

**Infrared Data Association
Infrared Universal Translator (IrUT)
For High-Speed Radio**



NEC
NTT DoCoMo
Matsushita/Panasonic
Sharp

Version 1.0

October 26, 1999

Authors:

Yusuke Kushida, kushida@cet.yrp.nttdocomo.co.jp (NTT DoCoMo)

Kazuya Anzawa, anzawa@cet.yrp.nttdocomo.co.jp (NTT DoCoMo)

Katsunori Hamada, hamada@cet.yrp.nttdocomo.co.jp (NTT DoCoMo)

Contributors:

Noriyuki Ogawa, oga@milh.mei.co.jp (Matsushita/Panasonic)

Kuniaki Sugimoto, sugimoto@milh.mei.co.jp (Matsushita/Panasonic)

Shuichiro Ono, ono@shpcsl.sharp.co.jp (SHARP)

Hiroshi Ono, ono@mcd.yh.nec.co.jp (NEC)

Special thanks to

Robert K.Lockhart, lob.lockhart@mot.com (Motorola)

Editor:

Yusuke Kushida

Document Status:

- Infrared Universal Translator (IrUT) for High-speed Radio version 1.0 (October25, 1999)

INFRARED DATA ASSOCIATION (IrDA) - NOTICE TO THE TRADE -SUMMARY:

Following is the notice of conditions and understandings upon which this document is made available to members and non-members of the Infrared Data Association. Availability of Publications, Updates and Notices

- Full Copyright Claims Must be Honored
- Controlled Distribution Privileges for IrDA Members Only
- Trademarks of IrDA - Prohibitions and Authorized Use
- No Representation of Third Party Rights
- Limitation of Liability
- Disclaimer of Warranty
- Certification of Products Requires Specific Authorization from IrDA after Product Testing for IrDA Specification Conformance

IrDA PUBLICATIONS and UPDATES:

IrDA publications, including notifications, updates, and revisions, are accessed electronically by IrDA members in good standing during the course of each year as a benefit of annual IrDA membership. Electronic copies are available to the public on the IrDA web site located at irda.org. IrDA publications are available to non-IrDA members for a pre-paid fee. Requests for publications, membership applications or more information should be addressed to: Infrared Data Association, P.O. Box 3883, Walnut Creek, California, U.S.A. 94598; or e-mail address: info@irda.org; or by calling John LaRoche at (510) 943-6546 or faxing requests to (510) 934-5600.

COPYRIGHT:

1. Prohibitions: IrDA claims copyright in all IrDA publications. Any unauthorized reproduction, distribution, display or modification, in whole or in part, is strictly prohibited.
2. Authorized Use: Any authorized use of IrDA publications (in whole or in part) is under NONEXCLUSIVE USE LICENSE ONLY. No rights to sublicense, assign or transfer the license are granted and any attempt to do so is void.

DISTRIBUTION PRIVILEGES for IrDA MEMBERS ONLY:

IrDA Members Limited Reproduction and Distribution Privilege: A limited privilege of reproduction and distribution of IrDA copyrighted publications is granted to IrDA members in good standing and for sole purpose of reasonable reproduction and distribution to non-IrDA members who are engaged by contract with an IrDA member for the development of IrDA certified products. Reproduction and distribution by the non-IrDA member is strictly prohibited.

TRANSACTION NOTICE to IrDA MEMBERS ONLY:

Each and every copy made for distribution under the limited reproduction and distribution privilege shall be conspicuously marked with the name of the IrDA member and the name of the receiving party. Upon reproduction for distribution, the distributing IrDA member shall promptly notify IrDA (in writing or by e-mail) of the identity of the receiving party. A failure to comply with the notification requirement to IrDA shall render the reproduction and distribution unauthorized and IrDA may take appropriate action to enforce its copyright, including but not limited to, the termination of the limited reproduction and distribution privilege and IrDA membership of the non-complying member.

TRADEMARKS:

1. Prohibitions: IrDA claims exclusive rights in its trade names, trademarks, service marks, collective membership marks and certification marks (hereinafter collectively "trademarks"), including but not limited to the following trademarks: INFRARED DATA ASSOCIATION (word mark alone and with IR logo), IrDA (acronym mark alone and with IR logo), IR logo, IR DATA CERTIFIED (composite mark), and MEMBER IrDA (word mark alone and with IR logo). Any unauthorized use of IrDA trademarks is strictly prohibited.
2. Authorized Use: Any authorized use of a IrDA collective membership mark or certification mark is by NONEXCLUSIVE USE LICENSE ONLY. No rights to sublicense, assign or transfer the license are granted and any attempt to do so is void.
3. Third party brands, trademarks, registered trademarks, service marks, and names are the property of their respective owners.

NO REPRESENTATION of THIRD PARTY RIGHTS:

IrDA makes no representation or warranty whatsoever with regard to IrDA member or third party ownership, licensing or infringement/non-infringement of intellectual property rights. Each recipient of IrDA publications, whether or not an IrDA member, should seek the independent advice of legal counsel with regard to any possible violation of third party

rights arising out of the use, attempted use, reproduction, distribution or public display of IrDA publications. IrDA assumes no obligation or responsibility whatsoever to advise its members or non-members who receive or are about to receive IrDA publications of the chance of infringement or violation of any right of an IrDA member or third party arising out of the use, attempted use, reproduction, distribution or display of IrDA publications.

LIMITATION of LIABILITY:

BY ANY ACTUAL OR ATTEMPTED USE, REPRODUCTION, DISTRIBUTION OR PUBLIC DISPLAY OF ANY IrDA PUBLICATION, ANY PARTICIPANT IN SUCH REAL OR ATTEMPTED ACTS, WHETHER OR NOT A MEMBER OF IrDA, AGREES TO ASSUME ANY AND ALL RISK ASSOCIATED WITH SUCH ACTS, INCLUDING BUT NOT LIMITED TO LOST PROFITS, LOST SAVINGS, OR OTHER CONSEQUENTIAL, SPECIAL, INCIDENTAL OR PUNITIVE DAMAGES. IrDA SHALL HAVE NO LIABILITY WHATSOEVER FOR SUCH ACTS NOR FOR THE CONTENT, ACCURACY OR LEVEL OF ISSUE OF AN IrDA PUBLICATION.

DISCLAIMER of WARRANTY:

All IrDA publications are provided "AS IS" and without warranty of any kind. IrDA (and each of its members, wholly and collectively, hereinafter "IrDA") EXPRESSLY DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND WARRANTY OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. IrDA DOES NOT WARRANT THAT ITS PUBLICATIONS WILL MEET YOUR REQUIREMENTS OR THAT ANY USE OF A PUBLICATION WILL BE UN-INTERRUPTED OR ERROR FREE, OR THAT DEFECTS WILL BE CORRECTED. FURTHERMORE, IrDA DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING USE OR THE RESULTS OR THE USE OF IrDA PUBLICATIONS IN TERMS OF THEIR CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE. NO ORAL OR WRITTEN PUBLICATION OR ADVICE OF A REPRESENTATIVE (OR MEMBER) OF IrDA SHALL CREATE A WARRANTY OR IN ANY WAY INCREASE THE SCOPE OF THIS WARRANTY.

LIMITED MEDIA WARRANTY:

IrDA warrants ONLY the media upon which any publication is recorded to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of distribution as evidenced by the distribution records of IrDA. IrDA's entire liability and recipient's exclusive remedy will be replacement of the media not meeting this limited warranty and which is returned to IrDA. IrDA shall have no responsibility to replace media damaged by accident, abuse or misapplication.

ANY IMPLIED WARRANTIES ON THE MEDIA, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF DELIVERY. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM PLACE TO PLACE.

CERTIFICATION and GENERAL:

Membership in IrDA or use of IrDA publications does NOT constitute IrDA compliance. It is the sole responsibility of each manufacturer, whether or not an IrDA member, to obtain product compliance in accordance with IrDA rules for compliance. All rights, prohibitions of right, agreements and terms and conditions regarding use of IrDA publications and IrDA rules for compliance of products are governed by the laws and regulations of the United States. However, each manufacturer is solely responsible for compliance with the import/export laws of the countries in which they conduct business. The information contained in this document is provided as is and is subject to change without notice.

TABLE OF CONTENTS

1.	Introduction	1
1.1	Scope.....	1
1.2	Framework.....	1
1.3	References	3
1.4	Acronyms.....	3
1.5	Byte Ordering	3
2.	Infrared Universal Translator (IrUT)	4
2.1	Data transmission overview.....	4
2.2	Basic requirements.....	5
2.3	Frame Format.....	5
2.3.1	Frame Format of UT-Units Field	5
2.3.2	UT-Unit for Real-Time Data Transmission.....	7
2.3.2.1	UT-Unit for Real-Time Data Control	7
2.3.2.2	UT-Unit for Real-Time Data Transmission.....	9
2.3.3	UT-Units for Non-real-time Data Transmission.....	10
2.3.3.1	UT-Unit for Non-real-time Data Control	10
2.3.3.2	UT-Data Unit for Non-real-time Data Transmission	10
2.4	IrUT service primitives	11
2.4.1	Connect Service	11
2.4.2	Disconnect Service	12
2.4.3	Data Service.....	12
2.4.4	Real-time Data Service	12
2.4.5	Status Service	13
2.4.6	Local Flow Service	13
2.5	Internal Organization of the IrUT layer	14
2.6	State Charts.....	15
2.6.1	The State Chart of UT Link Management Section	16
2.6.1.1	State Transition Diagram	16
2.6.1.2	State Transition Tables.....	17
2.6.1.2.1	IDLE State	17
2.6.1.2.2	SETUP State.....	18
2.6.1.2.3	CONN State	19
2.6.1.2.4	NRDTR State	20

2.6.1.2.5	RDTR State	21
2.6.1.2.6	DISC State	22
2.6.1.2.7	State Definitions	23
2.6.1.2.8	State Variables	23
2.6.1.2.9	Event Descriptions	24
2.6.1.2.10	Action Descriptions	24
2.6.2	The State Chart of UT Frame Management Section (UTF)	25
2.6.2.1	State Transmission Diagram	25
2.6.2.2	State Transmission Tables	26
2.6.2.2.1	IDLE State	26
2.6.2.2.2	SETUP State	27
2.6.2.2.3	CONN State	27
2.6.2.2.4	ROLE_EX State	28
2.6.2.2.5	ROLE_EX WAIT State	28
2.6.2.2.6	RECONN State	29
2.6.2.2.7	RECONN_WAIT State	30
2.6.2.2.8	DTR_PEND State	31
2.6.2.2.9	RDTR State	32
2.6.2.2.10	NRDTR State	35
2.6.2.2.11	DISC State	38
2.6.2.2.12	State Definitions	38
2.6.2.2.13	Event Descriptions	39
2.6.2.2.14	State Variables	40
2.6.2.2.15	Action Descriptions	40
2.7	Connection Establishment Procedure	42
2.7.1	The first IrUT link establishment procedure (see Figure 2.7-1)	42
2.7.2	Procedure to add new non-real-time data link (See figure 2.7-2)	44
2.7.3	Negotiation Parameters	46
2.7.3.1	Data transmission cycle	46
2.7.3.2	Maximum data length	46
3	ISDN	47
3.1	Data Transmission Overview	47
3.2	Frame Format	47
3.2.1	BRI Frame Structure	47
3.2.2	PRI frame structure	48

3.2.3	Frame Descriptions	48
3.2.3.1	Common flag field 1	49
3.2.3.2	Common flag field 2	50
3.2.3.3	Bit stream ordering	50
3.2.3.4	Contents of UT-Parameters for ISDN	51
3.3	Basic requirements	52
3.4	Hardware requirement	52
3.4.1	Bit synchronization	52
4.	IrOBEX	53
4.1	Data transmission overview	53
4.2	Frame format example	53
5.	IrCOMM	54
5.1	Data transmission overview	54
5.2	Frame format example	54
6.	Service sequence example	55
6.1	Primary / Secondary exchange	55
7.	IAS	56
7.1	IAS Entries for Class IrDA:IrUT	56
7.1.1	IrDA:TinyTP:LsapSel	56
7.1.2	Parameters	56
7.2	IAS Entries for Class IrDA:IrUT:IrOBEX	59
7.2.1	IrDA:TinyTP:LsapSel	59
7.2.2	Other Attributes	59
7.3	IAS Entries for Class IrDA:IrUT:IrCOMM	59
7.3.1	IrDA:TinyTP:LsapSel	59
7.3.2	Other Attributes	59
7.4	Service Hint Bit	60
Appendix A	Overview of procedure for errors	61
Appendix B	The Detailed Sequence of IrUT Connection	62
B-1	The First Link Connection Establishment Procedure	62
B-2	Data Transmission Procedure	63

B-3 Communication Error Occurs64

B-4 The Second Link Connection Establishment Procedure.....65

B-5 The Last Link Disconnect Procedure.66

1. Introduction

1.1 Scope

This document describes the real-time data transmission function for phase 2 IrMC specification. This specification set new protocol layer named IrUT, which offers a data translation function between an infrared interface and other interfaces. With IrUT, an infrared interface can provide real-time data services such as video conferencing.

1.2 Framework

The IrMC (the phase 1 IrMC specifications) defines the rules for object exchange, voice communication, and call control utilization through an infrared interface. In this specification, the targeted data transmission speed is 4 Mbps (or higher).

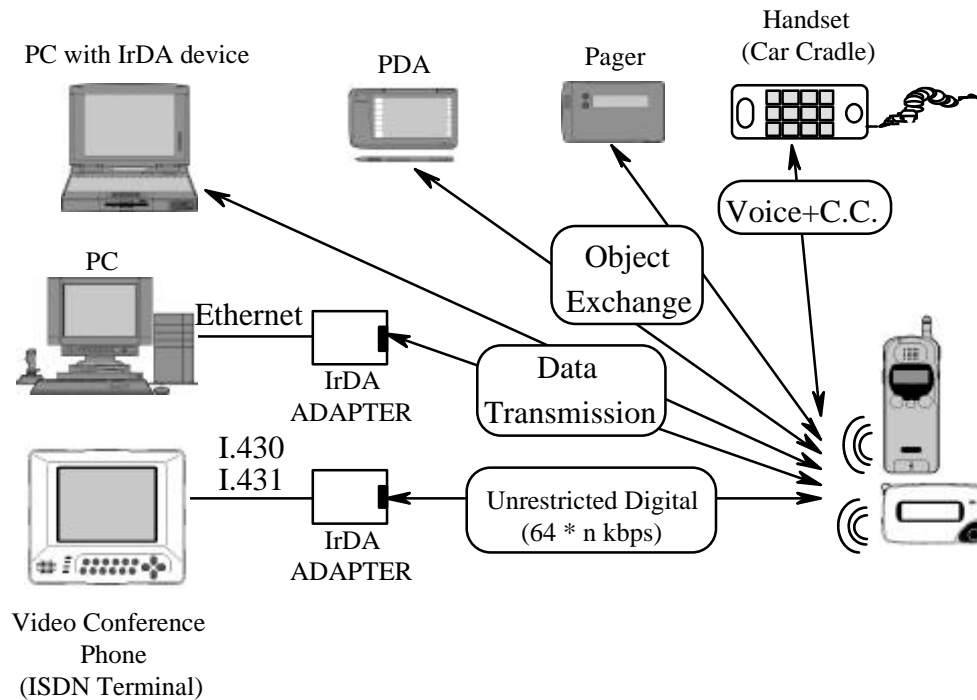
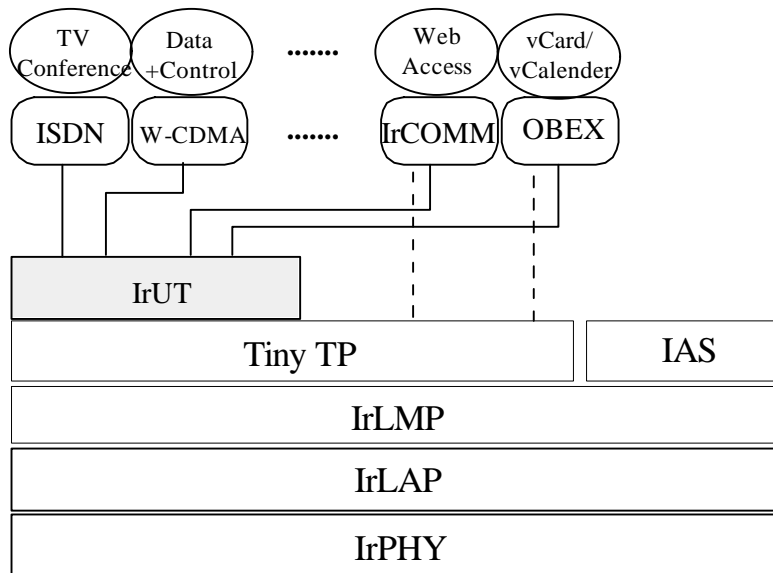


Figure 1-1 Possible services provided in IrMC phase 2 specification

Figure 1-1 shows the infrared data transmission services that will be provided by phase 2 IrMC specification. As the figure shows, phase2 IrMC can handle real-time applications such as TV conference through the ISDN network. Of course, applications supported by the phase 1 IrMC, such as IrOBEX, IrCOMM, etc., should be supported in this phase.

In order to ensure real-time data transmission, devices should continue to send / receive infrared data frame at the beginning of every fixed cycle. During a cycle, no data from other applications can be transmitted because doing so would change the period of the cycle. However, this restriction is not practical as capability to transmit non-real-time data, such as vCard, during the real-time data transmission. To cope with such a situation, one can define a function that combines the data of every application into a single frame.

Figure 1-2 shows the practical implementation of the IrUT layer; the IrUT layer fits on top of the IrDA protocol stack. As shown by the solid line, not only real-time applications, but also the data issued by the phase 1 IrMC applications (IrOBEX and IrCOMM), can be wrapped together into the same IrUT frame. Combines these applications, in this way, ensures compatibility with phase 1 IrMC devices. Of course, such data can be transmitted directly to the TinyTP layer, without going through IrUT layer (This case is shown by the broken line.)



—— The case of multi-application transmission through IrUT layer.

----- The case of existing IrDA application transmission without IrUT layer.

Figure 1-2 Protocol stack

1.3 References

[IrMC]	Specifications for Ir Mobile Communications ver 1.1
[IrPHY]	Serial Infrared Physical Layer Link Specification, ver 1.3, Infrared Data Association
[IrLAP]	IrDA Serial Infrared Link Access Protocol ver 1.1
[IrLMP]	IrDA Link Management Protocol ver 1.1
[ITU-T BRI]	ITU-T Recommendation BRI, Basic user-network interface Layer 1, International Telecommunication Union
[ITU-T PRI]	ITU-T Recommendation PRI, Primary rate user-network interface Layer 2, International Telecommunication Union

1.4 Acronyms

BRI	Basic Rate Interface
H&L	Handling and Length
IrUT	Infrared Universal Translator
LSB	Least Significant Bit
MSB	Most Significant Bit
PRI	Primary Rate Interface
TTPSAP	TinyTP Service Access Point
UDL	Unit Data Length
VLsap	Virtual Link Service Access Points

1.5 Byte Ordering

This document represents IrUT frames as collections of octets (bytes), with each octet being composed of 8 bits numbered 0-7. Bit 0 is always the least significant bit (LSB), and bit 7 is always the most significant bit (MSB). In some cases, IrLMP frames contain components that are composed of multiple bytes. These larger components are represented as $n \times 8$ bits, where n is the number of bytes in the component. The least significant bit is numbered bit 0 while the most significant bit is numbered $(n \times 8) - 1$. The least significant byte of a multi-byte component is defined to be the byte that contains bits 0-7.

2. Infrared Universal Translator (IrUT)

2.1 Data transmission overview

IrUT provides an emulation function that can convert the signals of various types of interfaces into infrared signals. The main feature, offered by this layer, is the real-time data transmission function. The targeted data transmission speed is 4 Mbps (or higher).

IrUT has two operation modes, called the multi-mode and the single-mode. As described in previous section, non-real-time application data such as IrOBEX and IrCOMM can be combined with the real-time data and transmitted through the IrUT layer. This multi-application transmission method is defined as the multi-mode. On the other hand, in the single-mode operation, only one real-time application data can be transmitted. This mode is defined to lighten IrUT implementation in consideration of less capable devices (in terms of computing performance).

If neither IrUT layers (in the communicating state) support multi-mode operation, IrUT operation can only be in single-mode. Likewise, if only one device supports multi-mode. Thus, both devices work in single-mode. When both devices support it, multi-mode operation is possible. Table 2.1-1 summarizes the possible work modes.

Supported Mode		Primary Station	
		Single Mode	Multi Mode
Secondary Station	Single Mode	(Work mode: single) Real-time application	(Work mode: single) Real-time application
	Multi Mode	(Work mode: single) Real-time application	(Work mode: multi) Real-time application

Table 2.1-1 Possible work modes supported by IrUT

2.2 Basic requirements

- IrUT data is transmitted in “T” formatted frames.
- The window size of the IrUT transmission is set to one.

2.3 Frame Format

Figure 2.3-1 shows the IrLAP frame payload data. IrUT frames are fitted into the user data field of TTP-PDU. This field is called UT-Units field. Every UT-Units are wrapped into this field. These units are defined in the next section.

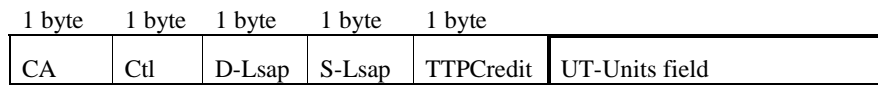
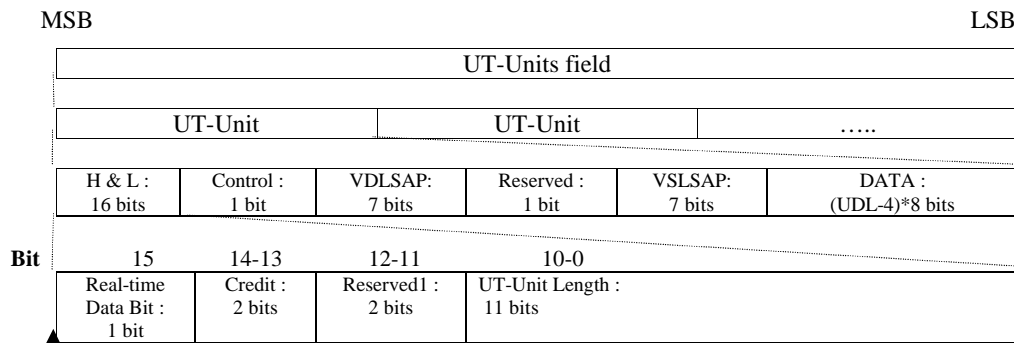


Figure 2.3-1 Frame format

2.3.1 Frame Format of UT-Units Field

Figure 2.3-2 shows the contents of UT-Units field. In multi mode operation, the UT-Units field is to be shared by every active application.



First byte of payload data delivered to / received from the TTP layer.

*H&L: Handling and Length

Figure 2.3-2 UT-Units field

Table 2.3-1 shows the contents of every bit in the UT-Units field. The first two octets of this unit are Handling and Length bits, which handle IrUT data transmission and indicate the length of the UT-Unit field. The Real-time data bit is used to distinguish whether the UT-Unit is used for real-time data or not. The credit bits provide the flow control function for non-real-time data. These bits are used in the same manner as TTP credits are used. In the UT-Units of real-time data, this value is fixed at 0. The UT-Unit length bits indicate the length of the UT-Units including the Handling and Length field.

The Control Bit is used to distinguish the type of the UT-Unit. The UT-Unit can be divided two types, control unit and data unit. UT-Control unit is used to transmit data that control UT_Connect service and UT_Disconnect service. The data issued by the upper application layer is transmitted by in the form of the UT_Data service and the UT_Rdata service. Details of these Units are given in the following section.

Virtual Lsaps are used same as LsapSel of IrLMP. Every application that is handled by IrUT layer has inherent virtual Lsaps. Virtual Lsaps are described in the IAS class IrUT. With this information, IrUT can classify the multi UT-Units and deliver them to the correct destination.

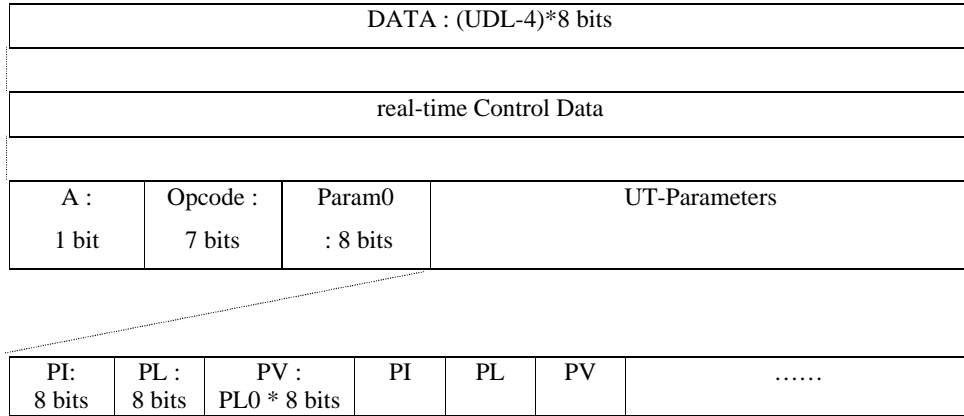
Contents	Size [bit]	Bit	Description
Handling and Length	16		
Real-Time data bit	1	15	1: UT-Units for real-time data 0: UT-Unit for non-real-time data
Credit	2	14-13	In the case of real-time data: Fixed to 0 In the case of non-real-time data: Used to provide flow control function of non-real-time data. These bits are used in the same manner as the Credit of TinyTP is used.
Reserved 1	2	12-11	Fixed to 0
UT-Unit length [Byte]	11	10-0	UT-Unit length (Including the length of handling & length)
Control bit	1	7	1: UT-Units for link control 0: UT-Units for user data
Virtual Destination Lsap-SEL	7	6-0	Bit 6-0 This parameter is used in the same manner as the Lsap-SEL of IrLMP.
Reserved 2	1	7	Fixed to 0
Virtual Source Lsap-SEL	7	6-0	Bit 6-0 This parameter is used in the same manner as the Lsap-SEL of IrLMP.
DATA	(UDL-4)*8		UT-Unit data field

Table 2.3-1 Description of IrUT bits

2.3.2 UT-Unit for Real-Time Data Transmission

2.3.2.1 UT-Unit for Real-Time Data Control

The UT-Unit for real-time data control is used to transmit UT service primitives, the UT_Connect service and the UT_Disconnect service. The content of this unit is shown in Figure 2.3-3. As shown in this figure, this unit is composed of “A” bit, opcode, and UT-Parameters. The “A” bit is used to indicate the generic types of command primitives. This bit and OP code work in same way as the bits of the IrLMP link control frame. These bits control the connect / disconnect procedure between peer IrUT layers. The UT-parameters are composed of three fields: PI, PL, and PV. These parameters are used to negotiate the real-time data transmission conditions, such as data transmission cycle. The details of the negotiation process are described in the section 2.7.

**Figure 2.3-3 UT-Unit for control the real-time link**

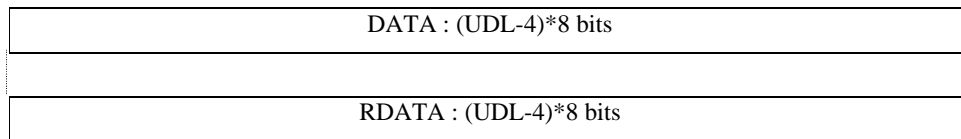
Contents	Size [bit]	Bit	Description
Real-time Control Data	(UDL-4)*8		Control data for real-time data transmission
A	1	7	Command request / Command indication bit Same function as an IrLMP Link control frame "A" bit. 0 : Signifies a command request at the source side and should be interpreted as a command indication at the destination side. 1 : Signifies the command response at the source side and a command confirmation at the destination side.
Opcode	7	6-0	Operation code Same function as an IrLMP Link control frame opcode. (Used here only as a connect or disconnect code.)
Param0	8		Same functions as rsvd (at Opcode : connect) and reason (at Opcode : disconnect) defined in "parameters" of IrLMP Link Control Frames.
PARAMS	(UDL-6)*8		IrUT control parameters
PI	8		Parameter ID
PL	8		PV length
PV	PL*8		Parameter value

Table 2.3-2 Contents of UT-Unit for control the real-time link

PI	PI name	PL	PV data type	PV description
0x00	2-Way Real Time Data	4 * combination	n bytes	Defines data transmission cycle and maximum data length in each cycle. This PV is Combination of 4-byte data unit stream. The first two bytes = The data transmission cycle in millisecond. (Higher byte first) The second two bytes = The number of data length in this cycle. (Higher byte first) (Another 4 bytes unit for other allowable combination repeatedly follows this 4 bytes data unit.)
0x10	User Defined Data	n	n bytes	User defines this PV.
Other	Reserved			

Table 2.3-3 Contents of UT-Parameters**2.3.2.2 UT-Unit for Real-Time Data Transmission**

Real-time data issued by the application layer is transmitted as an UT_Rdata service. The structure and content of this unit are shown below. As shown in the figure, real-time application data should be placed in the RDATA field without any headers.

**Figure 2.3-4 UT-Unit for real-time data transmission**

Content	Size [bit]	Description
RDATA	(UDL-4)*8	User data for real-time data transmission.

Table 2.3-4 Content of UT-Unit for real-time data Transmission

2.3.3 UT-Units for Non-real-time Data Transmission

2.3.3.1 UT-Unit for Non-real-time Data Control

This unit is used to control non-real-time data transmission. The content of this unit is the same as in the A bit, opcode, and parameters described in the link control frames of IrLMP.

It is necessary to set the same value with the initial credit, which is specified by the Credit of “Handling & Length” field, to the InitialCredit field of virtual Connect TTP-PDU of IrUT layer.



Figure 2.3-5 UT-Unit for control of non-real-time link

Content	Size [bit]	Description
NRCD	(UDL-4)*8	Non-real-time control data Same functions as “A”bit, opcode, and parameters defined in IrLMP control frame.

Table 2.3-5 Contents of UT-Unit for control of non-real-time link

2.3.3.2 UT-Data Unit for Non-real-time Data Transmission

This unit is used to transmit non-real-time data, issued by the application layer as UT_Data service. The content of this Unit is shown in Figure 2.3-6. As shown in the figure, non-real-time data can be placed in the DATA field without any headers.

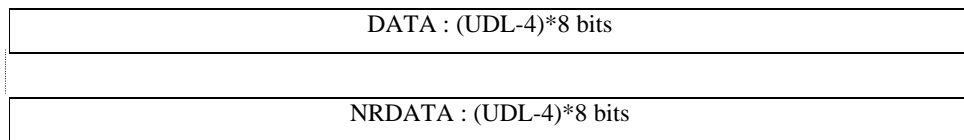


Figure 2.3-6 UT-Unit for non-real-time data transmission

Contents	Size [bit]	Description
NRDATA	(UDL-4)*8	User data for non-real-time data

Table 2.3-6 Contents of UT-Unit for non-real-time data transmission

2.4 IrUT service primitives

2.4.1 Connect Service

UT_Connect.request (CalledLsap, CalledvLsap, RequestedQos, RequestedUT-Qos, Client Data)

UT_Connect.indication (CallingLsap, CallingvLsap, ResultantQos, TentativeUT-Qos, Client Data)

UT_Connect.response (CallingLsap, CallingvLsap, RespondedUT-Qos, Client Data)

UT_Connect.confirm (CalledLsap, CalledvLsap, ResultantQos, ResultantUT-Qos, Client Data)

Parameters:

CalledLsap, CallingLsap	TTPSAP address (Lsap address)
CalledvLsap	This is the virtual Lsap-SEL of the application that receives the <i>UT_Connect.req.</i>
CallingvLsap	This is the virtual Lsap-SEL of the application that issues the <i>UT_Connect req.</i>
RequestedQos	The device which issued the connect request uses this parameter to indicate the requested communication quality of the TinyTP link.
ResultantQos	This parameter indicates the communication quality of the TinyTP link.
RequestedUT-Qos	The device which issued the connect request uses this parameter to indicate the requested communication quality of IrUT link.
TentativeUT-Qos	This parameter indicates the tentative Qos of the IrUT link. This parameter is the result of negotiation by the application layer which received the connect request from the device on the other side. This parameter is determined by comparing the requested Qos of the application which issued the connect request and the Qos of the IrUT layer which executes the negotiation.
RespondedUT-Qos	The device, which received the <i>UT_Connect.ind.</i> , uses this parameter to indicate the requested communication quality of the IrUT link.
ResultantUT-Qos	This parameter indicates the resultant Qos of the IrUT link. This parameter is the result of negotiation by the application layer which received the connect request from the device on the other side. This parameter is determined by comparing the requested Qos of the application which received the connect indication and the TentativeUT-Qos.
Client Data	Data that a service user wants to send along in the connection packet.

Description: The connect services are used to establish an IrUT connection with a peer application.

2.4.2 Disconnect Service

UT_Disconnect.request (Client data, Reason)

UT_Disconnect.indication (Client data, Reason)

Parameters

Client Data	Shown in 2.4.1.
Reason	This parameter indicates the reason why a link is disconnected or why a connection is refused. The 'Reason' should be one of the following:
Unspecified Reason	The reason is unspecified in this document.
User disconnect	This value is used when the responder refuses to make an IrUT connection, or when an IrUT user wishes to disconnect the existing connection.
Provider disconnect	This value is used when the provider of the IrUT connection (IrUT or underlying protocol stack) causes the disconnection.

Description: The disconnect service is used to end the connection with a peer application.

2.4.3 Data Service

UT_Data.request (Data)

UT_Data.indication (Data)

Parameters:

Data	A detailed "Data" description method is defined for every application.
-------------	--

Description: The data service is used to transmit non-real-time data.

2.4.4 Real-time Data Service

UT_Rdata.request (Real-Time Data)

UT_Rdata.indication (Real-Time Data)

Parameters:

Real-TimeData A detailed “Data” description method is defined for every application.

Description: The real-time data service is used to transmit real-time data. Upper application layer should issue this service in fixed period determined by a negotiation.

2.4.5 Status Service

UT_Status.indication (Status)

Parameters:

Status This parameter indicates the status of the IrUT link. This parameter may takes the following value

Status=TxDiscard Indicates the status that the real-time data unit which will be sent is discarded.

Description: The status service is used to inform the status of IrUT link to upper applications.

2.4.6 Local Flow Service

UT_LocalFlow.request (Flow=on|off)

Parameters:

Flow **Flow=on** enables the flow of received UT-Unit for non-real-time data to pass from the receiving UT entity to the local UT client via the local invocation of *UT_Data.indication* primitives.

Flow=off halts the flow of received UT-Unit for non-real-time data to the local UT client. Inbound data is held backlogged within the receiving UT entity, which may apply backpressure to halt the data flow at the sending peer UT entity.

Description: The LocalFlow service is used to control the flow of received UT-Unit for non-real-time data between the receiving UT entity and its local client.

2.5 Internal Organization of the IrUT layer

This section exposes the fabric of the IrUT layer. Figure 2.5-1 gives a detailed breakdown of the service primitives that are handled inside the IrUT modules.

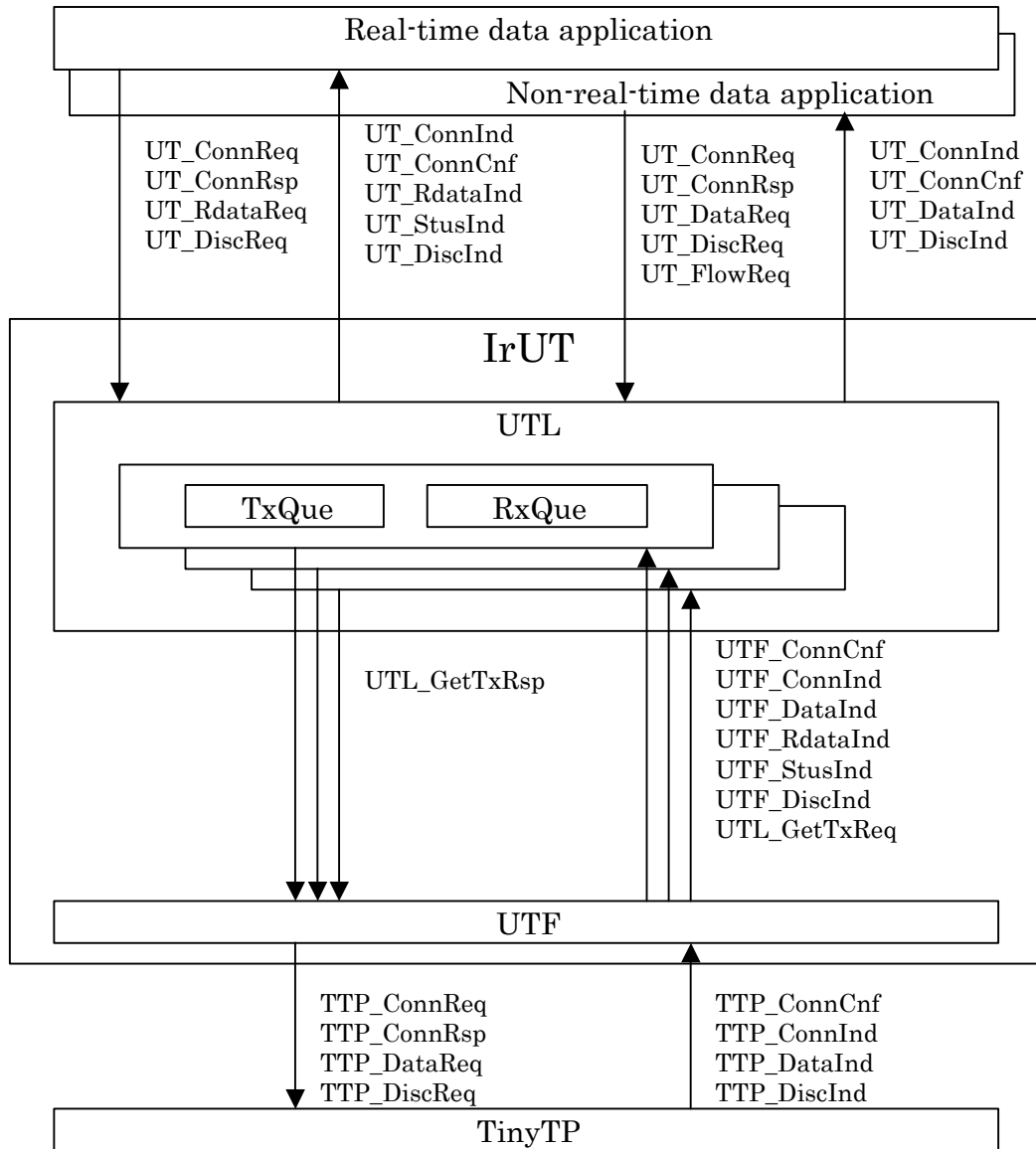


Figure 2.5-1 IrUT internal organization

Acronyms and Definitions

UTL (IrUT Link Manager)

This module manages the IrUT link. The function of this module is to control the multi-link operation of IrUT. Every IrUT link is called UT links. There are as many UTL instances as there are UT links.

TxQue (Transmission Queue)

TxQue is the transmission queue for every UT link.

RxQue (Reception Queue)

RxQue is reception queue for every UT link.

UTF (IrUT Frame Manager)

This module manages the IrUT frame. The functions of this module are as follows.

It provides IrUT link management function of the TinyTP layer.

It divides the TinyTP user data, transmitted by TTP_DataIndication, into UT-Units and conveys these units to every UTL instance.

It combines UT-Units, issued by every UT link, into one TinyTP user data unit. Then it issues this data unit to TinyTP with the TTP_DataRequest.

It manages the real-time data transmission.

2.6 State Charts

The following sections specify in detail the IrUT operating procedures. These procedures define the behavior of the IrUT layer during each phase of operation. The operation procedures include the UTL state chart and the UTF state chart.

• Codes

These state charts described in following sections are written with codes similar to the codes of C language. The following special codes are also used:

!x	Negation of “x”.
x && y	The logical “AND” of “x” and “y”.
x y	The logical “OR” of “x” and “y”.
x == y	Indicates “x” equal to “y”.
{ x }	Indicates a set which has elements "x".
{ }	Indicates an empty set.
NULL	Indicates “nothing”.

2.6.1 The State Chart of UT Link Management Section

2.6.1.1 State Transition Diagram

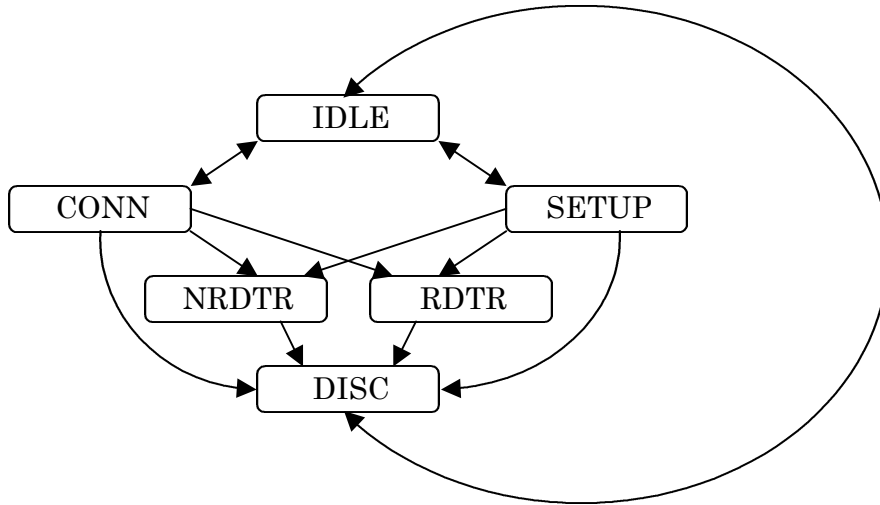


Figure 2.6-1 State Transition Diagram of UTL

2.6.1.2 State Transition Tables

2.6.1.2.1 IDLE State

IDLE state is the initial state of UTL.

Event	Action	Next State
UT_ConnReq && (UTF state is IDLE or DISC) && (ConnReq is a real time unit)	PutTxQue(ConnReq unit); IsRealTimeLink = true; StartWdTimer(SetupTime);	SETUP
UT_ConnReq(params) && (UTF state is IDLE or DISC) && !(ConnReq is a real time unit)	Error; /* First link is a real - time link only */	IDLE
UT_ConnReq(params) && !(UTF state is IDLE or DISC) && (ConnReq is a real time unit)	Error; /* Real-time link is only one */	IDLE
UT_ConnReq(params) && !(UTF state is IDLE or DISC) && !(ConnReq is a real time unit) && GetConnectProcLink() == NULL	PutTxQue(ConnReq unit); IsRxBusy = false; IsRealTimeLink = false; StartWdTimer(SetupTime);	SETUP
UT_ConnReq(params) && !(UTF state is IDLE or DISC) && !(ConnReq is a real time unit) && GetConnectProcLink() != NULL	Error; /* The other link is in connect proccess */	IDLE
UTF_ConnInd && GetConnectProcLink() == Own link	if(ConnInd indicates a real time link){ IsRealTimeLink = true; } else{ SendCdt = Initial credit of the ConnInd; IsRxBusy = false; IsRealTimeLink = false; } UT_ConnInd;	CONN
UTF_ConnInd && GetConnectProcLink() != Own link	/* Issue a disconnect request when other link is in connect proccess */ if(ConnInd indicates a real time link){ IsRealTimeLink = true; } else{ SendCdt = Initial credit of the ConnInd; IsRxBusy = false; IsRealTimeLink = false; } PutTxQue(DiscReq unit);	DISC

2.6.1.2.2 SETUP State

Event	Action	Next State
UT_FlowReq(On) && !IsRealTimeLink	IsRxBusy = false;	SETUP
UT_FlowReq(Off) && !IsRealTimeLink	IsRxBusy = true;	SETUP
UT_DiscReq	EmptyTxQue(); EmptyRxQue(); PutTxQue(DiscReq unit);	DISC
UTF_ConnCnf && IsRealTimeLink	Qos = the qos of the ConnCnf; UT_ConnCnf;	RDTR
UTF_ConnCnf && !IsRealTimeLink	SendCdt = Initial credit of the ConnCnf; Qos = the qos of the ConnCnf; UT_ConnCnf;	NRDTR
UTF_DiscInd	UT_DiscInd; EmptyTxQue(); EmptyRxQue();	IDLE
UTF_ConnInd	/* Conflict connect establishment process */ UT_DiscInd; EmptyTxQue(); EmptyRxQue(); PutTxQue(DiscReq unit);	DISC
UTL_GetTxReq && IsRealTimeLink	unit = GetTxQueHead(); /* The "unit" should be a ConnReq unit */ SetCredit(0, unit); UTL_GetTxRsp(unit);	SETUP
UTL_GetTxReq && !IsRealTimeLink	AvailCdt = 0; SendCdt = 0; n = INITIAL_CDT; /* Local policy */ if(n > 3){ AvailCdt = n - 3; n = 3; } RemoteCdt = n; unit = GetTxQueHead(); /* The "unit" should be a ConnReq unit */ SetCredit(n, unit); UTL_GetTxRsp(unit);	SETUP
WdTimerExpired	EmptyTxQue(); EmptyRxQue(); PutTxQue(DiscReq unit);	DISC

2.6.1.2.3 CONN State

Event	Action	Next State
UT_ConnRsp	IsRxBusy = false; PutTxQue(ConnRsp unit);	CONN
UT_DiscReq	EmptyTxQue(); EmptyRxQue(); PutTxQue(DiscReq unit);	DISC
UTF_DiscInd	UT_DiscInd; EmptyTxQue(); EmptyRxQue();	IDLE
UTL_GetTxReq && IsRealTimeLink	unit = GetTxQueHead(); /* The "unit" should be a ConnRsp unit */ SetCredit(0, unit); Qos = the qos of the "unit"; UTL_GetTxRsp(unit);	RDTR
UTL_GetTxReq && !IsRealTimeLink	AvailCdt = 0; n = INITIAL_CDT; /* Local policy */ if(n > 3){ AvailCdt = n - 3; n = 3; } RemoteCdt = n; unit = GetTxQueHead(); /* The "unit" should be a ConnRsp unit */ SetCredit(n, unit); Qos = the qos of the "unit"; UTL_GetTxRsp(unit);	NRDTR

2.6.1.2.4 NRDTR State

This state is available on non-real-time links (IsRealTimeLink == false) only.

Event	Action	Next State
UT_DataReq(data)	PutTxQue(DataReq unit);	NRDTR
UT_FlowReq(On)	IsRxBusy = false;	NRDTR
UT_FlowReq(Off)	IsRxBusy = true;	NRDTR
UT_DiscReq	EmptyRxQue(); PutTxQue(DiscReq unit);	DISC
UTF_DataInd(unit)	SendCdt = SendCdt + credit of the "unit"; if(unit != dataless unit){ RemoteCdt = RemoteCdt - 1; PutRxQue(unit); }	NRDTR
UTF_DiscInd	PutRxQue(DiscInd unit); EmptyTxQue();	DISC
UTL_GetTxReq && TxQueHead != NULL && SendCdt > 0	n = AvailCdt; AvailCdt = 0; if(n > 3){ AvailCdt = n - 3; n = 3; } RemoteCdt = RemoteCdt + n; SendCdt = SendCdt - 1; unit = GetTxQueHead(); SetCredit(n, unit); UTL_GetTxRsp(unit);	NRDTR
UTL_GetTxReq && TxQueHead == NULL && SendCdt > 0	UTL_GetTxRsp(NULL);	NRDTR
UTL_GetTxReq && SendCdt == 0 && RemoteCdt <= LowThreshold && AvailCdt > 0	n = AvailCdt; AvailCdt = 0; if(n > 3){ AvailCdt = n - 3; n = 3; } RemoteCdt = RemoteCdt + n; unit = dataless unit; SetCredit(n, unit); UTL_GetTxRsp(unit);	NRDTR
UTL_GetTxReq && SendCdt == 0 && !(RemoteCdt <= LowThreshold && AvailCdt > 0)	UTL_GetTxRsp(NULL);	NRDTR
RxQueHead == DataInd unit && !IsRxBusy	unit = GetRxQueHead(); AvailCdt = AvailCdt + 1; UT_DataInd(UserData of the "unit");	NRDTR

2.6.1.2.5 RDTR State

This state is available on real-time links (IsRealTimeLink == true) only.

Event	Action	Next State
UT_RdataReq(data)	UTF_RdataReq(RdataReq unit);	RDTR
UT_DiscReq	EmptyRxQue(); PutTxQue(DiscReq unit);	DISC
UTF_RdataInd(unit)	UT_RdataInd(UserData of the "unit");	RDTR
UTF_StusInd	UT_StusInd;	RDTR
UTF_DiscInd	PutRxQue(DiscInd unit); EmptyTxQue();	DISC
UTL_GetTxReq	unit = GetTxQueHead(); SetCredit(0, unit); UTL_GetTxRsp(unit);	RDTR

2.6.1.2.6 DISC State

Event	Action	Next State
UT_FlowReq(On) && !IsRealTimeLink	IsRxBusy = false;	DISC
UT_FlowReq(Off) && !IsRealTimeLink	IsRxBusy = true;	DISC
UTL_GetTxReq && TxQueHead == DiscReq unit	unit = GetTxQueHead(); EmptyTxQue(); EmptyRxQue(); UTL_GetTxRsp(unit);	IDLE
UTL_GetTxReq && TxQueHead != DiscReq unit && TxQueHead != DataReq unit	/* Discard a TxQue head element */ unit = GetTxQueHead(); UTL_GetTxRsp(NULL);	DISC
UTF_GetTxReq && TxQueHead == DataReq unit && SendCdt > 0 && !IsRealTimeLink	n = AvailCdt; AvailCdt = 0; if(n > 3){ AvailCdt = n - 3; n = 3; } RemoteCdt = RemoteCdt + n; SendCdt = SendCdt - 1; unit = GetTxQueHead(); SetCredit(n, unit); UTL_GetTxRsp(unit);	DISC
UTL_GetTxReq && SendCdt == 0 && RemoteCdt <= LowThreshold && AvailCdt > 0 && !IsRealTimeLink	n = AvailCdt; AvailCdt = 0; if(n > 3){ AvailCdt = n - 3; n = 3; } RemoteCdt = RemoteCdt + n; unit = dataless unit; SetCredit(n, unit); UTL_GetTxRsp(unit);	DISC
UTL_GetTxReq && SendCdt == 0 && !(RemoteCdt <= LowThreshold && AvailCdt > 0) && !IsRealTimeLink	UTL_GetTxRsp(NULL);	DISC
RxQueHead == DataInd unit && !IsRxBusy	unit = GetRxQueHead(); UT_DataInd(UserData of the "unit");	DISC
RxQueHead == DiscInd unit	unit = GetRxQueHead(); EmptyTxQue(); EmptyRxQue(); UT_DiscInd;	IDLE
RxQueHead != DataInd unit && RxQueHead != DiscInd unit && RxQueHead != NULL	/* Discard a RxQue head element */ unit = GetRxQueHead();	DISC

2.6.1.2.7 State Definitions

IDLE	The device is waiting for the <i>UT_ConnectRequest</i> or <i>UT_ConnectIndication</i> .
SETUP	The device is waiting for the <i>UT_ConnectCnfirm</i> , after the issuing of the <i>UT_ConnectRequest</i> .
CONN	The device is waiting for the <i>UT_ConnectResponse</i> , after the receipt of the <i>UT_ConnectIndication</i> .
NRDTR	The non-real-time UT link is established, and the station is in the state of non-real-time data transmission. Only non-real-time UT links can be in this state.
RDTR	The real-time UT link is established, and the station is in the state of real-time data transmission. Only real-time UT links can be in this state.
DISC	After the receipt of the <i>UT_DisconnectRequest</i> , the station should be in this state before it transmits a UT-Unit that includes a DisconnectRequest to the UTF module. Or, after the receipt of a DisconnectIndication from the UTF module, the station should be in this state before the <i>UT_DisconnectIndication</i> is sent to upper applications.

2.6.1.2.8 State Variables

IsRealTimeLink	This variable containing the UT link type. This variable determines whether it is real-time link or not.
IsRxBusy	This variable contains the flow control flag of the received data. This is adopted only non-real-time-link.
INITIAL_CDT	This variable contains the initial credit value to be given to the peer device. This is adopted only non-real-time-link.
AvailCdt	This variable contains the credit value that could be used to set to the peer devices credit value. This is adopted only for non-real-time-link.
RemoteCdt	This variable contains the credit value that corresponds to the data unit number that peer device can send. This is adopted only for non-real-time-link.
SendCdt	This variable contains the credit value that corresponds to the data unit number that the device can send. This is adopted only for non-real-time-link.
Qos	This variable contains the UT-Qos of the UT link.

2.6.1.2.9 Event Descriptions

UTF_ConnInd	UTF informs the local link that it issued a UT_ConnectIndication.
UTF_ConnCnf	UTF informs the local link that it issued a UT_ConnectConfirm.
UTF_RdataInd	UTF informs the local link that it issued a UT_RdataIndication.
UTF_DataInd	UTF informs the local link that it issued a UT_DataIndication.
UTF_StusInd	UTF informs the local link that it issued a UT_StatusIndication.
UTF_DiscInd	UTF informs the local link that it issued a UT_DisconnectIndication.
UTL_GetTxReq	UTF sends a request to get the first element of the TxQue or UT-unit consisting of only credit to the local link.

GetConnectProcLink() == Own link

The variable of UTF module “ConnectProcLink” indicates the link itself and this link is in the connect establishment procedure.

RxQueHead == x	Indicates that the first element of the RxQue is x.
TxQueHead == x	Indicates that the first element of the TxQue is x.
WdTimerExpired	Indicates that the Watch Dog timer is expired.

2.6.1.2.10 Action Descriptions

UTF_RdataReq(unit)	Transmit the UT_RdataRequest to the UTF. This information is sent only to the real-time link.
UTL_GetTxRsp(x)	In response to the UTL_GetTxReq, transmit the “x” to the UTF. In this case the “x” is to be the first element of the TxQue or the UT-Unit consisting of only credit value. If there is no data to be transmitted, then “x” is set to NULL.
SetCredit(n, unit)	Set the value “n” in the credit field of UT-Unit.
x = GetTxQueHead()	Set the first element of the TxQue to “x”. The first element of the TxQue should be deleted after this operation. If the TxQue is empty, then “x” is set to NULL.
x = GetRxQueHead()	Set the first element of the RxQue to “x”. The first element of the RxQue should be deleted after this operation. If the RxQue is empty, then “x” is set to NULL.
PutTxQue(x)	Add “x” to the TxQue as the last element.
EmptyTxQue()	Delete all elements in the TxQue.
EmptyRxQue()	Delete all elements in the RxQue.
StartWdTimer(Setup Time)	Start the WD timer, which counts the interval of the setup state from the issuing of the <i>UT_ConncetRequest</i> to the acceptance of the <i>UT_ConnectConfirm</i> .

2.6.2.2 State Transmission Tables

2.6.2.2.1 IDLE State

IDLE state is the initial state of UTF.

Event	Action	Next State
GetRtConnReqLink(ALL) != NULL	FirstLink = GetRtConnReqLink(ALL); ConnectProcLink = NULL; IsGetTx = false; TTP_ConnReq;	SETUP
TTP_ConnInd && IrLAP is Primary role && Primary role is expected	/* IrLAP is Primary role */ FirstLink = NULL; ConnectProcLink = NULL; IsGetTx = false; TTP_ConnRsp;	CONN
TTP_ConnInd && IrLAP is Primary role && !(Primary role is expected)	/* IrLAP is Primary role */ FirstLink = NULL; ConnectProcLink = NULL; IsGetTx = false; TTP_ConnRsp; /* Secondary role is expected */ StartWdTimer (RoleExWaitTime);	ROLE_EX_WAIT
TTP_ConnInd && !(IrLAP is Primary role) && Primary role is expected && IrLAP Role Exchange service is supported	/* IrLAP is Secondary role */ FirstLink = NULL; ConnectProcLink = NULL; IsGetTx = false; TTP_ConnRsp; LAP_PrimReq;	ROLE_EX
TTP_ConnInd && !(IrLAP is Primary role) && Primary role is expected && !(IrLAP Role Exchange service is supported)	/* IrLAP is Secondary role */ FirstLink = NULL; ConnectProcLink = NULL; IsGetTx = false; TTP_ConnRsp; TTP_DiscReq; WaitLapDisconnect; TTP_ConnReq;	RECONN
TTP_ConnInd && !(IrLAP is Primary role) && !(Primary role is expected)	/* IrLAP is Secondary role */ FirstLink = NULL; ConnectProcLink = NULL; IsGetTx = false; TTP_ConnRsp;	CONN

2.6.2.2.2 SETUP State

Event	Action	Next State
TTP_ConnCnf && IrLAP is Primary role && Primary role is expected && !IsGetTx	IsGetTx = true; UTL_GetTxReq(FirstLink);	SETUP
TTP_ConnCnf && IrLAP is Primary role && !(Primary role is expected) && !IsGetTx	StartWdTimer(RoleExWaitTime);	ROLE_EX_WAIT
TTP_ConnCnf && !(IrLAP is Primary role) && Primary role is expected && IrLAP Role Exchange service is supported && !IsGetTx	LAP_PrimReq;	ROLE_EX
TTP_ConnCnf && !(IrLAP is Primary role) && Primary role is expected && !(IrLAP Role Exchange service is supported) && !IsGetTx	TTP_DiscReq; WaitLapDisconnect; TTP_ConnReq;	RECONN
TTP_ConnCnf && !(IrLAP is Primary role) && !(Primary role is expected) && !IsGetTx	IsGetTx = true; UTL_GetTxReq(FirstLink);	SETUP
TTP_DiscInd && Primary role is expected	FirstLink = NULL; UTF_DiscInd(ALL);	IDLE
TTP_DiscInd && !(Primary role is expected)	StartWdTimer(ReconnWaitTime);	RECONN_WAIT
UTL_GetTxRsp(unit)	/* The "unit" should be a real time ConnReq unit */ ConnectProcLink = link of the "unit"; IsGetTx = false; TTP_DataReq(unit); StartWdTimer(DtrPendTime);	DTR_PEND

2.6.2.2.3 CONN State

Event	Action	Next State
TTP_DataInd(data) && data == Real time ConnInd unit	FirstLink = link of the "data"; UTF_ConnInd(FirstLink); ConnectProcLink = FirstLink; StartWdTimer(DtrPendTime);	DTR_PEND
TTP_DiscInd	FirstLink = NULL; UTF_DiscInd(ALL);	IDLE

2.6.2.2.4 ROLE_EX State

Event	Action	Next State
LAP_PrimCnf(deny) && !IsGetTx	TTP_DiscReq; WaitLapDisconnect; TTP_ConnReq;	RECONN
LAP_PrimCnf(not deny) && FirstLink != NULL && !IsGetTx	IsGetTx = true; UTL_GetTxReq(FirstLink);	ROLE_EX
LAP_PrimCnf(not deny) && FirstLink == NULL && !IsGetTx	/* Wait a real time ConnInd unit */	ROLE_EX
TTP_DataInd(data) && data == Real time ConnInd unit && IrLAP is Primary role && !IsGetTx	FirstLink = link of the "data"; UTF_ConnInd(FirstLink); ConnectProcLink = FirstLink; StartWdTimer(DtrPendTime);	DTR_PEND
TTP_DiscInd	UTF_DiscInd(ALL);	IDLE
UTL_GetTxRsp(unit)	/* The "unit" should be a real time ConnReq unit */ ConnectProcLink = link of the "unit"; IsGetTx = false; TTP_DataReq(unit); StartWdTimer(DtrPendTime);	DTR_PEND

2.6.2.2.5 ROLE_EX WAIT State

Event	Action	Next State
LAP_PrimInd && FirstLink != NULL && !IsGetTx	LAP_PrimRsp; IsGetTx = true; UTF_GetTxReq(FirstLink);	ROLE_EX_WAIT
LAP_PrimInd && FirstLink == NULL && !IsGetTx	LAP_PrimRsp;	ROLE_EX_WAIT
TTP_DataInd(data) && data == Real time ConnInd unit && !(IrLAP is Primary role)	FirstLink = link of the "data"; UTF_ConnInd; ConnectProcLink = FirstLink; StartWdTimer(DtrPendTime);	DTR_PEND
TTP_DiscInd	StartWdTimer(ReconnWaitTime);	RECONN_WAIT
UTF_GetTxRsp(unit)	/* The "unit" should be a real time ConnReq unit */ ConnectProcLink = link of the "unit"; IsGetTx = false; TTP_DataReq(unit); StartWdTimer(DtrPendTime);	DTR_PEND
WdTimerExpired && !IsGetTx	TTP_DiscReq; UTF_DiscInd(ALL);	IDLE

2.6.2.2.6 RECONN State

Event	Action	Next State
TTP_ConnCnf && IrLAP is Primary role && FirstLink != NULL && !IsGetTx	IsGetTx = true; UTL_GetTxReq(FirstLink);	RECONN
TTP_ConnCnf && IrLAP is Primary role && FirstLink == NULL && !IsGetTx	/* Wait a real time ConnInd unit */	RECONN
TTP_ConnCnf && !(IrLAP is Primary role) && !IsGetTx	FirstLink = NULL; TTP_DiscReq; UTF_DiscInd(ALL);	IDLE
TTP_DataInd(data) && data == Real time ConnInd unit && IrLAP is Primary role && !IsGetTx	FirstLink = link of the "data"; ConnectProcLink = FirstLink; UTF_ConnInd(FirstLink); StartWdTimer(DtrPendTime);	DTR_PEND
TTP_DiscInd	FirstLink = NULL; UTF_DiscInd(ALL);	IDLE
UTL_GetTxRsp(unit)	/* The "unit" should be a real time ConnReq unit */ ConnectProcLink = link of the "unit"; IsGetTx = false; TTP_DataReq(unit); StartWdTimer(DtrPendTime);	DTR_PEND

2.6.2.2.7 RECONN_WAIT State

Event	Action	Next State
TTP_ConnInd && !(IrLAP is Primary role) && FirstLink != NULL && !IsGetTx	TTP_ConnRsp; IsGetTx = true; UTL_GetTxReq(FirstLink);	RECONN_WAIT
TTP_ConnInd && !(IrLAP is Primary role) && FirstLink == NULL	TTP_ConnRsp;	RECONN_WAIT
TTP_DataInd(data) && data == Real time ConnInd unit && !(IrLAP is Primary role) && !IsGetTx	FirstLink = link of the "data"; ConnectProcLink = FirstLink; UTF_ConnInd; StartWdTimer(DtrPendTime);	DTR_PEND
TTP_ConnInd && IrLAP is Primary role && !IsGetTx	FirstLink = NULL; TTP_DiscReq; UTF_DiscInd(ALL);	IDLE
TTP_DiscInd	UTF_DiscInd(ALL);	IDLE
UTL_GetTxRsp(unit)	/* The "unit" should be a real time ConnReq unit */ ConnectProcLink = link of the "unit"; IsGetTx = false; TTP_DataReq(unit); StartWdTimer(DtrPendTime);	DTR_PEND
WdTimerExpired && !IsGetTx	FirstLink = NULL; TTP_DiscReq; UTF_DiscInd(ALL);	IDLE

2.6.2.2.8 DTR_PEND State

Event	Action	Next State
TxQueHead(ConnectProcLink) == ConnRsp unit && !IsGetTx	IsGetTx = true; UTL_GetTxReq(ConnectProcLink);	DTR_PEND
TxQueHead(ConnectProcLink) == DiscReq unit && !IsGetTx	IsGetTx = true; UTL_GetTxReq(ConnectProcLink);	DTR_PEND
TTP_DataInd(data) && data == Real time ConnCnf unit of "ConnectProcLink" && !IsGetTx	NegotiateUtQos(the qos of the "data"); ConnectProcLink = NULL; UTF_ConnCnf;	RDTR
TTP_DataInd(data) && data == Real time DiscInd unit of "ConnectProcLink" && !IsGetTx	ConnectProcLink = NULL; UTF_DiscInd(FirstLink); StartWdTimer(DiscTime);	DISC
TTP_DiscInd && !IsGetTx	ConnectProcLink = NULL; UTF_DiscInd(ALL);	IDLE
UTL_GetTxRsp(unit) && unit == Real time ConnRsp unit	IsGetTx = false; NegotiateUtQos(the qos of the "unit"); Links = { }; ConnectProcLink = NULL; TTP_DataReq(unit);	RDTR
UTL_GetTxRsp(unit) && unit == Real time DiscReq unit	ConnectProcLink = NULL; IsGetTx = false; TTP_DataReq(unit); UTF_DiscInd(ALL); StartWdTimer(DiscTime);	DISC
GetIdleOrDiscLinks() == ALL && !IsGetTx	StartWdTimer(DiscTime);	DISC
WdTimerExpired && !IsGetTx	UTF_DiscInd(ALL); StartWdTimer(DiscTime);	DISC

2.6.2.2.9 RDTR State

Event	Action	Next State
TTP_DataInd(data) && !IsGetTx	<pre> nextState = RDTR; unit = GetAndRemoveHeadUnit(data); while(unit != NULL){ if(unit == ConnInd unit){ if(ConnectProcLink == NULL){ ConnectProcLink = link of the "unit"; } UTF_ConnInd; /* UTL issues a DiscReq if ConnectProcLink != NULL */ } else if(unit == ConnCnf unit){ NegotiateUtQos(the qos of the "unit"); if(link of the "unit" == ConnectProcLink){ ConnectProcLink = NULL; } UTF_ConnCnf; } else if(unit == RdataInd unit){ UTF_RdataInd; } else if(unit == DataInd unit){ UTF_DataInd; } else if(unit == DiscInd unit){ if(link of the "unit" == ConnectProcLink){ ConnectProcLink = NULL; } UTF_DiscInd(link of the "unit"); if(unit == Real time unit){ nextState = NRDTR; } } } /* Other kind unit is discarded */ unit = GetAndRemoveHeadUnit(data); </pre>	nextState

UTF_RdataReq(unit) && IsTtpDataReqConditionOk && !IsGetTx	<pre> if(ConnectProcLink != NULL){ Links = ALL - GetControlLinks() + { ConnectProcLink }; } else{ Links = GetTxLinks() - {real time link}; } TxTtpUserData = unit; link = GetAndRemoveLink(Links); while(link != NULL && TxQueHeadUnitSize(link) > max size of Data TTP-PDU UserData - size of TxTtpUserData){ link = GetAndRemoveLink(Links); } if(link == NULL){ TTP_DataReq(TxTtpUserData); } else{ IsGetTx = true; UTL_GetTxReq(link); } </pre>	RDTR
UTF_RdataReq(unit) && ! IsTtpDataReqConditionOk && !IsGetTx	UTF_StusInd(link of the "unit", TxDiscard);	RDTR
TxQueHead(real time link) == DiscReq unit && IsTtpDataReqConditionOk && !IsGetTx	<pre> TxTtpUserData = NULL; Links = {ALL} - {real time link}; IsGetTx = true; UTL_GetTxReq(real time link); </pre>	RDTR
A real time link UTL state is IDLE && GetIdleOrDiscLinks() != ALL && !IsGetTx	<pre> /* Do nothing */ /* Established UT links are non real time links only */ </pre>	NRDTR
GetIdleOrDiscLinks() == ALL && !IsGetTx	StartWdTimer(DiscTime);	DISC
TTP_DiscInd	UTF_DiscInd(ALL);	IDLE

UTL_GetTxRsp(unit)	<pre> if(unit == Non real time ConnRsp unit){ if(link of the "unit" == ConnectProcLink){ NegotiateUtQos(the qos of the "unit"); ConnectProcLink = NULL; } } else if(unit == Non real time ConnReq unit){ if(ConnectProcLink == NULL){ ConnectProcLink = link of the "unit"; } } IsGetTx = false; TxTtpUserData = TxTtpUserData + unit; link = GetAndRemoveLink(Links); while(link != NULL && TxQueHeadUnitSize(link) > max size of Data TTP-PDU UserData - size of TxTtpUserData){ link = GetAndRemoveLink(Links); } if(link == NULL){ TTP_DataReq(TxTtpUserData); } else{ IsGetTx = true; UTL_GetTxReq(link); } </pre>	RDTR
----------------------	--	------

2.6.2.2.10 NRDTR State

Event	Action	Next State
TTP_DataInd(data) && !IsGetTx	<pre> nextState = NRDTR; unit = GetAndRemoveHeadUnit(data); while(unit != NULL){ if(unit == Non real time ConnInd unit){ if(ConnectProcLink == NULL){ ConnectProcLink = link of the "unit"; } UTF_ConnInd; } else if(unit == Real time ConnCnf unit){ NegotiateUtQos(the qos of the "unit"); if(link of the "unit" == ConnectProcLink){ ConnectProcLink = NULL; nextState = RDTR; } UTF_ConnCnf; } else if(unit == Non real time ConnCnf unit){ NegotiateUtQos(the qos of the "unit"); if(link of the "unit" == ConnectProcLink){ ConnectProcLink = NULL; } ConnectProcLink = link of the "unit". UTF_ConnCnf; } else if(unit == DataInd unit){ UTF_DataInd; } else if(unit == DiscInd unit){ if(link of the "unit" == ConnectProcLink){ ConnectProcLink = NULL; } UTF_DiscInd; } /* Other kind unit is discarded */ unit = GetAndRemoveHeadUnit(data); } </pre>	nextState

GetTxLinks() != NULL && IsTtpDataReqConditionOk && !IsGetTx	<pre> if(ConnectProcLink != NULL){ Links = GetTxLinks() - GetControlLinks() + { ConnectProcLink }; } else{ Links = GetTxLinks(); } TxTtpUserData = NULL; link = GetAndRemoveLink(Links); while(link != NULL && TxQueHeadUnitSize(link) > max size of Data TTP-PDU UserData - size of TxTtpUserData){ link = GetAndRemoveLink(Links); } if(link == NULL){ TTP_DataReq(TxTtpUserData); } else{ IsGetTx = true; UTL_GetTxReq(link); } </pre>	NRDTR
---	--	-------

UTL_GetTxRsp(unit)	<pre> nextState = NRDTR; if(unit == Real time ConnRsp unit){ if(link of the "unit" == ConnectProcLink){ NegotiateUtQos(qos of the "unit"); ConnectProcLink = NULL; nextState = RDTR; } } else if(unit == Non real time ConnRsp unit){ if(link of the "unit" == ConnectProcLink){ NegotiateUtQos(qos of the "unit"); ConnectProcLink = NULL; } } else if(unit == ConnReq unit){ if(ConnectProcLink == NULL){ ConnectProcLink = link of the "unit"; } } IsGetTx = false; TxTtpUserData = TxTtpUserData + unit; link = GetAndRemoveLink(Links); while(link != NULL && TxQueHeadUnitSize(link) > max size of Data TTP-PDU UserData - size of TxTtpUserData){ link = GetAndRemoveLink(Links); } if(link == NULL){ TTP_DataReq(TxTtpUserData); } else{ IsGetTx = true; UTL_GetTxReq(link); } </pre>	nextState
GetIdleOrDiscLinks() == ALL && !IsGetTx	StartWdTimer(DiscTime);	DISC
TTP_DiscInd	UTF_DiscInd(ALL);	IDLE

2.6.2.2.11 DISC State

Event	Action	Next State
TTP_DiscInd	/* Do Nothing */	IDLE
TTP_DataInd(data) && data == Real time ConnInd unit && !IsGetTx	UTF_ConnInd; IsGetTx = false; FirstLink = link of the "data"; ConnectProcLink = FirstLink; StartWdTimer(DtrPendTime);	DTR_PEND
GetRtConnReqLink(ALL) != NULL && !IsGetTx	link = GetRtConnReqLink(ALL); FirstLink = link; IsGetTx = true; UTL_GetTxReq(link);	DISC
UTL_GetTxRsp(unit)	/* The "unit" should be a real time ConnReq unit */ ConnectProcLink = link of the "unit"; IsGetTx = false; TTP_DataReq(unit); StartWdTimer(DtrPendTime);	DTR_PEND
WdTimerExpired && !IsGetTx	TTP_DiscReq;	IDLE

2.6.2.2.12 State Definitions

IDLE	The station is waiting for either the <i>UT_ConnectRequest</i> or the <i>TTP_ConnectIndication</i> .
SETUP	The station is waiting for the <i>TTP_ConnectConfirm</i> after the issuing of the <i>TTP_ConnectRequest</i> .
CONN	The station is waiting for the <i>UT_ConnectIndication</i> after the issuing of the <i>TTP_ConnectResponse</i> .
ROLE_EX	The station is in the state of role exchange with the receipt of <i>IrLAP_PrimaryRequest</i> . Only the station that is to be a primary station can be in this state.
RECONN	The station is in the state of role exchange with the receipt of either the <i>TTP_DisconnectRequest</i> or the <i>TTP_ConnectRequest</i> when one or both of the communicating devices do not support the role exchange service. Only the station that is to be a primary station can be in this state.
ROLE_EX_WAIT	The station is waiting for the <i>IrLAP_PrimaryIndication</i> or the <i>TTP_DisconnectIndication</i> from the peer device during the role exchange procedure. Only the station that is to be a secondary station can be in this state.
RECONN_WAIT	The station is waiting for either the reconnect action (<i>TTP_ConnectIndication</i>) from the peer device when one or both of the

	communicating devices do not support the role exchange service. Only the station that is to be a secondary station can be in this state.
DTR_PEND	The station is waiting for the <i>UT_ConnectResponse</i> or <i>UT_ConnectConfirm</i> .
RDTR	A real-time data link is already established, and the station is in a communicating state.
NRDTR	There are only non-real-time data links between peer devices.
DISC	The station is holding to disconnect the TTP link after all UT links have been disconnected.

2.6.2.2.13 Event Descriptions

UTF_RdataReq(unit)	Send the <i>UT_RdataRequest</i> to the UTF. This information is sent only to the real-time.
UTL_GetTxRsp(x)	This command is issued as a response to the <i>UTL_GetTxReq</i> . In this case the “x” is to be the first element of the TxQue or the UT-Unit consisting of only credit value. If there is no data to be transmitted, then “x” is set to NULL.
GetIdleOrDiscLinks() == ALL	Indicates that all UT links (UTL instance) are in IDLE or DISC states.
GetRtConnReqLink(ALL)!=NULL	Indicates that there is a UT link whose first element of the TxQue is a UT-Unit containing real-time connect request.
GetTxLinks()!=NULL	Indicates that there is a UT link which has first element of the TxQue or the UT-Unit consisting of only credit to be transmitted.
IsTtpDataReqConnditionOk	Indicates the existence of the condition to reduce the delay of data transmission time and to keep the interval of frame exchange period constant.
WdTimerExpired	Indicates that the WD timer is expired.

2.6.2.2.14 State Variables

FirstLink	This variable indicates the first UT link. It is used during the first UT link establishment procedure.
ConnecetProcLink	This variable indicates the UT link which is in the link establishment procedure.
Links	This variable keeps the set of UT link for a while.
TxTtpUsrData	This variable is used for combining any UT-Units. It is Data TTP-PDU user data for <i>TTP_DataRequest</i> .
IsGetTx	This variable indicates whether the UTF is waiting for the <i>UTL_GetTxRsp</i> after the issuing of the <i>UTL_GetTxReq</i> .
DiscTime	This variable indicates the time it takes for the issue of <i>TTP_DisconnectRequest</i> after all UT links to disconnect.
RoleExWaitTime	This variable indicates the waiting interval for the <i>IrLAP_PrimaryIndication</i> or <i>TTP_DisconnectIndication</i> , in the <i>ROLE_EX_WAIT</i> state.
ReconnWaitTime	This Variable indicates the waiting interval for the <i>UT_ConnectIndication</i> or <i>UT_ConnectRequest</i> , in the <i>RECONN_WAIT</i> state.
DtrPendTime	This variable indicates the waiting time for the <i>UT_ConnectConfirm</i> or the <i>UT_ConnectResponse</i> at the <i>DTR_PEND</i> state.

2.6.2.2.15 Action Descriptions

UTF_ConnInd(link)	Sends the <i>UT_ConnectIndication</i> to the UT link designated by “link”.
UTF_ConnCnf(link)	Sends the <i>UT_ConnectcConfirm</i> to the UT link designated by “link”.
UTF_RdataInd(link)	Sends the <i>UT_RdataIndication</i> to the UT link designated by “link”.
UTF_DataInd(link)	Sends the <i>UT_DataIndication</i> to the UT link designated by “link”.
UTF_StusInd(link)	Sends the <i>UT_StatusIndication</i> to the UT link designated by “link”.
UTF_DiscInd(link)	Sends the <i>UT_DisconnectIndication</i> to the UT link designated by “link”.
UTL_GetTxReq(link)	Sends a request to get the first element of the TxQue or the UT-Unit consisting of only credit to the UT link indicated by “link”.
links = GetIdleOrDiscLinks()	Set the UT links (UTL instances) that are in <i>IDLE</i> or <i>DISC</i> states to “links”.
links = GetTxLinks()	Sets the UT links (UTL instances) that have the element in the TxQue or the UT-Unit consisting of only credit to be transmitted to the set of “links”.

links = GetControlLinks()

Set the UT links (UTL instances) to “links” whose first element is a UT-Unit for control.

link = GetRtConnReqLink(x)

Sets a link from the set of link “x” whose first element of TxQue is a real-time ConnectRequest UT-Unit, to “link”. If the set of link “x” is set to “All”, this process should be applied to all UT links.

link = GetAndRemoveLink(links)

Sets the UT link that is in the set of “links” to “link” and deletes this link from the set of “links”.

unit = GetAndRemoveHeadUnit(data)

Sets the UT-Unit, which is an element of the "data", to the "unit". The "data" is a Data TTP-PDU UserData. And the unit, which is already set to "unit", is deleted from the "data".

TxQueHeadUnitSize(link)

Gets the UT-Unit length of the first element of the TxQue or of the UT-Unit consisting of only credit to be transmitted. If there is no first element of the TxQue or the UT-Unit consisting of only credit, then gets zero.

NegotiateUtQos(qos)

If necessary, adjusts "qos" to the appropriate value.

WaitLapDisconnect

Wait till the IrLAP is disconnected.

StartWdTimer(x)

Start the Watch Dog timer for “x”.

2.7 Connection Establishment Procedure

2.7.1 The first IrUT link establishment procedure (see Figure 2.7-1)

This procedure starts by issuing of the *UT_Connect.req* from the application layer that wants to start data transmission through the IrUT layer. This service primitive transmits the information used to establish the data link between IrUT layers. The infrared link establishment procedure proceeds if the infrared connection under the TinyTP layer is still not established. The negotiation parameters (data transmission cycle and max data length) are transmitted on this service primitive. If there is the necessity of role exchange, the role exchange procedure should be done at this time.

The content of the *UT_Connect.req* is transmitted on the *TTP_Data.req*. The real-time data transmission condition is negotiated by the application layer, which receives this request. The result of this negotiation is put into the *UT_Connect.rsp* and transmitted to peer application layer. IrUT layer. After the receipt of this information, the application layer that issued the connect request, starts data transmission.

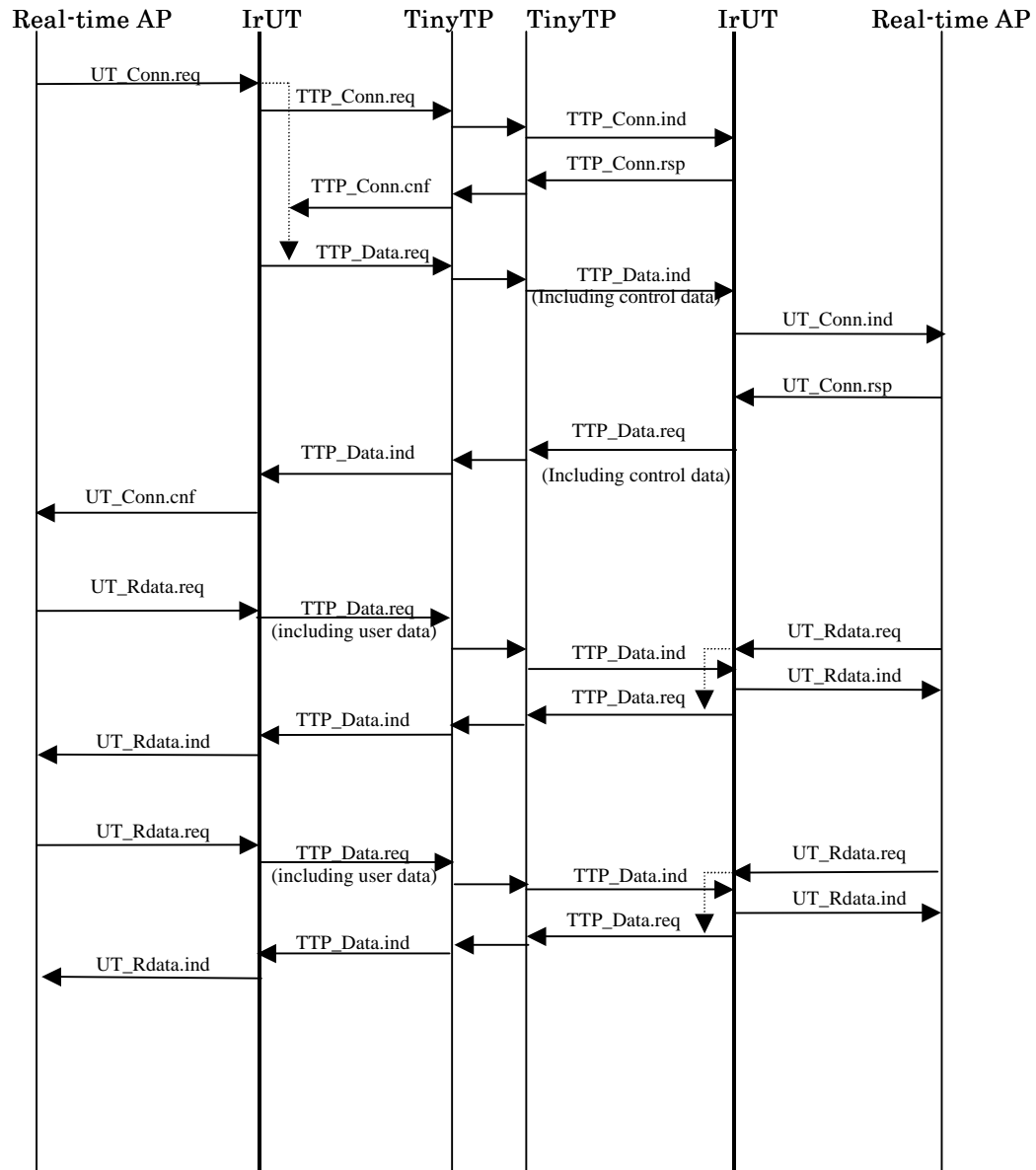


Figure 2.7-1 UT connection establishment procedure

2.7.2 Procedure to add new non-real-time data link (See figure 2.7-2)

In the multi-mode operation, there is the case to add new non-real-time data link.

After the new *UT_Connect.req* is generated, it is put into the same *TTP_Data.req* as the real-time data and sent to the device on the other side of the link. In this case, the real-time data transmission, which is already proceeding, should not to be interrupted. The negotiation of the data transmission condition is done at the application layer on the other side. The negotiated parameter is the data length. The negotiated parameter is sent back to the device from which the request originated at the time the real-time data request is issued. With the arrival of this negotiated parameter, the non-real-time application starts to issue the *UT_Data.req*. The non-real-time data is put into the *TTP_Data.req* and transmitted with the real-time data.

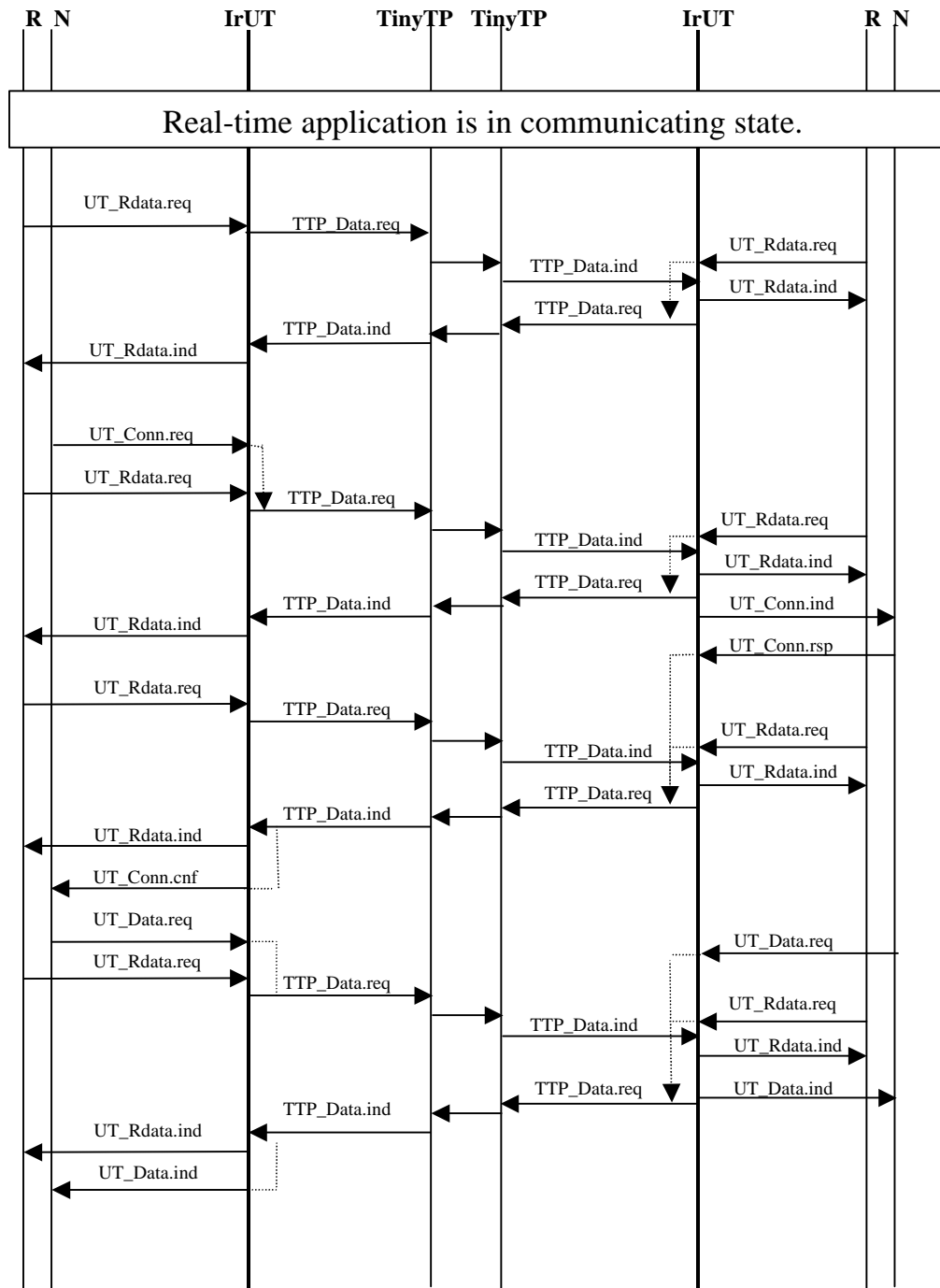


Figure 2.7-2 Procedure to add new non-real-time data link

2.7.3 Negotiation Parameters

2.7.3.1 Data transmission cycle

In order to ensure real-time data transmission, the data transmission cycles of both stations should be the same. To determine this cycle, negotiation should be done before the data transmission starts. In the negotiation phase, the actual value of data transmission cycle is determined according to the capabilities of both application layers.

(1) Negotiation case: one real-time application

When the negotiation about one real-time application A proceeds, the application layer that does the negotiation determines a common value for the data transmission cycle between both devices. If there is a common value, the value of data transmission cycle is decided according to the rule of each application. If not, the negotiation for the application A will fail.

(2) Negotiation case: one real-time application and other non-real-time application

If there is a demand to transmit application A (real-time) and application B (non-real-time) through the IrUT, the negotiation is executed at the application layer, which received the *UT_Connect.indication*. The negotiated parameter is the data length. The available data length is within the remaining range of the user data field.

2.7.3.2 Maximum data length

When one or more non-real-time data are transmitted with real-time type data through the IrUT layer, the data length of non-real-time data should be limited to the rest of user data range. In this case, the non-real-time application should know the permitted maximum data length. This value will be determined and be informed to each application after the negotiation

If it is necessary to inform the maximum data length clearly, it should be set to the MaxSdu of virtual TinyTP Connect TTP-PDU transmitted by UT-Unit for control. In this case, the maximum data length is informed to application layer after limited to the length, which can be transmitted, by the IrUT layer.

3 ISDN

3.1 Data Transmission Overview

IrUT will provide reliable 64 kbps*n full-duplex user data transmission by converting an ISDN electric interface (BRI or PRI wired interface) to an infrared interface. This section describes the structure of the IrUT frame format.

For the ISDN data transmission service, IrUT service primitives, which are defined as services used for the real-time data transmission, are used entirely. These services are shown in section “2.4 IrUT service primitives”. In the following sections, only formats of user data, which are used by these services, are defined.

The frame format, described in the next section, indicates the format of "Real-Time Data", which is a parameter of real-time data service primitive. (see section 2.4.4)

The format of ISDN parameters, transmitted to the peer device encapsulated in the UT-Parameters field of UT-Unit for control, is shown in section "3.2.3.4 Contents of UT-Parameters for ISDN".

3.2 Frame Format

3.2.1 BRI Frame Structure

BRI is the layer 1 protocol, called the basic rate interface, between the Network Terminal (NT) and Terminal Equipment (TE). BRI is a bi-directional (2B+D: B:64 kbps, D:16 kbps) transmission, and the frame structure is different for each direction of transmission. In order to support BRI in the Ir interface, the data shown in Table 3.3-1 must be transmitted.

Table 3.2-1

D	D-channel bits	16 kbps
B1	B1-channel bits	64 kbps
B2	B2-channel bits	64 kbps
F _A	Assist frame bit	4 kbps
M	Multi-frame bit (Only down link: NT to TE)	4 kbps
S	Reserved bit (Fix 0) (Only down link: NT to TE)	4 kbps

3.2.2 PRI frame structure

PRI is the layer 1 protocol, called the primary rate interface, between NT and TE. It serves 24 time slots (TSs) with 64 kbps of data each.

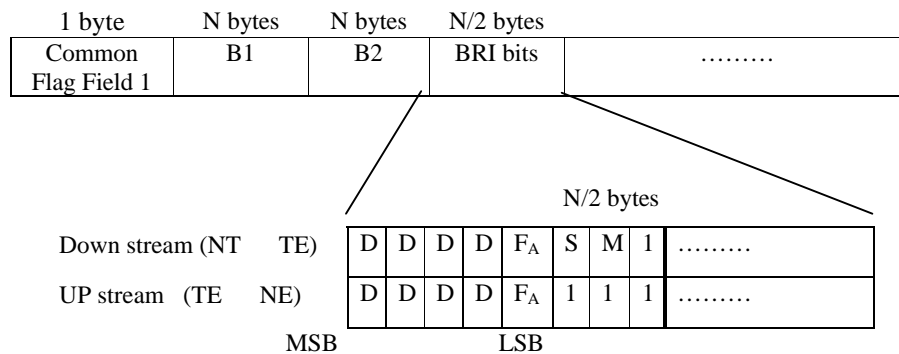
3.2.3 Frame Descriptions

The unified frame format is defined for both BRI and PRI. Figure 3.3-1 shows the proposed frame structure for the IrUT for ISDN data transmission. For effective data transmission there are three common flag fields (CFF) in front of the user data for effective data transmission. These frames are sent alternately on the Ir connection between the TE and the NT.

CCF1 indicates which channels of the BRI interface are used. When more than two BRI lines are used, the 7th bit (Extension byte) is set to 1, and the flag field is extended.

CFF2 indicates which TSs of the PRI interface is used. 24 TSs are defined in proposed frame structure.

BRI Frame



PRI Frame

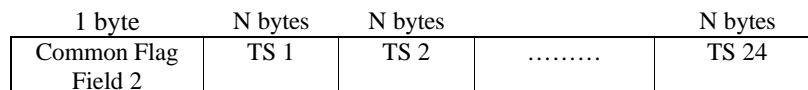


Figure 3.2-1 Proposed IrUT frame structure (IrUT payload data).

3.2.3.1 Common flag field 1

This flag field indicates what channels of the BRI interface are in use.

Table 3.2-2 Contents of common flag field 1.

1st Byte

Bit	Function	Value	Description
0	BRI first line	0	None
		1	Use
1	BRI first line B1 channel	0	None
		1	Use
2	BRI first line B2 channel	0	None
		1	Use
3	BRI second line	0	None
		1	Use
4	BRI second line B1 channel	0	None
		1	Use
5	BRI second line B2 channel	0	None
		1	Use
6	0	0	(Fixed)
7	Extension byte	0	None
		1	Use

2nd Byte

Bit	Function	Value	Description
0	BRI third line	0	None
		1	Use
1	BRI third line B1 channel	0	None
		1	Use
2	BRI third line B2 channel	0	None
		1	Use
3-5	Reserved	0	
6	0	0	(Fixed)
7	Extension byte	0	None
		1	Use

3.2.3.2 Common flag field 2

This flag field indicates which TSs of the PRI is in use.

Table 3.2-3 Contents of common flag field 2.

1st Byte

Bit	Function	Value	Description
0	TS1	0 1	None Use
• •	• •	• •	• •
7	TS8	0 1	None Use

2nd Byte

Bit	Function	Value	Description
0	TS9	0 1	None Use
• •	• •	• •	• •
7	TS16	0 1	None Use

3rd Byte

Bit	Function	Value	Description
0	TS17	0 1	None Use
• •	• •	• •	• •
7	TS24	0 1	None Use

3.2.3.3 Bit stream ordering

The transmitted ISDN bit stream is packed into the user data field of the UT-Unit in order from MSB.

Figure 3.3-2 gives the order of the ISDN bit stream and packed IrUT user data. In Fig 3.3-2, the time increases from left to right. Of course, the same method is adopted to D-Channel bits.

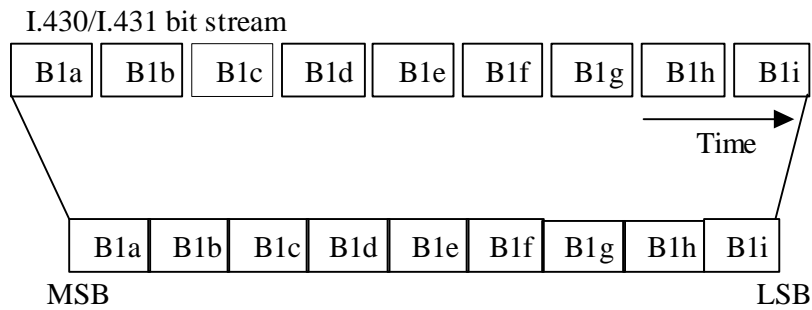
BRI bit stream**Figure 3.2-2 The order of ISDN bit stream and packed IrUT user data****3.2.3.4 Contents of UT-Parameters for ISDN**

Table 3.3-3 and 3.3-4 defines the contents of UT-Parameters for ISDN. These parameters are encapsulated into the PV field of parameter identified by PI “0x10” of the UT-Parameters. The detail of this parameter is shown in Table 2.3-3.

PI	PL	PV Description
0x00	1	Number of Supported I.430 lines
0x01	1	Number of Supported B-ch in 1st I.430 line
0x02	1	Number of Supported B-ch in 2nd I.430 line
0x03	1	Number of Supported B-ch in 3rd I.430 line
Other	Reserved	

Table 3.2-3 User Defined Data in case of I.430 Virtual Lsap-SEL

PI	PL	Contents
0x00	1	Number of Supported TS-ch (including TS24 ch)
Other	Reserved	

Table 3.2-4 User Defined Data in case of I.431 Virtual Lsap-SEL

3.3 Basic requirements

- If more than one line is used to transmit BRI data, the bits that were transmitted on the same ISDN frame in every channel should be transmitted in the same Ir frame.
- The NT-side device must be the primary station because during the real-time data transmission, the secondary station sets the clock frequency from the transmitted signals sent by mobile network. During non-synchronous data transmission periods, this procedure is necessary preparation for synchronous data transmission.

3.4 Hardware requirement

3.4.1 Bit synchronization

The difference in accuracy between the transmission clock frequency and reception clock frequency causes bit slip in the user data. In cases like voice transmission, bit slips are not so big a problem. However, in communications that require more reliability, bit slips must be extinguished.

The following mechanism is needed. The NT-side device (primary station) must send infrared signal that is synchronized with the signal from the radio network. The TE-side device (secondary station) must adjust its clock by the data signal sent from the NT device. As a result, both side devices can be synchronized with the radio network.

If there is no severe bit slip restriction, the TE-side device generates the clock.

4. IrOBEX

4.1 Data transmission overview

IrOBEX data is to be transmitted through the IrUT layer when both stations support the multi-application mode. The details of IrOBEX communication through IrUT follow the IrOBEX specification, except to using the data link through the IrUT layer not through the TinyTP layer and using the different frame format. IrOBEX data in the IrUT frame corresponds to IrOBEX frames in the TinyTP user data field.

4.2 Frame format example

Figure 4.3-1 shows an example of the frame format of IrOBEX data transmission through the IrUT layer. In this example, IrOBEX data is transmitted with BRI data.

BRI Unit						IrOBEX Unit					
H&L	Con trol	VD Lsap	Rsvd	VS Lsap	BRI Data	H&L	Con trol	VD Lsap	Rsvd	VS Lsap	IrOBEX Data

Note: H&L is Handling and Length bits.

Figure 4.2-1 Frame format example (BRI and IrOBEX data transmission)

5. IrCOMM

5.1 Data transmission overview

IrCOMM data is to be transmitted through the IrUT layer when both stations support the multi-application mode. The details of IrCOMM communication through IrUT follow the IrCOMM specification, except to using the data link through the IrUT layer not through the TinyTP layer and using the different frame format. IrCOMM data in the IrUT frame corresponds to IrCOMM frames in the TinyTP user data field.

5.2 Frame format example

Figure 5.3-1 shows an example of the frame format of IrCOMM data transmission through the IrUT layer. In this example, IrCOMM data is transmitted with BRI data.

BRI Unit						IrCOMM Unit					
H&L	Con trol	VD Lsap	Rsvd	VS Lsap	BRI Data	H&L	Con trol	VD Lsap	Rsvd	VS Lsap	IrCOMM Data

Note: H&L is Handling and Length bits.

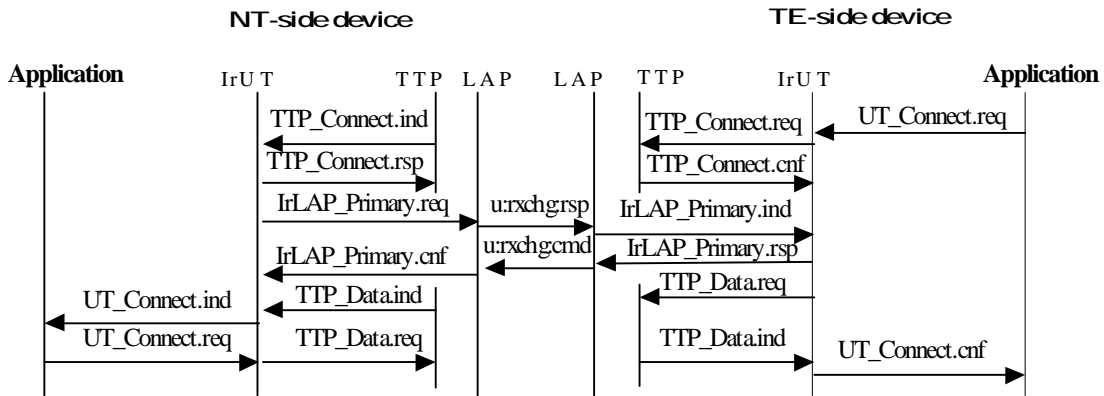
Figure 5.2-1 Frame format example (BRI and IrCOMM data transmission)

6. Service sequence example

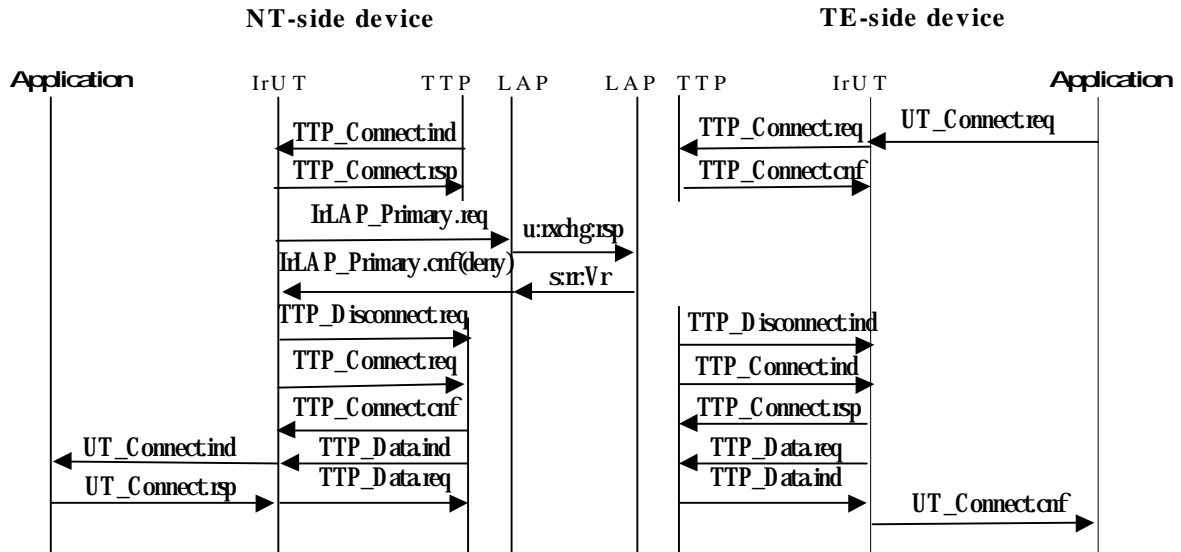
6.1 Primary / Secondary exchange

When the TE-side device initiates the IrLAP connection with the radio-network-side device, and the radio-network-side device acts as the secondary station, the radio-network-side device should exchange the roles of the primary and secondary stations. First, it tries to exchange primary/secondary roles, which is an optional procedure of IrLAP (case 1). If this fails, the portable phone shuts down the IrLAP connection and re-starts it immediately (case 2).

Case 1: The TE-side device supports IrLAP role exchange service.



Case 2: The TE-side device does not support IrLAP role exchange service.



Field	Value Type	Description
PI – Parameter Identifier	UINT8	A unique parameter identifier
PL – Parameter Length	UINT8	The length of the PV field in bytes.
PV – Parameter Value	UINT8 sequence	Value, whose meaning depends on the PI.

The Parameters of the various services of the IrDA:IrUT class are listed in the following sections. If a service is not supported the service parameters may be omitted.

The IAS parameters should be added, according to the addition of applications which works on IrUT layer.

The following table indicates actual contents of the attribute named parameter.

Table 7.1-2 Contents of parameters

PI	PI name	PL	PV data type	PV description
0x00	Multi application mode support	1	Byte (bit mask) bit 0 bit 1-7	0 : Single application mode 1 : Multi application mode Reserved
0x01	IrUT version	2	Two octets	1st octet = Major version number 2nd octet = Minor version number
0x02	Device type	1	Byte (bit mask) bit 0 bit 1-7	0 : Equipment act as NT side 1 : Equipment act as TE side Reserved
0x03	Role Exchange Support	1	Byte (bit mask) Bit 0 bit 1-7	0 : Not supporting Role Exchange 1 : Supporting Role Exchange Reserved
0x10	Virtual Lsap for ISDN BRI	1	One octet	Virtual Lsap for ISDN BRI
0x11	Virtual Lsap for ISDN PRI	1	One octet	Virtual Lsap for ISDN PRI
0x12	Virtual Lsap for IrOBEX	1	One octet	This value is same to the VlsapSel of the IAS class IrDA:IrUT:IrOBEX
0x13	Virtual Lsap for IrCOMM	1	One octet	This value is same to the VlsapSel of the IAS class IrDA:IrUT:IrCOMM
other	Reserved			

Descriptions

Multi-application mode support

This parameter indicates the presence of a multi application mode.

IrUT version	This two-octet field is provided to ensure compatibility with future versions. The first octet is the major version number and the second octet is the minor version number. This parameter is optional.
Device type	The bit 0 indicates the supportable side of the adapter (Radio network side or TE side). Connection between the same-side adapters must not be implemented. This parameter is mandatory.
Role exchange support	This parameter indicates whether the device supports the role exchange function or not.
Virtual LSAPs	These parameters indicate the virtual LSAPs for every application. With these parameters, the other device can know the supported application of this IrUT layer.

7.2 IAS Entries for Class IrDA:IrUT:IrOBEX

The information needed for accessing the IrOBEX service over IrUT layer is included in the IAS class IrDA:IrUT:IrOBEX. The following tables define the attributes associated with this class.

7.2.1 IrDA:TinyTP:LsapSel

This attribute contains the virtual LsapSel of IrOBEX over IrUT for the service. This attribute must be present for connection-oriented use.

Attribute Name	Value Type	Description
VLsapSel	Integer (0x01)	This value is the virtual LsapSel for IrOBEX

7.2.2 Other Attributes

Other attributes are same to these described in the IrOBEX specification. For further information, please refer to the IrOBEX specification.

7.3 IAS Entries for Class IrDA:IrUT:IrCOMM

The information needed for accessing the IrCOMM service over IrUT layer is included in the IAS class IrDA:IrUT:IrCOMM. The following tables define the attributes associated with this class.

7.3.1 IrDA:TinyTP:LsapSel

This attribute contains the virtual LsapSel of IrCOMM over IrUT for the service. This attribute must be present for connection-oriented use, whatever the class name is.

Attribute Name	Value Type	Description
VLsapSel	Integer (0x01)	This value is the virtual LsapSel for IrCOMM

7.3.2 Other Attributes

Other attributes are same to these described in the IrCOMM specification. For further information, please refer to the IrCOMM specification.

7.4 Service Hint Bit

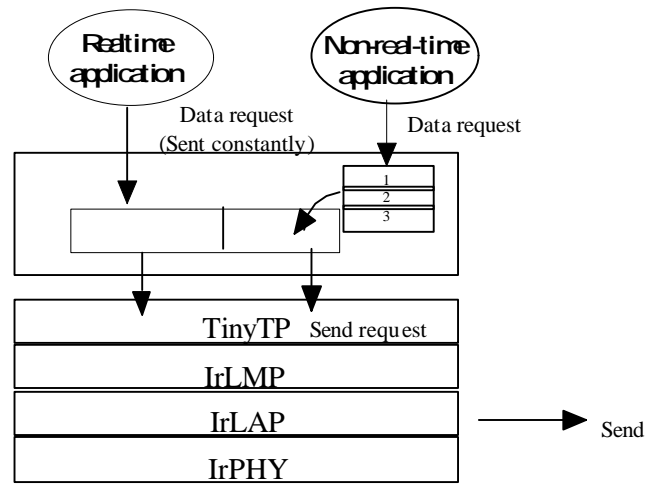
The Telephony service hint bit (bit 8) of the IrLMP service hints issued in the device discovery to inform about the Telecom application capabilities of the device. [IrLMP]

It should be noted that the Telephony bit does not indicate the type of device in question. It only points out that the device supports some Telecom services specified in the phase 1 IrMC specifications and this document (the phase 2 IrMC specifications). Consequently, all PCs, pagers and other non-phone devices are also expected to indicate their Telecom capability with the Telephony bit.

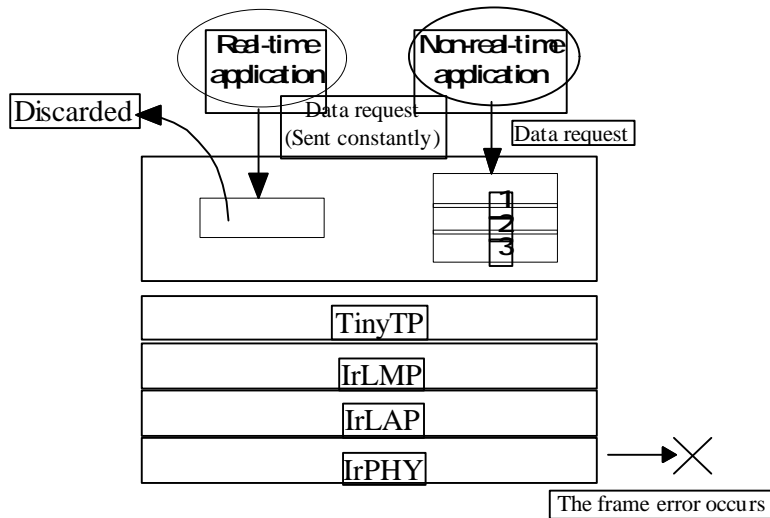
Appendix A Overview of procedure for errors

This section describes the procedure when the frame error occurs. During the data transmission, there are possibilities of the frame error like CRC error.

Case 1: During normal communication



(a)



Case 2: During error occurring

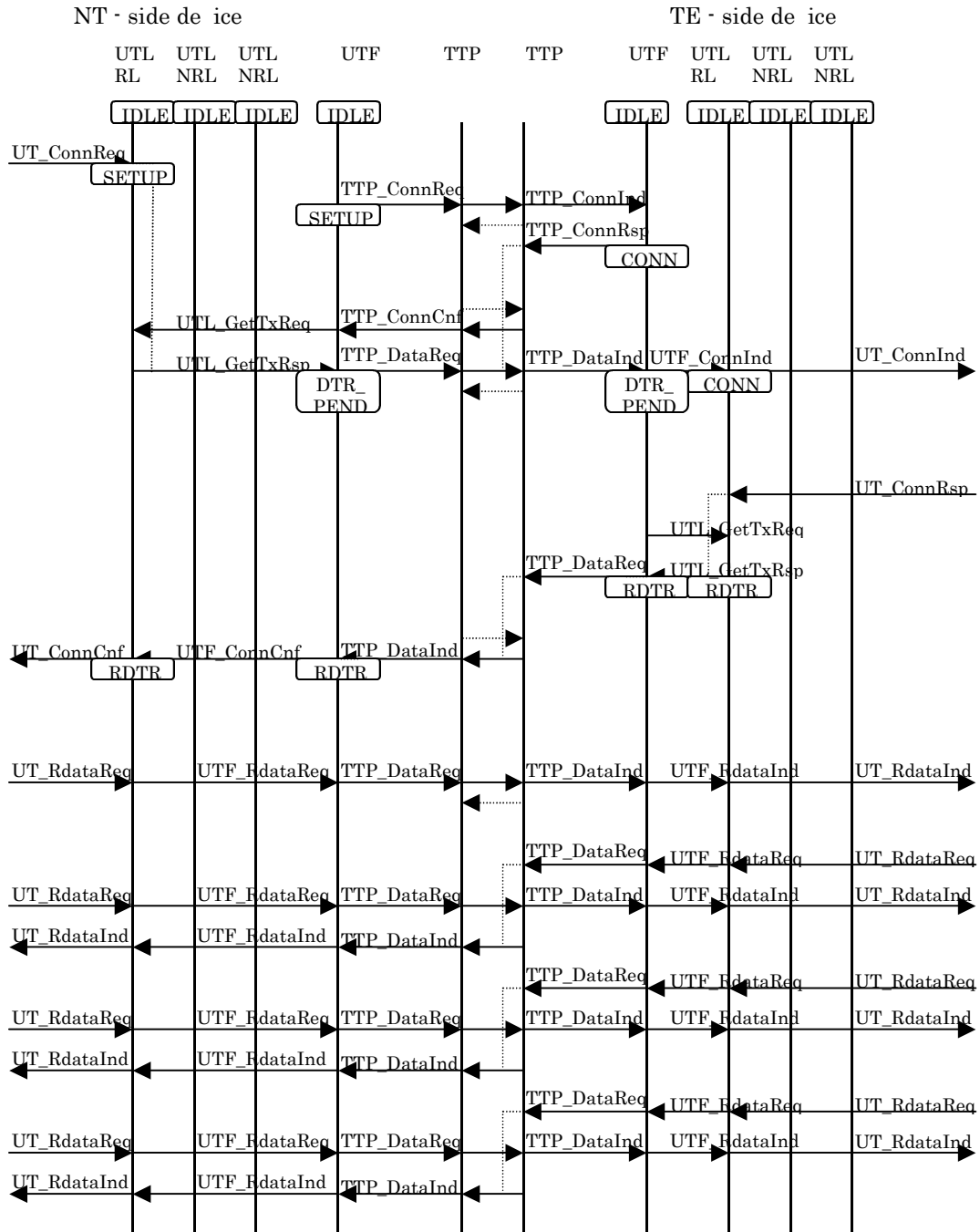
(b)

Figure A-1 Overview of procedure for errors

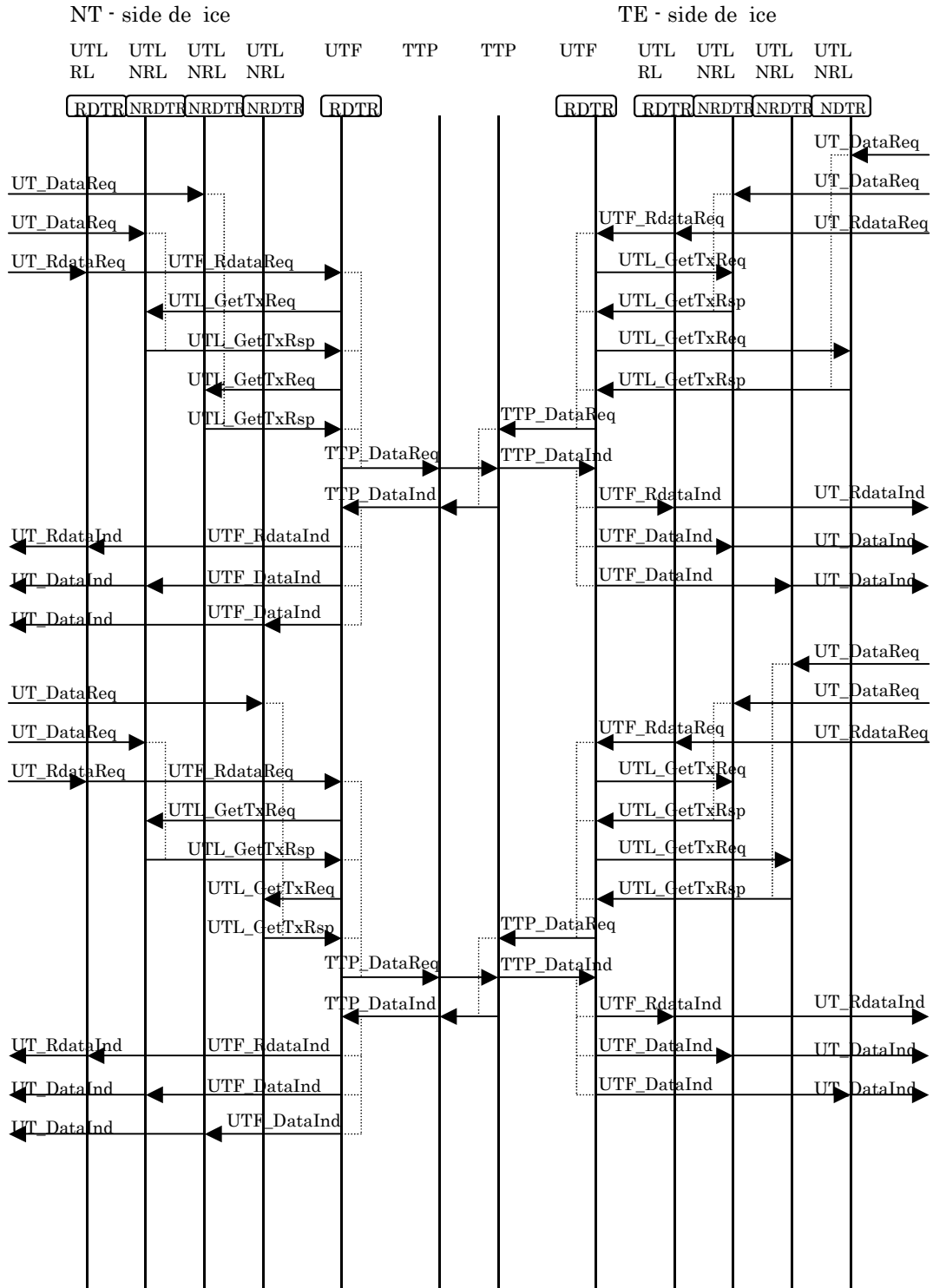
Appendix B The Detailed Sequence of IrUT Connection

These sequences are drawn based on the State Transition Table. In the sequence diagrams, the "RL" represents a real-time link, and the "NRL" represents a non-real-time link.

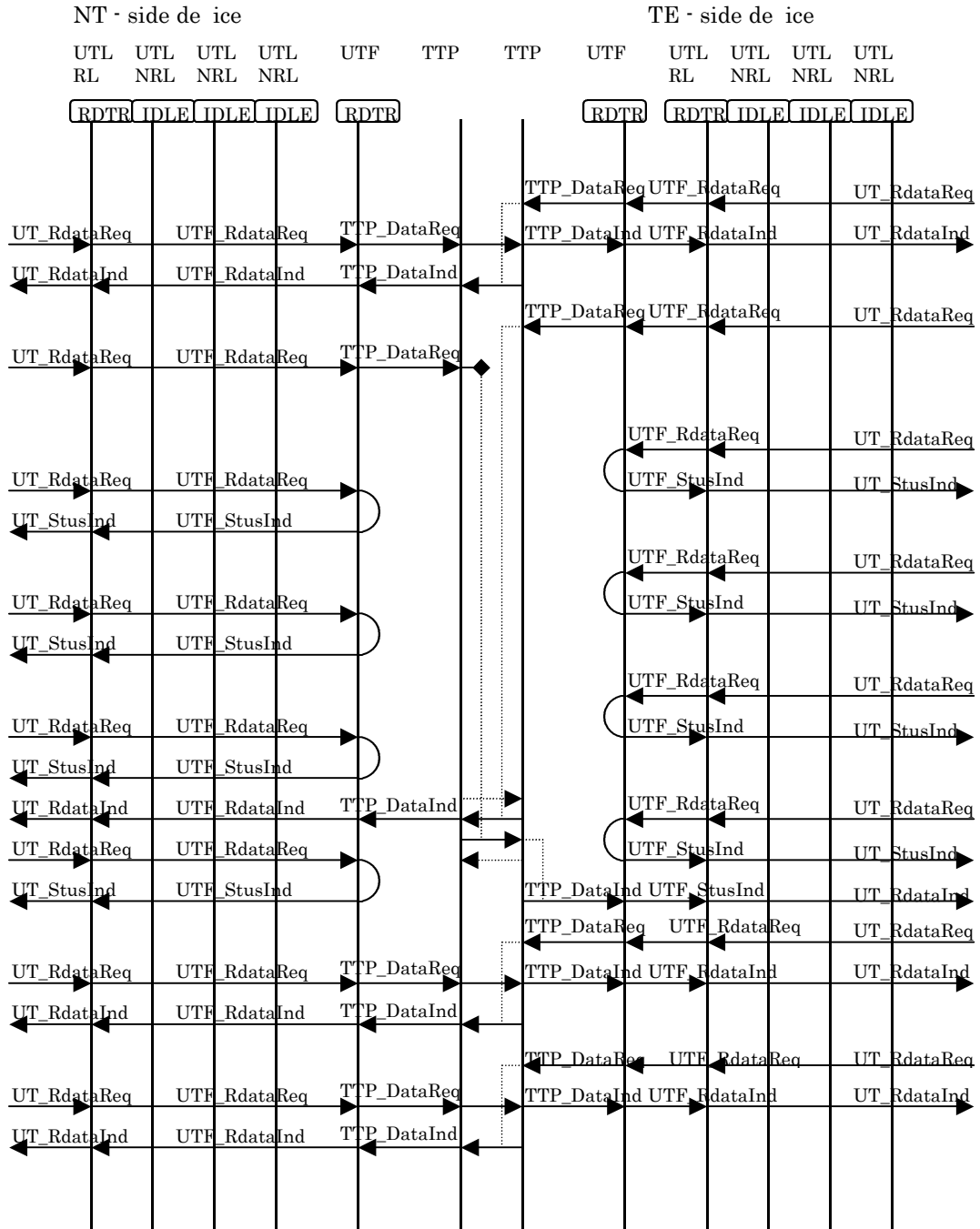
B-1 The First Link Connection Establishment Procedure.



B-2 Data Transmission Procedure



B-3 Communication Error Occurs



NT - side de ice

UTL RL UTL NRL UTL NRL UTL NRL UTF TTP TTP UF UTL RL UTL NRL UTL NRL UTL NRL

RDTR IDLE IDLE IDLE RDTR RDTR RDTR IDLE IDLE IDLE

SETUP

CONN

NRDTR

NRDTR

B-5 The Last Link Disconnect Procedure.

