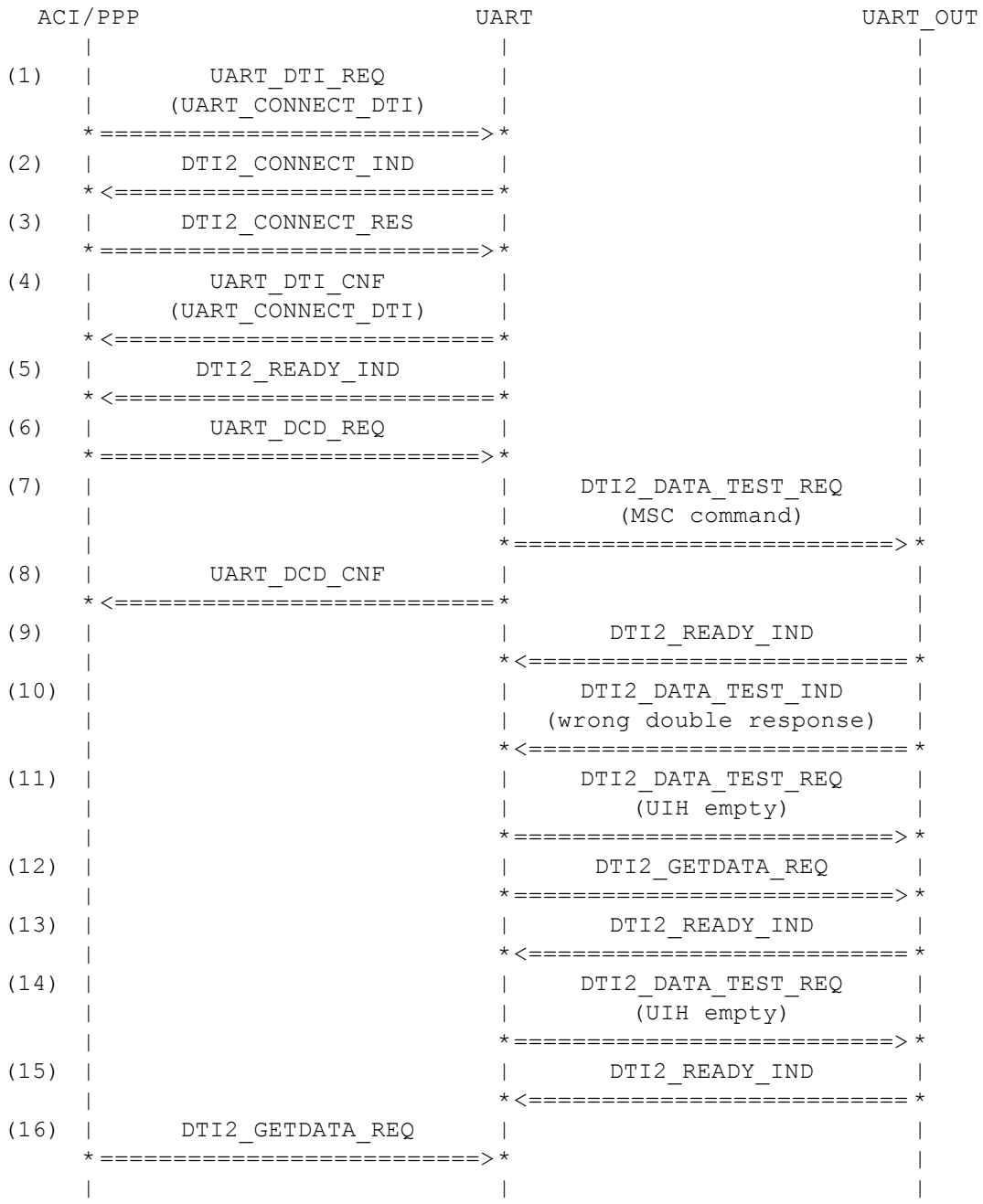
3.2.7 UART129: DCD request after DTI configuration

Description:

After successfully establishing a DLC the ACI assigns a DTI connection to PPP.

Preamble:

UART124B



Parametrization:

Primitive	Parameter	Value
(1) UART_DTI_REQ		
	dti_conn	UART_CONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_A
	direction	FALSE
	link_id	LINK_ID_1
	entity_name	DTI_CHANNELNAME_PPP
(2) DTI2_CONNECT_IND		
	link_id	LINK_ID_1
	version	DTI_VERSION_10

(3) DTI2_CONNECT_RES		
	link_id	LINK_ID_1
	version	DTI_VERSION_10
(4) UART_DTI_CNF		
	dti_conn	UART_CONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_A
(5) DTI2_READY_IND		
	link_id	LINK_ID_1
(6) UART_DCD_REQ		
	device	UART_DEVICE
	dlci	DLCI_A
	line_state	UART_LINE_ON
(7) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSC_CMD_UE2TE_A
(8) UART_DCD_CNF		
	device	UART_DEVICE
	dlci	DLCI_A
(9) DTI2_READY_IND		
	link_id	LINK_WRITEDATA_PORT_1
(10) DTI2_DATA_TEST_IND		
	link_id	LINK_READDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSCMSC_RES_A
(11) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_CORRUPTED_RES_EMPTY
(12) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(13) DTI2_READY_IND		
	link_id	LINK_WRITEDATA_PORT_1
(14) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_CORRUPTED_RES_EMPTY
(15) DTI2_READY_IND		
	link_id	LINK_WRITEDATA_PORT_1
(16) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1

History:

14-May-2001	STW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.3 Multiplexed DLC Data Transmission

3.3.1 UART131: Sending Data on Single DLC (UE->TE)

Description:

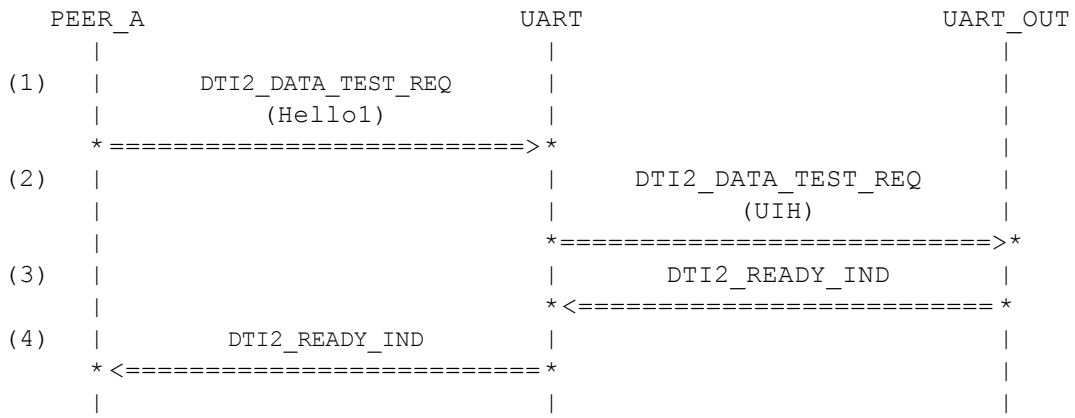
The peer entity A sends data on DLCI A. The multiplexer builds an information frame and forwards it to the UART test output interface.

Variants:

<A>....<G>

Preamble:

<A>UART127A
<B>UART127B
<C>UART127C
<D>UART127D
<E>UART128A
<F>UART128B
<G>UART129



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_PEER_A
	sdu	STRING_HELLO1_SDU
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	SDU_FRAME_HELLO1_UE2TE_A
(3) DTI2_READY_IND	link_id	LINK_UART_OUT_PORT_1
(4) DTI2_READY_IND		
	link_id	LINK_ID_1

History:

18-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.3.2 UART132: Receiving Data on Single DLC (TE->UE)

Description:

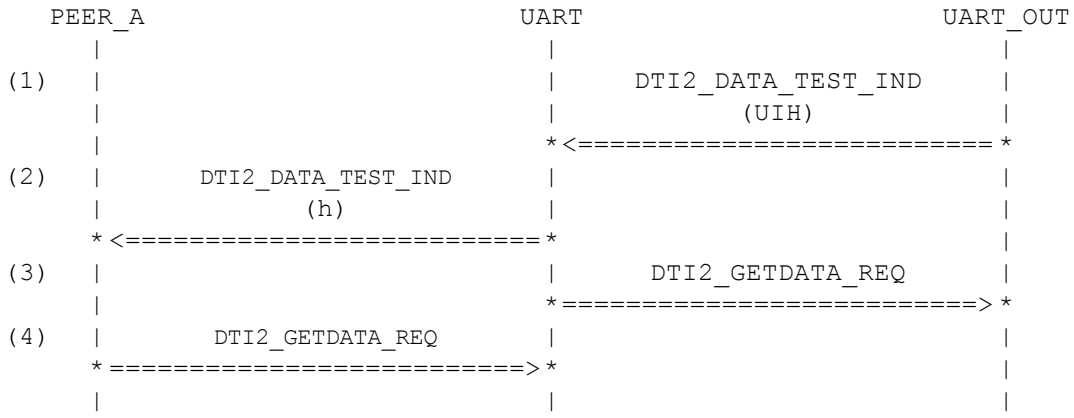
A data frame arrives on DLCI A. The multiplexer extracts the information from the arrived frame and forwards the encapsulated string "Hello2" to peer entity A.

Variants:

<A>....<F>

Preamble:

<A>UART128A
<B>UART128B
<C>UART131A
<D>UART131B
<E>UART131E
<F>UART131F



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_FRAME_H_TE2UE_A
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_PEER_A
	sdu	STRING_H_SDU
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1

History:

18-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.3.3 UART133: Mixed Sending/Receiving of Data on Two DLCs

Description:

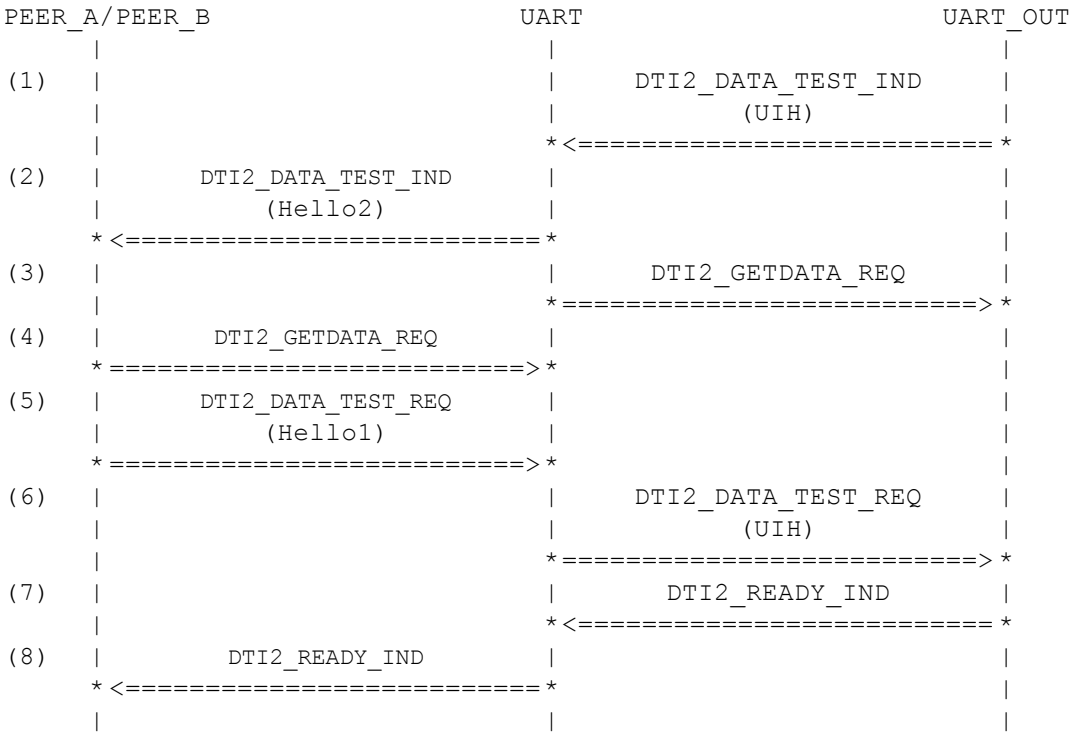
A data frame arrives on DLCI A. The multiplexer extracts the information from the arrived frame and forwards the encapsulated string "Hello2" to peer entity A. Next the peer B wants to transmit some data. It is encapsulated in an appropriate UIH frame and sent out.

Variants:

<A>....<F>

Preamble:

<A>UART128A
<B>UART128B
<C>UART131E
<D>UART131F
<E>UART132E
<F>UART132F



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
<A>	sdu	SDU_FRAME_HELLO2_TE2UE_A
<B>	sdu	SDU_FRAME_HELLO2_TE2UE_A
<C>	sdu	SDU_FRAME_HELLO2_TE2UE_A

<D>	sdu	SDU_FRAME_HELLO2_TE2UE_A
<E>	sdu	SDU_FRAME_HELLO2_TE2UE_A
<F>	sdu	SDU_FRAME_64_TE2UE_A
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_PEER_A
<A>	sdu	STRING_HELLO2_SDU
<B>	sdu	STRING_HELLO2_SDU
<C>	sdu	STRING_HELLO2_SDU
<D>	sdu	STRING_HELLO2_SDU
<E>	sdu	STRING_HELLO2_SDU
<F>	sdu	STRING_64_SDU
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(5) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_2
	parameters	DTI_PARAMETER_PEER_B
	sdu	STRING_HELLO1_SDU
(6) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_PEER_B
	sdu	SDU_FRAME_HELLO1_UE2TE_B
(7) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(8) DTI2_READY_IND		
	link_id	LINK_ID_2

History:

18-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.3.4 UART134: Receiving Data on Single DLC (TE->UE) with Flow Control

Description:

A long data frame arrives on DLCI A. The multiplexer extracts the information from the arrived frame. The DTX entity disables the data flow since its buffer is full. A modem status command for the channel is sent out. DTX forwards the encapsulated string "Hello2" to peer entity A and enables the flow again.

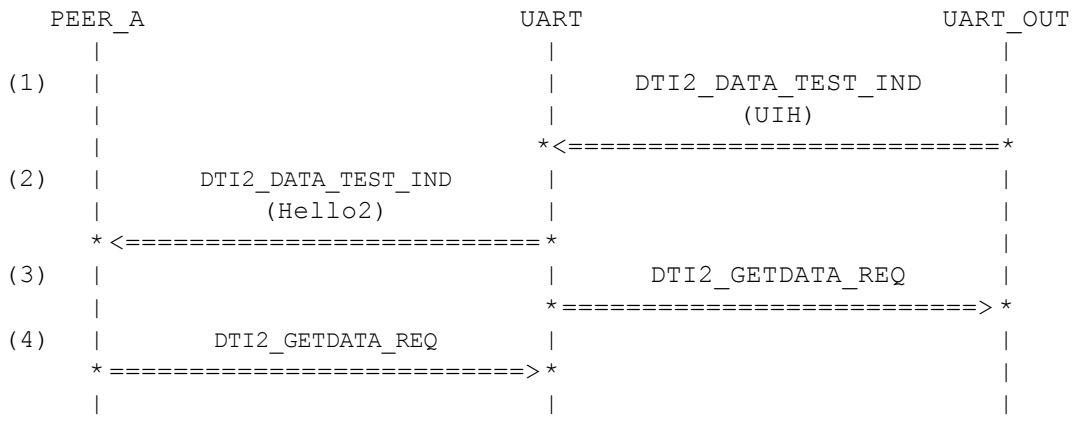
TEST CASE IS STILL INCOMPLETE!

Variants:

<A>....<C>

Preamble:

<A>UART128A
<B>UART131A
<C>UART131C



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_FRAME_HELLO2_TE2UE_A
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_PEER_A
	sdu	STRING_HELLO2_SDU
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1

History:

18-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.4 Unsupported Features/Commands

3.4.1 UART140: Reception of Unsupported Command

Description:

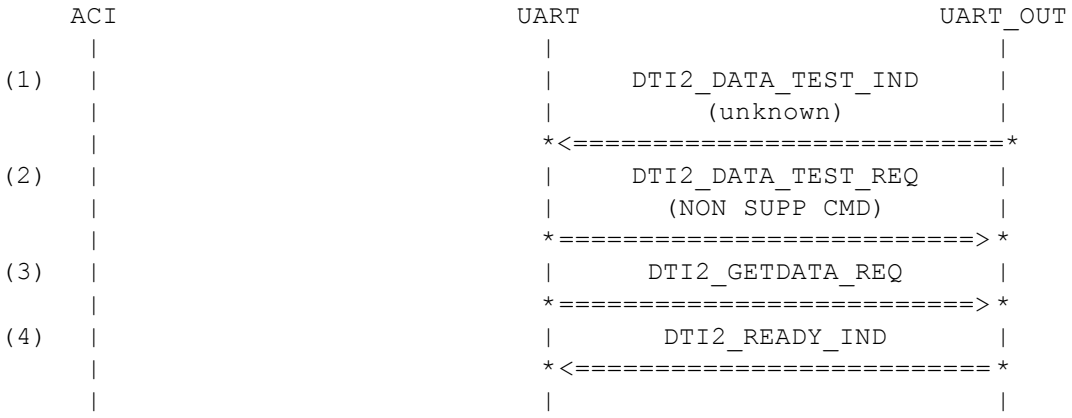
An unsupported command is received from TE. In variant A it is received for an open DLC in variant B it is received with an unassigned DLCI. A non supported command response frame is sent back to the peer entity in both cases.

Variants:

<A>...<B>

Preamble:

<A> UART128A
<B> UART103



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_READDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_UNKNOWN_CMD_TE2UE_A
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_UNSUPP_CMD_UE2TE_A
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_READY_IND		
	link_id	LINK_WRITEDATA_PORT_1

History:

24-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.5 Modem Status Commands

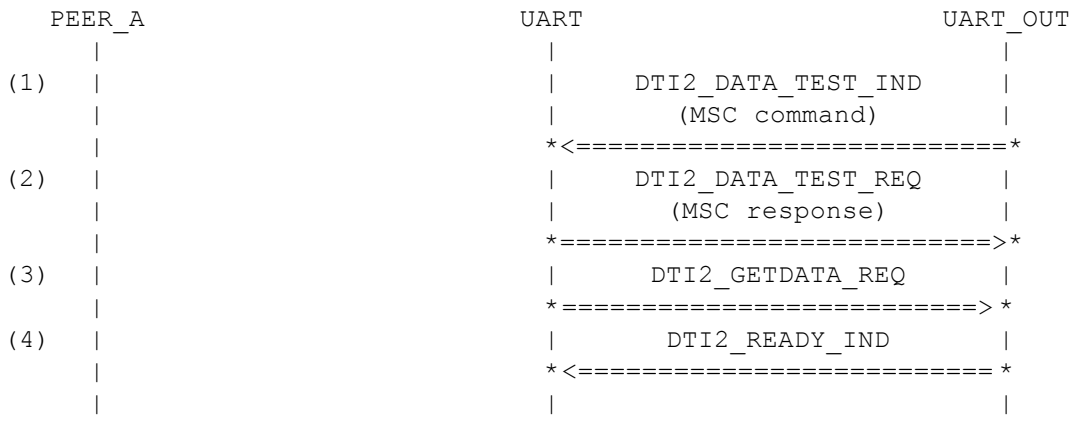
3.5.1 UART150: Reception of MSC

Description:

The peer sends a modem status command for an established DLC. Since it does not contain a ring or break signal, no notification is sent to the ACI.

Preamble:

UART128A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSC_CMD_TE2UE_A
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSC_RES_UE2TE_A
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1

History:

24-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

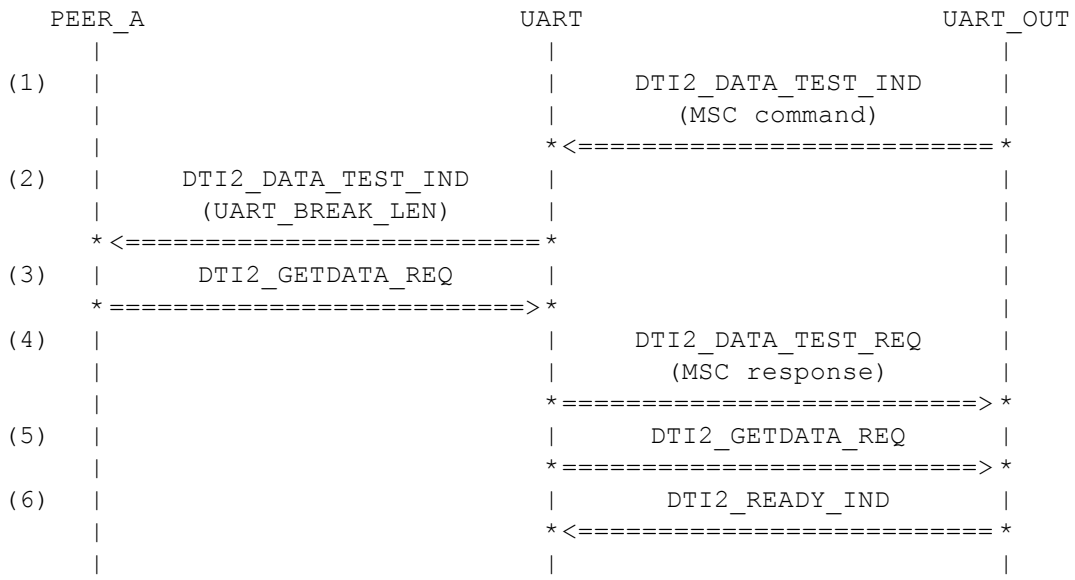
3.5.2 UART151: Reception of Break-MSC

Description:

The peer sends a modem status command for an established DLC indicating a break signal for DLCI A. The MMI/ACI is notified by the UART entity with an empty DTI2_DATA_IND primitive.

Preamble:

UART150



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSCBREAK_CMD_TE2UE_A
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME_BREAK
	sdu	EMPTY_STRING_SDU
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(4) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSCBREAK_RES_UE2TE_A
(5) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(6) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1

History:

24-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

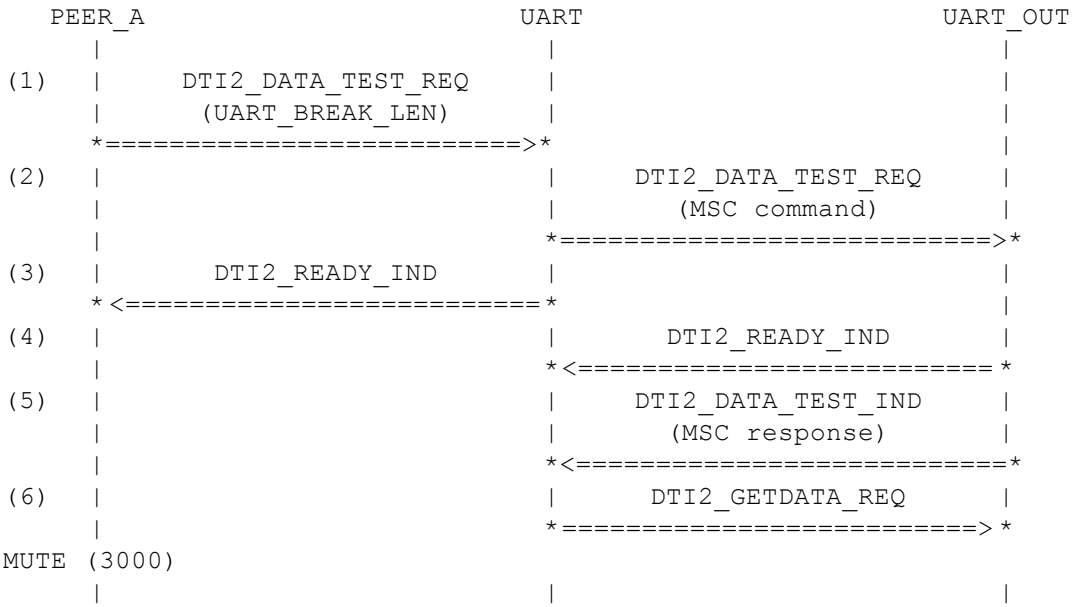
3.5.3 UART152: Sending Break-MSC

Description:

The device A requests that a break signal is to be sent on DLCI_A. An appropriate MSC frame is sent to the peer multiplexer.

Preamble:

UART150



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME_BREAK
	sdu	EMPTY_STRING_SDU
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSCBREAK_CMD_UE2TE_A
(3) DTI2_READY_IND		
	link_id	LINK_ID_1
(4) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(5) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSCBREAK_RES_TE2UE_A
(6) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1

History:

24-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

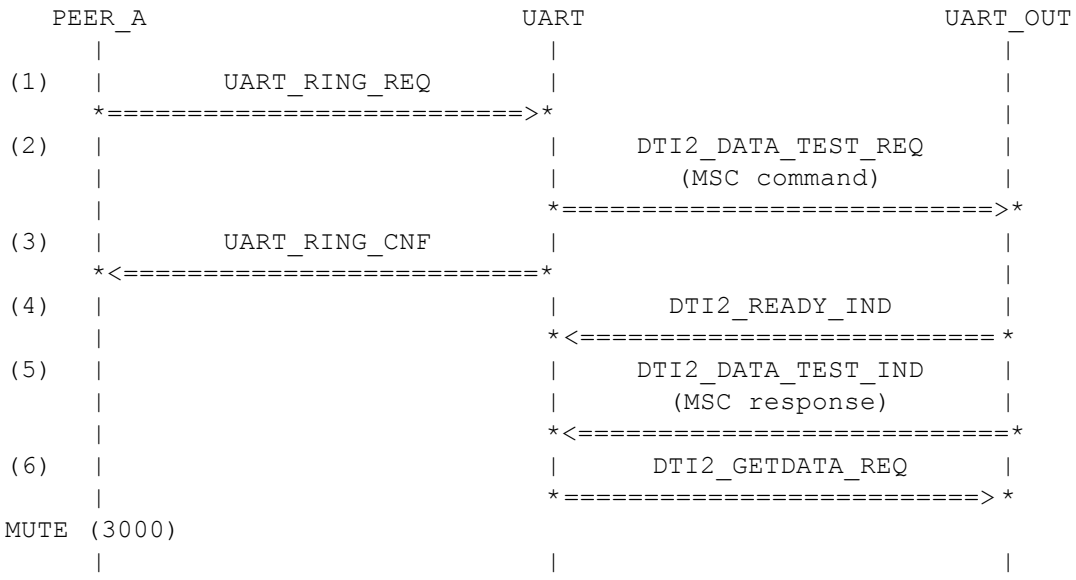
3.5.4 UART153: Sending Ring-MSC

Description:

The device A requests that an incomming call indication is to be sent on DLCI_A. An appropriate MSC frame is sent to the peer multiplexer.

Preamble:

UART150



Parametrization:

Primitive	Parameter	Value
(1) UART_RING_REQ		
	device	UART_DEVICE
	dlci	DLCI_A
	line_state	UART_LINE_ON
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSCRING_CMD_UE2TE_A
(3) UART_RING_CNF		
	device	UART_DEVICE
	dlci	DLCI_A
(4) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(5) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSCRING_RES_TE2UE_A

(6)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1

History:

24-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

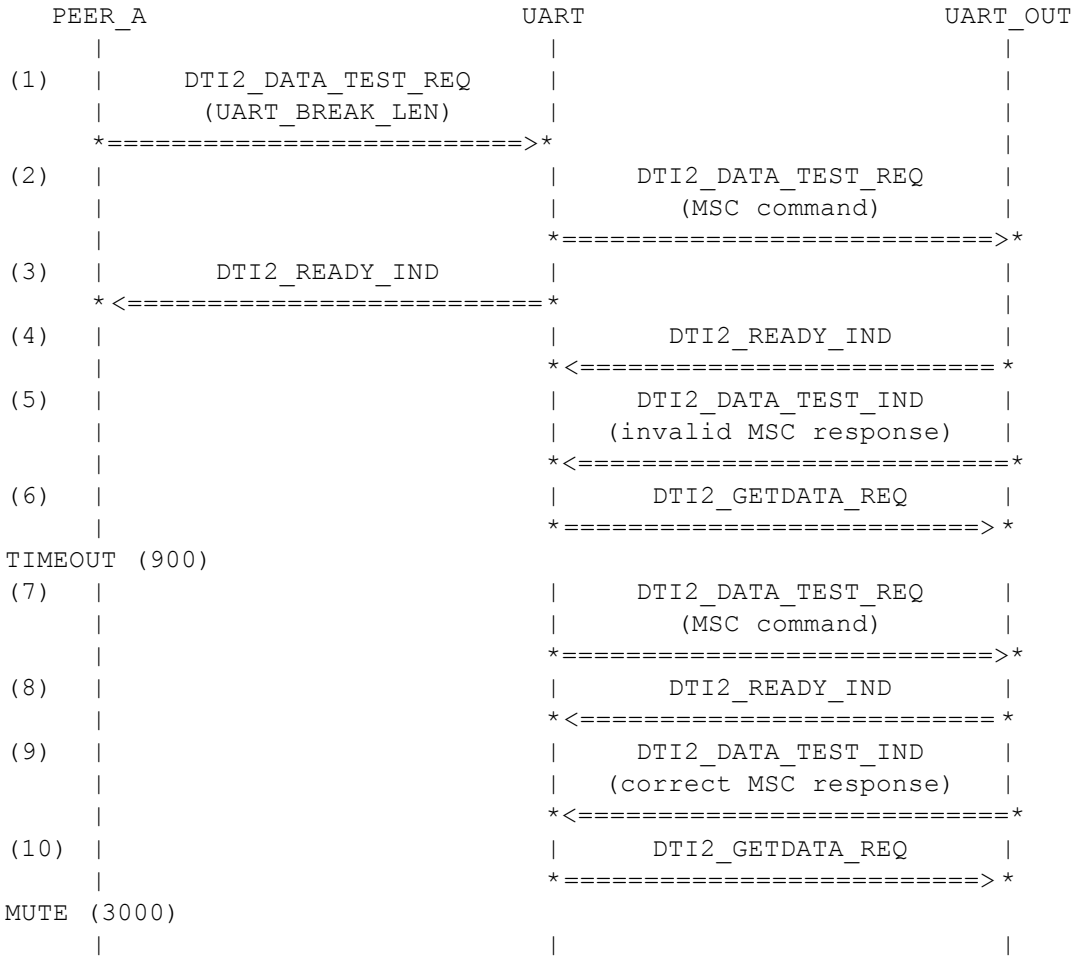
3.5.5 UART154: Retransmission of Break-MSC

Description:

The device A requests that a break signal is to be sent on DLCI_A. An appropriate MSC frame is sent to the peer multiplexer. The received MSC response is different from the expected response, therefore the MSC command is retransmitted when the timer T2 expires. Then the correct response message is received and the timer T2 is stopped.

Preamble:

UART150



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_1

	parameters	DTI_PARAMETER_FRAME_BREAK
	sdu	EMPTY_STRING_SDU
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSCBREAK_CMD_UE2TE_A
(3) DTI2_READY_IND		
	link_id	LINK_ID_1
(4) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(5) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSCBREAK_RES_TE2UE_A_INVALID
(6) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(7) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSCBREAK_CMD_UE2TE_A
(8) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(9) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSCBREAK_RES_TE2UE_A
(10) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1

History:

21-Mar-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

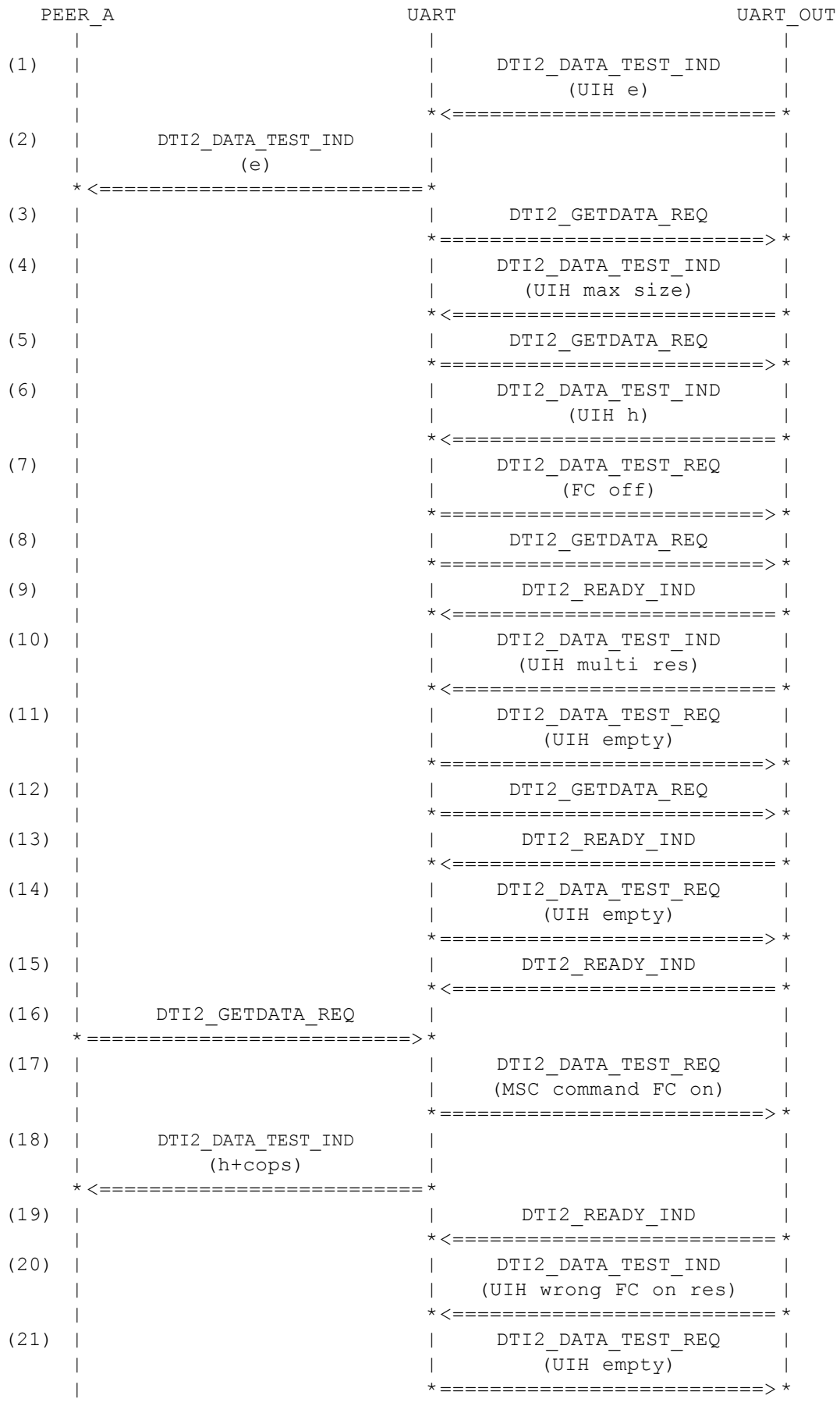
3.5.6 UART155: Flow-Control-MSC on reception side

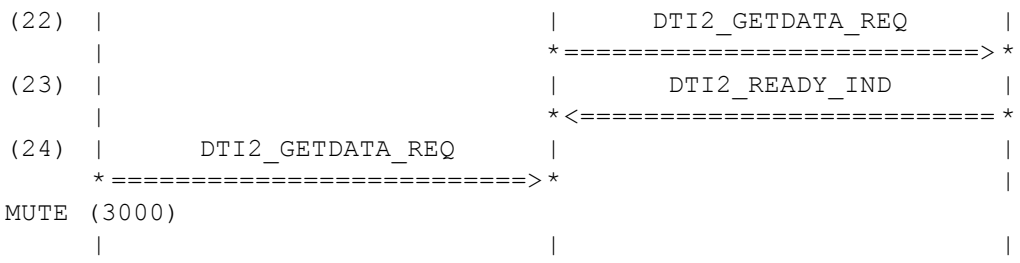
Description:

DLCI_A receives more characters than it can process. It sends a Flow-Control-MSC to the peer to stop sending.

Preamble:

UART132D


PEER_A
UART
UART_OUT
(1)
DTI2_DATA_TEST_IND
(UIH e)
(2)
DTI2_DATA_TEST_IND
(e)
(3)
DTI2_GETDATA_REQ
(4)
DTI2_DATA_TEST_IND
(UIH max size)
(5)
DTI2_GETDATA_REQ
(6)
DTI2_DATA_TEST_IND
(UIH h)
(7)
DTI2_DATA_TEST_REQ
(FC off)
(8)
DTI2_GETDATA_REQ
(9)
DTI2_READY_IND
(10)
DTI2_DATA_TEST_IND
(UIH multi res)
(11)
DTI2_DATA_TEST_REQ
(UIH empty)
(12)
DTI2_GETDATA_REQ
(13)
DTI2_READY_IND
(14)
DTI2_DATA_TEST_REQ
(UIH empty)
(15)
DTI2_READY_IND
(16)
DTI2_GETDATA_REQ
(17)
DTI2_DATA_TEST_REQ
(MSC command FC on)
(18)
DTI2_DATA_TEST_IND
(h+cops)
(19)
DTI2_READY_IND
(20)
DTI2_DATA_TEST_IND
(UIH wrong FC on res)
(21)
DTI2_DATA_TEST_REQ
(UIH empty)



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_READDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_FRAME_E_TE2UE_A
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_PEER_A
	sdu	STRING_E_SDU
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_READDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_FRAME_64_TE2UE_A
(5) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_READDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_FRAME_H_TE2UE_A
(7) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_MSCFCOFF_CMD_UE2TE_A
(8) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(9) DTI2_READY_IND		
	link_id	LINK_WRITEDATA_PORT_1
(10) DTI2_DATA_TEST_IND		
	link_id	LINK_READDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DMSCMSCDDDD_RES_A
(11) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_CORRUPTED_RES_EMPTY
(12) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1

(13)	DTI2_READY_IND		
		link_id	LINK_WRITEDATA_PORT_1
(14)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CORRUPTED_RES_EMPTY
(15)	DTI2_READY_IND		
		link_id	LINK_WRITEDATA_PORT_1
(16)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1
(17)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_MSC_CMD_UE2TE_A
(18)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_PEER_A
		sdu	STRING_PLUSCOPS_SDU
(19)	DTI2_READY_IND		
		link_id	LINK_WRITEDATA_PORT_1
(20)	DTI2_DATA_TEST_IND		
		link_id	LINK_READDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CORRUPT_MSC_RES_A
(21)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CORRUPTED_RES_EMPTY
(22)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(23)	DTI2_READY_IND		
		link_id	LINK_WRITEDATA_PORT_1
(24)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1

History:

10-May-2001	STW	Initial
11-Sep-2001	STW	new minimum size_multiplier=3
11-Oct-2001	TVO	adapted for use with DTI2

3.6 DLC Release

3.6.1 UART160: DLC Release Requested by ACI

Description:

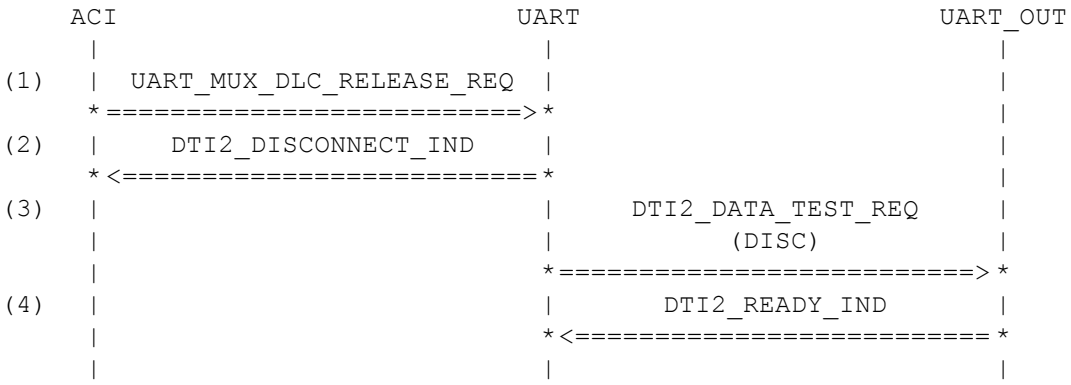
The ACI wants to release a data link connection (DLC) with an allocated data link connection identifier (DLCI). A DISC frame is transmitted in order to release the connection.

Variants:

<A>....<C>

Preamble:

<A> UART127A
<B> UART127C
<C> UART128A



Parametrization:

	Primitive	Parameter	Value
(1)	UART_MUX_DLC_RELEASE_REQ		
		device	UART_DEVICE
		dlci	DLCI_A
(2)	DTI2_DISCONNECT_IND		
		link_id	LINK_ID_1
		cause	DTI_CAUSE_NORMAL_CLOSE
(3)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_DISC_DLCI_A_UE2TE
(4)	DTI2_READY_IND		
		link_id	LINK_UART_OUT_PORT_1

History:

17-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2
16-Jan-2003	STW	error correction

3.6.2 UART161: Confirmation of DLC Release (initiated by UE)

Description:

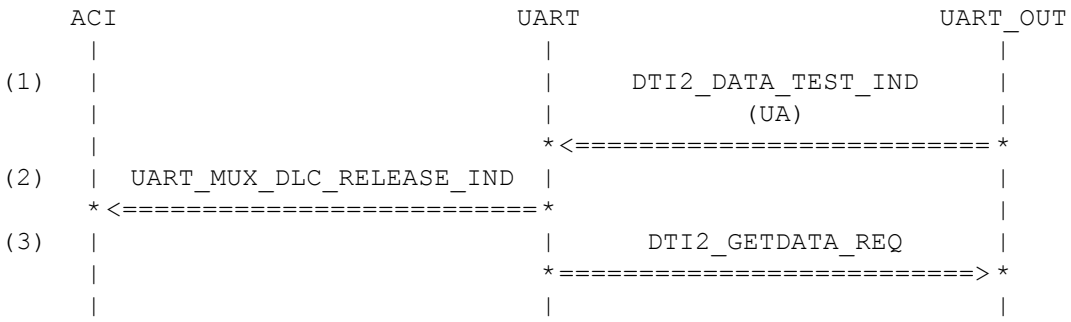
The confirmation for a previously requested release of a DLC arrives.

Variants:

<A>....<F>

Preamble:

<A> UART160A
<B> UART160B
<C> UART160C
<D> UART169A
<E> UART169B
<F> UART169C



Parametrization:

	Primitive	Parameter	Value
(1)	DTI2_DATA_TEST_IND		
		link_id	LINK_READDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_UA_TE2UE_DLCI_A
(2)	UART_MUX_DLC_RELEASE_IND		
		device	UART_DEVICE
		dlci	DLCI_A
(3)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1

History:

17-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

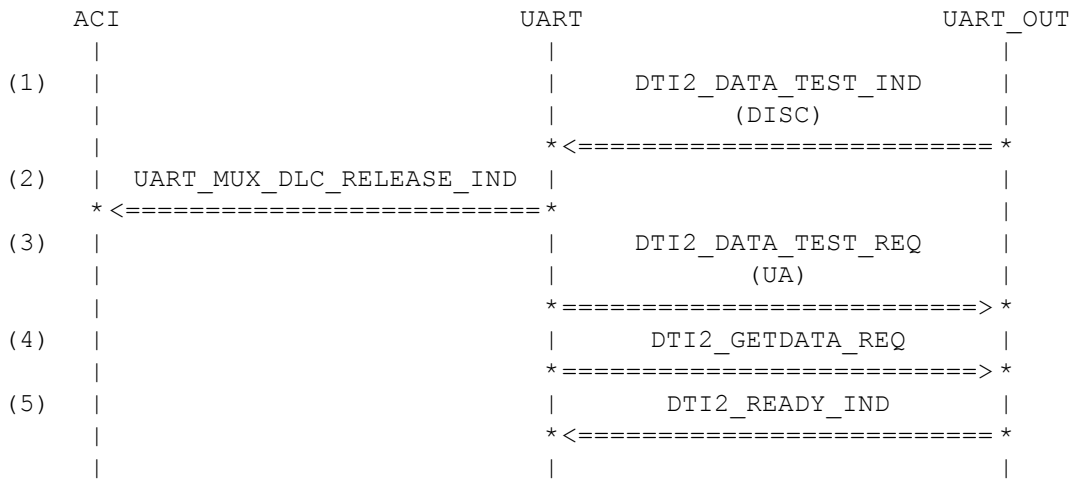
3.6.3 UART162: DLC Release Requested by TE

Description:

The TE wants to release a data link connection (DLC) with an allocated data link connection identifier (DLCI). A DISC frame is received, a confirmation is sent and the MMI is notified of the DLC release.

Preamble:

UART127C



Parametrization:

	Primitive	Parameter	Value
(1)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_DISC_DLCI_B_TE2UE
(2)	UART_MUX_DLC_RELEASE_IND		
		device	UART_DEVICE
		dlci	DLCI_B
(3)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_UA_UE2TE_DLCI_B
(4)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(5)	DTI2_READY_IND		
		link_id	LINK_UART_OUT_PORT_1

History:

17-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.6.4 UART163: Data Received for Already Released Channel

Description:

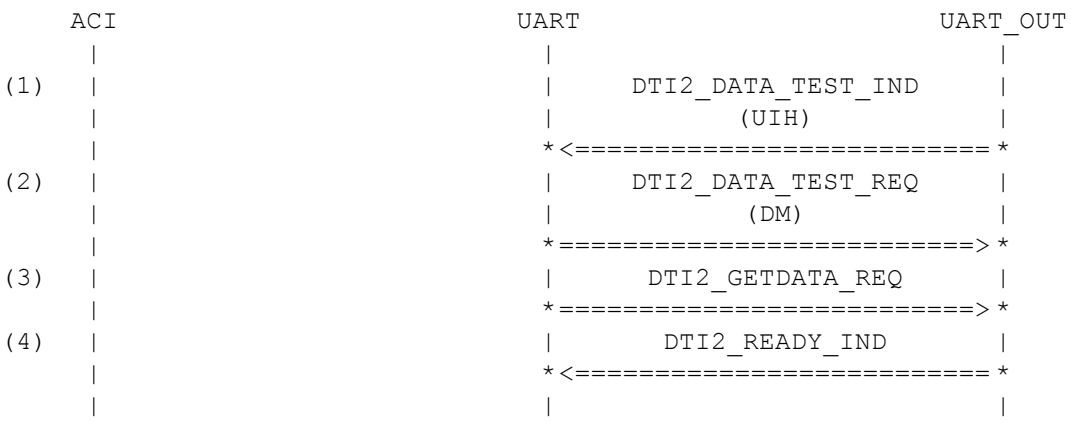
Although the DLC A has previously been released, more data is received. Since the specified channel is in disconnected mode the UART responds with a DM frame.

Variants:

<A>....<F>

Preamble:

<A> UART161A
<B> UART161B
<C> UART161C
<D> UART161D
<E> UART161E
<F> UART161F



Parametrization:

	Primitive	Parameter	Value
(1)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_FRAME_HELLO2_TE2UE_A
(2)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_DM_UE2TE_A_0F
(3)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(4)	DTI2_READY_IND		
		link_id	LINK_UART_OUT_PORT_1

History:

24-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.6.5 UART164: No Response to Retransmissions of DLC DISC Frame

Description:

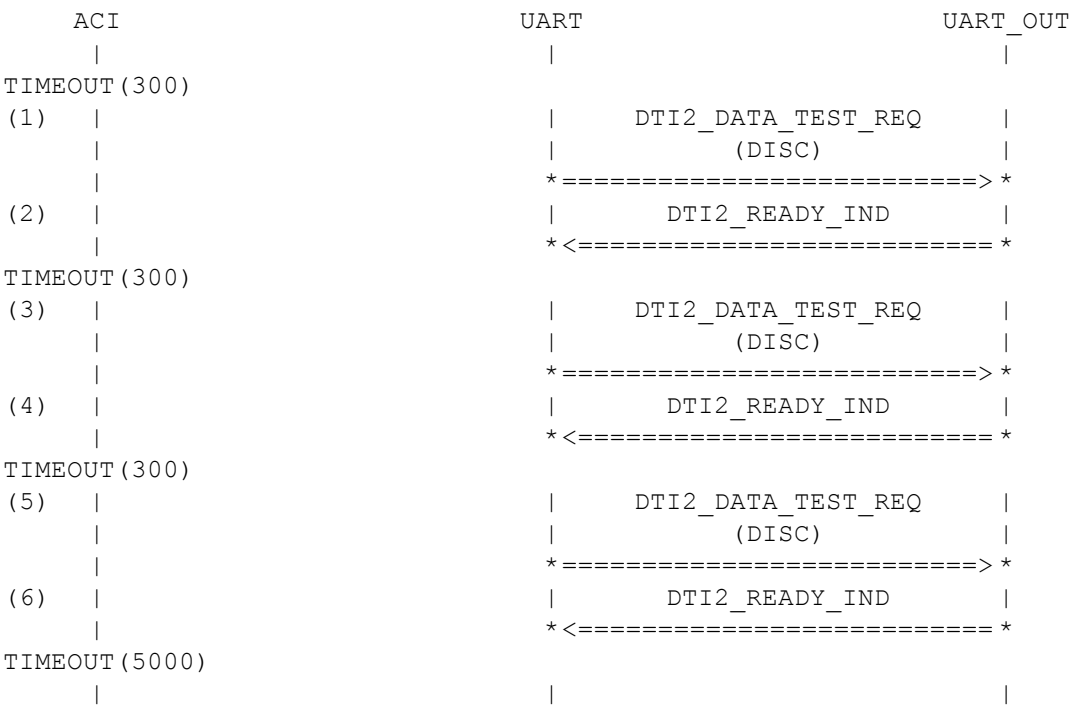
In test case UART160 a DISC for DLCI_A was sent. Now the timer T1 expires. Because the counter for retransmissions in not zero yet the UART transmitts the DISC frame again and decreases the retransmission counter by one. After three retransmissions the max. number of retransmissions is reached and the channel is close without the response.

Variants:

<A>....<F>

Preamble:

<A> UART160A
<B> UART160B
<C> UART160C
<D> UART169A
<E> UART169B
<F> UART169C



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DISC_DLCI_A_UE2TE
(2) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(3) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DISC_DLCI_A_UE2TE
(4) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(5) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DISC_DLCI_A_UE2TE
(6) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1

History:

24-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

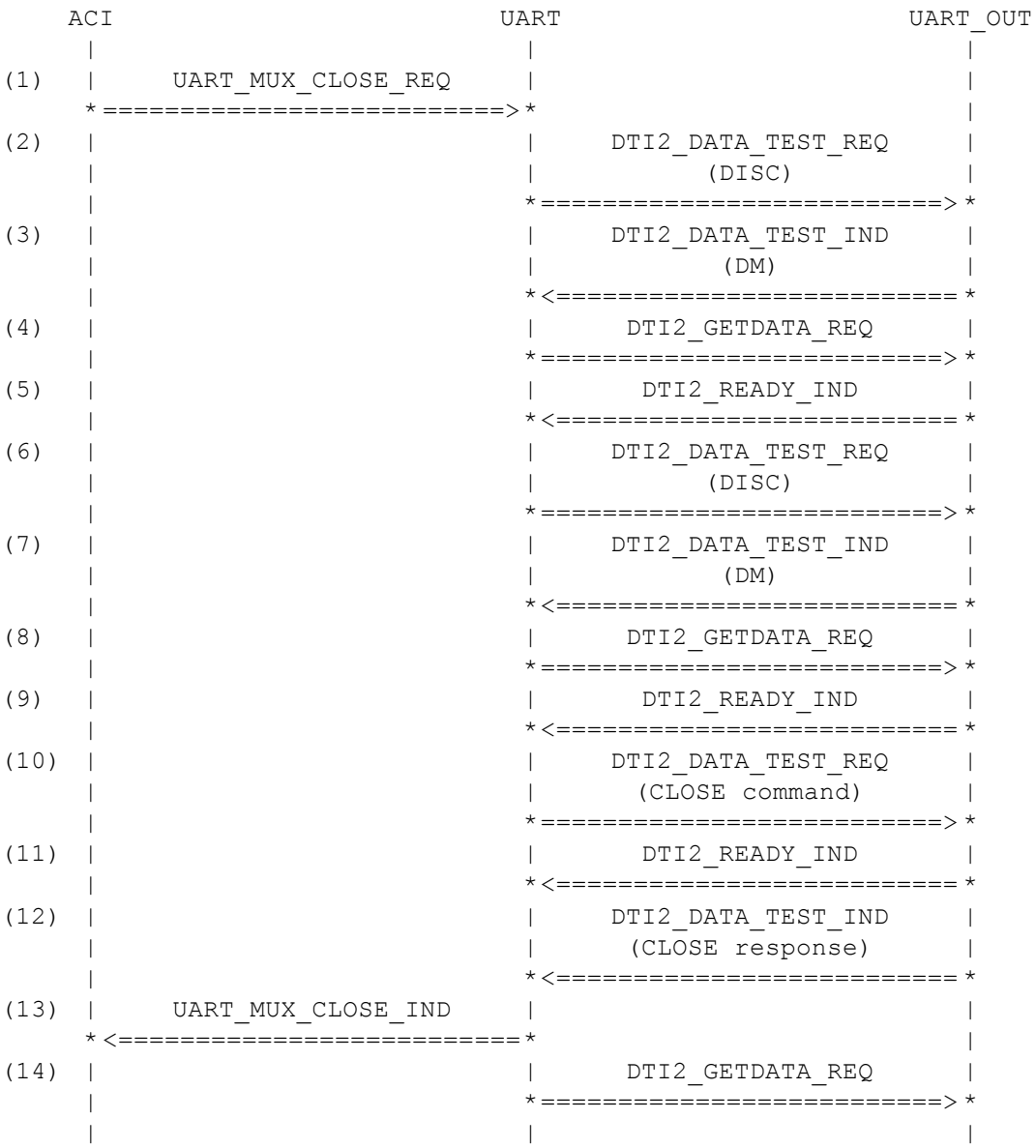
3.6.6 UART165: ACI Initiated Multiplexer Close-down (icl. DLC release)

Description:

The ACI wants to end the multiplexer operation and sends a UART MUX CLOSE request to the UART entity. Then the UART releases all DLCs currently in use (DLCI_A, DLCI_B) and uses the usual close down procedure (see UART105) to return to the normal AT mode.

Preamble:

UART126A



Parametrization:

	Primitive	Parameter	Value
(1)	UART_MUX_CLOSE_REQ		
		device	UART_DEVICE
(2)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_DISC_DLCI_A_UE2TE
(3)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_DM_TE2UE_A
(4)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(5)	DTI2_READY_IND		
		link_id	LINK_UART_OUT_PORT_1
(6)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_DISC_DLCI_B_UE2TE
(7)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_DM_TE2UE_B
(8)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(9)	DTI2_READY_IND		
		link_id	LINK_UART_OUT_PORT_1
(10)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_CMD_UE2TE_CLOSE
(11)	DTI2_READY_IND		
		link_id	LINK_UART_OUT_PORT_1
(12)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_RES_TE2UE_CLOSE
(13)	UART_MUX_CLOSE_IND		
		device	UART_DEVICE
(14)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1

History:

24-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

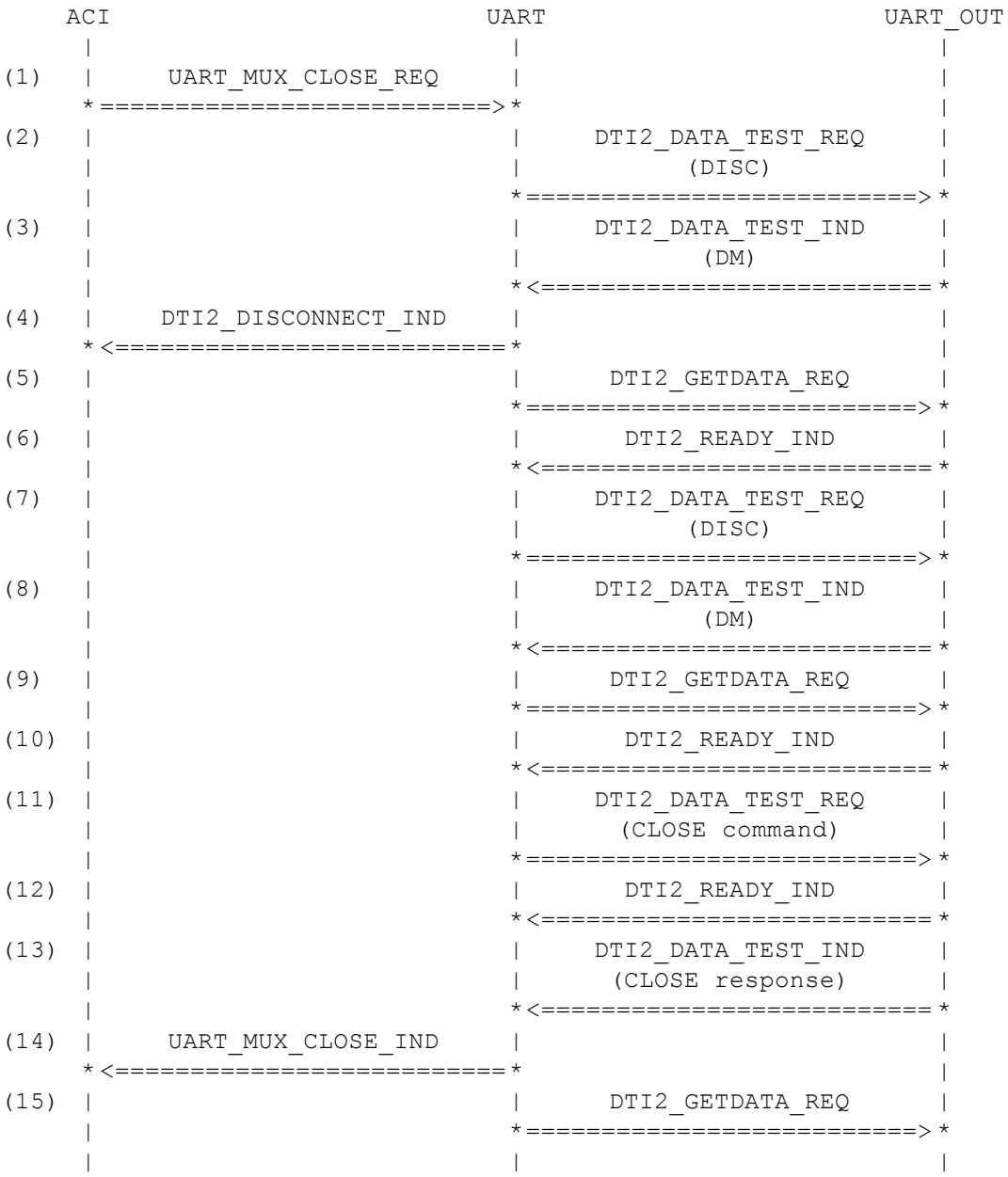
3.6.7 UART166: ACI Initiated Multiplexer Close-down (icl. DLC release and DTI channel)

Description:

The ACI wants to end the multiplexer operation and sends a UART MUX CLOSE request to the UART entity. Then the UART releases all DLCs currently in use (DLCI_A, DLCI_B) and uses the usual close down procedure (see UART105) to return to the normal AT mode.

Preamble:

UART127C



Parametrization:

Primitive	Parameter	Value

(1) UART_MUX_CLOSE_REQ		
	device	UART_DEVICE
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DISC_DLCI_A_UE2TE
(3) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DM_TE2UE_A
(4) DTI2_DISCONNECT_IND		
	link_id	LINK_ID_1
	cause	DTI_CAUSE_NORMAL_CLOSE
(5) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(6) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(7) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DISC_DLCI_B_UE2TE
(8) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DM_TE2UE_B
(9) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(10) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(11) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_CMD_UE2TE_CLOSE
(12) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(13) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_RES_TE2UE_CLOSE
(14) UART_MUX_CLOSE_IND		
	device	UART_DEVICE
(15) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1

History:

10-Oct-2001	TVO	Initial, adapted from UART165
11-Oct-2001	TVO	adapted for use with DTI2
16-Jan-2003	STW	Error corrections

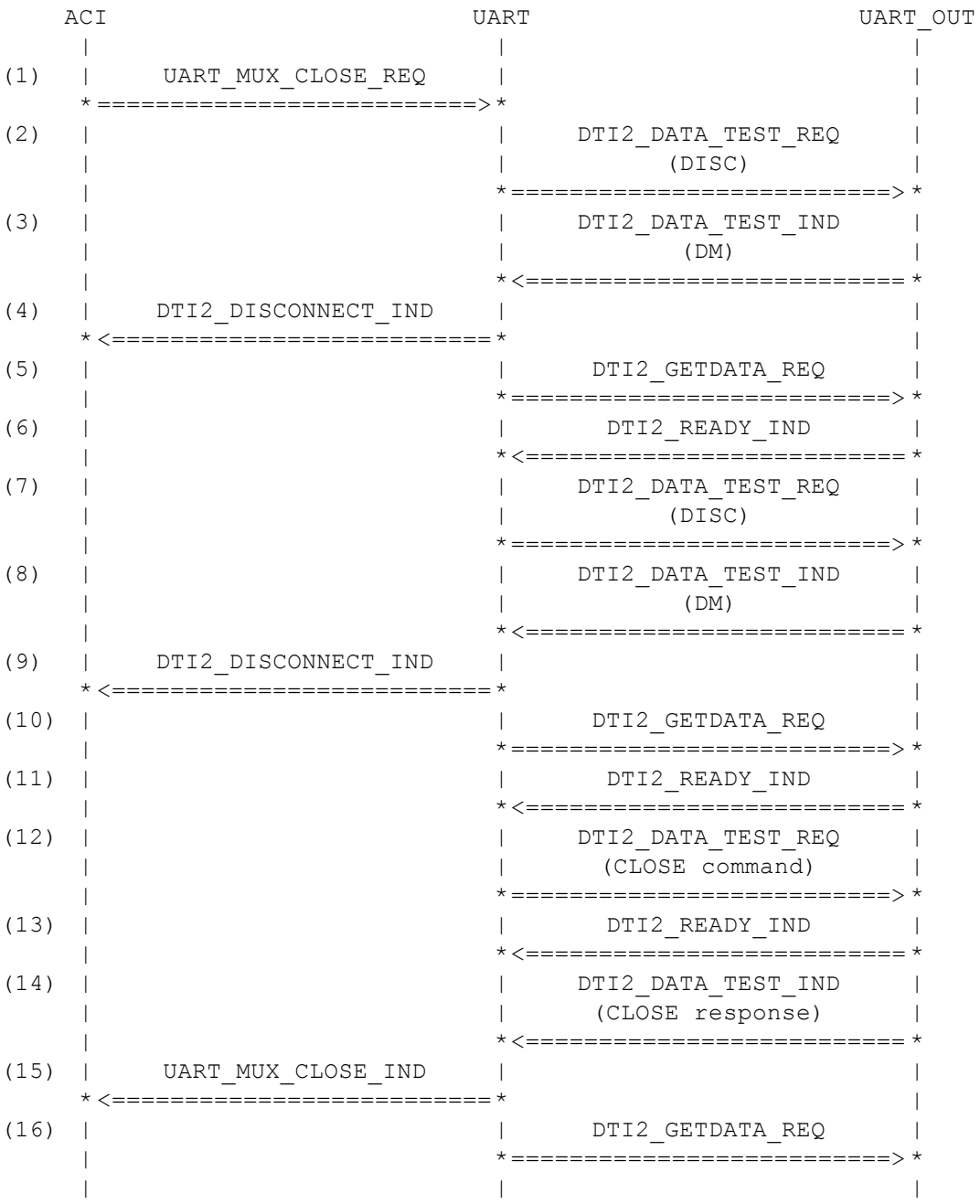
3.6.8 UART167: ACI Initiated Multiplexer Close-down (icl. DLC release and two DTI channels)

Description:

The ACI wants to end the multiplexer operation and sends a UART MUX CLOSE request to the UART entity. Then the UART releases all DLCs currently in use (DLCI_A, DLCI_B) and uses the usual close down procedure (see UART105) to return to the normal AT mode.

Preamble:

UART128A



Parametrization:

Primitive	Parameter	Value
(1) UART_MUX_CLOSE_REQ		
	device	UART_DEVICE
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DISC_DLCI_A_UE2TE
(3) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1

	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DM_TE2UE_A
(4) DTI2_DISCONNECT_IND		
	link_id	LINK_ID_1
	cause	DTI_CAUSE_NORMAL_CLOSE
(5) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(6) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(7) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DISC_DLCI_B_UE2TE
(8) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DM_TE2UE_B
(9) DTI2_DISCONNECT_IND		
	link_id	LINK_ID_2
	cause	DTI_CAUSE_NORMAL_CLOSE
(10) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(11) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(12) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_CMD_UE2TE_CLOSE
(13) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(14) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_RES_TE2UE_CLOSE
(15) UART_MUX_CLOSE_IND		
	device	UART_DEVICE
(16) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1

History:

10-Oct-2001	TVO	Initial, adapted from UART165
11-Oct-2001	TVO	adapted for use with DTI2

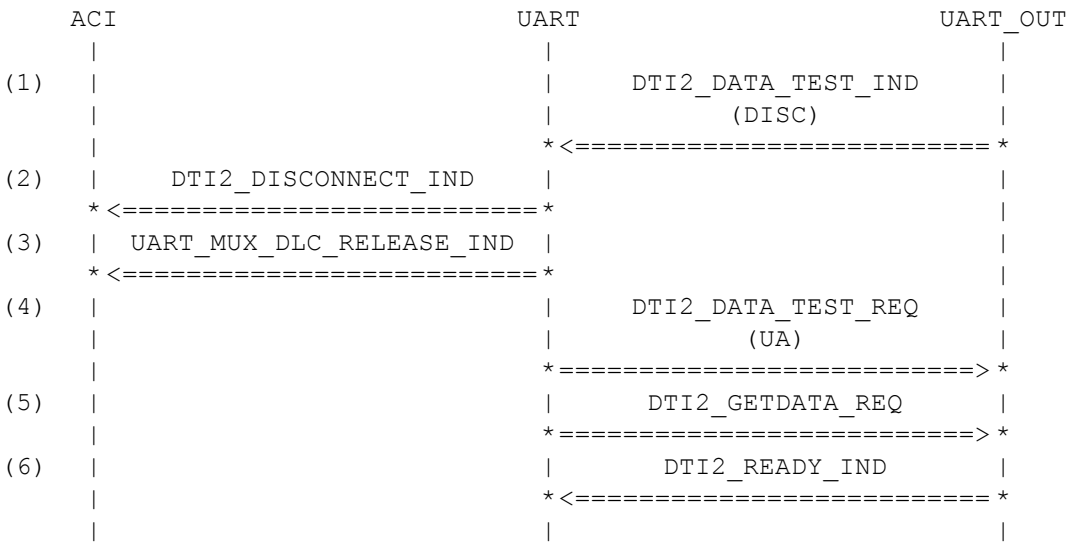
3.6.9 UART168: DLC Release Requested by TE (two MUX channels)

Description:

The TE wants to release a data link connection (DLC) with an allocated data link connection identifier (DLCI). A DISC frame is received, a confirmation is sent and the MMI is notified of the DLC release.

Preamble:

UART128A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DISC_DLCI_B_TE2UE
(2) DTI2_DISCONNECT_IND		
	link_id	LINK_ID_2
	cause	DTI_CAUSE_NORMAL_CLOSE
(3) UART_MUX_DLC_RELEASE_IND		
	device	UART_DEVICE
	dlci	DLCI_B
(4) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_UA_UE2TE_DLCI_B
(5) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(6) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1

History:

17-Jan-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2
16-Jan-2003	STW	Error corrections

3.6.10 UART169: DLC Release Requested by ACI, following a DTI2_DISCONNECT_REQ

Description:

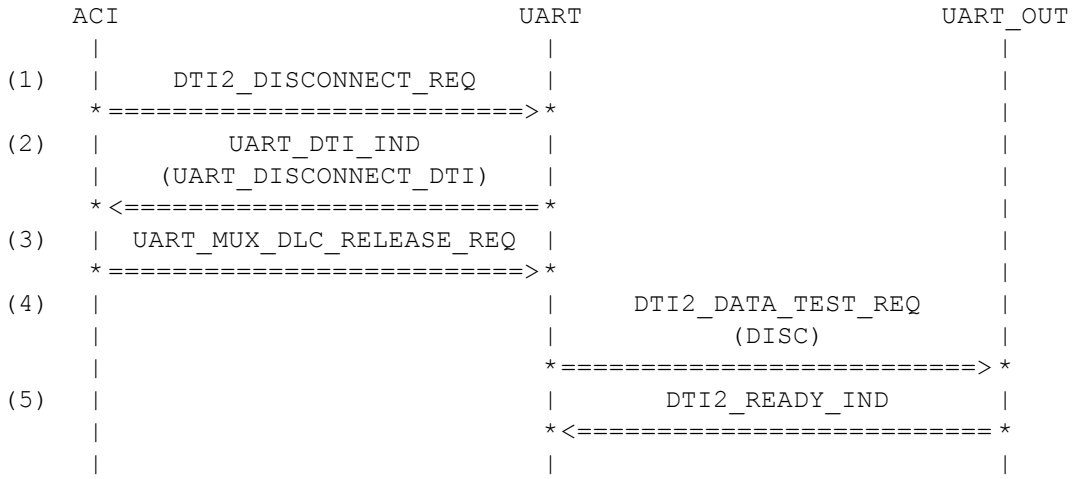
The ACI wants to release a data link connection (DLC) with an allocated data link connection identifier (DLCI). A DISC frame is transmitted in order to release the connection.

Variants:

<A>....<C>

Preamble:

<A> UART127A
<B> UART127C
<C> UART128A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DISCONNECT_REQ		
	link_id	LINK_ID_1
	cause	DTI_CAUSE_NORMAL_CLOSE
(2) UART_DTI_IND		
	dti_conn	UART_DISCONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_A
(3) UART_MUX_DLC_RELEASE_REQ		
	device	UART_DEVICE
	dlci	DLCI_A
(4) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_DISC_DLCI_A_UE2TE

(5)	DTI2_READY_IND		
		link_id	LINK_UART_OUT_PORT_1

History:

12-Oct-2001	TVO	Initial
16-Jan-2003	STW	Error corrections

3.7 Handling of Invalid Frames

3.7.1 UART170: Reception of Corrupted Frame

Description:

The UART entity receives invalid data. It ignores all of it.

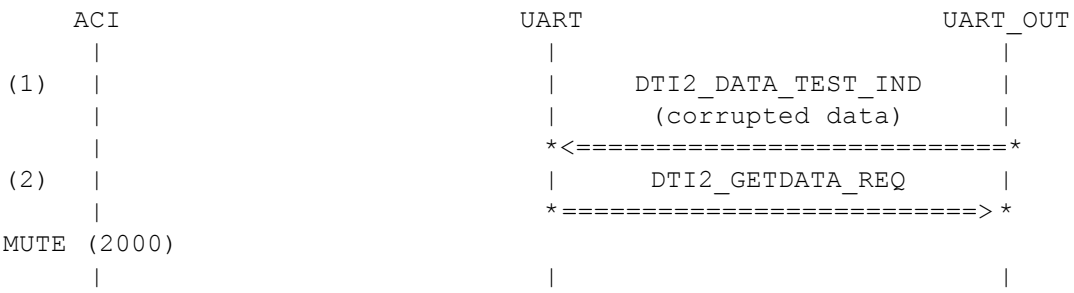
<A>	no flags in data
<B>	upper flag corrupted
<C>	lower flag corrupted
<D>	only flags received
<E>	FCS invalid
<F>	only two octets between flags
<G>	EA bit in address not set
<H>	control octet not valid
<I>	SABM frame with P bit set to 0
<J>	DISC frame with P bit set to 0
<K>	UA frame with F bit set to 0
<L>	EA bit in type octet is not set (type > 1 octet not supported)
<M>	EA bit in length octet is not set (length > 1 octet not supported)
<N>	frame longer than 64 octets

Variants:

<A>....<N>

Preamble:

UART128A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
<A>	sdu	SDU_CORRUPTED_DATA1
<B>	sdu	SDU_CORRUPTED_DATA2

<C>	sdu	SDU_CORRUPTED_DATA3
<D>	sdu	SDU_CORRUPTED_DATA4
<E>	sdu	SDU_CORRUPTED_DATA6
<F>	sdu	SDU_CORRUPTED_DATA7
<G>	sdu	SDU_CORRUPTED_DATA9
<H>	sdu	SDU_CORRUPTED_DATA11
<I>	sdu	SDU_CORRUPTED_DATA12
<J>	sdu	SDU_CORRUPTED_DATA13
<K>	sdu	SDU_CORRUPTED_DATA14
<L>	sdu	SDU_CORRUPTED_DATA16
<M>	sdu	SDU_CORRUPTED_DATA17
<N>	sdu	SDU_CORRUPTED_DATA21
(2) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1

History:

14-Mar-2001	LW	Initial
27-Aug-2001	STW	add frame longer than 64 octets
11-Oct-2001	TVO	adapted for use with DTI2

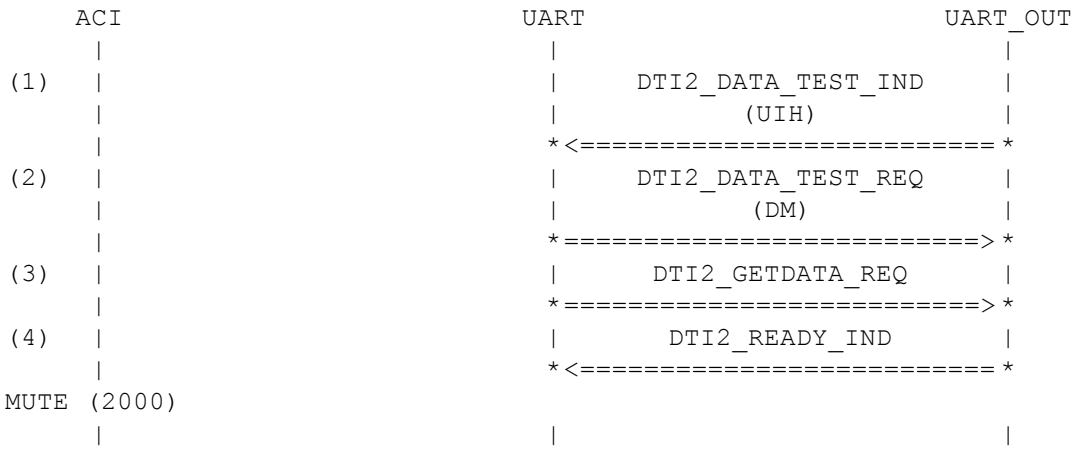
3.7.2 UART171: Reception of Frame for Unknown Channel

Description:

The UART entity receives data for an unknown dlci. A DM frame (type 0F since it is a response for a information and not a command frame) is sent to the peer.

Preamble:

UART128A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_CORRUPTED_DATA8

(2)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
		sdu	SDU_DM_CORRUPTED_DATA8
(3)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(4)	DTI2_READY_IND		
		link_id	LINK_UART_OUT_PORT_1

History:

14-Mar-2001	LW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.7.3 UART172: Reception of Corrupted Message

Description:

The UART entity receives invalid data. It responses an empty UIH Control frame.

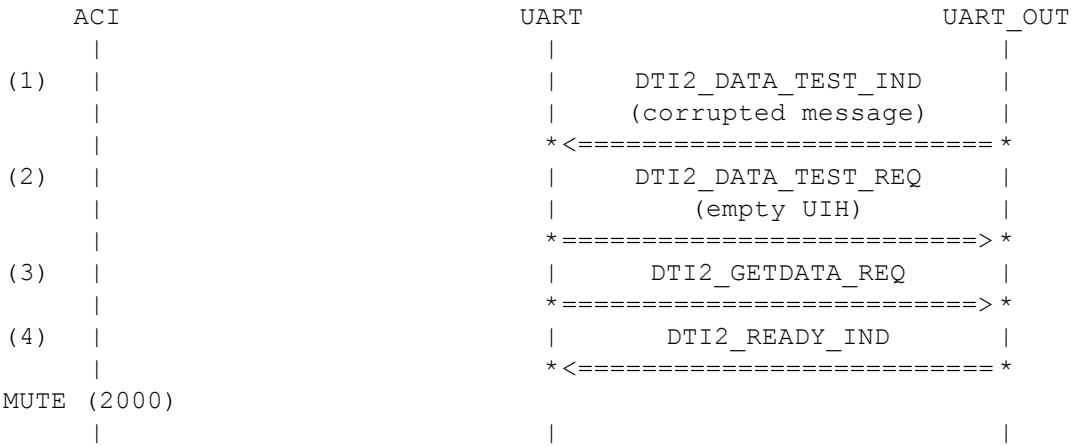
<A>	corrupt length field in message
<B>	message information shorter than length (MSC)
<C>	message information longer than length (MSC)

Variants:

<A>....<C>

Preamble:

UART128A



Parametrization:

	Primitive	Parameter	Value
(1)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_UART_OUT
<A>		sdu	SDU_CORRUPTED_DATA18
<B>		sdu	SDU_CORRUPTED_DATA19
<C>		sdu	SDU_CORRUPTED_DATA20

(2) DTI2_DATA_TEST_REQ

link_id	LINK_WRITEDATA_PORT_1
parameters	DTI_PARAMETER_UART_OUT
sdu	SDU_CORRUPTED_RES_EMPTY

(3) DTI2_GETDATA_REQ

link_id	LINK_READDATA_PORT_1

(4) DTI2_READY_IND

link_id	LINK_UART_OUT_PORT_1

History:

23-Mar-2001	STW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

3.8 HDLC Escape Characters in Multiplexer Frames

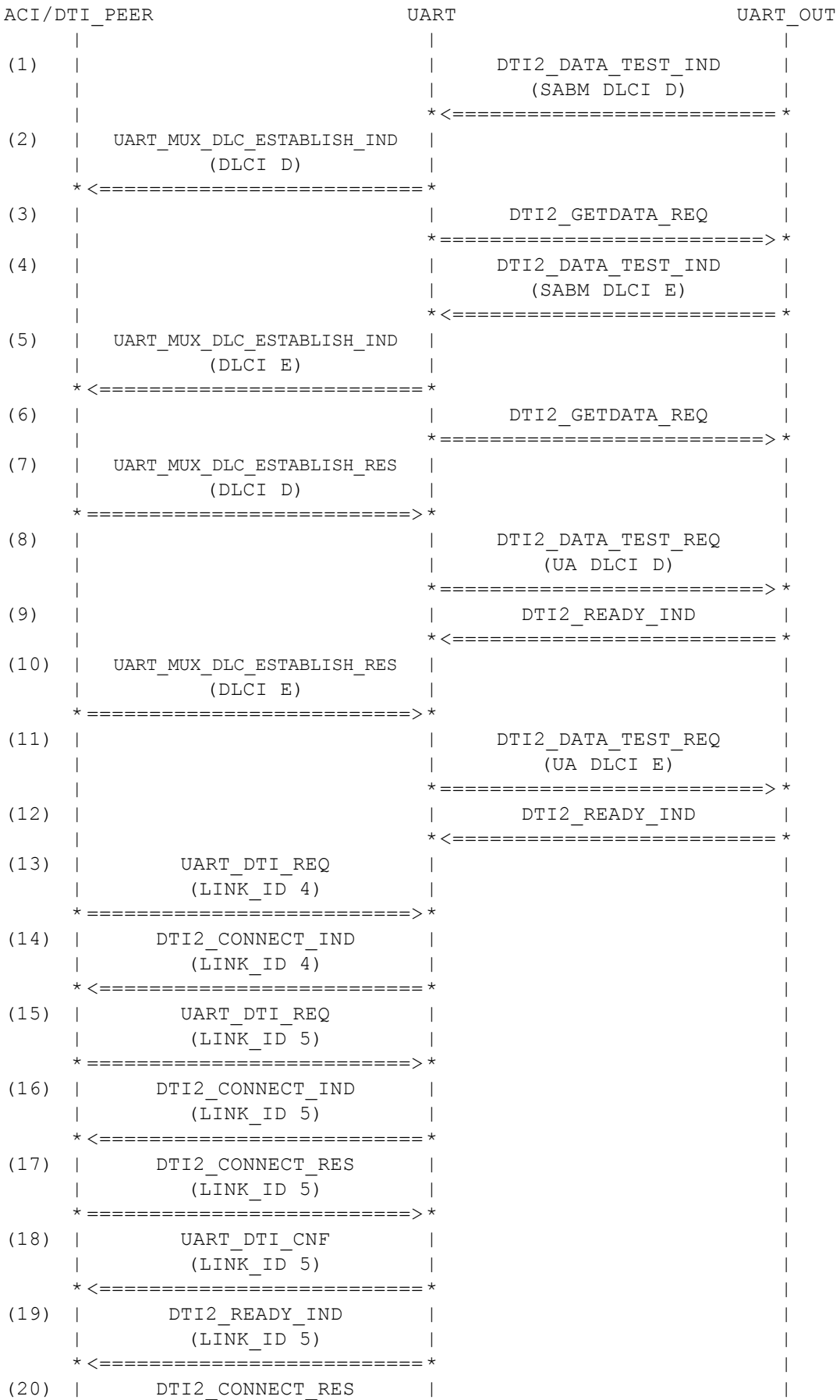
3.8.1 UART180: Establishment of two additional Multiplexer connections

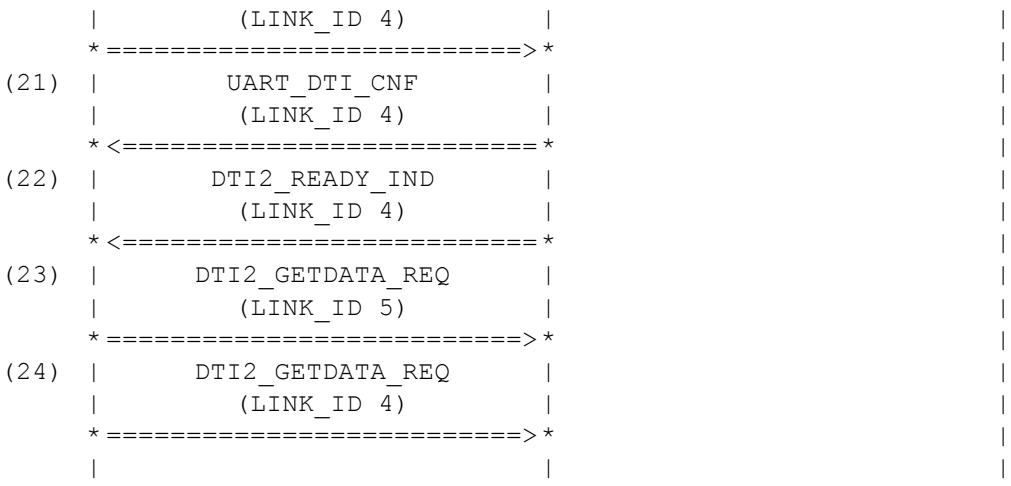
Description:

Two Multiplexer connections are active. In this testcase two additional DLCs are set up. These additional DLCs have special DLCIs to force the use of HDLC escape characters.

Preamble:

UART128B


ACI/DTI_PEER
UART
UART_OUT
(1)
DTI2_DATA_TEST_IND
(SABM DLCI D)
(2)
UART_MUX_DLC_ESTABLISH_IND
(DLCI D)
(3)
DTI2_GETDATA_REQ
(4)
DTI2_DATA_TEST_IND
(SABM DLCI E)
(5)
UART_MUX_DLC_ESTABLISH_IND
(DLCI E)
(6)
DTI2_GETDATA_REQ
(7)
UART_MUX_DLC_ESTABLISH_RES
(DLCI D)
(8)
DTI2_DATA_TEST_REQ
(UA DLCI D)
(9)
DTI2_READY_IND
(10)
UART_MUX_DLC_ESTABLISH_RES
(DLCI E)
(11)
DTI2_DATA_TEST_REQ
(UA DLCI E)
(12)
DTI2_READY_IND
(13)
UART_DTI_REQ
(LINK_ID 4)
(14)
DTI2_CONNECT_IND
(LINK_ID 4)
(15)
UART_DTI_REQ
(LINK_ID 5)
(16)
DTI2_CONNECT_IND
(LINK_ID 5)
(17)
DTI2_CONNECT_RES
(LINK_ID 5)
(18)
UART_DTI_CNF
(LINK_ID 5)
(19)
DTI2_READY_IND
(LINK_ID 5)
(20)
DTI2_CONNECT_RES



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_ESTABLISH_DLCI_D_TE2UE
(2) UART_MUX_DLC_ESTABLISH_IND		
	device	UART_DEVICE
	dlci	DLCI_D
	convergence	UART_MUX_CONVERGENCE_UOS
	n1	UART_MUX_N1_ADVANCED_DEFAULT
	service	UART_MUX_SERVICE_DEFAULT
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_ESTABLISH_DLCI_E_TE2UE
(5) UART_MUX_DLC_ESTABLISH_IND		
	device	UART_DEVICE
	dlci	DLCI_E
	convergence	UART_MUX_CONVERGENCE_UOS
	n1	UART_MUX_N1_ADVANCED_DEFAULT
	service	UART_MUX_SERVICE_DEFAULT
(6) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(7) UART_MUX_DLC_ESTABLISH_RES		
	device	UART_DEVICE
	dlci	DLCI_D
	n1	UART_MUX_N1_ADVANCED_DEFAULT
(8) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1

	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_UA_UE2TE_DLCI_D
(9) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(10) UART_MUX_DLC_ESTABLISH_RES		
	device	UART_DEVICE
	dlci	DLCI_E
	n1	
	UART_MUX_N1_ADVANCED_DEFAULT	
(11) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_UA_UE2TE_DLCI_E
(12) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(13) UART_DTI_REQ		
	dti_conn	UART_CONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_D
	direction	FALSE
	link_id	LINK_ID_4
	entity_name	DTI_CHANNELNAME_PPP
(14) DTI2_CONNECT_IND		
	link_id	LINK_ID_4
	version	DTI_VERSION_10
(15) UART_DTI_REQ		
	dti_conn	UART_CONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_E
	direction	FALSE
	link_id	LINK_ID_5
	entity_name	DTI_CHANNELNAME_PPP
(16) DTI2_CONNECT_IND		
	link_id	LINK_ID_5
	version	DTI_VERSION_10
(17) DTI2_CONNECT_RES		
	link_id	LINK_ID_5
	version	DTI_VERSION_10
(18) UART_DTI_CNF		
	dti_conn	UART_CONNECT_DTI
	device	UART_DEVICE
	dlci	DLCI_E
(19) DTI2_READY_IND		
	link_id	LINK_ID_5
(20) DTI2_CONNECT_RES		
	link_id	LINK_ID_4
	version	DTI_VERSION_10
(21) UART_DTI_CNF		
	dti_conn	UART_CONNECT_DTI

		device	UART_DEVICE
		dlci	DLCI_D
(22)	DTI2_READY_IND		
		link_id	LINK_ID_4
(23)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_5
(24)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_4

History:

23-Jan-2003	STW	Initial

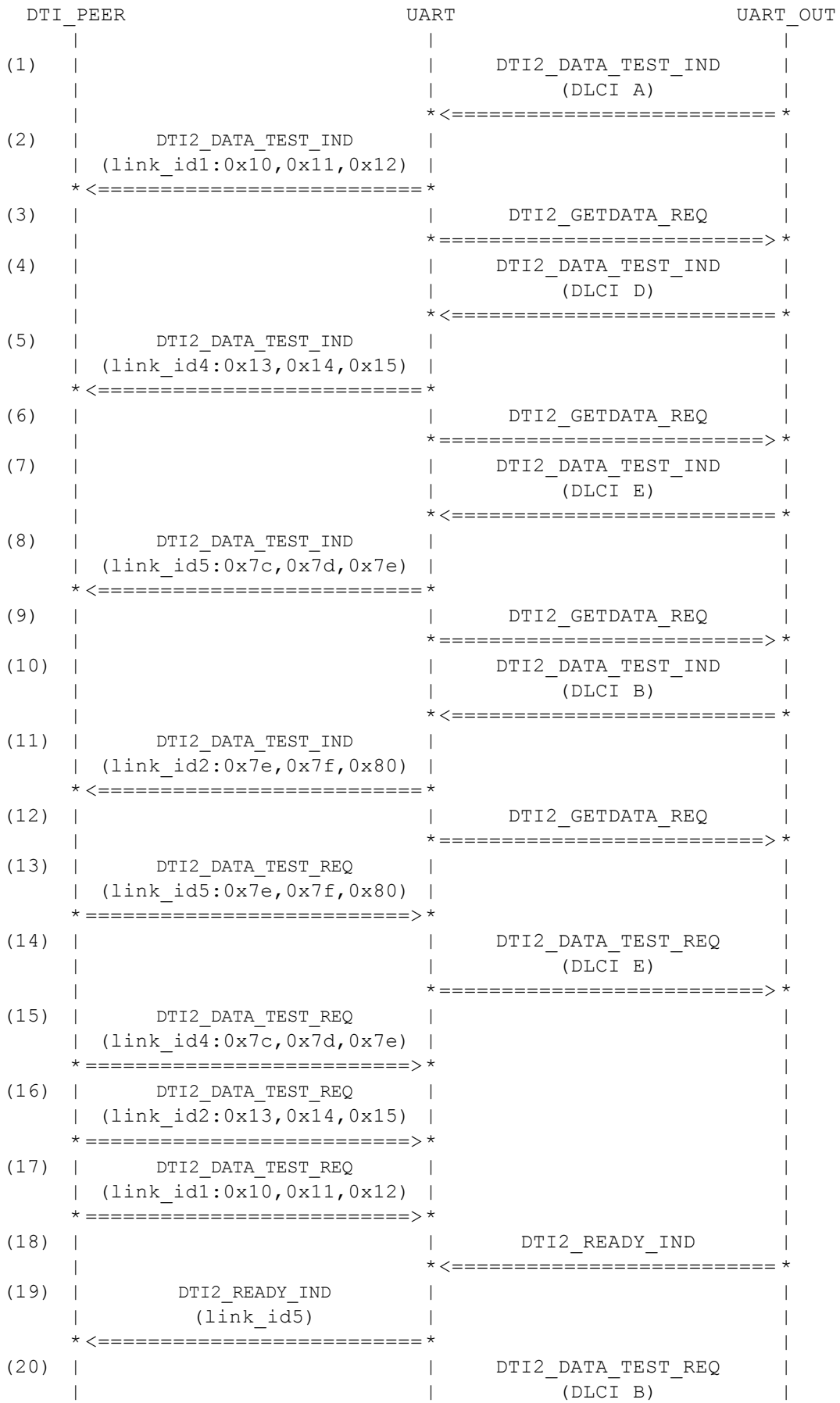
3.8.2 UART181: HDLC Escape character in Address field and Information field

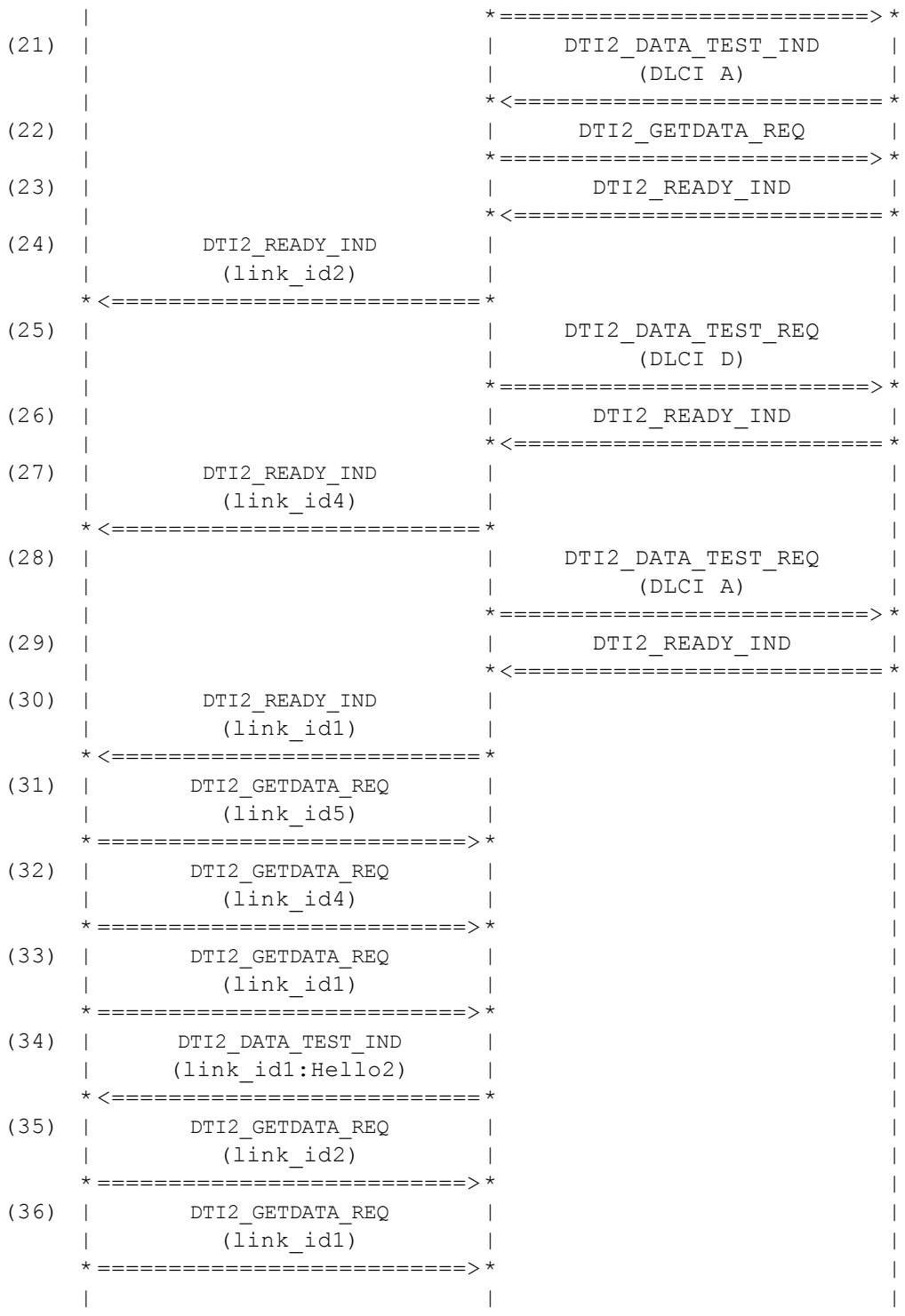
Description:

Mixed data transfer on all active DLCs. The data contains XON (0x11) and XOFF (0x13) characters, the HDLC Escape (0x7d) character and the HDLC Flag (0x7e) character. So the Information field must be extended with HDLC Escape characters. In case of DLCI D and E the Address field must also be extended with HDLC Escape characters.

Preamble:

UART180


DTI_PEER
UART
UART_OUT
(1)
DTI2_DATA_TEST_IND
(DLCI A)
(2)
DTI2_DATA_TEST_IND
(link_id1:0x10,0x11,0x12)
(3)
DTI2_GETDATA_REQ
(4)
DTI2_DATA_TEST_IND
(DLCI D)
(5)
DTI2_DATA_TEST_IND
(link_id4:0x13,0x14,0x15)
(6)
DTI2_GETDATA_REQ
(7)
DTI2_DATA_TEST_IND
(DLCI E)
(8)
DTI2_DATA_TEST_IND
(link_id5:0x7c,0x7d,0x7e)
(9)
DTI2_GETDATA_REQ
(10)
DTI2_DATA_TEST_IND
(DLCI B)
(11)
DTI2_DATA_TEST_IND
(link_id2:0x7e,0x7f,0x80)
(12)
DTI2_GETDATA_REQ
(13)
DTI2_DATA_TEST_REQ
(link_id5:0x7e,0x7f,0x80)
(14)
DTI2_DATA_TEST_REQ
(DLCI E)
(15)
DTI2_DATA_TEST_REQ
(link_id4:0x7c,0x7d,0x7e)
(16)
DTI2_DATA_TEST_REQ
(link_id2:0x13,0x14,0x15)
(17)
DTI2_DATA_TEST_REQ
(link_id1:0x10,0x11,0x12)
(18)
DTI2_READY_IND
(19)
DTI2_READY_IND
(link_id5)
(20)
DTI2_DATA_TEST_REQ
(DLCI B)



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_FRAME_101112_TE2UE_A

(2) DTI2_DATA_TEST_IND

link_id	LINK_ID_1
parameters	DTI_PARAMETER_PEER_A
sdu	STRING_101112_SDU

(3) DTI2_GETDATA_REQ

link_id	LINK_READDATA_PORT_1

(4) DTI2_DATA_TEST_IND

link_id	LINK_UART_OUT_PORT_1
parameters	DTI_PARAMETER_UART_OUT
sdu	SDU_FRAME_131415_TE2UE_D

(5) DTI2_DATA_TEST_IND

link_id	LINK_ID_4
parameters	DTI_PARAMETER_PEER_A
sdu	STRING_131415_SDU

(6) DTI2_GETDATA_REQ

link_id	LINK_READDATA_PORT_1

(7) DTI2_DATA_TEST_IND

link_id	LINK_UART_OUT_PORT_1
parameters	DTI_PARAMETER_UART_OUT
sdu	SDU_FRAME_7c7d7e_TE2UE_E

(8) DTI2_DATA_TEST_IND

link_id	LINK_ID_5
parameters	DTI_PARAMETER_PEER_A
sdu	STRING_7c7d7e_SDU

(9) DTI2_GETDATA_REQ

link_id	LINK_READDATA_PORT_1

(10) DTI2_DATA_TEST_IND

link_id	LINK_UART_OUT_PORT_1
parameters	DTI_PARAMETER_UART_OUT
sdu	SDU_FRAME_7e7f80_TE2UE_B

(11) DTI2_DATA_TEST_IND

link_id	LINK_ID_2
parameters	DTI_PARAMETER_PEER_A
sdu	STRING_7e7f80_SDU

(12) DTI2_GETDATA_REQ

link_id	LINK_READDATA_PORT_1

(13) DTI2_DATA_TEST_REQ

link_id	LINK_ID_5
parameters	DTI_PARAMETER_PEER_B
sdu	STRING_7e7f80_SDU

(14) DTI2_DATA_TEST_REQ

link_id	LINK_WRITEDATA_PORT_1
parameters	DTI_PARAMETER_PEER_B
sdu	SDU_FRAME_7e7f80_UE2TE_E

(15) DTI2_DATA_TEST_REQ

link_id	LINK_ID_4
parameters	DTI_PARAMETER_PEER_B
sdu	STRING_7c7d7e_SDU

(16) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_2
	parameters	DTI_PARAMETER_PEER_B
	sdu	STRING_131415_SDU
(17) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_PEER_B
	sdu	STRING_101112_SDU
(18) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(19) DTI2_READY_IND		
	link_id	LINK_ID_5
(20) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_PEER_B
	sdu	SDU_FRAME_131415_UE2TE_B
(21) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_FRAME_HELLO2_TE2UE_A
(22) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(23) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(24) DTI2_READY_IND		
	link_id	LINK_ID_2
(25) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_PEER_B
	sdu	SDU_FRAME_7c7d7e_UE2TE_D
(26) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(27) DTI2_READY_IND		
	link_id	LINK_ID_4
(28) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_PEER_B
	sdu	SDU_FRAME_101112_UE2TE_A
(29) DTI2_READY_IND		
	link_id	LINK_UART_OUT_PORT_1
(30) DTI2_READY_IND		
	link_id	LINK_ID_1
(31) DTI2_GETDATA_REQ		
	link_id	LINK_ID_5
(32) DTI2_GETDATA_REQ		
	link_id	LINK_ID_4

(33)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1
(34)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_PEER_A
		sdu	STRING_HELLO2_SDU
(35)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_2
(36)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1

History:

24-Jan-2003	STW	Initial

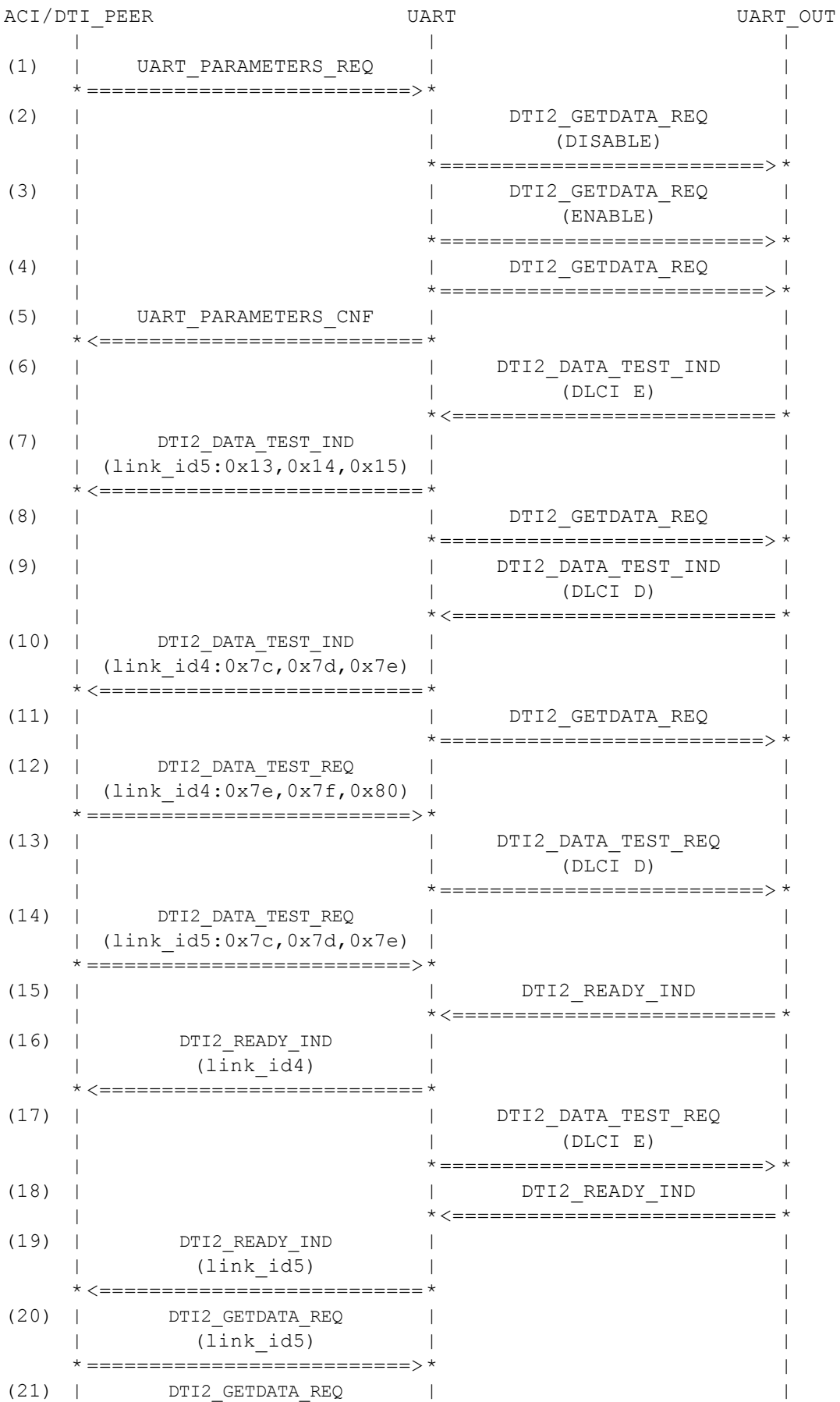
3.8.3 UART182: HDLC Escape character in Control field and FCS field

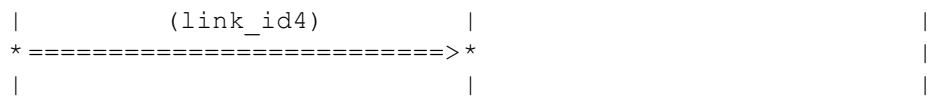
Description:

Mixed data transfer on all active DLCs. The data contains XON (0x11) and XOFF (0x13) characters, the HDLC Escape (0x7d) character and the HDLC Flag (0x7e) character. So the Information field must be extended with HDLC Escape characters. In case of DLCI D and E the Address field must also be extended with HDLC Escape characters.

Preamble:

UART181


ACI/DTI_PEER
UART
UART_OUT
(1) UART_PARAMETERS_REQ
(2) DTI2_GETDATA_REQ (DISABLE)
(3) DTI2_GETDATA_REQ (ENABLE)
(4) DTI2_GETDATA_REQ
(5) UART_PARAMETERS_CNF
(6) DTI2_DATA_TEST_IND (DLCI E)
(7) DTI2_DATA_TEST_IND (link_id5:0x13,0x14,0x15)
(8) DTI2_GETDATA_REQ
(9) DTI2_DATA_TEST_IND (DLCI D)
(10) DTI2_DATA_TEST_IND (link_id4:0x7c,0x7d,0x7e)
(11) DTI2_GETDATA_REQ
(12) DTI2_DATA_TEST_REQ (link_id4:0x7e,0x7f,0x80)
(13) DTI2_DATA_TEST_REQ (DLCI D)
(14) DTI2_DATA_TEST_REQ (link_id5:0x7c,0x7d,0x7e)
(15) DTI2_READY_IND
(16) DTI2_READY_IND (link_id4)
(17) DTI2_DATA_TEST_REQ (DLCI E)
(18) DTI2_READY_IND
(19) DTI2_READY_IND (link_id5)
(20) DTI2_GETDATA_REQ (link_id5)
(21) DTI2_GETDATA_REQ



Parametrization:

Primitive	Parameter	Value
(1) UART_PARAMETERS_REQ		
	device	UART_DEVICE
	comPar	COMPAR_XON_CTRL_XOFF_FCS
(2) DTI2_GETDATA_REQ		
	link_id	LINK_DISABLE_PORT_1
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ENABLE_PORT_1
(4) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(5) UART_PARAMETERS_CNF		
	device	UART_DEVICE
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_FRAME_131415_TE2UE_E_X
(7) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_5
	parameters	DTI_PARAMETER_PEER_A
	sdu	STRING_131415_SDU
(8) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(9) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_UART_OUT
	sdu	SDU_FRAME_7c7d7e_TE2UE_D_X
(10) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_4
	parameters	DTI_PARAMETER_PEER_A
	sdu	STRING_7c7d7e_SDU
(11) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(12) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_4
	parameters	DTI_PARAMETER_PEER_B
	sdu	STRING_7e7f80_SDU
(13) DTI2_DATA_TEST_REQ		
	link_id	LINK_WRITEDATA_PORT_1
	parameters	DTI_PARAMETER_PEER_B
	sdu	SDU_FRAME_7e7f80_UE2TE_D_X
(14) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_5

		parameters	DTI_PARAMETER_PEER_B
		sdu	STRING_7c7d7e_SDU
(15)	DTI2_READY_IND		
		link_id	LINK_UART_OUT_PORT_1
(16)	DTI2_READY_IND		
		link_id	LINK_ID_4
(17)	DTI2_DATA_TEST_REQ		
		link_id	LINK_WRITEDATA_PORT_1
		parameters	DTI_PARAMETER_PEER_B
		sdu	SDU_FRAME_7c7d7e_UE2TE_E_X
(18)	DTI2_READY_IND		
		link_id	LINK_UART_OUT_PORT_1
(19)	DTI2_READY_IND		
		link_id	LINK_ID_5
(20)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_5
(21)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_4

History:

24-Jan-2003	STW	Initial

4 Escape Sequence Tests (UART300 - UART400)

4.1 Setup Escape Sequence Tests

4.1.1 UART300: Additional Setup for Escape Sequence Tests

Description:

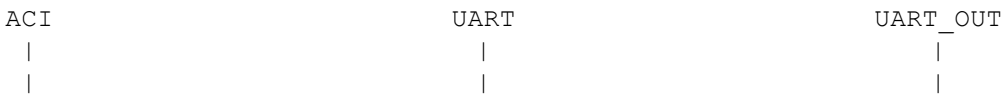
This test case may be used for additional parameter setup which is only required in Escape Sequence tests. Currently by ACI the Escape Sequence character and the Guard Period are set within UART_PARAMETERS_REQ.

Variants:

<A>....<B>

Preamble:

<A>UART020B
<B>UART128B



Parametrization:

Primitive	Parameter	Value

History:

19-Jul-2001	IH	Initial
11-Oct-2001	TVO	adapted for use with DTI2
16-Oct-2001	TVO	moved parameters to UART_PARAMETERS_REQ

4.2 Errors During Escape Sequence Detection

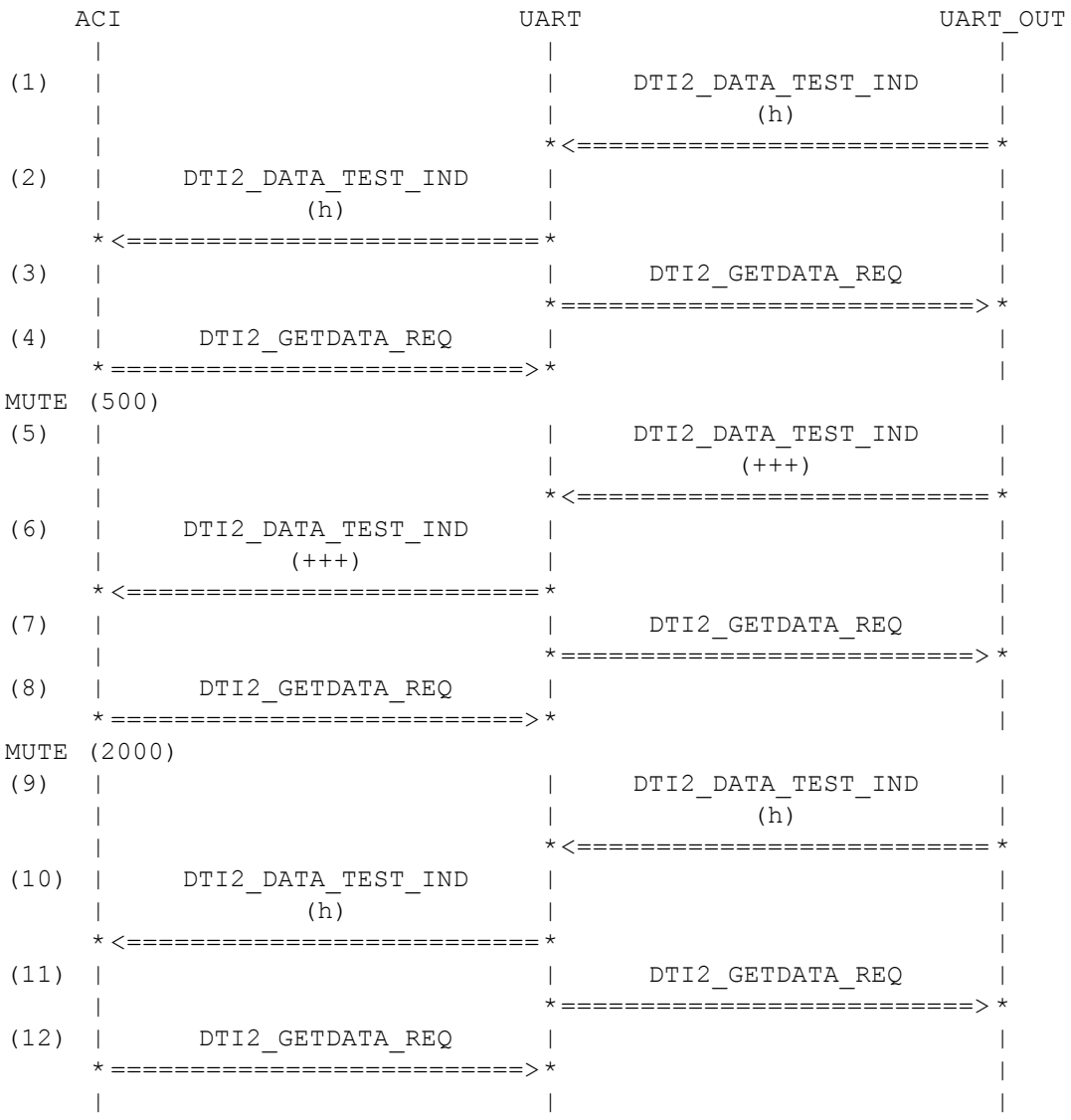
4.2.1 UART301: Leading Guard Period Too Short

Description:

The UART receives the Escape Sequence character, but the Guard Period between the last data (h) and the first Escape Sequence character was shorter than requested with the element "guardPeriod" of the UART_PARAMETERS_REQ primitive. The state of the Escape Sequence Detection is reset.
Within the second timeout no DTI2_DATA_TEST_IND primitive (indicating the Escape Sequence Detection) is expected.

Preamble:

UART300A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(5) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1

	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_ESC_SEQ_SDU
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_ESC_SEQ_SDU
(7) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(8) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(9) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(10) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(11) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(12) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1

History:

19-Jul-2001	IH	Initial
11-Sep-2001	STW	remove escape sequence if detected
11-Oct-2001	TVO	adapted for use with DTI2

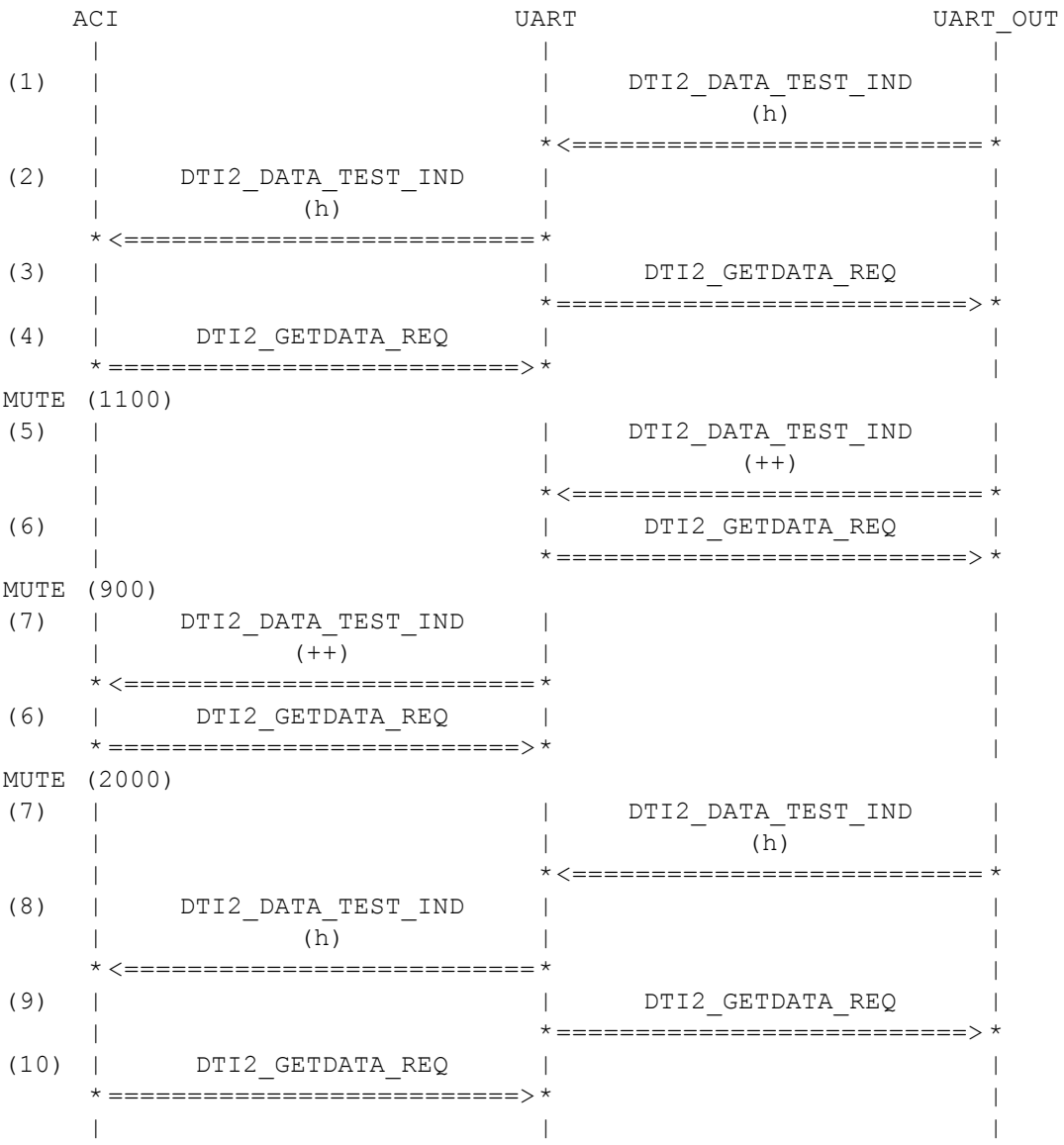
4.2.2 UART302: Escape Sequence Character Mismatch

Description:

After the minimum Guard Period the first Escape Sequence characters are received. The third character does not fit into the Escape Sequence (not an Escape Sequence character). The state of the Escape Sequence Detection is reset.
The third timeout afterwards (trailing guard period) confirms that no DTI2_DATA_TEST_IND primitive (indicating the Escape Sequence Detection) is received.

Preamble:

UART300A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1

(5)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_PLUSPLUS_SDU
(6)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(7)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_PLUSPLUS_SDU
(8)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1
(9)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_H_SDU
(10)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_H_SDU
(11)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(12)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1

History:

23-Jul-2001	IH	Initial
11-Sep-2001	STW	remove escape sequence if detected
11-Oct-2001	TVO	adapted for use with DTI2

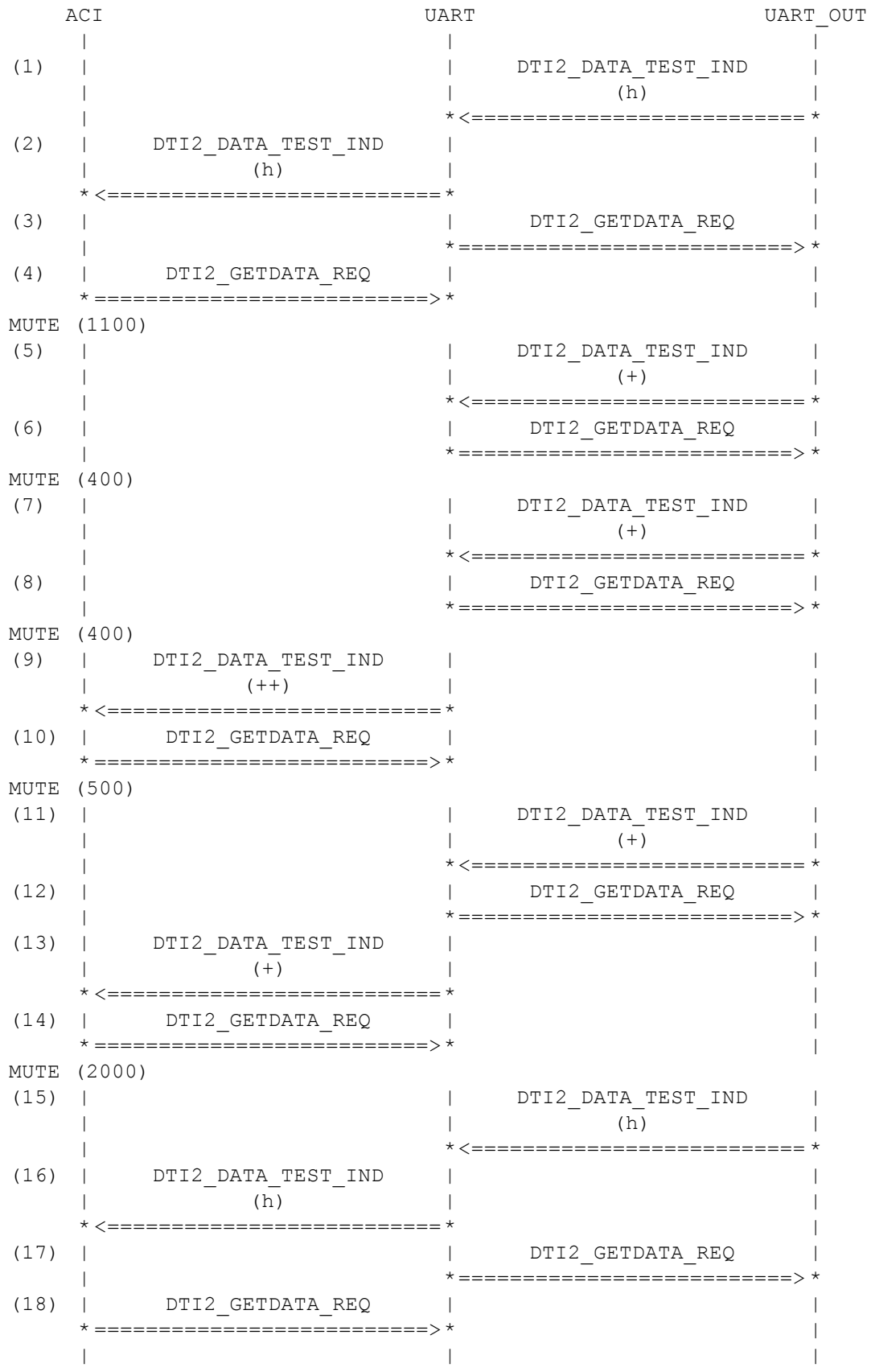
4.2.3 UART303: Escape Sequence Characters Duration Too Long

Description:

The three Escape Sequence characters are received within a period of time which is longer than the Guard Period. The state of the Escape Sequence Detection is reset.
Within the final timeout (trailing guard period) a DTI2_DATA_TEST_IND primitive (indicating the Escape Sequence Detection) must not be received.

Preamble:

UART300A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(5) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_PLUS_SDU
(6) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(7) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_PLUS_SDU
(8) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(9) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_PLUSPLUS_SDU
(10) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(11) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_PLUS_SDU
(12) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(13) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_PLUS_SDU
(14) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(15) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU

(16)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_H_SDU
(17)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(18)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1

History:

23-Jul-2001		IH Initial
11-Sep-2001	STW	remove escape sequence if detected
11-Oct-2001	TVO	adapted for use with DTI2 interchanged (12) and (13)

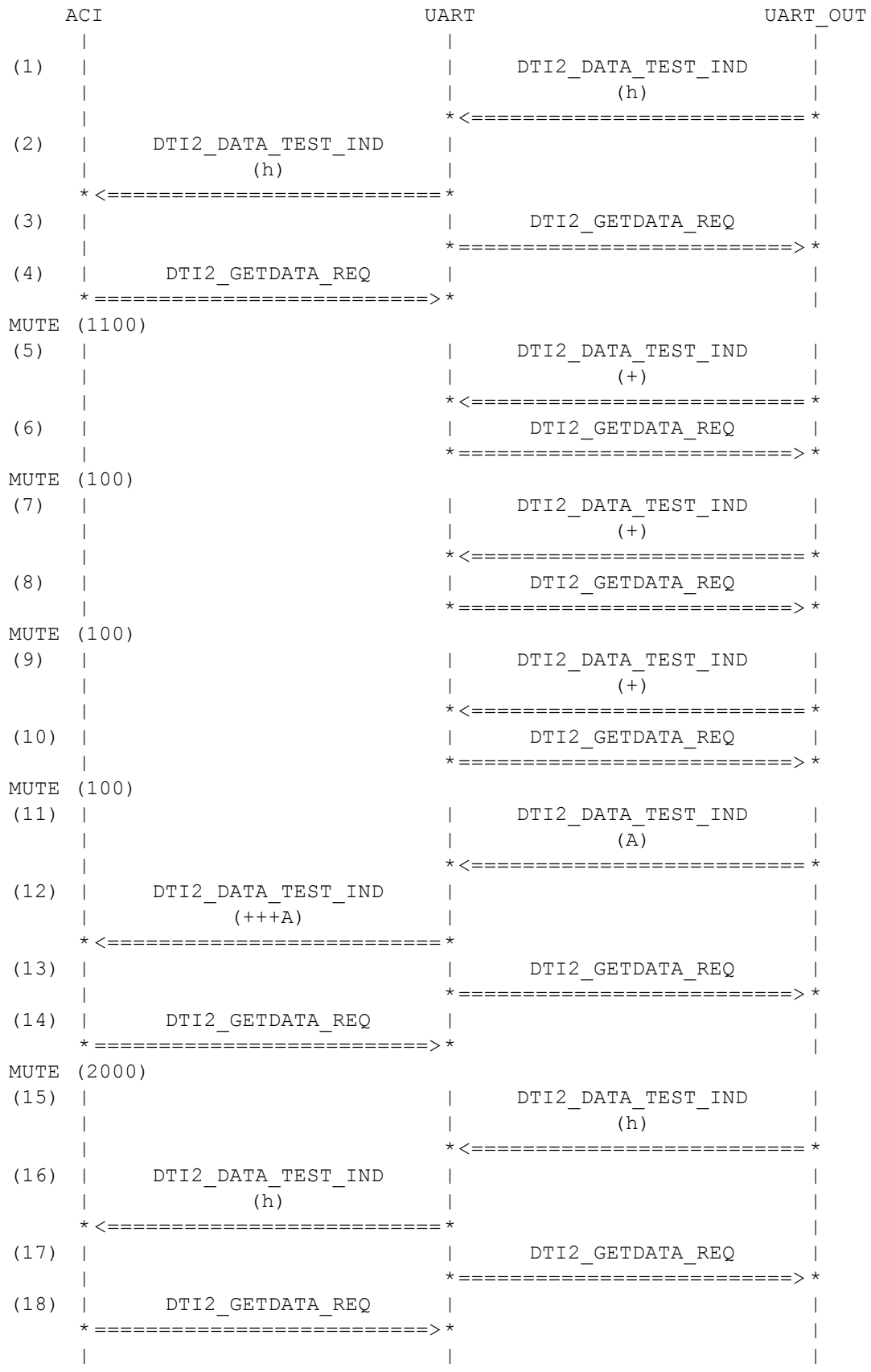
4.2.4 UART304: Trailing Guard Period Too Short

Description:

First Guard Period and reception of the Escape Sequence characters are in time. The first character of the following AT command however, is received within the trailing guard period. The state of the Escape Sequence Detection is reset. The DTI2_DATA_TEST_IND primitive (indicating the Escape Sequence Detection) must not be received within the final timeout.

Preamble:

UART300A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(5) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_PLUS_SDU
(6) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(7) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_PLUS_SDU
(8) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(9) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_PLUS_SDU
(10) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(11) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_A_SDU
(12) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_3PLUSA_SDU
(13) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(14) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(15) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU

(16)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_H_SDU
(17)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(18)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1

History:

23-Jul-2001	IH	Initial
11-Sep-2001	STW	remove escape sequence if detected
11-Oct-2001	TVO	adapted for use with DTI2

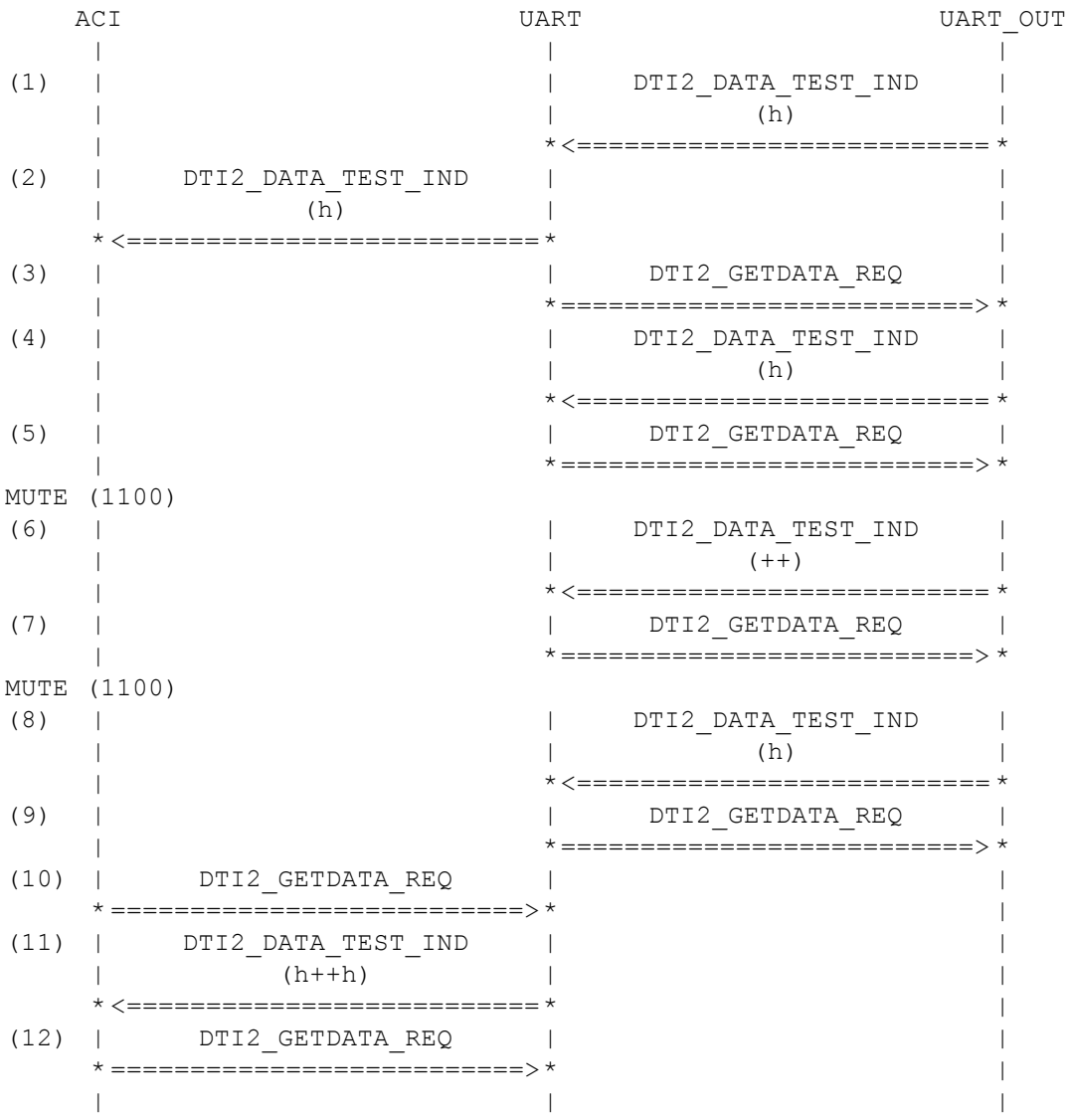
4.2.5 UART305: Escape Sequence Character Mismatch without Flow Control primitive

Description:

After the minimum Guard Period the first Escape Sequence characters are received. The third character does not fit into the Escape Sequence (not an Escape Sequence character). The state of the Escape Sequence Detection is reset.
Because the upper layer does not send a Flow Control pirimive, UART has to collect one previous sent character, two escape sequence characters and one character sent afterwards.

Preamble:

UART300A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU

(5) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_PLUSPLUS_SDU
(7) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(8) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(9) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(10) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(11) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_HPLUSPLUSH_SDU
(12) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1

History:

12-Sep-2001	STW	Initial
11-Oct-2001	TVO	adapted for use with DTI2

4.3 Successful Escape Sequence Detection

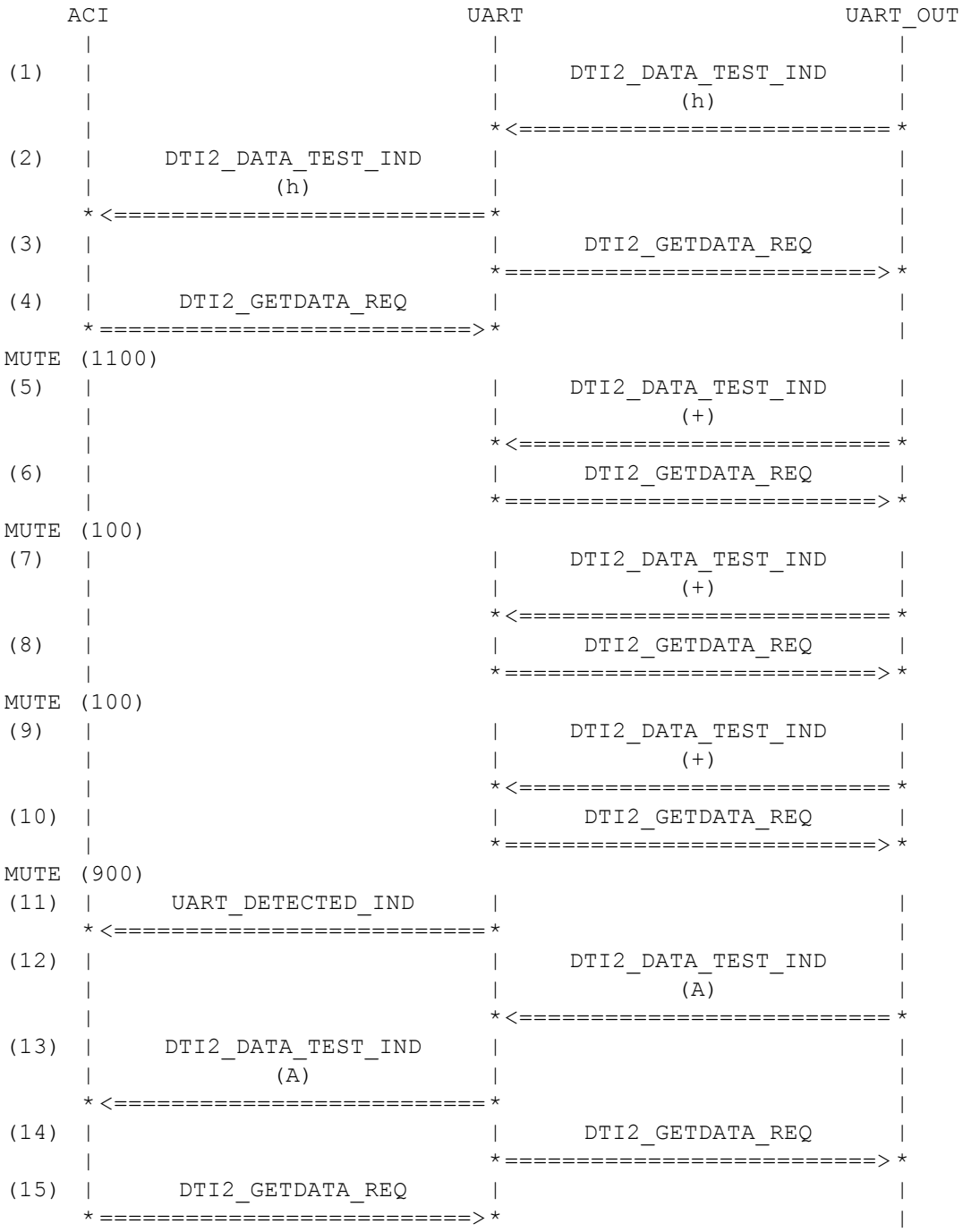
4.3.1 UART310: Single Connection

Description:

An Escape Sequence is detected on a single Multiplexer connection. The occurence of the Escape Sequence is indicated to upper layers with an UART_DETECTED_IND primitive. The first character of the following AT command, is forwarded with the next DTI2_DATA_IND primitive.

Preamble:

UART300A



| | | |

Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_H_SDU
(3) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1
(5) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_PLUS_SDU
(6) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(7) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_PLUS_SDU
(8) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(9) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_PLUS_SDU
(10) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(11) UART_DETECTED_IND		
	device	UART_DEVICE
	dlci	DLCI_CONTROL
	cause	UART_DETECT_ESC
(12) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_A_SDU
(13) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_1
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_A_SDU
(14) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1

(15) DTI2_GETDATA_REQ

link_id	LINK_ID_1

History:

23-Jul-2001	IH	Initial
11-Sep-2001	STW	remove escape sequence if detected
11-Oct-2001	TVO	adapted for use with DTI2

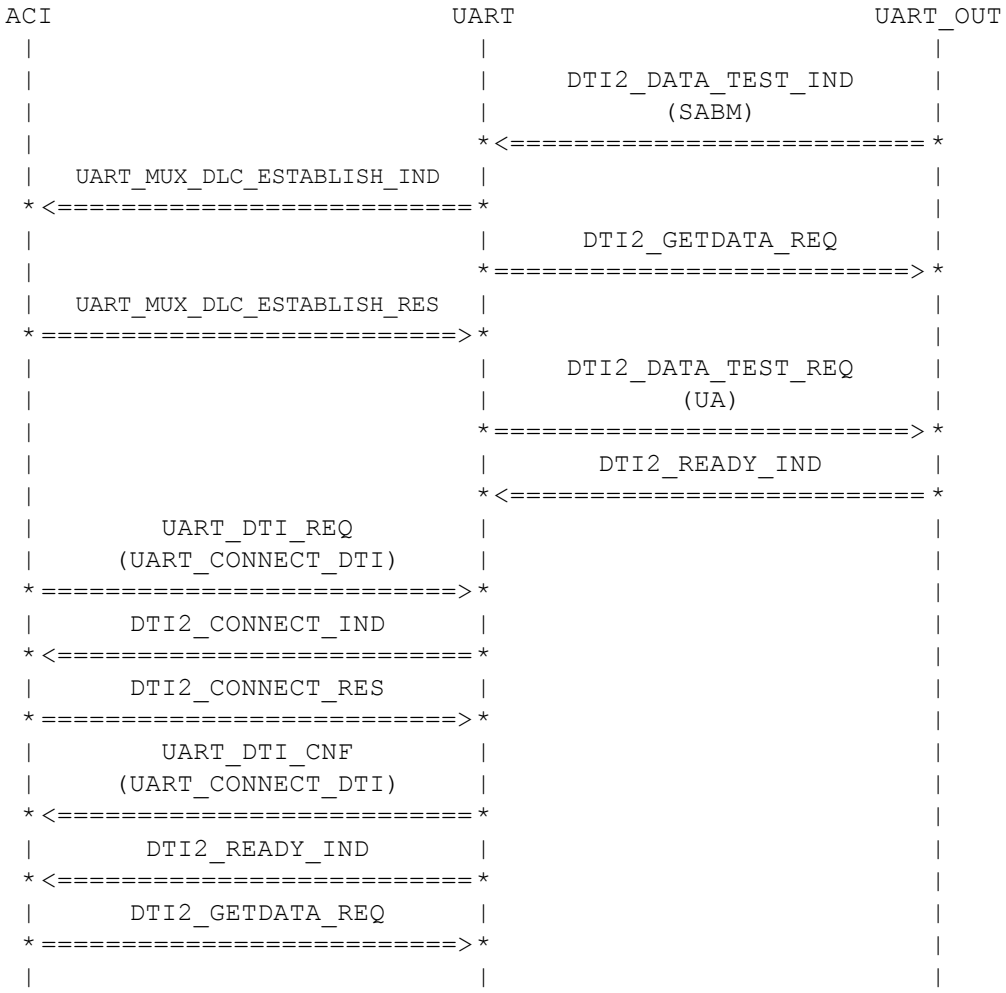
4.3.2 UART311: Establishment of A Third Multiplexer Connection

Description:

Two Multiplexer connections are active. In this testcase a third DLC is set up. The DTI connection is set and the Flow Control primitive DTI2_GETDATA_REQ is sent to enable data reception.

Preamble:

UART300B



Parametrization:

Primitive	Parameter	Value

(1) DTI2_DATA_TEST_IND

link_id	LINK_UART_OUT_PORT_1
parameters	DTI_PARAMETER_UART_OUT
sdu	SDU_ESTABLISH_DLCI_C_TE2UE

(2) UART_MUX_DLC_ESTABLISH_IND

device	UART_DEVICE
dlci	DLCI_C
convergence	UART_MUX_CONVERGENCE_UOS
n1	UART_MUX_N1_ADVANCED_DEFAULT
service	UART_MUX_SERVICE_DEFAULT

(3) DTI2_GETDATA_REQ

link_id	LINK_READDATA_PORT_1

(4) UART_MUX_DLC_ESTABLISH_RES

device	UART_DEVICE
dlci	DLCI_C
n1	UART_MUX_N1_ADVANCED_DEFAULT

(5) DTI2_DATA_TEST_REQ

link_id	LINK_WRITEDATA_PORT_1
parameters	DTI_PARAMETER_UART_OUT
sdu	SDU_UA_UE2TE_DLCI_C

(6) DTI2_READY_IND

link_id	LINK_UART_OUT_PORT_1

(7) UART_DTI_REQ

dti_conn	UART_CONNECT_DTI
device	UART_DEVICE
dlci	DLCI_C
direction	FALSE
link_id	LINK_ID_3
entity_name	DTI_CHANNELNAME_PPP

(8) DTI2_CONNECT_IND

link_id	LINK_ID_3
version	DTI_VERSION_10

(9) DTI2_CONNECT_RES

link_id	LINK_ID_3
version	DTI_VERSION_10

(10) UART_DTI_CNF

dti_conn	UART_CONNECT_DTI
device	UART_DEVICE
dlci	DLCI_C

(11) DTI2_READY_IND

link_id	LINK_ID_3

(12) DTI2_GETDATA_REQ

link_id	LINK_ID_3

History:

27-Jul-2001	IH	Initial
11-Oct-2001	TVO	adapted for use with DTI2

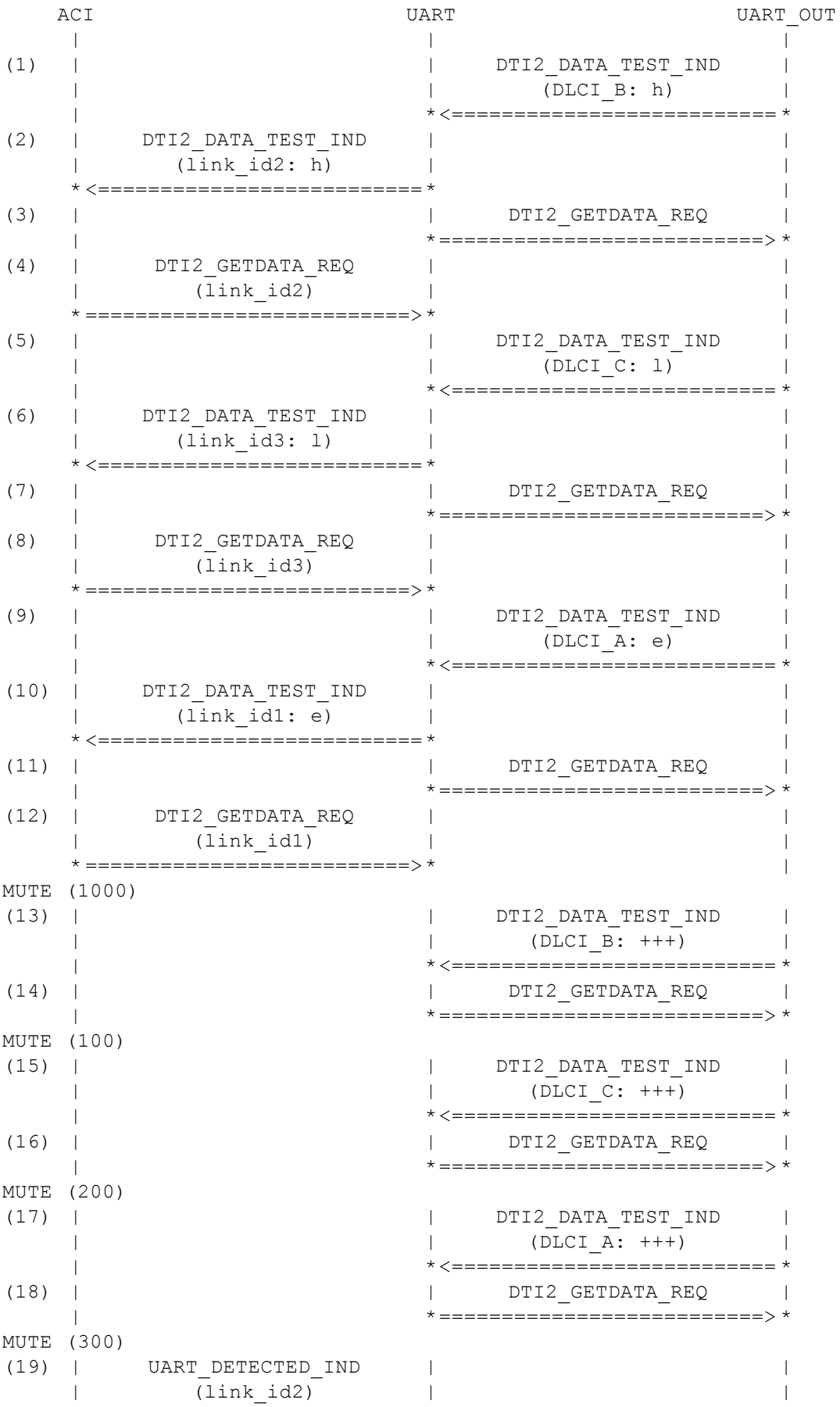
4.3.3 UART312: Escape Sequence on Three DLCs

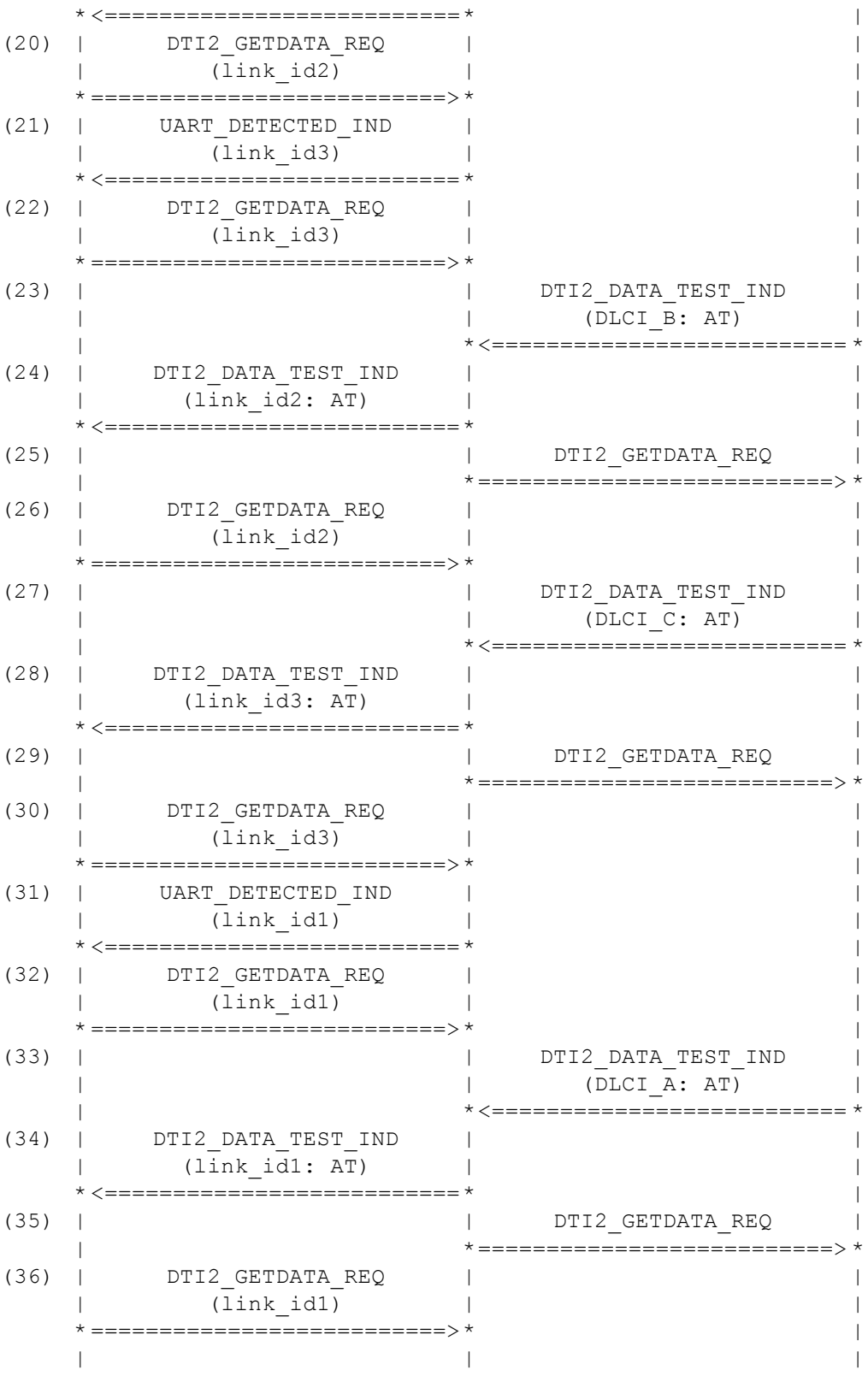
Description:

Three DLC are active in multiplexer mode of the UART. After the receipt of the last byte of data on each of the DLCs the first Guard Period is detected. The Escape Sequence characters and the trailing Guard Period are detected on each of the three Multiplexer connections nearly simultaneous. The order is DLCI_B, DLCI_C and DLCI_A.
The occurence of the Escape Sequence is indicated to upper layers with an empty DTI2_DATA_IND primitive. On DLCI_A the DTI2_GETDATA_REQ primitive is not sent after receipt of the last byte of data (e). That's why the DTI2_DATA_IND primitive is sent delayed.
The following AT commands, are forwarded with the DTI2_DATA_IND primitives after the Flow Control primitives DTI2_GETDATA_REQ.

Preamble:

UART311


ACI
UART
UART_OUT
(1)
DTI2_DATA_TEST_IND
(DLCI_B: h)
(2)
DTI2_DATA_TEST_IND
(link_id2: h)
(3)
DTI2_GETDATA_REQ
(4)
DTI2_GETDATA_REQ
(link_id2)
(5)
DTI2_DATA_TEST_IND
(DLCI_C: l)
(6)
DTI2_DATA_TEST_IND
(link_id3: l)
(7)
DTI2_GETDATA_REQ
(8)
DTI2_GETDATA_REQ
(link_id3)
(9)
DTI2_DATA_TEST_IND
(DLCI_A: e)
(10)
DTI2_DATA_TEST_IND
(link_id1: e)
(11)
DTI2_GETDATA_REQ
(12)
DTI2_GETDATA_REQ
(link_id1)
MUTE (1000)
(13)
DTI2_DATA_TEST_IND
(DLCI_B: +++)
(14)
DTI2_GETDATA_REQ
MUTE (100)
(15)
DTI2_DATA_TEST_IND
(DLCI_C: +++)
(16)
DTI2_GETDATA_REQ
MUTE (200)
(17)
DTI2_DATA_TEST_IND
(DLCI_A: +++)
(18)
DTI2_GETDATA_REQ
MUTE (300)
(19)
UART_DETECTED_IND
(link_id2)



Parametrization:

Primitive	Parameter	Value

(1) DTI2_DATA_TEST_IND

link_id	LINK_UART_OUT_PORT_1
parameters	DTI_PARAMETER_FRAME
sdu	SDU_FRAME_H_TE2UE_B

(2) DTI2_DATA_TEST_IND

link_id	LINK_ID_2
parameters	DTI_PARAMETER_FRAME
sdu	STRING_H_SDU

(3) DTI2_GETDATA_REQ

link_id	LINK_READDATA_PORT_1

(4) DTI2_GETDATA_REQ

link_id	LINK_ID_2

(5) DTI2_DATA_TEST_IND

link_id	LINK_UART_OUT_PORT_1
parameters	DTI_PARAMETER_FRAME
sdu	SDU_FRAME_L_TE2UE_C

(6) DTI2_DATA_TEST_IND

link_id	LINK_ID_3
parameters	DTI_PARAMETER_FRAME
sdu	STRING_L_SDU

(7) DTI2_GETDATA_REQ

link_id	LINK_READDATA_PORT_1

(8) DTI2_GETDATA_REQ

link_id	LINK_ID_3

(9) DTI2_DATA_TEST_IND

link_id	LINK_UART_OUT_PORT_1
parameters	DTI_PARAMETER_FRAME
sdu	SDU_FRAME_E_TE2UE_A

(10) DTI2_DATA_TEST_IND

link_id	LINK_ID_1
parameters	DTI_PARAMETER_FRAME
sdu	STRING_E_SDU

(11) DTI2_GETDATA_REQ

link_id	LINK_READDATA_PORT_1

(12) DTI2_GETDATA_REQ

link_id	LINK_ID_1

(13) DTI2_DATA_TEST_IND

link_id	LINK_UART_OUT_PORT_1
parameters	DTI_PARAMETER_FRAME
sdu	SDU_FRAME_ESC_SEQ_B

(14) DTI2_GETDATA_REQ

link_id	LINK_READDATA_PORT_1

(15) DTI2_DATA_TEST_IND

link_id	LINK_UART_OUT_PORT_1
parameters	DTI_PARAMETER_FRAME
sdu	SDU_FRAME_ESC_SEQ_C

(16) DTI2_GETDATA_REQ

link_id	LINK_READDATA_PORT_1

(17) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	SDU_FRAME_ESC_SEQ_A
(18) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(19) UART_DETECTED_IND		
	device	UART_DEVICE
	dlci	DLCI_B
	cause	UART_DETECT_ESC
(20) DTI2_GETDATA_REQ		
	link_id	LINK_ID_2
(21) UART_DETECTED_IND		
	device	UART_DEVICE
	dlci	DLCI_C
	cause	UART_DETECT_ESC
(22) DTI2_GETDATA_REQ		
	link_id	LINK_ID_3
(23) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	SDU_FRAME_AT_TE2UE_B
(24) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_2
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_AT_SDU
(25) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(26) DTI2_GETDATA_REQ		
	link_id	LINK_ID_2
(27) DTI2_DATA_TEST_IND		
	link_id	LINK_UART_OUT_PORT_1
	parameters	DTI_PARAMETER_FRAME
	sdu	SDU_FRAME_AT_TE2UE_C
(28) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_3
	parameters	DTI_PARAMETER_FRAME
	sdu	STRING_AT_SDU
(29) DTI2_GETDATA_REQ		
	link_id	LINK_READDATA_PORT_1
(30) DTI2_GETDATA_REQ		
	link_id	LINK_ID_3
(31) UART_DETECTED_IND		
	device	UART_DEVICE
	dlci	DLCI_A
	cause	UART_DETECT_ESC
(32) DTI2_GETDATA_REQ		
	link_id	LINK_ID_1

(33)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_FRAME
		sdu	SDU_FRAME_AT_TE2UE_A
(34)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_AT_SDU
(35)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(36)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1

History:

27-Jul-2001	IH	Initial
11-Sep-2001	STW	remove escape sequence if detected
11-Oct-2001	TVO	adapted for use with DTI2
16-Jan-2003	STW	Error corrections

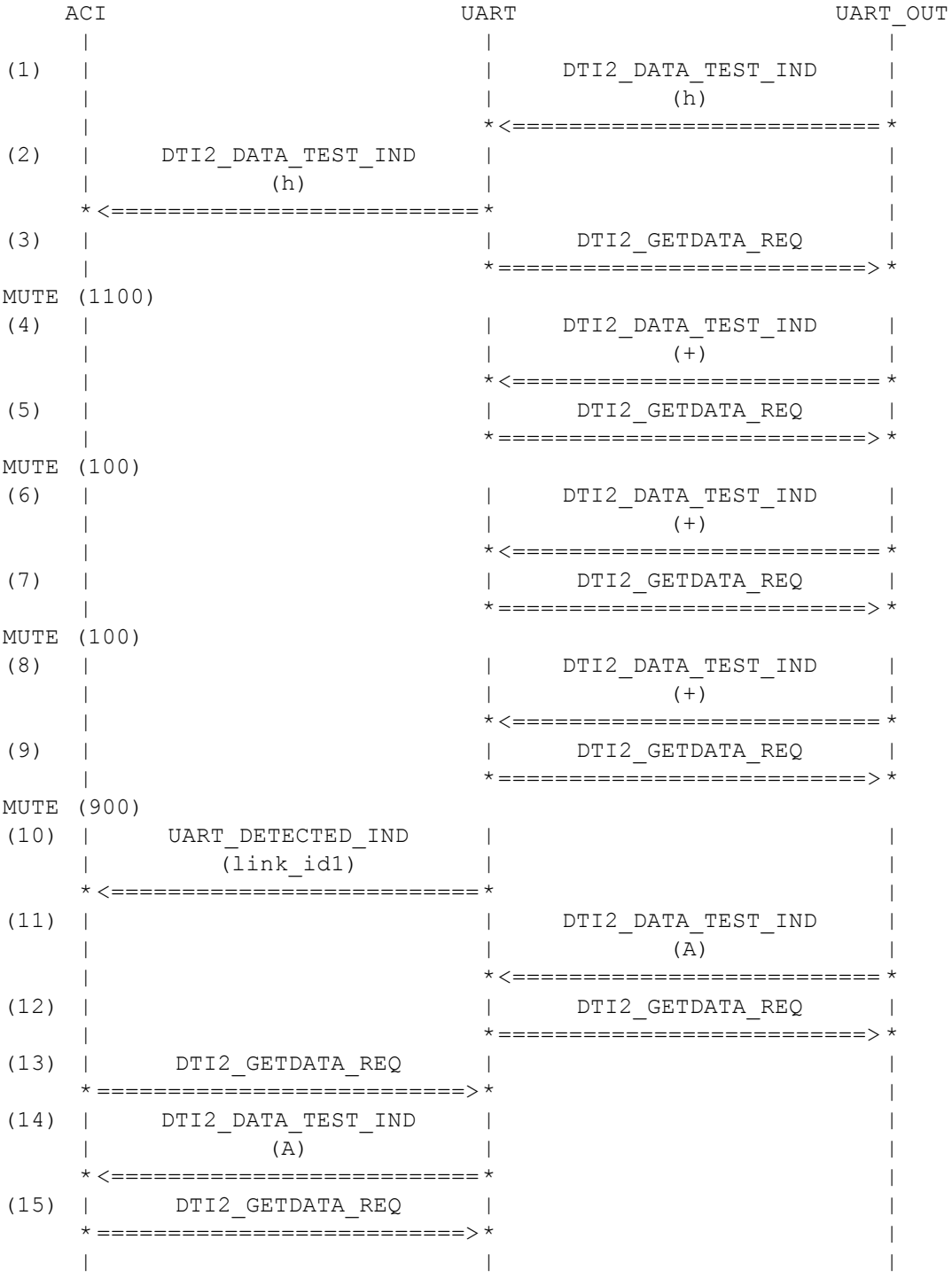
4.3.4 UART313: Single Connection without FLow Control primitive

Description:

An Escape Sequence is detected on a single connection. In former versions of the uart SAP, the occurence of the Escape Sequence could not be indicated immediately to upper layers because there was no Flow Control primive from the upper layer. Escape Sequence and subsequent data were forwarded with one DTI2_DATA_IND primitive. Now, this is basically a simpler form of UART312.

Preamble:

UART300A



Parametrization:

	Primitive	Parameter	Value
(1)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_H_SDU
(2)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_H_SDU
(3)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(4)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_PLUS_SDU
(5)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(6)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_PLUS_SDU
(7)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(8)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_PLUS_SDU
(9)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(10)	UART_DETECTED_IND		
		device	UART_DEVICE
		dlci	DLCI_CONTROL
		cause	UART_DETECT_ESC
(11)	DTI2_DATA_TEST_IND		
		link_id	LINK_UART_OUT_PORT_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_A_SDU
(12)	DTI2_GETDATA_REQ		
		link_id	LINK_READDATA_PORT_1
(13)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_1
(14)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_1
		parameters	DTI_PARAMETER_FRAME
		sdu	STRING_A_SDU

(15) DTI2_GETDATA_REQ

link_id	LINK_ID_1

History:

12-Sep-2001	STW	Initial
11-Oct-2001	TVO	adapted for use with DTI2