ETSI EN 300 797 V1.2.1 (2005-05)

European Standard (Telecommunications series)

Digital Audio Broadcasting (DAB);
Distribution interfaces;
Service Transport Interface (STI)

European Broadcasting Union Union Européenne de Radio-Télévision

EBU·UER

Reference

REN/JTC-DAB-37

Keywords

audio, broadcasting, DAB, data, digital, interface, transport

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00   Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This European Standard (Telecommunications series) has been produced by Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI).

NOTE 1: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of broadcasting and related fields. Since 1995 the JTC Broadcast became a tripartite body by including in the Memorandum of Understanding also CENELEC, which is responsible for the standardization of radio and television receivers. The EBU is a professional association of broadcasting organizations whose work includes the co-ordination of its members' activities in the technical, legal, programme-making and programme-exchange domains. The EBU has active members in about 60 countries in the European broadcasting area; its headquarters is in Geneva.

European Broadcasting Union
CH-1218 GRAND SACONNEX (Geneva)
Switzerland
Tel: +41 22 717 21 11
Fax: +41 22 717 24 81

The Eureka Project 147 was established in 1987, with funding from the European Commission, to develop a system for the broadcasting of audio and data to fixed, portable or mobile receivers. Their work resulted in the publication of a European Standard, EN 300 401 [1], for DAB (see note 2) which now has world-wide acceptance. The members of the Eureka Project 147 are drawn from broadcasting organizations and telecommunication providers together with companies from the professional and consumer electronics industry.

NOTE 2: DAB is a registered trademark owned by one of the Eureka Project 147 partners.

National transposition dates	
Date of adoption of this EN:	27 May 2005
Date of latest announcement of this EN (doa):	31 August 2005
Date of latest publication of new National Standard or endorsement of this EN (dop/e):	28 February 2006
Date of withdrawal of any conflicting National Standard (dow):	28 February 2006

Introduction

The present document is one of a set associated with DAB. EN 300 401 [1] describes the transmitted signal; the interface between the broadcaster's transmitters and the listener's receiver. The associated documents, EN 300 798 [2] and ETS 300 799 [3] describe additional interfaces which can be used by broadcasters or network providers to build DAB networks.

Figure 1 shows a DAB network in outline. For convenience, the Network is split into a number of different parts, each managed by a different entity. The different entities are; the Programme/Data provider, the Service Component provider, the Ensemble provider and the Transmission Network provider.

NOTE: A Service Component provider may be generating a full DAB service or a component of a DAB service. For the purposes of the present document, the terms Service provider and Service Component provider are interchangeable.

Programme/Data provider

The Programme/Data provider is the originator of the audio programme or the data being carried within the DAB Service Component. The format for the output of the Programme/Data provider may take many different forms and should be agreed between the Programme/Data provider and the Service Component provider.

Service Component provider

The Service Component provider is producing one or more complete service components which may form the complete DAB Service, but may not. Data from the Service Component provider will comprise three different parts:

- Service Component data which is to be inserted into the DAB Main Service Channel (MSC);
- Service Information related to the Service Component data which is to be inserted into the Fast Information Channel (FIC);
- other data, not intended for transmission, including status monitoring or control.

The interface between the Service Component provider and the Ensemble provider is known as the Service Transport Interface (STI) and is the subject of the present document.

Ensemble provider

The Ensemble provider receives a set of service components from one or more Service Component providers. He then formats the FIC, and generates an unambiguous description of the full DAB ensemble.

The ensemble description is passed to the Transmission Network provider via an interface called the Ensemble Transport Interface (ETI) which is defined in ETS 300 799 [3].

Transmission Network provider

The Transmission Network provider generates the DAB Ensemble and transmits it to the receiver. The output of the Transmission Network provider is defined by EN 300 401 [1]. The Transmission Network provider is usually the final recipient of the ETI and is responsible for turning it into the DAB transmission signal using an OFDM generator.

In some cases, as an intermediate step, the Transmission Network provider may find it convenient to generate a baseband representation of the signal to be transmitted. The baseband representation, known as the Digital baseband I/Q Interface (DIQ), is a set of digital samples defining the In-phase (I) and Quadrature (Q) components of the final carrier. This interface is defined in EN 300 798 [2], and provides a convenient interface between digital processing equipment and radio-frequency modulating equipment.

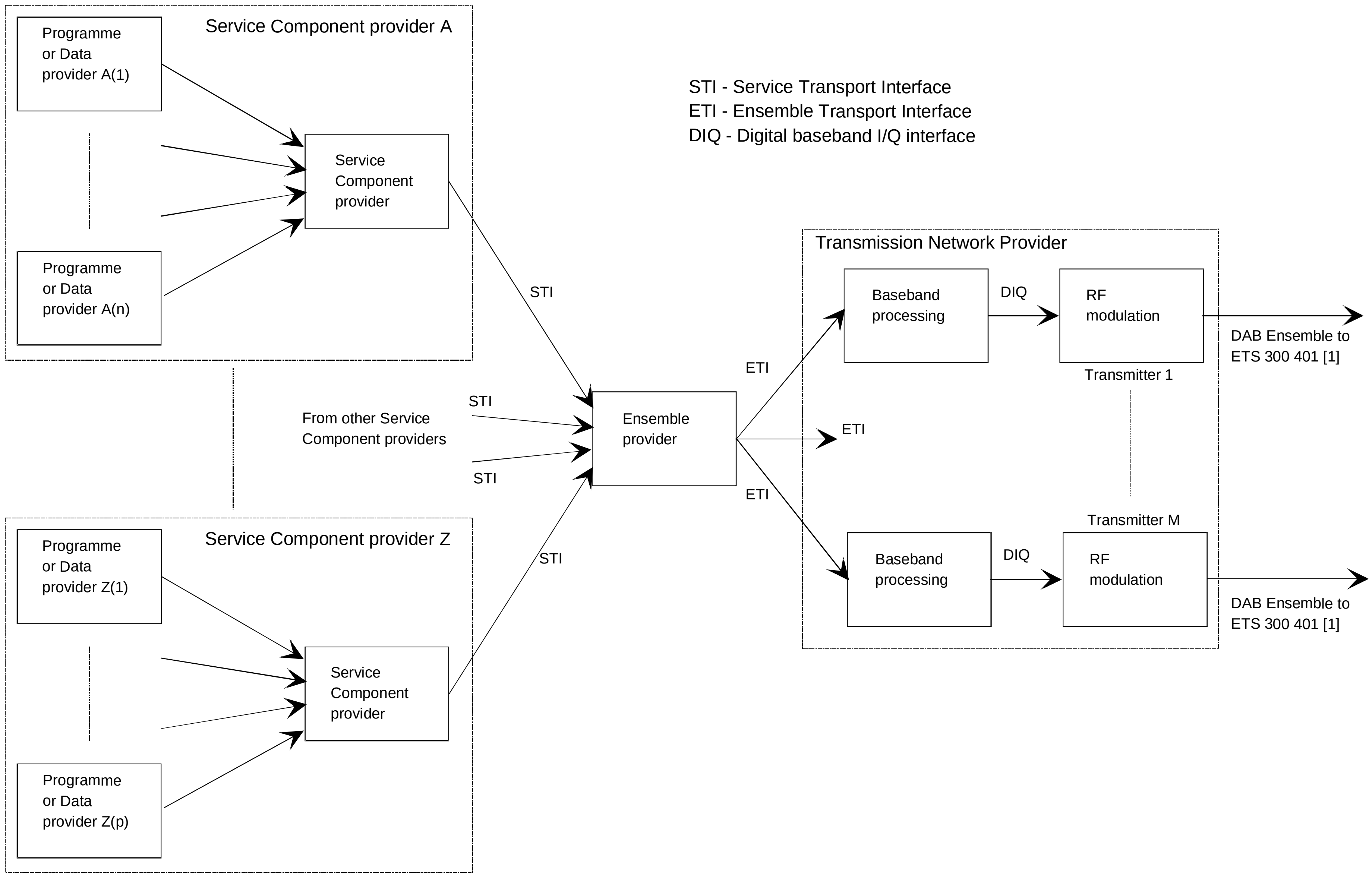

Figure 1: DAB network outline

1 Scope

The present document establishes a standard method for transporting Service components (audio and data) produced by Service providers at their own studios to the DAB multiplexing equipment located at the Ensemble provider's centre.

EN 300 401 [1] established a broadcasting standard for a DAB system. Broadcasters who implement DAB networks require methods for transporting DAB signals, or the component parts of a DAB signal, between studios, where the programme or data service originates, and the transmitter sites from which the signal will be radiated. The network of circuits connecting the studios to the Ensemble provider's ensemble multiplexer is generally known as the Collection Network. The network of circuits connecting the ensemble multiplexer to the transmitters is generally known as the Distribution Network.

The present document is applicable to Collection Networks used in a DAB System. It describes the characteristics of a signal suitable for transporting Service Components, Service Information and control data between a Service provider and an Ensemble provider. The interface is suitable for use on a number of different physical media and telecommunication networks. Provision is made for the inclusion of appropriate error detection and correction and for the management of network transit delay.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

[1] ETSI EN 300 401: "Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers".

[2] ETSI EN 300 798: "Digital Audio Broadcasting (DAB); Distribution interfaces; Digital baseband In-phase and Quadrature (DIQ) Interface".

[3] ETSI ETS 300 799: "Digital Audio Broadcasting (DAB); Distribution interfaces; Ensemble Transport Interface (ETI)".

[4] ITU-T Recommendation G.703: "Physical/electrical characteristics of hierarchical digital interfaces", (section 6: "Interface at 2048 kbit/s).

[5] ITU-T Recommendation X.24: "List of definitions for interchange circuits between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) on public data networks".

[6] ITU-T Recommendation V.11: "Electrical characteristics for balanced double-current interchange circuits operating at data signalling rates up to 10 Mbit/s".

[7] ITU-T Recommendation G.704: "Synchronous frame structures used at 1544, 6312, 2048, 8448 and 44 736 kbit/s hierarchical levels", (section 2.3: "Basic frame structure at 2048 kbit/s).

[8] ITU-T Recommendation G.706: "Frame alignment and cyclic redundancy check (CRC) procedures relating to basic frame structures defined in Recommendation G.704".

[9] ITU-T Recommendation V.24: "List of definitions for interchange circuits between data terminal equipment (DTE) and data circuit-terminating equipment (DCE)".

[10] ITU-T Recommendation V.28: "Electrical characteristics for unbalanced double-current interchange circuits".

[11] ITU-T Recommendation H.221: "Frame structure for a 64 to 1920 kbit/s channel in audiovisual teleservices".

[12] ITU-T Recommendation H.242: "System for establishing communication between audiovisual terminals using digital channels up to 2 Mbit/s".

[13] IEC 60958 (all parts): "Digital audio interface".

[14] ISO/IEC 646: "Information technology - ISO 7-bit coded character set for information interchange".

3 Definitions, abbreviations, symbols and terminology

3.1 Definitions

For the purposes of the present document, the terms and definitions given in EN 300 401 [1] and the following apply:

asynchronous FIB insertion: method of providing Fast Information Blocks (FIBs) in the STI-D(LI) and to incorporate them in the Fast Information Channel (FIC)

NOTE: The FIBs shall be inserted in the FIC asynchronously and no constant time relation shall be expected between generation and insertion of a FIB.

block: component part of an STI(PI, G.704/1) multiframe consisting of eight G.704 frames

NOTE: Each block comprises 256 bytes.

codeword: Reed-Solomon codeword, as used by STI(PI, G.704/1), comprises 240 bytes

NOTE: Some of these bytes are data bytes, others are check bytes.

coding array: array used in the conceptual description of STI(PI, G.704/1)

collection network: network of circuits in a DAB network that connects Service providers to an Ensemble provider, or Service providers with each other

command (CMD): part of STI-C(LI) that defines the command carried in a message

command extension (EXT): part of STI-C(LI) giving information in addition to the command (CMD)

configuration: description of services, service components and the way they shall be incorporated in the DAB multiplex

Control Frame field (CF): part of the generic transport frame, STI(PI, X), that carries the transport adapted control part

Control Frame Size field(CFS): part of the generic transport frame, STI(PI, X), that defines the length of the control frame field

data exchange session: mechanism used in STI-C(LI) to exchange a group of messages

Data Frame field (DF): part of the generic transport frame, STI(PI, X), that carries the data part of the STI

Data Frame Count Field (DFCT): part of STI-D(LI) containing the data frame count

NOTE: The data frame count consists of a higher part (DFCTH) and a lower part (DFCTL).

Data Frame Dadding field (DFPD): part of the generic transport frame, STI(PI, X), that carries padding in data frame field

Data Frame Size (DFS): part of the generic transport frame, STI(PI, X), that defines the length of the data frame field

Data Length field (DL): part of STI-D(LI) giving information about the length of the data part frame

data link layer: layer of the control part transport adaptation, STI-C(TA), that contains data link packets and provides framing and error protection

distribution network: network of circuits in a DAB network that connects the Ensemble provider's ensemble multiplexer to transmitters of Transmission Network providers

downstream entity: one of the two entities involved in the exchange of STI-D(LI) and STI-C(LI) information

NOTE: The downstream entity shall be the receiver of the STI-D(LI) flow provided by the upstream entity.

End of Frame field (EOF): part of STI-D(LI) containing End-Of-Frame information

End of Header field (EOH): part of STI-D(LI) containing End-Of-Header information

ensemble multiplex: set of data which describes the component parts of the DAB ensemble

ensemble multiplexer: multiplexer which generates an ensemble multiplex

FIB grid: vector of boolean values used by the Ensemble provider to predefine in which CIFs a Service provider may insert FIBs using the synchronous FIB insertion method

NOTE: The FIB grid can be exchanged using the STI-C(LI) messages.

FIG change control: mechanism that the creator of Fast Information Groups (FIGs) applies when parts of a FIG's information change in order to signal the change to DAB receivers

FIG file: file that can be exchanged using STI-C(LI) messages

FIG repetition: mechanism that the creator of the Fast Information Channel (FIC) applies to FIGs, where FIGs are periodically repeated

NOTE: The period for the repetition typically depends on the content of the FIG, e.g. the importance of the content for a DAB receiver, and the capacity of the FIC.

Frame Characterization field (FC): part of STI-D(LI) containing frame characterization data

Frame Padding field (FRPD): part of the generic transport frame, STI(PI, X), containing frame padding

Frame Synchronization field (FSYNC): synchronization field of the generic transport frame, STI(PI, X)

generic transport frame, STI(PI, X): generic frame structure for the transport of the STI data and control part on physical interfaces

NOTE: The generic transport frame can either contain the data part, the control part or both.

GF(2^8): mathematical entity (a Gallois Field of 256 entries) used in the process of producing Reed-Solomon error protection bytes

header cyclic redundancy checksum (CRCH): part of STI-D(LI) containing a cyclic redundancy checksum for header information

individual stream characterization field ($ISTC_n$): part of STI-D(LI) giving the stream characterization of the individual stream *n*, carried in the STI-D(LI)

individual stream data field ($ISTD_n$): part of the STI-D(LI) main stream data, carrying an individual stream

local entity: one of the two entities involved in an STI-C(LI) message exchange

NOTE: The local entity shall be the sender of the message to be received by the remote entity.

Logical Interface (LI): definition of the STI which contains all the elements to be carried by the interface, but has no physical manifestation

logical interface layer: upper layer of the STI managing the logical interface definition of the STI

NOTE: The logical layer manages STI-C(LI) messages or STI-D(LI) frames.

logical layer: layer of the control part transport adaptation, STI-C(TA), that manages the transfer of messages between the STI-C(LI) and the transport layer

Main Stream data field (MST): part of STI-D(LI) carrying the collection of the individual stream data originated by the Service provider

message: syntactical unit managed by the STI-C(LI)

NOTE: A message is a string of characters beginning with a *CMD* field and ending with a *DELIM* field.

multiframe: composite frame structure used in STI(PI, G.704/1) to map the generic STI(PI, X) transport frame onto the elemental G.704 frames

network layer: layer of the control part transport adaptation, STI-C(TA), that contains network packets and provides the indication of source and destination addresses

Number of Streams field (NST): part of STI-D(LI) giving information about the number of streams being carried

open connection: state of the STI interface allowing upstream and downstream entities to exchange STI-C(LI) messages

packet: basic unit of information carried on the different layers of the STI-C(TA)

packet number field (PKTNUM): part of the STI-C(TA) carrying a sequential number attached to logical packets

Physical Interface (PI): generic term to name the physical implementation of the STI

reconfiguration: procedure allowing to modify a configuration and, as a consequence, the DAB Ensemble multiplex

Reed-Solomon forward error coding: form of coding which allows the correction of transmission errors

remote entity: one of the two entities involved in an STI-C(LI) message exchange

NOTE: The remote entity shall be the receiver of the message sent by the local entity.

separator field (SEP): part of STI-C(LI) managing the separation of the data fields of a message

Service Provider Identifier (SPID): part of STI-D(LI) allowing the recipient of the STI to uniquely identify the originator of the STI

Status field (STAT): part of the STI-D(LI) carrying status information about the STI-D(LI)

NOTE: The STAT field can be modified by physical interfaces to allow status information to be updated as the signal is carried through the collection network.

STI-C(LI): logical definition of the STI control part. It is composed of several message categories which allow the STI to be managed. It has no physical manifestation

STI-C(TA): protocol layers providing transport adaptation for safe and reliable transport of the STI-C(LI)

STI-D(LI): logical definition of the STI data part

NOTE: It defines the syntax of the frames used to carry the Service provider's broadcast data. It has no physical manifestation.

STI(PI, X): generic transport frame structure used in all physical implementations of the STI

Stream Characterization field (STC): part of STI-D(LI) carrying the collection of the individual stream characterization information originated by the Service provider

stream CRC ($CRCST_n$): cyclic redundancy checksum calculated on the individual stream data field *n*

stream CRC flag (CRCSTF): flag in STI-D(LI) indicating the presence of the stream cyclic redundancy checksum

Stream identifier (STID): part of STI-D(LI) used to uniquely identify each individual data stream

Stream length field (STL_n): part of STI-D(LI) carrying the length in bytes of the individual data stream *n*

Synchronization field (SYNC): A part of STI(PI, X) which carries status information and signifies the start of the frame

synchronous FIB insertion: method of providing Fast Information Blocks (FIBs) in the STI-D(LI) and to incorporate them in the Fast Information Channel (FIC)

NOTE: The FIB insertion is governed by the use of the CIF count and a FIB grid.

Time stamp field (TIST): part of STI-D(LI) comprising a 24-bit timestamp

transmit packet stack: storage memory, organized as a stack of transport packets, allowing the implementation of the retransmission feature provided by the STI-C(TA)

Transport Adaptation (TA): adaptation of the STI-C(LI) allowing safe and reliable transport of STI-C(LI) messages

Transport Frame Header (TFH): part of STI(PI, X) carrying information about the lengths of the data and control parts of the STI

Transport Frame Length (TFL): length in bytes of the STI(PI, X) frame

transport layer: layer of the control part transport adaptation, STI-C(TA), that contains transport packets and provides safe and reliable transport

Type Identifier (TID): part of STI-D(LI) giving information about the type of an individual stream data field

type identifier extension (TIDext): part of STI-D(LI) giving additional information about each type of individual stream data field

upstream entity: one of the two entities involved in the exchange of STI-D(LI) and STI-C(LI) information

NOTE: The upstream entity shall be the originator of the STI-D(LI) flow provided to the downstream entity.

WG1/2 interface: physical interface, designed by the EUREKA 147 project, to carry several service components on a local point-to-point connection

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in EN 300 401 [1] and the following apply:

ACK	ACKnowledge
ACKNUM	ACKnowledge NUMber
ASS	ASynchronous Signalling
C-TADATA	Control part Transport Adaptation DATA
CF	Control Frame
CFS	Control Frame Size
CRC	Cyclic Redundancy Checksum
CRCH	Header Cyclic Redundancy Checksum
CRCSTF	Cyclic Redundancy Checksum STream Flag
CRCST	Stream Cyclic Redundancy Checksum
CTL	Control part Transport adapted Length
D-LIDATA	Data part Logical Interface DATA
DIQ	Digital baseband In-phase / Quadrature Interface
DAD	Destination ADdress
DE	Downstream Entity
DF	Data Frame
DFCT	Data Frame CounT
DFCTH	Data Frame CounT - Higher part
DFCTL	Data Frame CounT - Lower part
DFPD	Data Frame PaDding
DFS	Data Frame Size

DL	Data Length
EOF	End Of Frame
EOH	End Of Header
EPID	Ensemble Provider IDentifier
FC	Frame Characterization
FL	Frame Length
FRPD	FRame PaDding
FSS	Frame Synchronous Signalling
FSYNC	Frame SYNChronization
ISTC	Individual STream Characterization field
ISTD	Individual Stream Data
ITU	International Telecommunication Union
LI	Logical Interface
LSb	Least Significant bit
MSb	Most Significant bit
MST	Main STream data
NASC	Network Adapted Signalling Channel
NST	Number of STreams
PA	Packet Address
PI	Physical Interface
PKTNUM	PacKet NUmber
REP	REPetition index
Rfa	Reserved for future addition
Rfu	Reserved for future use
SAD	Source ADdress
SEP	SEParator
SPID	Service Provider IDentifier
STAT	STATus
STC	STream Characterization
STI	Service Transport Interface
STID	STream IDentifier
STL	STream Length
SYNC	SYNChronization
TA	Transport Adaptation
TFH	Transport Frame Header
TFL	Transport Frame Length
TID	Type IDentifier
TIDext	Type IDentifier extension
TIST	TIme STamp
UE	Upstream Entity

3.3 Symbols

For the purposes of the present document, the following mathematical symbols apply:

3.3.1 Numerical ranges

[m...n]	denotes the numerical range m, m + 1, m + 2, ... , n, where m and n are positive integers with n > m
X_{16}	the subscript "16" is used to denote hexadecimal numbers

3.3.2 Bit and byte numbering

b_n	denotes bit number *n*. *n* is in the range [0...7]
$b_{k..n}$	denotes bit numbers *k* to *n*. *k* and *n* are in the range [0...7] and $k < n$
$b_{k,n}$	denotes bit numbers *k* and *n*. *k* and *n* are in the range [0...7] and $k < n$
$B_{m,f}$	denotes byte number *m* in frame number *f*
$B_{m,f}(b_n)$	denotes bit number *n* of byte *m* in frame *f*
x	is used to denote an arbitrary binary value (0 or 1)

NOTE: *f* may sometimes be omitted where no ambiguity results.

3.3.3 Arithmetic operators

+	Addition
×	Multiplication
m DIV p	denotes the quotient part of the division of m by p (m and p are positive integers)
m MOD p	denotes the remainder of the division of m by p (m and p are positive integers)
$\sum_{i=p}^{q} f(i)$	denotes the sum: $f(p) + f(p + 1) + f(p + 2) ... + f(q)$
$\prod_{i=p}^{q} f(i)$	denotes the product: $f(p) \times f(p + 1) \times f(p + 2) ... \times f(q)$

3.3.4 Logical operators

AND	Logical AND function
OR	Logical OR function
XOR	Exclusive-OR function

3.3.5 STI-C(LI) Field Types

string	a sequence of characters
char	a single character
dec	decimal number
hex	hexadecimal number

3.4 Ordering of bytes and bits for transmission

The bytes of each STI frame shall be transmitted sequentially with the lowest numbered byte being transmitted first e.g. byte $B_{0,f}$ is transmitted before byte $B_{1,f}$ and so on. The highest numbered byte of frame *f* shall be transmitted earlier than the lowest numbered byte of frame $f + 1$.

The bits of each byte shall be transmitted sequentially with the lowest numbered bit being transmitted first e.g. $B_{0,1}(b_0)$ is transmitted before byte $B_{0,1}(b_1)$, and so on.

Unless otherwise stated in the associated text, data shall be carried with its MSb in the lowest numbered bit of the lowest numbered byte. The next most significant bit shall be carried in the next lowest numbered bit of the lowest numbered byte, and so on. This implies that the MSb of any data byte shall be received earlier than the LSb.

3.5 Reserved bits

In some fields of the STI, unused bits may be found. These are designated as:

Rfa: Reserved for future addition. The future use of Rfa bits shall not modify the usage of other bits in the same field as the Rfa bits.

Rfu: Reserved for future use. The future use of Rfu bits can modify the usage of other bits in the same field as the Rfu bits.

Unless otherwise specified, the values of bits designated as either Rfa or Rfu shall be set to zero.

3.6 STI-C character set

The STI-C character set that shall be used for information exchange shall be made of 8-bit characters where the most significant bit shall be set to zero and the remaining 7 bits shall be coded as defined in ISO/IEC 646 [14]. The International Reference Version (IRV) shall be used.

3.6.1 STI-C(LI) message character set

STI-C(LI) messages shall be restricted to use only the following subset of characters:

- the SPACE character (bit combination 2/0) used only as a separator between data fields;
- all the G0 characters (bit combinations 2/1 to 7/14) with the restriction that the semicolon (represented by bit combination 3/11) shall only be used as the delimiter of STI-C(LI) messages.

A decimal character shall be coded as a character in the range of "0" to"9" (bit combination 3/0 to 3/9).

A decimal number shall be coded as a string of decimal characters. Leading zeros shall not be coded.

A hexadecimal character shall be coded as a character in the range of "0" to "9" and "A" to "F" (bit combination 3/0 to 3/9 and 4/1 to 4/6).

A hexadecimal number shall be coded as a string of hexadecimal characters. Leading zeros shall be coded (e.g.: "0A23", "00F7").

3.6.2 STI-C(TA) character set

STI-C(TA) shall be restricted to use only the following subset of characters:

- all the characters used by the STI-C(LI) messages as defined in clause 3.6.1;
- the carriage return (CR) character (bit combination 0/13);
- the line feed (LF) character (bit combination 0/10).

4 Overview of the Service Transport Interface definition

4.1 Conceptual model of the Service Transport Interface

Figure 2 is based on a figure from EN 300 401 [1] and shows the conceptual block diagram of the first part of the DAB emission system. The conceptual locations of the Service Transport Interface (STI) and the Ensemble Transport Interface (ETI) (see bibliography) are shown on this diagram.

The Service Transport Interface (STI) has been defined to provide a standardized way of transporting DAB service components, service information and control messages in a DAB collection network. It consists of two parts: the data part, STI-D, which carries data intended for broadcast, and the control part, STI-C, which carries management messages, for control and monitoring purposes, and which is not intended for broadcast.

The STI-D can carry audio components, stream mode and packet mode data components, Fast Information Blocks (FIBs) and Fast Information Groups (FIGs). As all of those are broadcast elements, STI-D is unidirectional in nature.

The STI-C can carry configuration information, requests for reconfiguration, monitoring information and management information. Since this type of information requires a dialogue between the two entities involved, STI-C is bi-directional in nature.

The present document uses a layered approach to specify the two parts of the STI:

- The Logical Interface (LI) layer, specifies the syntax of the two STI parts: the STI-D frame and STI-C messages.

- The Transport Adaptation (TA) layer is specified only to be used with the STI-C part, to ensure an error-free transport of STI-C messages and that they will be received in the order they were sent.

The Physical Interface (PI) layer, provides a physical manifestation of the STI and since STI is intended to be used in a wide variety of collection networks, various physical implementations - for the envisaged types of collection networks - are defined to transport the two logical parts of the STI. These two parts may be carried together or separately depending on the chosen collection network topology.

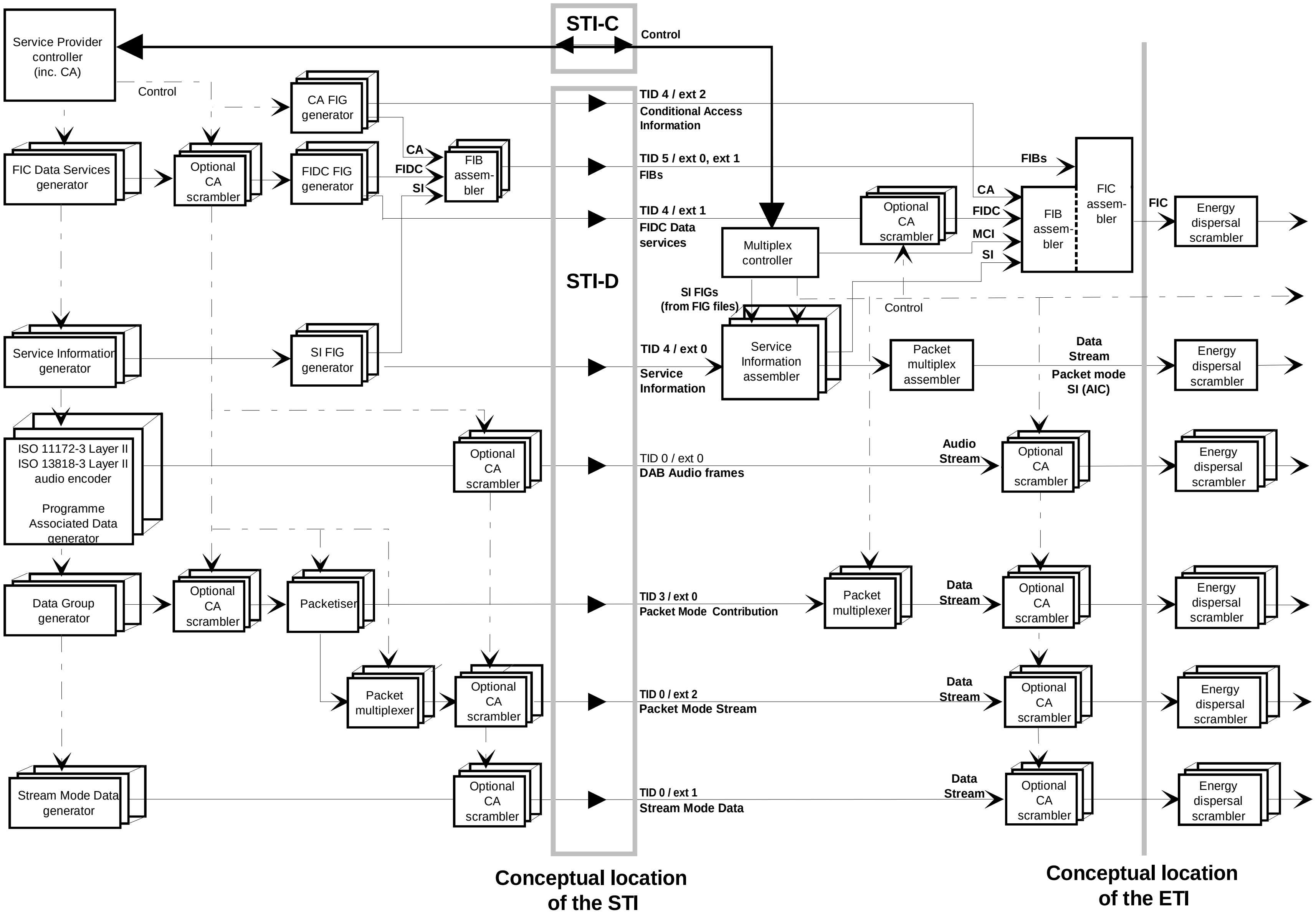

Figure 2: Conceptual DAB emission block diagram showing the location of the STI

4.2 The logical model of the STI

Figure 3 shows the logical connection of the STI. It is point to point between the upstream entity (UE), a Service provider and the downstream entity (DE), typically an Ensemble provider. Upstream and downstream are defined in relation to the STI-D(LI) data flow.

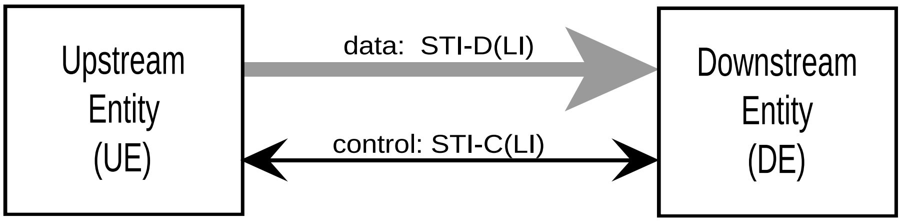

Figure 3: Logical connection of STI

The STI identifies each entity in the collection network with an identifier. These identifiers are used in the STI-D(LI) to identify the source of the data, and in the STI-C(TA) to identify both the source and the destination of the messages. These identifiers are called the service provider identifier (SPID) and the ensemble provider identifier (EPID). Typically the EPID should be the DAB Ensemble Identifier (EId), but it need not be.

The STI-D(LI) includes all possible service component formats to be inserted in the MSC channel, but also Service Information, Fast Information Data Channel service components and Conditional Access (CA) information that will be transported in the FIC. The STI-D(LI) is based on a 24 ms frame structure.

The data streams transported by the STI-D(LI) are classified as follows:

- MSC sub-channel:
    - an audio service component;
    - a data stream service component;
    - a data stream made of one or more packet mode service components (and padding).
- MSC sub-channel contribution:
    - a packet mode data service component.
- FIC FIG stream:
    - Service Information;
    - Fast Information Data Channel service component;
    - CA information.
- FIC FIB stream:
    - Asynchronous FIB insertion;
    - Synchronous FIB insertion.

The STI-C(LI) provides a set of messages that allows both entities to setup and control the behaviour of the information provided in the STI-D(LI). The STI-C(LI) messages are composed of characters and do not have a 24 ms frame structure.

The messages transported by the STI-C(LI) are classified as follows:

- Action messages used to handle reconfigurations;
- Configuration messages used to exchange configuration information;
- FIG file messages used to exchange FIG files and to manage their use;
- FIB grid messages used to exchange a FIB grid and to manage its use;
- Resource messages used to exchange information about scarce resources;
- Information messages used to exchange information concerning capabilities and counters;
- Supervision messages used for monitoring and alarm purposes.

4.3 The layered model of the STI

The STI is defined in three layers as shown in figure 4.

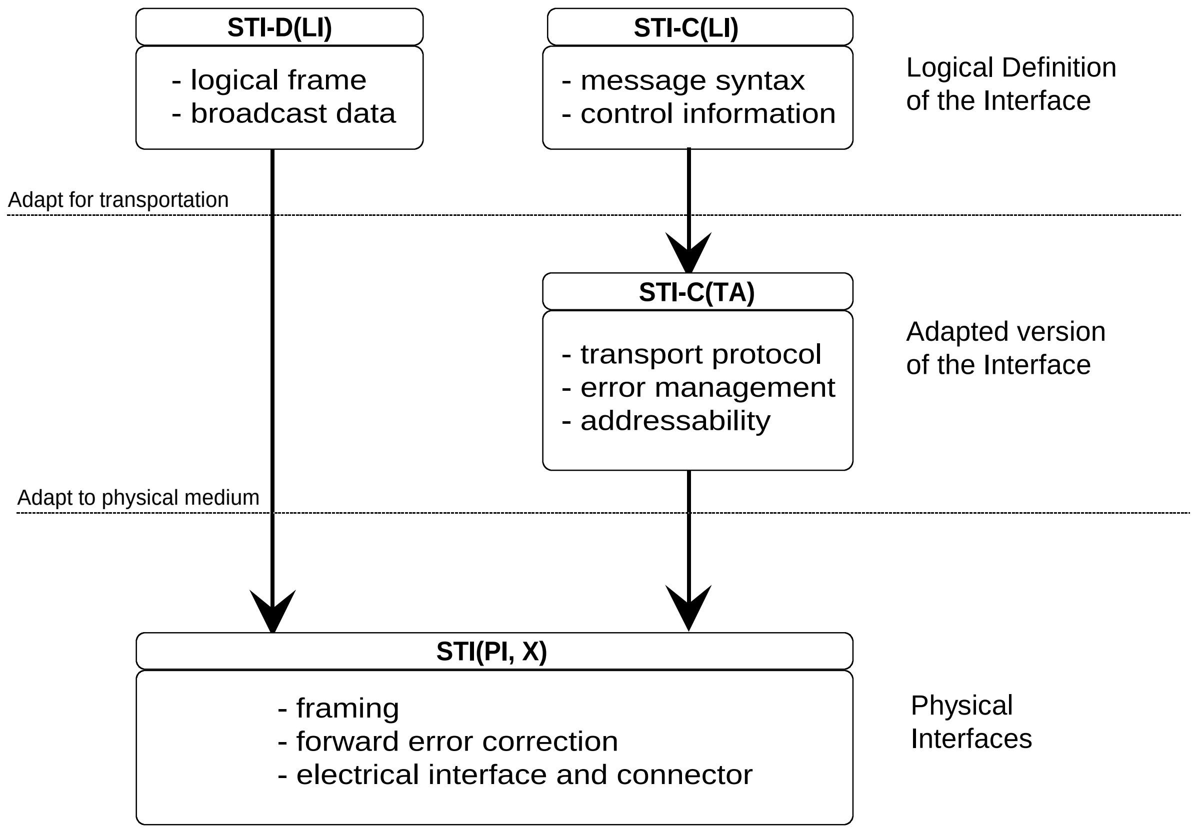

Figure 4: The layers of the STI

The uppermost layer in figure 4 shows the **Logical Interface (LI)** layer, which consists of the two logical parts, a data part and a control part as described in clause 4.2. These two parts provide a definition of the STI interface at its simplest level and has no physical manifestation. The data part, STI-D(LI), contains the service components to be broadcast, and is organized in a set of streams, each representing the different contributions to the DAB multiplex. The control part, STI-C(LI), defines the syntax and semantics of a set of messages, used to exchange control information between the upstream and downstream entities.

The middle layer, the **Transport Adaptation (TA)** layer, addresses the requirement of a safe transport of the STI-C(LI) messages and shall ensure that all messages are received error-free and in the order in which they were sent. Only one transport adaptation, STI-C(TA), is defined in the present document, and shall be used if STI-C(LI) is carried in any of the physical interfaces defined in the present document.

NOTE: This does not prevent the use of other transport adaptations, e.g. carrying STI-C(LI) separately using the TCP/IP protocol.

The lowest layer, **Physical Interface (PI)** layer, provides a set of physical manifestations of the logical interface, suitable for connection to various collection networks. The set of physical interfaces are referred to as STI(PI, X), where "X" signifies the particular type of network interface (e.g. V.11, G.703, G.704/1, etc.).

4.4 The implementation model of the STI

The STI is able to be used in various collection network topologies using a wide range of telecommunication links.

4.4.1 Examples of network topologies

There are many network topologies possible for using the STI. Two examples are provided below.

Figure 5 shows the selection of different types of physical interfaces to transport each part of the logical interface. The STI-D is adapted to a V.11 interface to allow transport in a PDH or a fixed telecommunication connection chosen for a fixed capacity given the type of service components carried by the STI-D frame. The STI-C is adapted to a V.24 interface and carried using a dial-up modem connection.

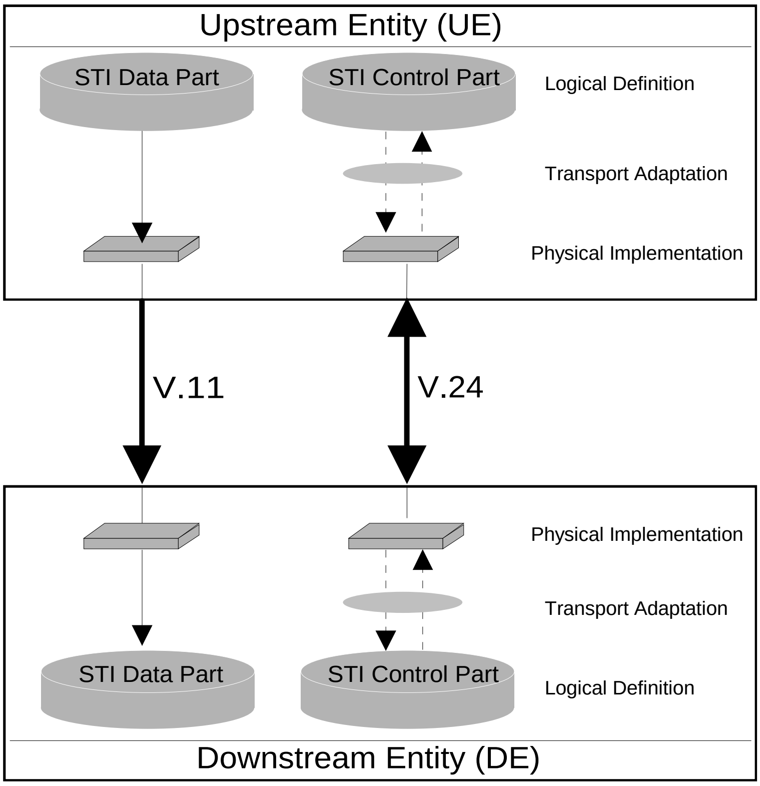

Figure 5: Example of STI connection with data and control parts carried in different types of physical interfaces

Figure 6 shows the selection of the same type of physical interface to transport each part of the logical interface. Two G.703 connections are used. One connection carries both the data and control parts in the downstream direction. The other connection carries the control part in the upstream direction.

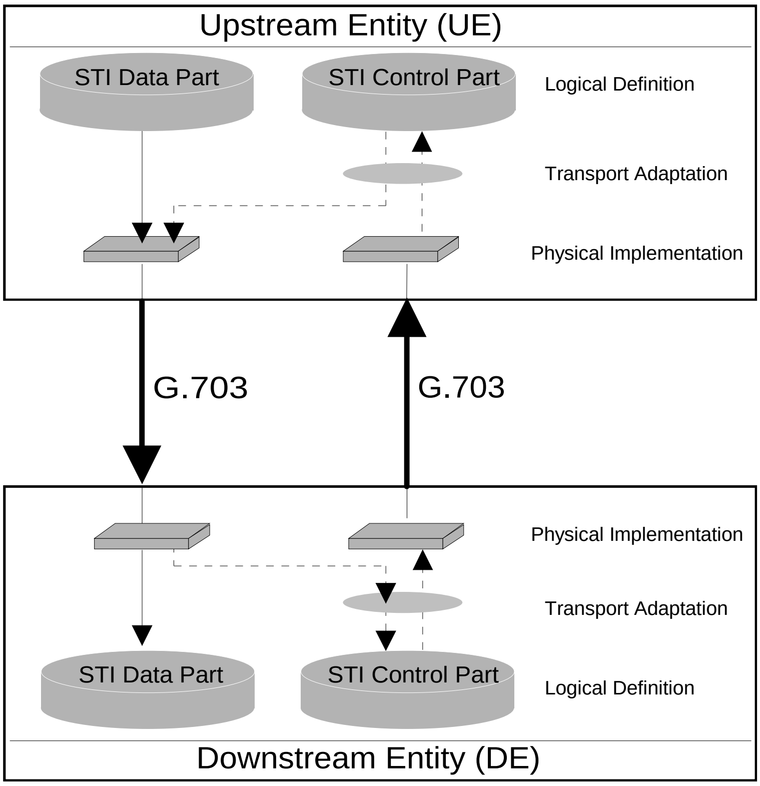

Figure 6: Example of STI connection with data and control parts carried in the same type of physical interface

4.4.2 Hierarchical collection networks

The STI is applicable equally to connections between a Service provider and an Ensemble provider and to connections between Service providers in a hierarchical collection network, as shown in figure 7.

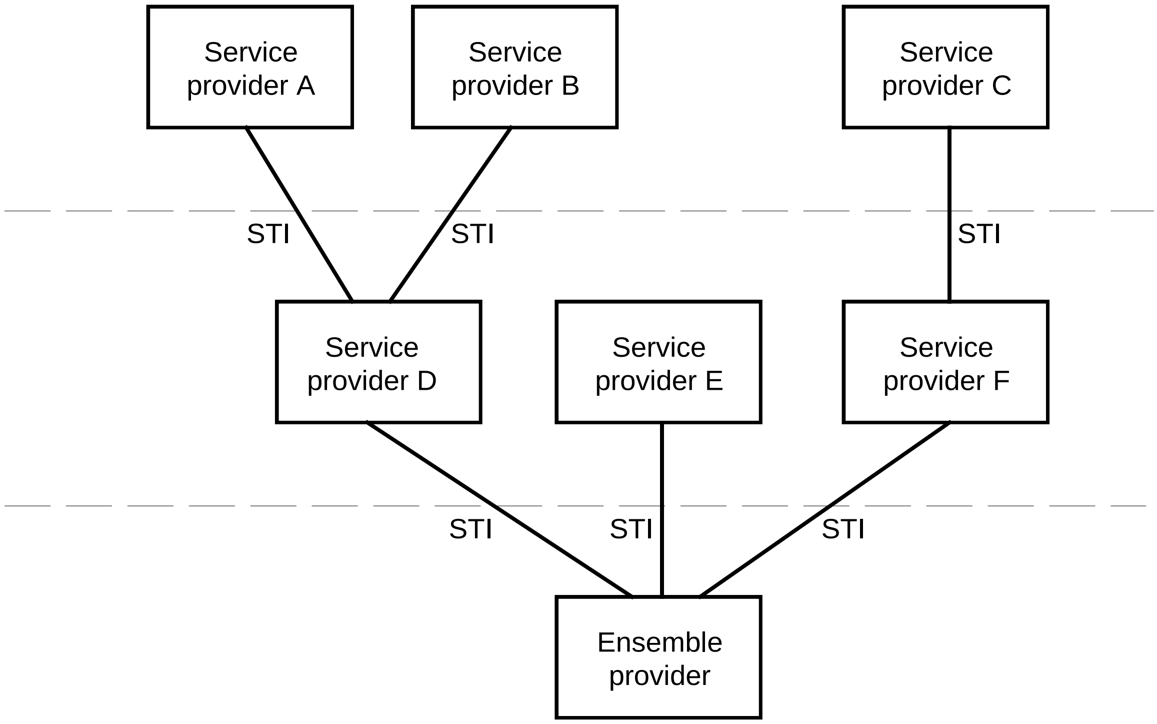

Figure 7: Example of hierarchical collection network

Service providers A and B send their service components, service information, etc., to Service provider D, who, as the downstream entity, acts in regard to them as an Ensemble provider. Service provider D is accordingly responsible for managing reconfiguration requests, etc., from the upstream Service providers.

Similarly Service provider C sends data to Service provider F.

Service providers D, E and F each provide a group of service components, service information, etc., to the Ensemble provider. The Ensemble provider is not aware of the existence of the Service providers A, B and C.

4.4.3 Multicasting

A Service provider may wish to send identical data contained in the STI-D(LI) to a number of different Ensemble providers (e.g. an audio service being broadcast in different regional DAB networks). Rather than having to install a separate connection (link) to each Ensemble provider, the STI-D(LI) can be distributed by satellite or other appropriate broadcast media. This is defined as multicasting of the STI.

STI multicasting means that one STI signal, carrying an STI-D part, is sent to all the Ensemble providers over the broadcast medium. The downstream STI-C flow may be carried within the same physical signal, in this case the Transport Adaptation layer provides the necessary addressing mechanism to ensure that the STI is still treated as a point-to-point connection at the Logical Interface layer.

NOTE: The use of multicasting of STI requires an implicit coordination between the upstream entity and the downstream entities and this can introduce a loss of flexibility in the use of the STI. As an example, a reconfiguration may require the change of an MSC sub-channel size. It may prove to be impossible to reconfigure if one of the downstream entities does not accept the reconfiguration request.

5 Logical definition of the STI Data Part, STI-D(LI)

The STI-D(LI) layer is the logical definition of the STI data part and shall be composed of logical frames. For MSC sub-channel streams, each logical frame shall contain the information needed to generate a 24 ms period of the component in the DAB multiplex. For FIC FIB streams using the synchronous insertion method, the content contained shall be related to that particular 24 ms period. For other components carried in the logical frame, the contents need not be related to the 24 ms period.

The STI-D(LI) logical frame shall consist of a status field and a data field as shown in figure 8. The status field shall give information about the quality of the collection network and may be changed by any physical layer of the STI. The data field shall carry information that is transparent to all physical layers of the STI and its content shall not be changed by other layers in an error free transmission.

Status field (STAT)	Data field (D-LIDATA)

Figure 8: The structure of an STI-D(LI) frame

5.1 General structure

The STI-D(LI) shall be composed of logical frames that consist of a variable number of bytes, where the length of each logical frame is defined by the Data Length field as defined in clause 5.3.3. The basic frame structure is illustrated in figure 9 and shall consist of:

- a status field of 1 byte, STAT, which shall consist of an 8-bit error status field ERR;
- a data field, D-LIDATA, which comprises:
    - a frame characterization field, FC, of 8 bytes;
    - a stream characterization field, STC, of variable size;
    - an end-of-header field, EOH, of 4 bytes;
    - a main stream data field, MST, of variable size;
    - an end of frame field, EOF, of 4 bytes;
    - a time stamp field, TIST, of 4 bytes.

$B_{0,p}(b_0)$ is the first bit of frame *p*

$B_{STAT}(b_0..b_7)$	**STAT** 1 byte	**ERR** 8 bits		
$B_{0,p}(b_0)..B_{1,p}(b_7)$	**D-LIDATA** DL bytes	**FC** 8 bytes	**SPID** 16 bits	
$B_{2,p}(b_0..b_2)$			**rfa** 3 bits	
$B_{2,p}(b_3)..B_{3,p}(b_7)$			**DL** 13 bits	
$B_{4,p}(b_0..b_7)$			**rfa** 8 bits	
$B_{5,p}(b_0..b_7)$			**DFCT** 13 bits	**DFCTL** 8 bits
$B_{6,p}(b_0..b_4)$				**DFCTH** 5 bits
$B_{6,p}(b_5)..B_{7,p}(b_7)$			**NST** 11 bits	
$B_{8,p}\cdots$		**STC** $4 \times$ NST bytes	**$ISTC_1$** 4 bytes	**TID** 3 bits
				STL 13 bits
				TIDext 3 bits
				CRCSTF 1 bit
				STID 12 bits
			: :	
			$ISTC_{NST}$ 4 bytes	**TID** 3 bits
				STL 13 bits
				TIDext 3 bits
				CRCSTF 1 bit
$..B_{4xNST+7,p}$				**STID** 12 bits
$B_{4xNST+8,p}\cdot\cdot$		**EOH** 4 bytes	**rfa** 16 bits	
$B_{4xNST+10,p}\cdot\cdot$			**CRCH** 16 bits	
$B_{4xNST+12,p}\cdot\cdot$		**MST** m bytes	**$ISTD_1$** $STL_1 - 2 \times CRCSTF_1$ bytes	
			$CRCST_1$ $2 \times CRCSTF_1$ bytes (optional, only present when $CRCSTF_1$ is set to 1)	
		$m = STL_1 + STL_2 + .. + STL_{NST}$	: :	
			$ISTD_{NST}$ $STL_{NST} - 2 \times CRCSTF_{NST}$ bytes	
$...B_{4xNST+m+11,p}$			**$CRCST_{NST}$** $2 \times CRCSTF_{NST}$ bytes (optional, only present when $CRCSTF_{NST}$ is set to 1)	

$B_{4xNST+m+12,p}$ ..			**EOF** 4 bytes	**rfa** 32 bits
$B_{4xNST+m+16,p}$ .. ..$B_{4xNST+m+19,p}$			**TIST** 4 bytes	$B_{4xNST+m+19,p}(b_7)$ is the last bit of frame *p*

The shaded fields indicate data which will be radiated as part of the DAB Ensemble
The un-shaded fields are STI-D(LI) signalling data.

Figure 9: The structure of an STI-D(LI) 24 ms logical frame

5.2 Error field (ERR)

The ERR field may convey information about the error status of the data in the same STI-D(LI) frame. This field may be filled by error information derived from the PI layer. Four levels of errors shall be defined as given in table 1.

Table 1: Definition of error status

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	Error status
1	1	1	1	1	1	1	1	Error level 0
1	1	1	1	0	0	0	0	Error level 1
0	0	0	0	1	1	1	1	Error level 2
0	0	0	0	0	0	0	0	Error level 3

The definition of the Error Levels will depend on the specific implementation but the following general guidelines should be used.

Error level 0: D-LIDATA contains no detected errors (which shall be the default state, set at the origin of D-LIDATA).

Error level 1: D-LIDATA contains errors which can be ignored by the processing equipment.

Error level 2: D-LIDATA contains errors which should not be ignored by the processing equipment but which may be mitigated by additional processing within that equipment (for example in the case of streams of constant length the header information from previous frames could be used).

Error level 3: D-LIDATA is not valid and should not be processed.

5.3 Frame characterization field (FC)

The FC field shall carry the common data characteristics that apply to the whole D-LIDATA field.

The FC for frame *p* shall be carried in bytes, $B_{[0...7],p}$. It shall contain a SPID field identifying the origin of the data part, a DL field indicating the total length of the STI-D(LI) frame, a DFCT field and a NST field that indicates the number of individual data streams carried in the MST field.

5.3.1 Service provider identifier field (SPID)

The SPID field shall carry a 16-bit number that shall be used by the downstream entity to identify the source of the STI-D(LI) logical frame, i.e. the upstream entity. The SPID shall be unique within the collection network connected to the downstream entity.

The SPID shall be carried in $B_{0,p}(b_0...b_7)$ and $B_{1,p}(b_0...b_7)$.

5.3.2 Reserved bits

Bits $B_{2,p}(b_0...b_2)$ shall be reserved for future addition. They shall be set to zero until they are defined.

5.3.3 Data length field (DL)

The DL field shall indicate the total length of the D-LIDATA field. The length shall be indicated by a 13-bit number which shall give the total number of bytes carried in the D-LIDATA field as follows:

For NST = 0, $DL = 20$

For NST ≠ 0, $DL = 20 + NST \times 4 + \sum_{Strm=1}^{NST} STL_{Strm}$

where:

- *NST* shall be the number of streams carried in the main stream data field (see clause 5.3.6);
- *Strm* shall be the ordinal number of the stream in the MST field;
- STL_{Strm} shall be the stream length of the stream with the ordinal number *Strm* (see clause 5.4.1.2).

The DL for frame *p* shall be carried in $B_{2,p}(b_3...b_7)$ to $B_{3,p}(b_0...b_7)$.

5.3.4 Reserved bits

Bits $B_{4,p}(b_0...b_7)$ shall be reserved for future addition. They shall be set to zero until they are defined.

5.3.5 Data frame count field (DFCT)

The DFCT is a modulo-5 000 counter and shall be arranged in two parts and shall be incremented by one every frame. The higher part, DFCTH, shall be a modulo-20 counter (0 to 19) and the lower part, DFCTL, shall be a modulo-250 counter (0 to 249).

The DFCTL for frame *p* shall be carried in $B_{5,p}(b_0...b_7)$.

The DFCTH for frame *p* shall be carried in $B_{6,p}(b_0...b_4)$.

5.3.6 Number of streams field (NST)

The NST field shall give the NST carried in the MST field of the STI-D(LI) frame. The number shall be carried as an 11-bit number in $B_{6,p}(b_5...b_7)$ and $B_{7,p}(b_0...b_7)$.

It shall be possible for NST to have the value zero, which shall indicate that there are no streams present in the MST field.

The NST field in the STI-D(LI) should not change except at reconfiguration, see annex E.

5.4 Stream characterization field (STC)

The STC field shall provide information that characterizes each stream carried in the MST field. Provided that NST ≠ 0, bytes $B_{[8...(4 \times NST+7)],p}$ shall carry this information.

5.4.1 Individual stream characterization field ($ISTC_{Strm}$)

Each individual stream, $ISTD_{Strm}$, shall be characterized by its own $ISTC_{Strm}$ field, where *Strm* shall be the ordinal number of the ISTC field carried in the STC field. *Strm* shall take the values 1...NST. Four bytes shall be used to describe each stream, giving a 3-bit type identifier, a 13-bit number for the length of the associated stream carried in the MST field, a 3-bit type identifier extension, one bit used to indicate if a CRC is present after the data stream and a 12-bit stream identifier.

With the exception of the STL, which may vary, the fields of the $ISTC_{Strm}$ field shall be constant from frame to frame.

5.4.1.1 Type identifier field (TID)

The TID field shall give information about the type of data carried in the MST field with ordinal number *Strm*. The TID shall be a 3-bit number and carried in $B_{4+4\times Strm,p}(b_0...b_2)$. The TID is coded as shown in table 2.

Table 2: Coding of TID

TID	b_0	b_1	b_2	Type of data stream
0	0	0	0	MSC sub-channel
1	0	0	1	rfa
2	0	1	0	rfa
3	0	1	1	MSC sub-channel contribution
4	1	0	0	FIC FIG stream
5	1	0	1	FIC FIB stream
6	1	1	0	rfa
7	1	1	1	In-house data (user definable)

5.4.1.2 Stream length field (STL)

The STL field shall give the length in bytes of the stream carried in the MST field with ordinal number *Strm*, $ISTD_{Strm}$. The length shall include the CRCST field, if present. The STL shall be a 13-bit number and carried in $B_{4+4\times Strm,p}(b_3...b_7)$ and $B_{5+4\times Strm,p}(b_0...b_7)$. Reference to an empty $ISTD_{Strm}$ shall be permitted, in which case STL_{Strm} shall have the value 0.

STL shall have values in the range 0 to 8 191.

The length of a $ISTD_{Strm}$ field may vary between two consecutive STI-D(LI) frames, but as long as the $ISTD_{Strm}$ stream is in use at the downstream entity (e.g. the ISTD is being used in the current DAB multiplex), the stream shall logically remain open (i.e. the STL_{Strm} shall be set to zero if no data is carried in the $ISTD_{Strm}$).

5.4.1.3 Type identifier extension field (TIDext)

The TIDext shall give additional information relevant to the type of data carried in the MST field with ordinal number *Strm*. The TIDext shall be 3-bit number and carried in $B_{6+4\times Strm,p}(b_0...b_2)$ and the interpretation of this field shall depend on the value of the TID field.

The interpretation of TIDext for different TIDs shall be as given in table 3.

Table 3: Coding of TIDext

TID	TIDext	b_0	b_1	b_2	Significance
0	0	0	0	0	MSC audio stream
	1	0	0	1	MSC data stream
	2	0	1	0	MSC packet mode stream
	3	0	1	1	rfa
	4...7	1	x	x	rfa
1...2	0...7	x	x	x	rfa
3	0	0	0	0	MSC packet mode contribution
	1	0	0	1	rfa
	2...3	0	1	x	rfa
	4...7	1	x	x	rfa
4	0	0	0	0	Service Information
	1	0	0	1	FIDC
	2	0	1	0	Conditional Access information
	3	0	1	1	rfa
	4...7	1	x	x	rfa
5	0	0	0	0	Asynchronous insertion mode
	1	0	0	1	Synchronous insertion mode
	2...3	0	1	x	rfa
	4...7	1	x	x	rfa
6...7	0...7	x	x	x	rfa

5.4.1.4 Stream cyclic redundancy checksum flag field (CRCSTF)

The CRCSTF field shall be a one-bit field that indicates the presence of a CRCST field directly after the associated single stream data field in the MST field. The CRCSTF field shall be carried in $B_{6+4\times Strm,p}(b_3)$ and its coding shall be as given in table 4.

Table 4: Coding of CRCSTF

CRCSTF	Coding
0	No CRCST present
1	CRCST present

5.4.1.5 Stream identifier field (STID)

A STID shall be chosen for each stream carried in the MST field and it shall be unique for a specific SPID. The STID field shall be a 12-bit field carried in $B_{6+4\times Strm,p}(b_4...b_7)$ and $B_{7+4\times Strm,p}(b_0...b_7)$.

The STID shall not be changed as long as the $ISTD_{Strm}$ stream is used in the current configuration.

5.5 End-of-header field (EOH)

The end-of-header field shall consist of four bytes. Two bytes shall be reserved for future addition and two bytes shall carry a cyclic redundancy checksum over the header data fields.

5.5.1 Reserved bytes

Bytes $B_{4\times NST+8,p}$ and $B_{4\times NST+9,p}$ shall be reserved for future addition. They shall be set to zero until they are defined.

5.5.2 Header cyclic redundancy checksum field (CRCH)

A cyclic redundancy checksum shall be carried out on the contents of the FC, the STC field and the two reserved bytes of the EOH field, i.e. bytes $B_{[0...(4\times NST+9)],p}$. The CRCH shall be carried in bytes $B_{4\times NST+10,p}$ and $B_{4\times NST+11,p}$.

The CRC calculation shall use the method given in annex A.

5.6 Main stream data field (MST)

The MST field shall carry all ISTD fields as defined by the ISTC fields in the STC field. For NST ≠ 0, MST shall occupy bytes $B_{4\times NST+12,p}$ to $B_{m+4\times NST+11,p}$, where m shall give the total length of the MST field in bytes and shall be calculated as follows:

For NST ≠ 0, $m = \sum_{Strm=1}^{NST} STL_{Strm}$

5.6.1 Individual stream data field ($ISTD_{Strm}$)

The $ISTD_{Strm}$ field shall carry the data stream as signalled in the corresponding $ISTC_{Strm}$ field. Further information about the data streams is given in clause 5.9. Each ISTD field, $ISTD_{Strm}$, shall be carried in $B_{[k...v],p}$. The value of *k* and *v* shall be calculated as follows:

For $Strm = 1$, $k = NST \times 4 + 12$

For $Strm > 1$, $k = NST \times 4 + 12 + \sum_{i=1}^{Strm-1} STL_i$

$v = k + STL_{Strm} - 1 - 2 \times CRCSTF_{Strm}$

5.6.2 Stream cyclic redundancy checksum field ($CRCST_{Strm}$)

When the $CRCSTF_{Strm}$ field is set to 1 in the $ISTC_{Strm}$ field, bytes $B_{v+1,p}$ and $B_{v+2,p}$ shall carry a cyclic redundancy checksum carried out on the contents of $ISTD_{Strm}$, $B_{[k...v],p}$. The value of *k* and *v* are given in clause 5.6.1.

The CRC calculation shall use the method given in annex A.

5.7 End-of-frame field (EOF)

The EOF field shall consist of four bytes carried in bytes $B_{m+4\times NST+12,p}$ to $B_{m+4\times NST+15,p}$. These bytes shall be reserved for future addition and shall be set to zero until they are defined. The value of *m* shall be as defined in clause 5.6.

5.8 STI-D(LI) time stamp field (TIST)

The four bytes $B_{m+4\times NST+16,p}$ to $B_{m+4\times NST+19,p}$ shall be used for an STI-D(LI) time stamp. The value of *m* is defined in clause 5.6. The time stamp coding and bit allocation shall be as given in annex B.

5.9 Details of the individual streams carried in the MST

5.9.1 MSC sub-channel streams

5.9.1.1 MSC audio stream

For TID in $ISTC_{Strm}$ equals 0 and TIDext equals 0, $ISTD_{Strm}$ shall carry an encoded audio programme (as defined in EN 300 401 [1]) that shall be inserted as an MSC stream mode audio sub-channel.

In the case of an audio encoded bitstream sampled with 48 kHz frequency, the complete audio frame shall be inserted in frame *p*.

An audio encoded bitstream sampled with 24 kHz frequency has a Frame Length (FL) of 48 ms. When carried in $ISTD_{Strm}$, each frame shall be split in two halves. The first half shall be carried in frame *p* and the second half in frame *p + 1*, (i.e. the following STI-D(LI) logical frame).

5.9.1.2 MSC data stream

For TID in $ISTC_{Strm}$ equals 0 and TIDext equals 1, $ISTD_{Strm}$ shall carry data to be inserted as an MSC stream mode data sub-channel.

5.9.1.3 MSC packet mode stream

For TID in $ISTC_{Strm}$ equals 0 and TIDext equals 2, $ISTD_{Strm}$ shall carry data to be inserted as a complete MSC packet mode sub-channel.

5.9.2 MSC sub-channel contributions

5.9.2.1 MSC packet mode data contributions

For TID in $ISTC_{Strm}$ equals 3 and TIDext equals 0, $ISTD_{Strm}$ shall carry data to be inserted asynchronously as a packet mode service component in a MSC sub-channel which may carry other packet data components as well. The packets shall be coded as defined in EN 300 401 [1]. $ISTD_{Strm}$ shall carry zero or more packets per frame and all packets carried in the stream shall have the same packet address.

NOTE: The co-ordination of capacity issues between the upstream entity and the downstream entity should be handled by the STI-C(LI), see clause 6.

5.9.3 FIC FIG stream

For TID in $ISTC_{Strm}$ equals 4, $ISTD_{Strm}$ shall carry FIGs to be inserted in the FIC and/or in the Auxiliary Information Channel (AIC) of a DAB multiplex. The TIDext field is used to indicate the type of information that the FIC FIG data stream carries. The TIDext field shall not change from frame to frame. The FIC FIG data stream provides the mechanism to carry SI, FIDC and CA information by using the format of a FIG. Zero or more FIGs shall be present in the $ISTD_{Strm}$ field and each FIG shall be coded according to EN 300 401 [1].

Multiplex Configuration Information (MCI), defined in EN 300 401 [1] shall not be transported in STI-D(LI).

The length of a FIC FIG stream field may vary between two consecutive STI-D(LI) frames, depending on the number of and length of each FIG carried in each logical frame. As long as the FIC FIG stream is used in the current configuration, the stream shall logically remain open. If no FIG is carried in a logical frame the STL_{Strm} shall be set to zero.

An upstream entity that generates an STI-D(LI) FIC FIG stream shall provide FIGs at the desired FIG repetition rates and shall expect the downstream entity to transport the FIGs transparently.

NOTE: The co-ordination of capacity issues between the upstream entity and the downstream entity should be handled by the STI-C(LI), see clause 6.

5.9.4 FIC FIB stream

For TID in $ISTC_{Strm}$ equals 5, $ISTD_{Strm}$ shall carry FIBs to be inserted in the FIC. The TIDext in $ISTC_{Strm}$ shall be used to indicate which of the two insertion methods is in use. The $ISTD_{Strm}$ field shall contain between zero and three FIBs if the FIBs are to be broadcast using DAB transmission modes I, II, and IV, and between zero and four FIBs if the FIBs are to be broadcast using DAB transmission mode III. Each FIB shall be coded according to EN 300 401 [1].

When TIDext equals 0, the $ISTD_{Strm}$ shall carry FIBs that shall be inserted in the FIC asynchronously. The Ensemble provider may delay the insertion of the received FIBs until capacity is available in the FIC.

NOTE 1: When using the asynchronous insertion method, the co-ordination of capacity issues between the upstream entity and the downstream entity should be handled by the STI-C(LI), see clause 6.

When TIDext equals 1, the $ISTD_{Strm}$ shall carry FIBs that shall be inserted in the FIC synchronously. The Ensemble provider shall insert the received FIBs in the FIC of the logical frame agreed by the upstream entity and downstream entity by means of a FIB grid. Only one FIC FIB stream shall be present in the STI-D(LI).

NOTE 2: When using the synchronous insertion method, the co-ordination of FIB grids between the upstream entity and downstream entity should be handled by the STI-C(LI), see clause 6.

The length of a FIC FIB data stream field may vary between two consecutive STI-D(LI) frames, depending on the number of FIBs carried in each logical frame. As long as the FIC FIB data stream is used in the current configuration, the stream shall logically remain open. If no FIBs are carried in a logical frame the STL_{Strm} shall be set to zero.

5.9.5 In-house data

For TID equals 7, $ISTD_{Strm}$ shall carry user defined in-house data. The format of this data is not subject to standardization.

6 Logical definition of the STI Control Part STI-C(LI)

The STI-C(LI) layer is the logical definition of the STI control part. STI-C(LI) provides asynchronous, bi-directional communication between upstream entity and downstream entity.

Whereas STI-D(LI), defined in clause 5, provides the transport of DAB broadcast data from an upstream entity to a downstream entity, STI-C(LI) allows the activities of both entities to be managed and supervised.

6.1 General Structure

Figure 10 presents the syntactical structure of messages on the STI-C(LI) interface, figure 11 gives the formatting details.

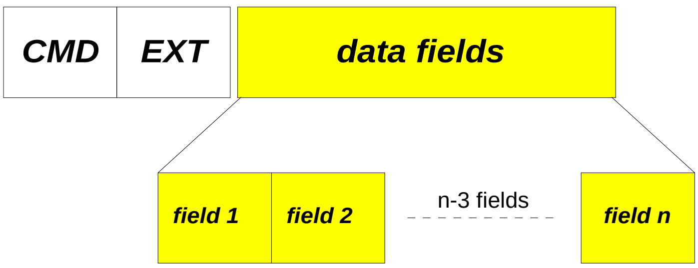

Figure 10: STI-C(LI) message

CMD, *EXT* and *data fields* shall carry the information of a STI-C(LI) message.

The message character set definition given in clause 3.6 shall be applied to the STI-C(LI) messages.

All messages shall contain a *CMD* and an *EXT* field. Depending on *CMD* and *EXT*, the *data fields* can be absent.

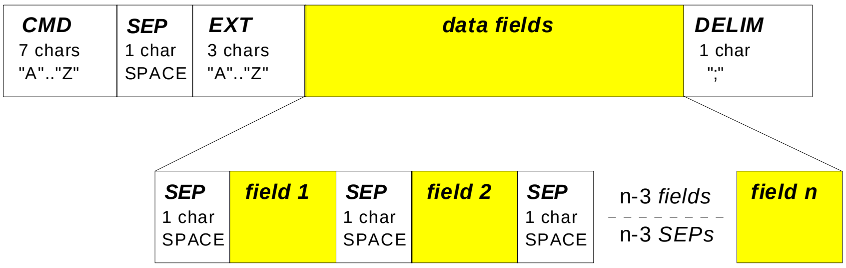

Figure 11: STI-C(LI) message format

Each STI-C(LI) message shall begin with a *CMD* field and shall end with a *DELIM* field.

The fields *CMD* and *EXT* govern structure and content of the *data fields* of the STI-C(LI) message.

Between fields one separator character *SEP* shall be inserted. *SEP* shall be the space character (bit combination 2/0).

NOTE: The clauses defining the different messages only give the contents of the *CMD*, *EXT* and *data fields*. Like figure 10, they refrain from mentioning the *SEP* and *DELIM* fields. Nevertheless, the rules given in this clause shall be applied, creating the format given in figure 11.

Table 5 presents the general definition of the STI-C(LI) message fields. Fields lengths are expressed as a number of characters.

Table 5: STI-C(LI) fields definition

Field Name	Meaning	Length	Field Type	Possible Value
CMD	command word	7	string	"A"..."Z" (upper case only)
EXT	command extension	3	string	"A"..."Z" (upper case only)
data fields	variable number of fields depending on the command	--	--	--
SEP	separator	1	char	" " space character (bit combination 2/0)
DELIM	delimiter	1	char	";" semicolon character (bit combination 3/11)

The following additional definitions shall be applied to the STI-C(LI) message fields:

CMD shall identify the command carried in the message.

EXT shall give additional information on the command carried in the message.

data fields these fields are governed by the type of message defined by the *CMD* and *EXT* fields. The number of fields is variable but the following definition shall be applied:
data fields shall not carry the space character (2/0) and the semicolon character (3/11).

6.2 Message handling

Each logical interface receiving STI-C(LI) messages shall analyse the command field (*CMD*) and shall process the message according to the command extension (*EXT*). It shall also handle error messages in relation to unrecognized commands, invalid formats, out of range errors, etc.

The STI-C(LI) provides bi-directional communication. Therefore upstream and downstream are not always appropriate in expressing the direction of the information flow. This text uses local entity and remote entity in cases where it is irrelevant whether an entity is the upstream or the downstream entity.

6.2.1 Data Exchange Sessions

Each STI-C(LI) message carries one unit of information that normally can be interpreted by the recipient in its own respect. Some information, however, is only meaningful in a context of several units of information. For the exchange of such information the STI-C(LI) provides data exchange sessions.

A data exchange session shall consist of a group of messages. The number of messages contained in the data exchange session shall either be fixed or shall be stated in the message that opens the session. Every session shall be explicitly closed after the data exchange session messages have been sent.

If errors are detected during, or immediately after, a data exchange session then these shall be indicated by the recipient by sending an error message. Sending an error message aborts the data exchange session, and rejects the content of the session or partial session. An aborted session is implicitly closed.

Only one data exchange session shall be open at a time.

STI-C(LI) defines the following types of data exchange sessions: configuration, FIG file, FIB grid, resource, configuration name and FIG file name.

Some STI-C(LI) messages shall only be used in data exchange sessions of a certain type.

6.3 STI-C(LI) message set

STI-C(LI) messages are identified by their command and command extension fields. The command field (*CMD*) and the command extension field (*EXT*) govern the method to produce and to analyse the data fields.

Table 6 presents the STI-C(LI) message set. The following categories are covered:

Action this category includes the messages used to manage the reconfiguration procedure. Different extension fields allow reconfigurations to be activated, cancelled, and monitored.

Configuration this category includes the messages used to exchange information on configurations between entities. They allow both entities to open and to close a data exchange session during which all the information related to one configuration can be read or defined.
A data exchange session consisting of a number of configuration messages shall not be disturbed by any action message.

FIG file this category includes messages to handle FIC information that is to be broadcast in a cyclic manner. FIG files can be exchanged between upstream entity and downstream entity. Selection and deselection of FIG files for broadcast is provided.

FIB grid this category includes messages to co-ordinate the delivery and insertion of Fast Information Blocks (FIBs) using the synchronous insertion method. A FIB grid provides a mechanism for Service Providers to determine in which transmission frame and at which position a FIB shall be broadcast. FIB grid messages provide exchange and management of FIB grids.

Resource this category includes messages to monitor scarce resources that should be shared by all Service Providers.

Information this category includes the messages used to co-ordinate the activity between entities by the exchange of general information.

Supervision this category includes messages used to indicate error and alarm states.

In general, messages can be issued by both upstream and downstream entities. However, some messages shall be issued only by the upstream entity (UE), and some shall be issued only by the downstream entity (DE). The columns UE and DE in table 6 indicate if messages may be issued by the corresponding entity (Y for yes, N for no).

Table 6: STI-C(LI) set of messages

Message Class	Message Name (CMD)	Extension (EXT)	UE	DE	Usage
ACTION	RCONFIG	REQ	Y	N	to request a reconfiguration
		DEF	N	Y	to define a reconfiguration and its time instant
		INF	Y	N	to request details about the next reconfiguration
		CAN	Y	N	to request cancellation of a pending reconfiguration
		ACK	N	Y	to acknowledge a cancellation request
		ERR	N	Y	to indicate an error
CONFIG	CONFDEF	INF	Y	Y	to request the definition of a configuration
		DEF	Y	Y	to open a configuration data exchange session
		END	Y	Y	to close a configuration data exchange session
		DEL	Y	Y	to delete a configuration
		ERR	Y	Y	to indicate an error (aborts session)
	SUBCHAN	DEF	Y	Y	to define a sub-channel as forming part of a configuration
	USESTRM	DEF	Y	Y	to define a data stream as forming part of a configuration
	CMPNENT	DEF	Y	Y	to define the a component as forming part of a configuration
	SERVICE	DEF	Y	Y	to define the composition of a service
	USEFIGF	DEF	Y	Y	to select a FIG file as forming part of a configuration
FIGFILE	FIGFILE	INF	Y	Y	to request the content of a FIG file
		DEF	Y	Y	to open a FIG file data exchange session
		REC	Y	Y	to transfer one FIG
		END	Y	Y	to close a FIG file data exchange session
		DEL	Y	Y	to delete a FIG file
		SEL	Y	N	to select a predefined FIG file for broadcast
		DES	Y	N	to deselect a previously selected FIG file
		ERR	Y	Y	to indicate an error
FIBGRID	FIBGRID	INF	Y	N	to request the definition of a FIB grid file
		DEF	N	Y	to open a FIB grid data exchange session
		REC	N	Y	to define the value of a FIB grid record
		END	N	Y	to close a FIB grid data exchange session
		ACT	N	Y	to activate a FIB grid
		ERR	Y	Y	to indicate an error
RESOURCE	RESOURC	INF	Y	N	to request information about allocated resources
		DEF	N	Y	to open a resource data exchange session
		END	N	Y	to close a resource data exchange session
		ERR	Y	N	to indicate an error concerning a resource data exchange session
	CHANCAP	DEF	N	Y	to define the transmission channel capacities allocated
	STLIMIT	DEF	N	Y	to define limitations on stream usage
	IDALLOC	DEF	N	Y	to define identifier allocations
	IDLIMIT	DEF	N	Y	to define identifier allocation usage limitations
	PACKCON	DEF	N	Y	to define a packet contribution allocation
	FIGBLCK	DEF	N	Y	to define restrictions on the use of FIGs
	ANNSEND	DEF	N	Y	to define announcement parameters

Message Class	Message Name (CMD)	Extension (EXT)	UE	DE	Usage
INFORMATION	CONINFO	INF	Y	Y	to request the number of configurations (maximum and currently in use)
		DEF	Y	Y	to report the number of configurations (maximum and currently in use)
	CONNAME	INF	Y	Y	to request the names of stored configurations
		DEF	Y	Y	to open a configuration name data exchange session
		REC	Y	Y	to transfer a configuration name
		END	Y	Y	to close a configuration name data exchange session
		ERR	Y	Y	to indicate an error concerning a configuration name data exchange session
	FIGINFO	INF	Y	Y	to request the number of FIG files (maximum and currently in use)
		DEF	Y	Y	to report the number of FIG files (maximum and currently in use)
INFORMATION	FIGNAME	INF	Y	Y	to request the names of stored FIG files
		DEF	Y	Y	to open a FIG file name data exchange session
		REC	Y	Y	to transfer a FIG file name
		END	Y	Y	to close a FIG file name data exchange session
		ERR	Y	Y	to indicate an error concerning a FIG file name data exchange session
	COUNTER	INF	Y	N	to request the counter relation
		DEF	N	Y	to report the counter relation
SUPERVISION	PRERROR	GBG	Y	Y	to signal reception of a garbage message
		UKN	Y	Y	to signal reception of a unknown message
		SYN	Y	Y	to signal a message with bad syntax
		SEM	Y	Y	to signal a message with bad contents
		PRT	Y	Y	to signal an incorrect sequence of messages
	ALARMST	INF	Y	N	to request the current alarm status
		DEF	N	Y	to report the current alarm status
	STERROR	INF	Y	N	to request the current error status of each stream
		DEF	N	Y	to report the current error status of each stream

6.4 Action messages

The action messages shall be used to co-ordinate the reconfiguration procedure between upstream entity and downstream entity.

A reconfiguration shall be the change from one configuration to another configuration at a time instant previously exchanged between upstream and downstream entity. Typically a reconfiguration procedure on a STI connection results in a DAB multiplex reconfiguration. Although closely related, the terms reconfiguration and DAB multiplex reconfiguration are not interchangeable. Annex E gives further information on the relation between a reconfiguration and the DAB multiplex reconfiguration.

A reconfiguration shall take place by the activation of a named configuration at a certain time instant. Named configurations and their exchange between upstream and downstream entity are defined in clause 6.5. This clause describes how action messages are used to initiate, monitor or cancel a reconfiguration procedure.

The action messages are presented in table 7 and are defined in the following clauses.

Table 7: Action messages of the STI-C(LI)

ACTION MESSAGES				
CMD	***EXT***	***data fields***		
RCONFIG	REQ	*Name*	*UTC*	*DFCT*
RCONFIG	DEF	*Name*	*UTC*	*DFCT*
RCONFIG	INF			
RCONFIG	CAN			
RCONFIG	ACK			
RCONFIG	ERR	*Cause*		

6.4.1 General rules to use action messages

The upstream entity can request a reconfiguration by supplying the required configuration name and time instant. The downstream entity shall respond with either an acceptance or a rejection.

The upstream entity may subsequently ask to cancel the reconfiguration. The downstream entity shall respond with either an acceptance or a rejection.

The downstream entity can also initiate a reconfiguration. The upstream entity cannot reject a reconfiguration initiated by the downstream entity.

NOTE: It is expected that either the upstream entity or the downstream entity (but not both) shall control initiation of reconfigurations over a particular STI connection.

From the time that the downstream entity initiated or accepted a reconfiguration, this reconfiguration is pending. It remains pending until the reconfiguration procedure takes place or until the reconfiguration is cancelled.

Only one reconfiguration procedure per upstream entity shall be pending at a time.

The upstream entity may request details about the pending reconfiguration. The downstream entity shall respond with either the configuration name and time instant if a reconfiguration is pending, or an error message if no reconfiguration is pending.

The *Name* fields of RCONFIG messages shall not carry the reserved values "CURRENT" and "NEXT".

6.4.2 RCONFIG messages

The RCONFIG messages allow the reconfiguration procedure to be managed. A reconfiguration can be engaged by either entity.

RCONFIG REQ shall be issued by the upstream entity to request a reconfiguration. The downstream entity shall respond with either an acceptance by issuing the RCONFIG DEF message stating the exact reconfiguration time, or a rejection by issuing an RCONFIG ERR message.

RCONFIG DEF shall be issued by the downstream entity to define the exact timing of a reconfiguration. The reconfiguration is pending from the time this message is issued.

RCONFIG INF shall be issued by the upstream entity to request information about a pending reconfiguration. The downstream entity shall respond with an RCONFIG DEF message if a reconfiguration is pending, or with an RCONFIG ERR message if no reconfiguration is pending.

RCONFIG CAN shall be issued by the upstream entity to request cancellation of a pending reconfiguration. The downstream entity shall respond with either an acceptance by issuing the RCONFIG ACK message or a rejection by issuing the RCONFIG ERR message.

RCONFIG ERR shall be issued by the downstream entity to indicate errors. This message is used to reject a request for a reconfiguration, or to reject a request for a cancellation, or to indicate that no reconfiguration is pending in response to a request for information.

RCONFIG ACK shall be issued by the downstream entity to accept a cancellation request.

6.4.2.1 RCONFIG REQ

Only the upstream entity shall issue the RCONFIG REQ message.

RCONFIG REQ shall be issued by the upstream entity to request a reconfiguration.

The downstream entity shall either accept the reconfiguration by issuing the RCONFIG DEF message or reject it by issuing an RCONFIG ERR message.

The reconfiguration shall be pending from the time that the downstream entity accepts the request and issues a RCONFIG DEF message. The upstream entity should ensure that there is sufficient time for the transport of both request and response, such that if a rejection is received the upstream entity has not already begun to reconfigure.

The downstream entity may reject the request for a number of reasons. The RCONFIG ERR message shall be issued to state the cause of the rejection.

Figure 12 presents the RCONFIG REQ message structure.

RCONFIG	**REQ**	*Name*	*UTC*	*DFCT*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*	*(field 3)*

Figure 12: RCONFIG REQ message structure

Table 8 presents the definition of the RCONFIG REQ message fields. Fields length is expressed as a number of characters.

Table 8: RCONFIG REQ message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Name	Name of the configuration	[1...16]	string	A string giving the name of an existing configuration
UTC	Reference to a two-minute window for which the reconfiguration is requested	8	string	"HH:MM:SS" "HH" in "00"..."23" for hours, "MM" in "00"..."59" for minutes, "SS" in "00"..."59" for seconds,
DFCT	Data part frame count when reconfiguration is requested	6	string	"UU:LLL" "UU" in "00"..."19" for upper part, (DFCTH) "LLL" in "000"..."249" for lower part, (DFCTL)
NOTE: The range of DFCT provides a two-minute window with a frame accurate timing reference. UTC allows the reconfiguration to be requested up to 24 hours in advance by providing a pointer to the appropriate two-minute window. Annex E gives details about the interpretation of the *UTC* and *DFCT* values.				

6.4.2.2 RCONFIG DEF

Only the downstream entity shall issue the RCONFIG DEF message.

RCONFIG DEF shall be issued by the downstream entity to define the exact instant of a reconfiguration. It shall normally be issued to accept a request from the upstream entity, but in some circumstances it may be issued by the downstream entity to enforce a reconfiguration on the upstream entity.

The issue of a RCONFIG DEF message shall be the moment from which a reconfiguration is pending.

On request of the upstream entity issuing the RCONFIG REQ message, the downstream entity shall issue RCONFIG DEF if the reconfiguration is accepted, or RCONFIG ERR if the reconfiguration is rejected.

Figure 13 presents the RCONFIG DEF message structure.

RCONFIG	DEF	*Name*	*UTC*	*DFCT*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*	*(field 3)*

Figure 13: RCONFIG DEF message structure

Table 9 presents the definition of the RCONFIG DEF message fields. Fields length is expressed as a number of characters.

Table 9: RCONFIG DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Name	Name of the configuration	[1...16]	string	A string giving the name of an existing configuration
UTC	Reference to a two-minute window in which to perform the reconfiguration.	8	string	"HH:MM:SS" "HH" in "00"..."23" for hours, "MM" in "00"..."59" for minutes, "SS" in "00"..."59" for seconds,
DFCT	Data part frame count when reconfiguration shall be performed	6	string	"UU:LLL" "UU" in "00"..."19" for upper part, (DFCTH) "LLL" in "000"..."249" for lower part, (DFCTL)
NOTE: The range of DFCT provides a two-minute window with a frame accurate timing reference. UTC allows the reconfiguration to be requested up to 24 hours in advance by providing a pointer to the appropriate two-minute window. Annex E gives details about the interpretation of the *UTC* and *DFCT* values.				

6.4.2.3 RCONFIG INF

Only the upstream entity shall issue the RCONFIG INF message.

RCONFIG INF shall be issued to request details about the next reconfiguration affecting the upstream entity.

The downstream entity will respond with a RCONFIG DEF message if a reconfiguration is pending or a RCONFIG ERR if there is no reconfiguration pending.

Figure 14 presents the RCONFIG INF message structure.

RCONFIG	INF
(CMD)	*(EXT)*

Figure 14: RCONFIG INF message structure

6.4.2.4 RCONFIG CAN

Only the upstream entity shall issue the RCONFIG CAN message.

RCONFIG CAN shall be issued to request cancellation of a pending reconfiguration.

The downstream entity shall respond with a RCONFIG ACK message to accept the cancellation or with a RCONFIG ERR message to reject the cancellation or to indicate that no reconfiguration is pending.

The reconfiguration is cancelled from the time that the downstream entity accepts the request and issues a RCONFIG ACK message. The upstream entity should ensure that there is sufficient time for the transport of both request and response, such that if a rejection is received the upstream entity is able to follow the configuration change.

The downstream entity may reject the request for a number of reasons. The RCONFIG ERR message shall be issued to state the cause of the rejection.

Figure 15 presents the RCONFIG CAN message structure.

RCONFIG	**CAN**
(CMD)	*(EXT)*

Figure 15: RCONFIG CAN message structure

6.4.2.5 RCONFIG ACK

Only the downstream entity shall issue the RCONFIG ACK message.

The RCONFIG ACK message shall be issued to acknowledge a RCONFIG CAN message.

The RCONFIG ACK message shall be issued by the downstream entity immediately that the reconfiguration procedure is cancelled.

Figure 16 presents the RCONFIG ACK message structure.

RCONFIG	**ACK**
(CMD)	*(EXT)*

Figure 16: RCONFIG ACK message structure

6.4.2.6 RCONFIG ERR

Only the downstream entity shall issue the RCONFIG ERR message.

The RCONFIG ERR message is issued to indicate an error. It is issued to reject a request for a reconfiguration, to reject a request for a cancellation, or to reject a request for information.

Figure 17 presents the RCONFIG ERR message structure.

RCONFIG	**ERR**	*Cause*
(CMD)	*(EXT)*	*(field 1)*

Figure 17: RCONFIG ERR message structure

Table 10 presents the definition of the RCONFIG ERR message fields. Fields length is expressed as a number of characters.

Table 10: RCONFIG ERR message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Cause	the reason for the rejection	4	string	for error reactions on RECONFIG REQ: "NAME" the specified configuration name is not recognized "INST" the specified reconfiguration instant is not possible "PEND" a reconfiguration is already pending "NRES" a resource referenced in the desired configuration is not available for error reactions on RCONFIG INF: "NONE" there is no reconfiguration pending for error reactions on RCONFIG CAN: "NONE" there is no reconfiguration pending "LATE" the cancellation request arrived too late to be accepted

6.5 Configuration messages

The configuration messages shall be used to exchange configurations between upstream entity and downstream entity.

A configuration shall describe services, service components and related information. The Ensemble provider uses this information to build the Multiplex Configuration Information as defined in EN 300 401 [1]. Each configuration shall be given a unique configuration name.

This clause describes how configuration messages shall be used to create, monitor and delete configurations.

The configuration messages are presented in table 11 and are defined in the following clauses.

Table 11: Configuration messages of the STI-C(LI)

CONFIGURATION MESSAGES										
CMD	**EXT**	**data fields**								
CONFDEF	INF	*Name*								
CONFDEF	DEF	*Name*	*NChan*	*NStrm*	*NComp*	*NServ*	*NFigf*			
CONFDEF	END									
CONFDEF	DEL	*Name*								
CONFDEF	ERR	*Name*	*Cause*							
SUBCHAN	DEF	*Sub-ChId*	L	*Option*	*Pro-tection*	*Size*				
			S	*Table*	*Index*					
USESTRM	DEF	*STID*	SUB	*SubChId*						
			FIG	*BPSmax*	*BPSave*					
			FIB	*BPSmax*	*BPSave*					
			PMC	*BPSmax*	*BPSave*	*PLen*	*PType*	*PLevel*		
CMPNENT	DEF	*CmpId*	A	*STID*	*SubChId*	*ASCTy*		*SCCA*		
			S	*STID*	*SubChId*	*DSCTy*	*ExtDSCTy*	*SCCA*		
			P	*STID*	*SubChId*	*DSCTy*	*ExtDSCTy*	*SCCA*	*SCId*	*PA*
			F	*STID*	*FIDCId*	*DSCTy*	*ExtDSCTy*	*SCCA*		
			E	*SPID*	*ExtCmpId*					
SERVICE	DEF	*SId*	*Local*	*CAId*	*NComp*	*Comp 1*	*...*	*Comp n*		
USEFIGF	DEF	*Name*								

6.5.1 General rules to use configuration messages

The configuration messages shall be used by both the upstream entity and the downstream entity to exchange the details of configurations.

Each configuration shall be given a name that has a minimum length of one character and a maximum length of 16 characters. This name shall be used to uniquely reference a configuration.

There are two special names which are reserved and shall not be used to name a configuration. These are the names "CURRENT" and "NEXT". These names are used for monitoring purposes only.

A complete configuration definition may be large in terms of the number of characters needed to describe it. The STI uses a configuration data exchange session (see clause 6.2.1) to permit the definition of a configuration as a number of shorter messages.

Configurations may be created and deleted but cannot be modified.

Configuration definitions may be requested by both the upstream entity and the downstream entity.

To create a new configuration the definition shall be given using a name that is not currently defined. The new configuration shall only be accepted if it is error free. The first message sent shall be a CONFDEF DEF message, which opens a configuration data exchange session. The number of messages of each category that will follow to provide the complete definition shall be specified. The configuration shall then be described in terms of the sub-channels, streams, components, services, and FIG files that are required. The session shall be closed by issuing a CONFDEF END message.

To permit early detection of errors, the configuration shall be specified in a hierarchical way such that items shall be defined before they are referenced, as follows:

1) sub-channel definition;
2) stream definition;
3) component definition;
4) service definition;
5) FIG file use definition.

The data exchange session shall be aborted by the recipient if an error is detected. The reason for the error shall be given. If the session is aborted during the creation of a configuration then the creation shall be abandoned.

6.5.2 CONFDEF messages

CONFDEF messages shall be used to define configurations, delete configurations, request information about configurations and indicate errors.

The CONFDEF messages are presented in table 11 and are defined in the following clauses.

The CONFDEF INF message shall be issued to request that the remote entity shall open a configuration data exchange session to provide the definition of the named configuration.

The CONFDEF DEF message shall be issued to open a configuration data exchange session. Configuration data exchange sessions shall be used to exchange the contents of configurations when creating or monitoring named configurations.

The CONFDEF END message shall be issued by the originator of the session to close it.

The CONFDEF DEL message shall be issued to delete the named configuration on the remote entity.

The CONFDEF ERR message shall be issued to abort an open configuration data exchange session, or to indicate another error.

6.5.2.1 CONFDEF INF

The CONFDEF INF message shall be issued to request the definition of a named configuration.

The requested entity shall open a configuration data exchange session or shall reject the demand by issuing a CONFDEF ERR message.

Figure 18 presents the CONFDEF INF message structure.

CONFDEF	**INF**	*Name*
(CMD)	*(EXT)*	*(field 1)*

Figure 18: CONFDEF INF message structure

Table 12 presents the definition of the CONFDEF INF message fields. Fields length is expressed as a number of characters.

Table 12: CONFDEF INF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Name	Name of the configuration	[1...16]	string	A string giving the name of a configuration, or the reserved names "CURRENT" or "NEXT"

The name "CURRENT" requests that the description of the configuration now being broadcast is returned. The name "NEXT" requests that the description of the configuration scheduled to be broadcast after the pending reconfiguration is returned. If no reconfiguration is pending then an error shall be indicated.

6.5.2.2 CONFDEF DEF

CONFDEF DEF shall be issued to open a configuration data exchange session.

The data fields of the CONFDEF DEF messages shall be used to define the name and the number of messages required to completely describe the configuration.

Following the issue of the CONFDEF DEF message, the exact number of messages needed to describe the configuration and listed in the CONFDEF DEF data fields shall be issued. The session shall be closed by issuing the CONFDEF END message.

Figure 19 presents the CONFDEF DEF message structure.

CONFDEF	**DEF**	*Name*	*NChan*	*NStrm*	*NComp*	*NServ*	*NFigf*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*	*(field 3)*	*(field 4)*	*(field 5)*	*(field 6)*

Figure 19: CONFDEF DEF message structure

Table 13 presents the definition of the CONFDEF DEF message fields. Fields length is expressed as a number of characters.

Table 13: CONFDEF DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Name	Name of the configuration	[1...16]	string	any string
NChan	Number of SUBCHAN definitions included in the session	[1...2]	dec	"0"..."64"
NStrm	Number of USESTRM definitions included in the session	[1...4]	dec	"0"..."2 047"
NComp	Number of CMPNENT definitions included in the session	[1...4]	dec	"0"..."9 999"
NServ	Number of SERVICE definitions included in the session	[1...4]	dec	"0"..."9 999"
NFigf	Number of USEFIGF definitions included in the session	[1...2]	dec	"0"..."64"

6.5.2.3 CONFDEF END

The CONFDEF END message shall be issued to close a configuration data exchange session by the entity that opened it.

This message shall be sent after all the messages announced by the CONFDEF DEF message have been issued.

Figure 20 presents the CONFDEF END message structure.

CONFDEF	**END**
(CMD)	*(EXT)*

Figure 20: CONFDEF END message structure

6.5.2.4 CONFDEF DEL

CONFDEF DEL shall be issued to delete a configuration stored at the remote entity by the creator of the configuration.

If the name of the configuration is not recognized, the deletion shall be rejected by issuing a CONFDEF ERR message.

Figure 21 presents the CONFDEF DEL message structure.

CONFDEF	**DEL**	*Name*
(CMD)	*(EXT)*	*(field 1)*

Figure 21: CONFDEF DEL message structure

Table 14 presents the definition of the CONFDEF DEL message fields. Fields length is expressed as a number of characters.

Table 14: CONFDEF DEL message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Name	Name of the configuration	[1...16]	string	The name of the configuration to be deleted

6.5.2.5 CONFDEF ERR

CONFDEF ERR shall be issued to indicate errors during the use of CONFDEF commands or errors concerning a configuration data exchange session. The issue of CONFDEF ERR shall abort an open configuration data exchange session.

The *Name* field in the CONFDEF ERR message shall identify the error source, the *Cause* field shall state error details

Figure 22 presents the CONFDEF ERR message structure.

CONFDEF	**ERR**	*Name*	*Cause*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*

Figure 22: CONFDEF ERR message structure

Table 15 presents the definition of the CONFDEF ERR message fields. Fields length is expressed as a number of characters.

Table 15: CONFDEF ERR message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Name	Name of the configuration	[1...16]	string	The name of the configuration for which the error is being reported.
Cause	Details of the error detected	[4...7]	string	for error reactions on CONFDEF INF or CONFDEF DEL: "UNKNOWN" the name is not known (including "NEXT" when no reconfiguration is pending) for errors concerning configuration data exchange sessions: "NOSPACE" all configuration stores have been used "USED" the name is already defined "NCHAN" the number of SUBCHAN DEFs is wrong "NSTRM" the number of USESTRM DEFs is wrong "NCOMP" the number of CMPNENT DEFs is wrong "NSERV" the number of SERVICE DEFs is wrong "NFIGF" the number of USEFIGF DEFs is wrong "USESTRM" there is an error in a USESTRM DEF "SUBCHAN" there is an error in a SUBCHAN DEF "CMPNENT" there is an error in a CMPNENT DEF "SERVICE" there is an error in a SERVICE DEF "USEFIGF" there is an error in a USEFIGF DEF

6.5.3 SUBCHAN messages

The SUBCHAN messages shall be used to define the sub-channels required by the service provider in the DAB multiplex.

SUBCHAN messages shall only be used during an open configuration data exchange session.

The SUBCHAN messages presented in table 11 are defined in the following clauses.

SUBCHAN DEF allows the originator of the data exchange session to describe the characteristics of a sub-channel which will be used in the configuration.

6.5.3.1 SUBCHAN DEF

SUBCHAN DEF shall be issued to define the characteristics of a sub-channel as part of a configuration.

The definition of a sub-channel shall be performed before the sub-channel is referenced in a component definition.

Figure 23 presents the SUBCHAN DEF message structure.

SUBCHAN	**DEF**	*SubChId*	**L** / **S**	*Option* / *Table*	*Protection* / *Index*	*Size*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*	*(field 3)*	*(field 4)*	*(field 5)*

Figure 23: SUBCHAN DEF message structure

Table 16 presents the definition of the SUBCHAN DEF message fields. Fields length is expressed as a number of characters.

The fields marked (*) contain parameters or terms defined in EN 300 401 [1]. Their values shall be compliant with those defined in EN 300 401 [1].

Table 16: SUBCHAN DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
SubChId	Sub-channel identifier*	[1...2]	dec	"0"..."63"
L	Long form	1	char	"L"
Option	Option*	1	dec	"0"..."7"
Protection	Protection level*	1	dec	"1"..."4"
Size	Sub-channel size*	[1...3]	dec	"4"..."864"
S	Short form	1	char	"S"
Table	Table switch*	1	dec	"0"..."1"
Index	Table index*	1	dec	"0"..."63"

6.5.4 USESTRM messages

The USESTRM messages shall be used to define STI-D(LI) FIG and FIB streams required in the DAB multiplex and components that are not referenced in the service configuration (e.g. a sub-channel carrying an audio announcement channel which is accessed only via announcement switching).

The USESTRM messages shall also be used to define packet mode contributions required in the DAB multiplex.

For FIG, asynchronous insertion method FIB and packet mode contributions, the USESTRM message shall define the requested maximum and average data rate in a one second time window. This information may be used by the downstream entity to manage the data received from the corresponding stream (e.g. to calculate the buffer size required).

For packet mode contributions the USESTRM message shall also indicate the required protection profile and packet length of the sub-channel that will carry the contribution.

USESTRM messages shall only be used during an open configuration data exchange session.

The USESTRM messages are presented in table 11 and are defined in the following clauses.

USESTRM DEF allows the originator of a configuration data exchange session to define a STI-D(LI) stream as a forming part of a configuration.

6.5.4.1 USESTRM DEF

USESTRM DEF shall be issued to define a data stream as forming a part of the configuration.

Figure 24 presents the USESTRM DEF message structure.

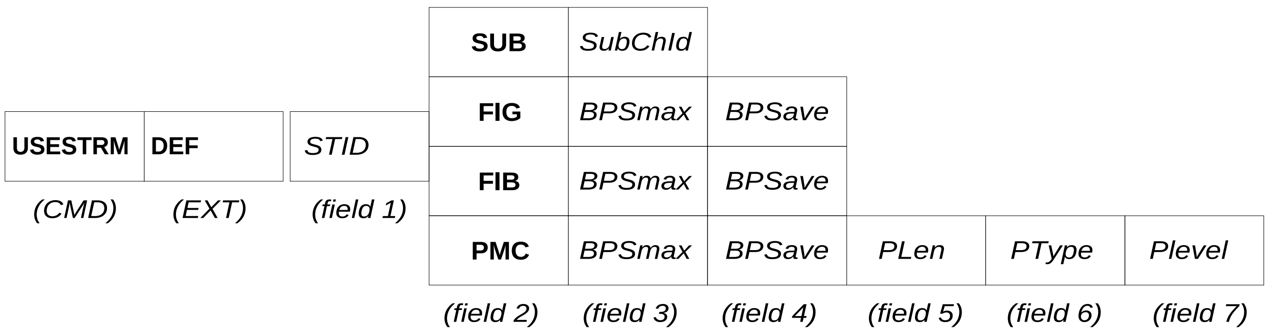

Figure 24: USESTRM DEF message structure

Table 17 presents the definition of the USESTRM DEF message fields. Fields length is expressed as a number of characters.

The fields marked (*) contain parameters or terms defined in EN 300 401 [1]. Their values shall be compliant with those defined in EN 300 401 [1].

Table 17: USESTRM DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
STID	identifier of the stream	3	hex	"000"..."FFF"
SUB	stream contains data destined for a sub-channel	3	string	"SUB"
SubChId	sub-channel identifier* in which stream shall be carried	[1...2]	dec	"0"..."63"
FIG	stream contains FIG contribution	3	string	"FIG"
FIB	stream contains FIBs for asynchronous insertion	3	string	"FIB"
PMC	stream contains packet mode contribution	3	string	"PMC"
BPSmax	maximum number of bytes per second	[1...6]	dec	"0"..."228 000"
BPSave	average number of bytes per second	[1...6]	dec	"0"..."228 000"
PLen	Packet length* (in bytes)	[1...2]	dec	"0", "24", "48", "72", "96" where "0" indicates that mixed packet lengths are used
PType	Protection type required	1	char	"U" for UEP "E" for EEP
PLevel	Protection level* (and option*) required	1 or 2	dec	"1"..."5" for UEP *LO* for EEP, where *L* is protection level* "1"..."4" *O* is protection option* "0"..."7"
NOTE 1: More than one FIG stream may be defined as contributing to the DAB multiplex. NOTE 2: More than one FIB stream may be defined as contributing to the DAB multiplex. NOTE 3: More than one PMC stream may be defined as contributing to the DAB multiplex. NOTE 4: Only sub-channels not referenced by the service configuration shall be defined.				

6.5.5 CMPNENT messages

The CMPNENT messages shall be used to describe service components and their characteristics.

A component identifier, which shall be unique within a configuration, shall be provided for referencing the component in the service description. The entity creating a configuration shall define the component identifier.

The source of the component data shall normally be given as the STI-D(LI) STID. However, components provided by other Service providers can also be referenced. These components are called external components and are referenced by the SPID and the component identifier used by the other Service provider. The component definition is also provided by the other Service provider.

CMPNENT messages shall only be used during an open configuration data exchange session.

The CMPNENT messages are presented in table 11 and are defined in the following clauses.

CMPNENT DEF allows the originator of a configuration data exchange session to identify a component and to describe its characteristics.

6.5.5.1 CMPNENT DEF

CMPNENT DEF shall be issued to define the characteristics of a DAB component as part of a configuration.

The definition of a component shall be performed before the component is referenced in a service definition.

Figure 25 presents the CMPNENT DEF message structure.

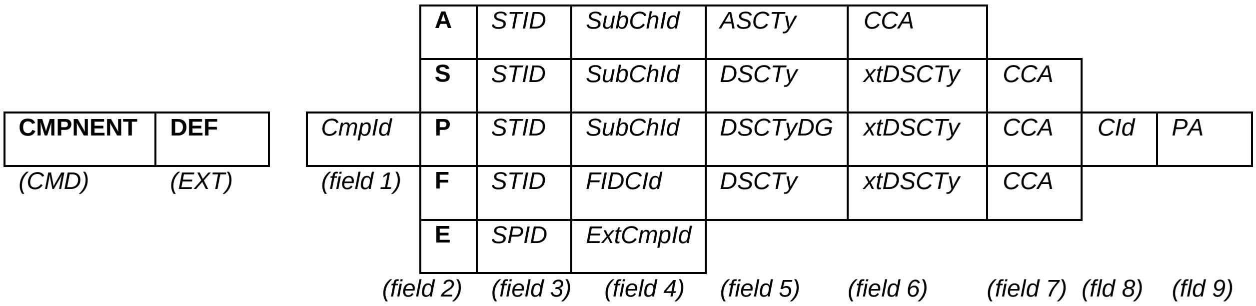

Figure 25: CMPNENT DEF message structure

Table 18 presents the definition of the CMPNENT DEF message fields. Fields length is expressed as a number of characters.

The fields marked (*) contain parameters defined in EN 300 401 [1]. Their values shall be compliant with those defined in EN 300 401 [1].

Table 18: CMPNENT DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
CmpId	component identifier	[1...4]	dec	"0"..."9 999"
A	audio component	1	char	"A"
S	stream mode component	1	char	"S"
P	packet mode component	1	char	"P"
F	fast info. data channel component	1	char	"F"
E	external component	1	char	"E"
STID	stream identifier	3	hex	"000"..."FFF"
SPID	Service provider identifier	4	hex	"0000"..."FFFF"
SubChId	Sub-channel identifier*	[1...2]	dec	"0"..."63"
FIDCId	Fast Info. Data Channel Id.*	[1...2]	dec	"0"..."63"
ExtCmpId	external component identifier	[1...4]	dec	"0"..."9 999"
ASCTy	Audio Service Component Type*	[1...2]	dec	"0"..."63"
DSCTy	Data Service Component Type*	[1...2]	dec	"0"..."63"
DSCTyDG	Data Service Component Type and DG flag*	[1…3]	dec	'0'...'63' or '128'…'191' (see note 3)
ExtDSCTy	DSCTy extension*	3 or 1	hex or char	"000"..."3FF" if used or "N" if not used
SCCA	Service Component CA*	4 or 1	hex or char	"0000"..."FFFF" if used or "N" if not used
SCId	Service Component Identifier*	3	hex	"000"..."FFF"
PA	Packet Address*	3	hex	"000"..."3FF"
NOTE 1: If the ExtDSCTy field is coded as "N" (not used) then no FIG(0/7) shall be generated for that component. NOTE 2: If the SCCA field is coded as "N" (not used) then the CA flag in the FIG(0/2) shall be set to zero. For stream mode or FIDC no FIG(0/4) shall be generated for the component. For packet mode the FIG(0/3) generated for that component shall have the SCCA flag set to zero and no SCCA field shall be present. NOTE 3: The value of the DSCTyDG field is calculated by taking the numerical value of the DSCTy and adding 128 if the DG flag is 1 and 0 if the DG flag is 0.				

6.5.6 SERVICE messages

The SERVICE messages shall be used to define a service as part of a configuration. The service shall be composed of components.

SERVICE messages shall only be used during an open configuration data exchange session.

The SERVICE messages presented in table 11 are defined in the following clauses.

SERVICE DEF allows the originator of the data exchange session to describe a service by providing the SId, the Local flag, the CAId, the number of components and the component list.

6.5.6.1 SERVICE DEF

SERVICE DEF shall be issued to define the composition of a DAB service.

Figure 26 presents the SERVICE DEF message structure and the structure of the component description.

A service definition using SERVICE DEF shall provide a list of components. The components referenced from the list shall form the service. The list shall contain *NComp* component references, each composed of a *SCIdS* and a *CmpId* field.

The first component reference in the component list shall define the primary service component as defined in EN 300 401 [1].

Component references shall only contain component identifiers of components previously defined.

Figure 26: SERVICE DEF message structure

Table 19 presents the definition of the SERVICE DEF message fields. Fields length is expressed as a number of characters. The fields marked (*) contain parameters defined in the EN 300 401 [1]. Their values shall be compliant with those defined in EN 300 401 [1].

Table 19: SERVICE DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
SId	Service Identifier*	4 or 8	hex	"0000"..."FFFF" or "00000000"..."FFFFFFFF"
Local	Local flag to indicate service area*	1	dec	"0"..."1"
CAId	Conditional Access Identifier*	1	dec	"0"..."7"
NComp	Number of components	[1..2]	dec	"1"..."12"
SCIdS	Service Component Id in Service*	1	hex or char	"0"..."F" if used or "N" if not used
CmpId	Component Identifier	[1..4]	dec	"0"..."9 999"

6.5.7 USEFIGF messages

The USEFIGF messages shall be used to define the contents of a FIG file as a forming part of the configuration.

USEFIGF messages shall only be used during an open configuration data exchange session.

The USEFIGF messages are presented in table 11 and are defined in the following clauses.

The USEFIGF DEF allows the originator of a configuration data exchange session to define a FIG file as forming part of a configuration.

6.5.7.1 USEFIGF DEF

The USEFIGF DEF message shall be issued to define a FIG file as forming part of a configuration.

The FIGs contained in the FIG file shall be broadcast under control of the downstream entity until the FIG file is deselected.

The downstream entity shall be responsible for FIG repetition and FIG change control.

Figure 27 presents the USEFIGF DEF message structure.

USEFIGF	**DEF**	*Name*
(CMD)	*(EXT)*	*(field 1)*

Figure 27: USEFIGF DEF message structure

Table 20 presents the definition of the USEFIGF DEF message fields. Fields length is expressed as a number of characters.

Table 20: USEFIGF DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Name	Name of the FIG file	[1...16]	string	Any string corresponding to a *Name* previously defined

6.6 FIG file messages

The FIG file messages provide a mechanism to exchange and handle FIC information between upstream entity and downstream entity.

FIG file messages are presented in table 21 and are defined in the following clauses.

Table 21: FIGFILE messages of the STI-C(LI)

FIGFILE MESSAGES			
CMD	***EXT***	***data fields***	
FIGFILE	INF	*Name*	
FIGFILE	DEF	*Name*	*NRecs*
FIGFILE	REC	*Data*	
FIGFILE	END		
FIGFILE	DEL	*Name*	
FIGFILE	SEL	*Name*	
FIGFILE	DES	*Name*	
FIGFILE	ERR	*Name*	*Cause*

6.6.1 General rules to use FIG file messages

The FIG file messages allow FIC information to be defined, monitored and managed.

FIG files shall contain Service Information FIGs only.

The upstream entity can define and monitor named FIG files for broadcast at the downstream entity. Each FIG file shall be defined by a unique name that consists of a minimum of 1 character and a maximum of 16 characters.

The downstream entity can use FIG files to inform the upstream entity about Service Information FIGs that are also being broadcast (e.g. ensemble related information).

FIG files can be created and deleted, but not modified.

To create a new FIG file the definition shall be given using a name that is not currently defined. The first message sent shall be a FIGFILE DEF message which opens a FIG file data exchange session. The number of FIGs to be transferred during the session shall be specified. The FIG file is then transferred. The session shall be closed by issuing a FIGFILE END message. If during the data exchange session errors are detected by the remote entity, the session can be aborted.

The upstream entity can select predefined FIG files for broadcast. The FIG information contained in the selected FIG file shall become part of the broadcast DAB multiplex until the file is deselected. A configuration can select one or more FIG files for broadcast. A reconfiguration shall deselect all FIG files that were selected before the time instant of the reconfiguration and select those that are specified by the new configuration. FIG files can also be selected and deselected outside of the reconfiguration process.

If the downstream entity accepts the selection of the FIG file, it shall broadcast all the FIGs contained within the file. The downstream entity is responsible for FIG repetition and FIG change control mechanism for all FIGs contained in FIG files.

NOTE: Agreement should be sought between the upstream and downstream entities on the action to be performed if insufficient capacity is available to broadcast all the requested FIG files at an adequate repetition rate.

6.6.2 FIGFILE messages

FIGFILE messages provide the mechanism to handle FIG information that an upstream entity wants the downstream entity to broadcast. An upstream entity can define and store a number of FIG files in the downstream entity using the mechanism described in this clause.

FIGFILE INF shall be issued to request that the remote entity opens a FIG file data exchange session.

FIGFILE DEF shall be issued to open a FIG file data exchange session to provide the contents of a named FIG file.

FIGFILE REC shall be issued to transfer one FIG.

FIGFILE END shall be issued to close a FIG file data exchange session.

FIGFILE DEL shall be issued to delete a FIG file at the remote entity.

FIGFILE SEL shall be issued by the upstream entity to select a FIG file for broadcast.

FIGFILE DES shall be issued by the upstream entity to deselect a FIG file and stop its contents being broadcast.

FIGFILE ERR shall be issued to abort an open FIG file data exchange session, or to indicate errors caused by the use of other FIGFILE messages.

6.6.2.1 FIGFILE INF

FIGFILE INF shall be issued to request the content of a named FIG file.

The remote entity shall either open a FIG file data exchange session to provide the requested FIG file or indicate an error if the file does not exist by issuing FIGFILE ERR.

Figure 28 presents the FIGFILE INF message structure.

FIGFILE	**INF**	*Name*
(CMD)	*(EXT)*	*(field 1)*

Figure 28: FIGFILE INF message structure

Table 22 presents the definition of the FIGFILE INF message fields. Fields length is expressed as a number of characters.

Table 22: FIGFILE INF message field definition

Field Name	Meaning	Length	Field type	Possible Value
Name	Name of the FIG file	[1...16]	string	The name of the FIG file required

6.6.2.2 FIGFILE DEF

FIGFILE DEF shall be issued to open a FIG file data exchange session for a named FIG file.

The data fields of the FIGFILE DEF messages are used to define the name and the number of records contained in the FIG file.

Following the issue of the FIGFILE DEF message, exactly *NRecs* FIGFILE REC messages shall be issued, and the session will be closed by issuing the FIGFILE END message. As one FIGFILE REC message shall contain one FIG, *NRecs* shall be the number of FIGs in the file.

Figure 29 presents the FIGFILE DEF message structure.

FIGFILE	**DEF**	*Name*	*NRecs*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*

Figure 29: FIGFILE DEF message structure

Table 23 presents the definition of the FIGFILE DEF message fields. Fields length is expressed as a number of characters.

Table 23: FIGFILE DEF message fields definition

Field Name	Meaning	Length	Field type	Possible Value
Name	Name of the FIG file	[1...16]	string	Any string
NRecs	Number of FIGFILE REC messages included in the session	[1...2]	dec	"1"..."99"

6.6.2.3 FIGFILE REC

FIGFILE REC shall be issued only when a FIG file data exchange session is open.

FIGFILE REC shall be issued to transfer one FIG as part of a named FIG file.

The data part of FIGFILE REC contains one FIG. Each byte of the FIG is coded as a two-digit hexadecimal number representing the numerical value of the byte. The bytes shall be coded in the same order as they are to be broadcast (i.e. the FIG header field, as defined in EN 300 401 [1], shall be coded first).

Figure 30 presents the FIGFILE REC message structure.

FIGFILE	**REC**	*Data*
(CMD)	*(EXT)*	*(field 1)*

Figure 30: FIGFILE REC message structure

Table 24 presents the definition of the FIGFILE REC message fields. Fields length is expressed as a number of characters.

Table 24: FIGFILE REC message fields definition

Field Name	Meaning	Length	Field type	Possible Value
Data	data for one FIG	[8...60]	string	all hexadecimal characters

6.6.2.4 FIGFILE END

FIGFILE END shall be issued only when a FIG file data exchange session is open.

FIGFILE END shall be issued to close a FIG file data exchange session by the entity that opened the session. The FIGFILE END message shall be sent after the complete number of records as stated in the FIGFILE DEF message have been transferred.

Figure 31 presents the FIGFILE END message structure.

FIGFILE	**END**
(CMD)	*(EXT)*

Figure 31: FIGFILE END message structure

6.6.2.5 FIGFILE DEL

FIGFILE DEL shall be issued to delete a FIG file that is stored at the remote entity. If the FIG file is selected for broadcast, then the file will be deselected.

If the name of the FIG file is not recognized then an error will be indicated.

Figure 32 presents the FIGFILE DEL message structure.

FIGFILE	**DEL**	*Name*
(CMD)	*(EXT)*	*(field 1)*

Figure 32: FIGFILE DEL message structure

Table 25 presents the definition of the FIGFILE DEL message fields. Fields length is expressed as a number of characters.

Table 25: FIGFILE DEL message field definition

Field Name	Meaning	Length	Field type	Possible Value
Name	Name of the FIG file	[1...16]	string	The name of the FIG file to be deleted

6.6.2.6 FIGFILE SEL

Only the upstream entity shall issue the FIGFILE SEL message.

FIGFILE SEL shall be issued to select a predefined FIG file for broadcast by the downstream entity. If the FIG file is already is selected, the message shall have no effect.

If the file does not exist, the downstream entity shall issue a FIGFILE ERR message, stating "UNKNOWN" as the cause.

If the required capacity for the FIG information is not available, the downstream entity should issue a FIGFILE ERR message, stating "NOSPACE" as the cause.

If the content of the file contains errors or is conflicting with other service information at the downstream entity, the downstream entity should issue a FIGFILE ERR message, stating "CONTENT" as the cause.

Figure 33 presents the FIGFILE SEL message structure.

FIGFILE	**SEL**	*Name*
(CMD)	*(EXT)*	*(field 1)*

Figure 33: FIGFILE SEL message structure

Table 26 presents the definition of the FIGFILE SEL message fields. Fields length is expressed as a number of characters.

Table 26: FIGFILE SEL message field definition

Field Name	Meaning	Length	Field type	Possible Value
Name	Name of the FIG file	[1...16]	string	The name of the FIG file to be selected

6.6.2.7 FIGFILE DES

Only the upstream entity shall issue the FIGFILE DES command.

FIGFILE DES shall be issued to deselect a previously selected FIG file.

If the specified file is not selected when issuing FIGFILE DES, the downstream entity shall issue a FIGFILE ERR message, stating "UNKNOWN" as the cause.

Figure 34 presents the FIGFILE DES message structure.

FIGFILE	**DES**	*Name*
(CMD)	*(EXT)*	*(field 1)*

Figure 34: FIGFILE DES message structure

Table 27 presents the definition of the FIGFILE DES message fields. Fields length is expressed as a number of characters.

Table 27: FIGFILE DES message field definition

Field Name	Meaning	Length	Field type	Possible Value
Name	Name of the FIG file	[1...16]	string	The name of the FIG file to be deselected

6.6.2.8 FIGFILE ERR

The FIGFILE ERR message shall be issued to indicate an error by issuing the FIGFILE SEL or the FIGFILE DES commands and errors that occur on a FIG file data exchange session.

The issue of a FIGFILE ERR during an open FIG file data exchange session aborts the session. An aborted session shall not create a FIG file.

Figure 35 presents the FIGFILE ERR message structure.

FIGFILE	**ERR**	*Name*	*Cause*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*

Figure 35: FIGFILE ERR message structure

Table 28 presents the definition of the FIGFILE DES message fields. Fields length is expressed as a number of characters.

Table 28: FIGFILE ERR message fields definition

Field Name	Meaning	Length	Field type	Possible Value
Name	Name of the FIG file	[1..16]	string	The name of the FIG file referenced
Cause	cause of the error	7	string	for errors concerning FIG file data exchange sessions: "NRECORD" wrong number of records "USED" the name is already defined "NOSPACE" all FIG file stores have been used for error reactions on FIGFILE INF, FIGFILE DEL: "UNKNOWN" unrecognized name for error reactions on FIGFILE SEL: "UNKNOWN" unrecognized name "FIGCAP" capacity not available "CONTENT" content error for error reactions on FIGFILE DES: "UNKNOWN" referenced file not selected

6.7 FIB grid messages

The FIB grid messages shall be used to co-ordinate the delivery and insertion of Fast Information Blocks (FIBs) using the synchronous insertion method.

A FIB grid, when agreed between upstream entity and downstream entity, shall represent a structure that allows the upstream entity to know exactly in which transmission frame a FIB will be broadcast.

The FIB grid messages shall be used to exchange and activate FIB grids as well as to indicate errors.

The FIBGRID messages are presented in table 29 and are defined in the following clauses.

Table 29: FIB grid messages of the STI-C(LI)

FIB grid messages		
CMD	***EXT***	***data fields***
FIBGRID	INF	
FIBGRID	DEF	
FIBGRID	REC	*Allocation*
FIBGRID	END	
FIBGRID	ACT	*UTC*
FIBGRID	ERR	*Cause*

6.7.1 General rules to use FIBGRID messages

The FIBGRID messages shall be used to manage and transfer a FIB grid from downstream entity to upstream entity.

The FIB grid shall be used simultaneously and synchronously by both entities, according to the rules given in this clause.

A FIB grid shall state for each FIB whether the FIB shall be provided by the upstream entity or the downstream entity. A FIB grid structure of 500 records shall be used, each record representing the FIB allocation associated with one CIF.

The FIB grid records shall be aligned to the CIF count value such that the first record of the FIB grid shall be associated with the DAB transmission frame where the lower nine bits of the CIF count are zero. The FIB grid shall be used cyclically with a period of 500 CIFs.

When FIB access is granted to the upstream entity, the upstream entity shall provide a FIB stream having TID field equal to 5 and TIDext field equal to 1. This stream shall contain FIBs in those STI-D(LI) frames that relate to transmission frames with FIB access for the upstream entity.

A FIB grid data exchange session shall be used to transfer a FIB grid from the downstream entity to the upstream entity. A FIB grid data exchange session shall contain exactly ten FIBGRID REC messages.

After a FIB grid data exchange session is completed and no errors are reported, the FIB grid shall be stored by both entities and be valid for activation.

6.7.2 FIBGRID messages

The FIBGRID messages shall allow the downstream entity to introduce a FIB grid to the upstream entity.

The use of the FIB grid shall be governed by the CIF count value of the DAB Ensemble. The COUNTER messages defined in clause 6.9.6 shall be used for the co-ordination of upstream entity and downstream entity frame counters.

FIBGRID INF shall be issued by the upstream entity to request the definition of a FIB grid from the downstream entity.

FIBGRID DEF shall be issued by the downstream entity to open a FIB grid data exchange session for the transfer of a FIB grid.

FIBGRID REC shall be issued by the downstream entity to specify the value of each FIB grid record. Ten FIBGRID REC messages shall be issued to define the FIB grid.

FIBGRID END shall be issued by the downstream entity to close a FIB grid data exchange session.

FIBGRID ERR shall be issued to signal an error in the content of the FIB grid and abort the FIB grid data exchange session.

FIBGRID ACT shall be issued by the downstream entity to activate a FIB grid.

6.7.2.1 FIBGRID INF

Only the upstream entity shall issue the FIBGRID INF message.

The FIBGRID INF message shall be issued to request the definition of a FIB grid from the downstream entity. The downstream entity shall respond by opening a FIB grid data exchange session or by issuing an error message to indicate that no FIB grid is available.

Figure 36 presents the FIBGRID INF message structure.

FIBGRID	**INF**
(CMD)	*(EXT)*

Figure 36: FIBGRID INF message structure

6.7.2.2 FIBGRID DEF

Only the downstream entity shall issue the FIBGRID DEF message.

The FIBGRID DEF message shall be issued to open a FIB grid data exchange session.

Following the issue of the FIBGRID DEF message, the downstream entity shall issue ten FIBGRID REC messages and then close the data exchange session by issuing the FIBGRID END message.

Figure 37 presents the FIBGRID DEF message structure.

FIBGRID	**DEF**
(CMD)	*(EXT)*

Figure 37: FIBGRID DEF message structure

6.7.2.3 FIBGRID REC

Only the downstream entity shall issue the FIBGRID REC message.

The FIBGRID REC message shall only be issued during an open FIB grid data exchange session.

FIBGRID REC messages shall be issued to transfer FIB grid records from downstream entity to upstream entity.

One FIBGRID REC message shall carry the definition of fifty successive FIB grid records. A set of ten FIBGRID REC messages shall be issued to define the whole FIB grid.

Each FIB grid record is coded as a hexadecimal character representing a 4-bit pattern defining the allocation of the FIBs. A bit value equal to 1 shall mean that access is granted to the upstream entity for the corresponding FIB, a bit value equal to 0 shall mean that access to the corresponding FIB is forbidden for the upstream entity. The most significant bit shall provide the allocation of the first FIB in the frame, the next most significant bit shall provide the allocation of the second FIB, and so on. In DAB transmission mode III each four FIBs are associated with each CIF and therefore the FIB grid record may take any value from "0", indicating no access to the upstream entity, to "F", indicating full access to the upstream entity. In the other DAB transmission modes three FIBs are associated with each CIF and therefore the FIB grid record shall always have the least significant bit set to zero. In this case the FIB grid record takes the values "0", "2", "4",…,"E".

The first record of the first FIBGRID REC message within a FIB grid data exchange session shall detail the FIB allocation of the DAB frame whose CIF count has the lower nine bits set to zero.

Figure 38 presents the FIBGRID REC message structure.

FIBGRID	**REC**	*Allocation*
(CMD)	*(EXT)*	*(field 1)*

Figure 38: FIBGRID REC message structure

Table 30 presents the definition of the FIBGRID REC message fields. Fields length is expressed as a number of characters.

Table 30: FIBGRID REC message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Allocation	FIB grid record for 50 successive CIFs	50	hex	"0...0".."F…F"

6.7.2.4 FIBGRID END

Only the downstream entity shall issue the FIBGRID END message.

The FIBGRID END message shall be issued to close a FIB grid data exchange session.

Figure 39 presents the FIBGRID END message structure.

FIBGRID	**END**
(CMD)	*(EXT)*

Figure 39: FIBGRID END message structure

6.7.2.5 FIBGRID ACT

Only the downstream entity shall issue the FIBGRID ACT message.

FIBGRID ACT shall be issued to activate the usage of a FIB grid by the upstream entity. The message shall only be issued when a valid FIB grid is stored at both entities. If the upstream entity does not have a FIB grid definition stored, then an error shall be indicated.

The FIB grid shall become active at the first instance that the lower nine bits of the CIF count are zero after the time given in the *UTC* field of FIBGRID ACT.

The activated FIB grid shall remain in use by the upstream and downstream entity until the activation time of another FIB grid.

Figure 40 presents the FIBGRID ACT message structure.

FIBGRID	**ACT**	*UTC*
(CMD)	*(EXT)*	*(field 1)*

Figure 40: FIBGRID ACT message structure

Table 31 presents the definition of the FIBGRID ACT message fields. Fields length is expressed as a number of characters.

Table 31: FIBGRID ACT message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
UTC	Time after which activation is effective	8	string	"HH:MM:SS" "HH" in "00".."23" for hours, "MM" in "00".."59" for minutes, "SS" in "00".."59" for seconds.

6.7.2.6 FIBGRID ERR

FIBGRID ERR shall be issued to signal errors concerning a FIB grid data exchange session or other errors concerning FIB grids.

Figure 41 presents the FIBGRID ERR message structure.

FIBGRID	**ERR**	*Cause*
(CMD)	*(EXT)*	*(field 1)*

Figure 41: FIBGRID ERR message structure

Table 32 presents the definition of the FIBGRID ERR message fields. Fields length is expressed as a number of characters.

Table 32: FIBGRID ERR message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Cause	Type of error detected	4	string	for errors concerning FIB grid data exchange sessions: "NREC" number of records is wrong for error reactions on FIBGRID INF or FIBGRID ACT: "GRID" no FIB grid available

6.8 Resource messages

The resource messages are used to monitor resources that are limited within a DAB ensemble and that should be shared by all service providers, like for example capacity of the Main Service Channel or sub-channel identifiers. The resources are allocated by the Ensemble provider.

The resource messages are presented in table 33 and are defined in the following clauses.

Table 33: Resource messages of the STI-C(LI)

RESOURCE MESSAGES						
CMD	***EXT***	***data fields***				
RESOURC	INF					
RESOURC	DEF	*NMsg*				
RESOURC	END					
RESOURC	ERR	*Cause*				
CHANCAP	DEF	MSC	*CapUnits*			
		FIC	*BPSmax*			
STLIMIT	DEF	SUB	*NSUBSt*	*BPSmax*		
		FIB	*NFIBSt*	*BPSave*	*BPSmax*	
		FIG	*NFIGSt*	*BPSave*	*BPSmax*	
		PMC	*NPMCSt*	*BPSave*	*BPSmax*	
IDALLOC	DEF	SUB	*SubIdLo*	*SubIdHi*		
		DS	*DSIdLo*	*DSIdHi*		
		PS	*PSIdLo*	*PSIdHi*		
		SC	*SCIdLo*	*SCIdHi*		
		FIDC	*FIDCIdLo*	*FIDCIdHi*		
		LIH	*LSNLo*	*LSNHi*		
		LIS	*LSNLo*	*LSNHi*		
		LNH	*LSNLo*	*LSNHi*		
		LNS	*LSNLo*	*LSNHi*		
IDLIMIT	DEF	SUB	*NSub*			
		DS	*NDS*			
		PS	*NPS*			
		SC	*NSC*			
		SCS	*NSCS*			
		FIDC	*NFIDC*			
PACKCON	DEF	*PLen*	*Ptype*	*PLevel*	*SubChId*	*PA*
FIGBLCK	DEF	*FIGtype*	*FIGext*			
ANNSEND	DEF	*ClusterId*	*AnnType*			

6.8.1 General rules to use resource messages

The resource messages can be used to transfer details about resource allocations.

The downstream entity shall use a resource data exchange session to transfer information about allocated resources to the upstream entity.

To request information about the resources that the downstream entity currently has allocated for the upstream entity's use, the upstream entity shall send a RESOURC INF message. The downstream entity shall then respond by opening a resource data exchange session.

NOTE: The information about allocated resources provided by the downstream entity shall be valid at the time the downstream entity issues the information.

The data exchange session can be aborted by the recipient if an error is detected. The reason for the error shall be stated.

6.8.2 RESOURC messages

RESOURC messages are used to exchange information about scarce resources.

The RESOURC messages are presented in table 33 and are defined in the following clauses.

The RESOURC INF message shall be issued by the upstream entity to request information about resources allocated to the upstream entity by the downstream entity.

The RESOURC DEF message shall be issued by the downstream entity to provide information about resources allocated by the downstream entity for the upstream entity. This message opens a resource data exchange session.

The RESOURC END message shall be issued by the downstream entity to close a data exchange session.

The RESOURC ERR message shall be issued to indicate errors.

6.8.2.1 RESOURC INF

Only the upstream entity shall issue the RESOURC INF message.

The RESOURC INF message shall be issued to request the definition of resources allocated by the downstream entity to the upstream entity. The downstream entity shall respond by opening a resource data exchange session.

Figure 42 presents the RESOURC INF message structure.

RESOURC	**INF**
(CMD)	*(EXT)*

Figure 42: RESOURC INF message structure

6.8.2.2 RESOURC DEF

Only the downstream entity shall issue the RESOURC DEF message.

RESOURC DEF shall be issued to open a resource data exchange session.

The data field of the RESOURC DEF message shall state the number of messages contained in the resource data exchange session.

Following the issue of the RESOURC DEF message, the exact number of messages as described in the RESOURC DEF data field shall be issued, and the session shall be closed by issuing the RESOURC END message.

Figure 43 presents the RESOURC DEF message structure.

RESOURC	**DEF**	*NMsg*
(CMD)	*(EXT)*	*(field 1)*

Figure 43: RESOURC DEF message structure

Table 34 presents the definition of the RESOURC DEF message fields. Fields length is expressed as a number of characters.

The fields marked (*) contain parameters defined in EN 300 401 [1]. Their values shall be compliant with those defined in EN 300 401 [1].

Table 34: RESOURC DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
NMsg	Total number of messages in the session i.e. between RESOURC DEF and RESOURC END	[1..3]	dec	"0".."999"

6.8.2.3 RESOURC END

Only the downstream entity shall issue the RESOURC END message.

The RESOURC END message shall be issued to close a resource data exchange session.

This message shall be sent after all the messages announced by the RESOURC DEF message have been issued.

Figure 44 presents the RESOURC END message structure.

RESOURC	**END**
(CMD)	*(EXT)*

Figure 44: RESOURC END message structure

6.8.2.4 RESOURC ERR

Only the upstream entity shall issue the RESOURC ERR message.

The RESOURC ERR message shall be issued to indicate an error concerning a resource data exchange session. The issue of RESOURC ERR shall abort the resource data exchange session.

Figure 45 presents the RESOURC ERR message structure.

RESOURC	**ERR**	*Cause*
(CMD)	*(EXT)*	*(field 1)*

Figure 45: RESOURC ERR message structure

Table 35 presents the definition of the RESOURC ERR message fields. Fields length is expressed as a number of characters.

Table 35: RESOURC ERR message fields definition

Field Name	Meaning	Length	Field Type	Possible Value	
Cause	Cause of the error	4	string	"NMSG"	a number of messages different from *NMsg* given in RESOURC DEF was received

6.8.3 CHANCAP messages

The CHANCAP messages shall be used to define the transmission channel capacities allocated to the upstream entity by the downstream entity. CHANCAP messages can be used to allocate Main Service Channel capacity and Fast Information Channel capacity.

CHANCAP messages shall only be used when a resource data exchange session is open. Only one CHANCAP message for the MSC and one for the FIC shall be used in a resource data exchange session.

The CHANCAP messages presented in table 33 are defined in the following clauses.

CHANCAP DEF shall be issued by the downstream entity to indicate the maximum transmission channel capacity which can be used by the upstream entity at the time the message is issued.

6.8.3.1 CHANCAP DEF

Only the downstream entity shall issue the CHANCAP DEF message.

CHANCAP DEF shall be issued to indicate a channel capacity which is allocated to the upstream entity at the time the message is issued.

Figure 46 presents the CHANCAP DEF message structure.

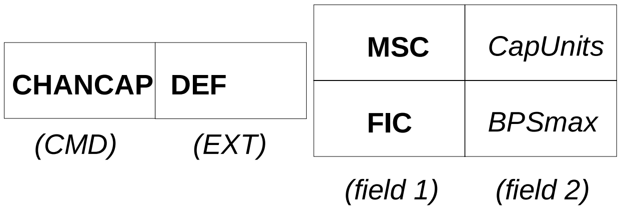

Figure 46: CHANCAP DEF message structure

Table 36 presents the definition of the CHANCAP DEF message fields. Fields length is expressed as a number of characters.

The fields marked (*) contain parameters defined in EN 300 401 [1]. Their values shall be compliant with those defined in EN 300 401 [1].

Table 36: CHANCAP DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
MSC	Main Service Channel is concerned	3	string	"MSC"
CapUnits	Capacity units* allocated	[1..3]	dec	"0".."864"
FIC	Fast Information Channel is concerned	3	string	"FIC"
BPSmax	maximum FIC capacity allocated, given in bytes per second. For the usage of this capacity both Service Information and Multiplex Configuration Information are taken into account.	[1..4]	dec	"0".."5 334"

6.8.4 STLIMIT messages

The STLIMIT messages shall be used to define limitations regarding usage of certain stream types allocated to the upstream entity by the downstream entity. Related to specific stream type the limitations apply to both maximum number of streams and bitrate which can be used.

STLIMIT messages shall only be used when a resource data exchange session is open. Only one STILIMIT message per stream type shall be used in a resource data exchange session.

The STLIMIT messages are presented in table 33 and are defined in the following clauses.

STLIMIT DEF shall be issued by the downstream entity to indicate limits for stream usage which are allocated to the upstream entity at the time the message is issued.

6.8.4.1 STLIMIT DEF

Only the downstream entity shall issue the STLIMIT DEF message.

STLIMIT DEF allows the downstream entity to indicate limits of stream usage which are allocated to the upstream entity at the time the message is issued.

Figure 47 presents the STLIMIT DEF message structure.

STLIMIT	DEF
(CMD)	(EXT)

(field 1)	(field 2)	(field 3)	(field 4)
SUB	*NSUBSt*	*BPSmax*	
FIB	*NFIBSt*	*BPSave*	*BPSmax*
FIG	*NFIGSt*	*BPSave*	*BPSmax*
PMC	*NPMCSt*	*BPSave*	*BPSmax*

Figure 47: STLIMIT DEF message structure

Table 37 presents the definition of the STLIMIT DEF message fields. Fields length is expressed as a number of characters.

Table 37: STLIMIT DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
SUB	MSC sub-channel streams allocated	3	string	"SUB"
NSUBSt	maximum number of MSC sub-channel streams	1	dec	"0".."99"
FIB	asynchronous FIB streams allocated	3	string	"FIB"
NFIBSt	maximum number of asynchronous FIB streams	1	dec	"0".."2"
FIG	FIG streams allocated	3	string	"FIG"
NFIGSt	maximum number of FIG streams	2	dec	"0".."99"
PMC	PMC streams allocated	3	string	"PMC"
NPMCSt	maximum number of PMC streams	3	dec	"0".."999"
BPSave	average number of bytes per second and per stream	[1..6]	dec	"0".."228 000"
BPSmax	maximum number of bytes per second and per stream	[1..6]	dec	"0".."228 000"
NOTE: Synchronous FIB streams are governed by use of the FIB grid.				

6.8.5 IDALLOC messages

The IDALLOC messages shall be used to define ranges of identifiers allocated to the upstream entity by the downstream entity. Simultaneous use of these identifiers can be limited by means of IDLIMIT messages (see clause 6.8.6).

IDALLOC messages shall only be used when a resource data exchange session is open. Several IDALLOC messages per identifier type can be used in a resource data exchange session.

The IDALLOC messages presented in table 33 are defined in the following clauses.

IDALLOC DEF shall be issued by the downstream entity to indicate an identifier range or list which is allocated to the upstream entity at the time the message is issued.

6.8.5.1 IDALLOC DEF

Only the downstream entity shall issue the IDALLOC DEF message.

IDALLOC DEF shall be issued to indicate a range of specific identifier values which are allocated to the upstream entity at the time the message is issued. The range specification allows for allocation of single identifiers too. In that case the corresponding value has to be specified twice i.e. lower bound equals upper bound of the range.

The IDALLOC DEF message shall only be issued when a resource data exchange session is open.

Figure 48 presents the IDALLOC DEF message structure.

(CMD)	(EXT)	(field 1)	(field 2)	(field 3)
		SUB	*SubIdLo*	*SubIdHi*
		DS	*DSIdLo*	*DSIdHi*
		PS	*PSIdLo*	*PSIdHi*
		SC	*SCIdLo*	*SCIdHi*
IDALLOC	**DEF**	**FIDC**	*FIDCIdLo*	*FIDCIdHi*
		LIH	*LSNLo*	*LSNHi*
		LIS	*LSNLo*	*LSNHi*
		LNH	*LSNLo*	*LSNHi*
		LNS	*LSNLo*	*LSNHi*

Figure 48: IDALLOC DEF message structure

Table 38 presents the definition of the IDALLOC DEF message fields. Fields length is expressed as a number of characters.

The fields marked (*) contain parameters defined in EN 300 401 [1]. Their values shall be compliant with those defined in EN 300 401 [1].

Table 38: IDALLOC DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
SUB	MSC Sub-channel identifier allocated	3	string	"SUB"
SubIdLo	lower bound of Sub-channel identifier* range	[1..2]	hex	"0".."3F"
SubIdHi	upper bound of Sub-channel identifier* range	[1..2]	hex	"0".."3F"
DS	Data Service identifier allocated	2	string	"DS"
DSIdLo	lower bound of Data Service identifier* range	8	hex	"00000000".."FFFFFFFF"
DSIdHi	upper bound of Data Service identifier* range	8	hex	"00000000".."FFFFFFFF"
PS	Program Service identifier allocated	2	string	"PS"
PSIdLo	lower bound of Program Service identifier* range	4	hex	"0000".."FFFF"
PSIdHi	upper bound of Program Service identifier* range	4	hex	"0000".."FFFF"
SC	Service Component identifier allocated	2	string	"SC"
SCIdLo	lower bound of Service Component identifier* range	[1..3]	hex	"0".."FFF"
SCIdHi	upper bound of Service Component identifier* range	[1..3]	hex	"0".."FFF"
FIDC	Fast Information Data Channel identifier allocated	4	string	"FIDC"
FIDCIdLo	lower bound of Fast Information Data Channel identifier* range	[1..2]	hex	"0".."3F"
FIDCdHi	upper bound of Fast Information Data Channel identifier* range	[1..2]	hex	"0".."3F"
LIH	allocated Linkage Set Number for International, Hard link *	3	string	"LIH"
LIS	allocated Linkage Set Number for International, Soft link *	3	string	"LIS"
LNH	allocated Linkage Set Number for National, Hard link *	3	string	"LNH"
LNS	allocated Linkage Set Number for National, Soft link *	3	string	"LNS"
LSNLo	lower bound of Linkage Set Number * range	[1..3]	hex	"0".."FFF"
LSNHi	upper bound of Linkage Set Number * range	[1..3]	hex	"0".."FFF"

6.8.6 IDLIMIT messages

The IDLIMIT messages shall be used to limit the number of identifiers of the same type which can be used simultaneously by the upstream entity. Limitation of identifier usage is allocated by the downstream entity. Allocation of identifiers can be done by means of IDALLOC messages (see clause 6.8.5).

IDLIMIT messages shall only be used when a resource data exchange session is open. Only one IDLIMIT message per identifier type shall be used in a resource data exchange session.

The IDLIMIT messages presented in table 33 are defined in the following clauses.

IDLIMIT DEF shall be issued by the downstream entity to indicate the limitation of identifier usage which is allocated to the upstream entity at the time the message is issued.

6.8.6.1 IDLIMIT DEF

IDLIMIT DEF shall be issued to indicate limitation of identifier usage which is allocated to the upstream entity at the time the message is issued.

NOTE 1: In case that identifier allocation has been done by the downstream entity by issuing IDALLOC DEF messages (see clause 6.8.5), the maximum number of the identifiers defined by IDLIMIT DEF cannot exceed the total number of corresponding identifiers of the same type defined by IDALLOC DEF messages.

NOTE 2: In case that no IDLIMIT DEF message has been provided for a specific type of identifier during a resource data exchange session, all allocated identifiers of that type can be used simultaneously by the upstream entity.

Figure 49 presents the IDLIMIT DEF message structure.

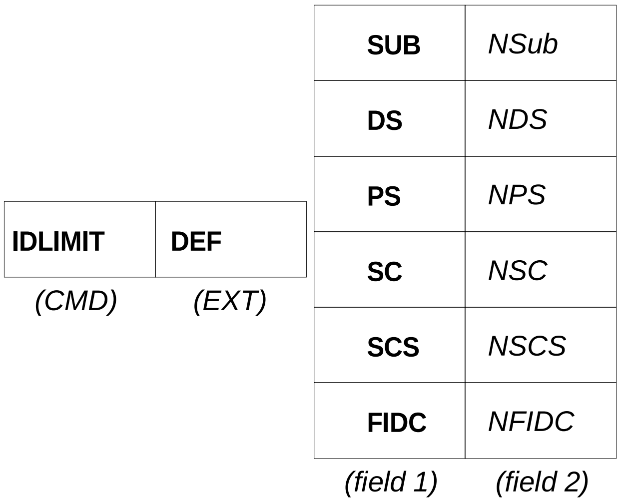

Figure 49: IDLIMIT DEF message structure

Table 39 presents the definition of the IDLIMIT DEF message fields. Fields length is expressed as a number of characters.

Table 39: IDLIMIT DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
SUB	Sub-Channel Identifiers concerned	3	string	"SUB"
NSub	maximum number of Sub-Channel Identifiers which can be used simultaneously	[1..2]	dec	"0".."64"
DS	Data Service Identifiers concerned	2	string	"DS"
NDS	maximum number of Data Service Identifiers which can be used simultaneously	[1..4]	dec	"0".."9999"
PS	Program Service Identifiers concerned	2	string	"PS"
NPS	maximum number of Program Service Identifiers which can be used simultaneously	[1..4]	dec	"0".."9999"
SC	Service Component Identifiers concerned	2	string	"SC"
NSC	maximum number of Service Component Identifiers which can be used simultaneously	[1..4]	dec	"0".."4096"
SCS	Service Components in Service concerned	3	string	"SCS"
NSCS	maximum number of Service Components in Service	[1..2]	dec	"1".."12"
FIDC	Fast information Data Channel Identifiers concerned	4	string	"FIDC"
NFIDC	maximum number of Fast Information Data Channel Identifiers which can be used simultaneously	[1..2]	dec	"0".."64"

6.8.7 PACKCON messages

The PACKCON messages shall be used to define the Packet Mode Contribution parameters allocated to the upstream entity by the downstream entity.

PACKCON messages shall only be used when a resource data exchange session is open.

The PACKCON messages are presented in table 33 and are defined in the following clauses.

PACKCON DEF shall be issued by the downstream entity to indicate a sub-channel identifier and packet address which are allocated to the upstream entity at the time the message is issued.

6.8.7.1 PACKCON DEF

Only the downstream entity shall issue the PACKCON DEF message.

PACKCON DEF shall be issued to indicate a sub-channel identifier and packet address which are allocated to the upstream entity at the time the message is issued.

The PACKCON DEF message shall only be issued when a resource data exchange session is open.

Figure 50 presents the PACKCON DEF message structure.

PACKCON	**DEF**	*PLen*	*PType*	*PLevel*	*SubChId*	*PA*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*	*(field 3)*	*(field 4)*	*(field 5)*

Figure 50: PACKCON DEF message structure

Table 40 presents the definition of the PACKCON DEF message fields. Fields length is expressed as a number of characters.

The fields marked (*) contain parameters defined in EN 300 401 [1]. Their values shall be compliant with those defined in EN 300 401 [1].

Table 40: PACKCON DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
PLen	Packet length* allocated	[1..2]	dec	"0", "24", "48", "72", "96" where "0" indicates that mixed packet lengths are used
PType	Protection type allocated*	1	char	"U" for UEP "E" for EEP
PLevel	Protection level* (and option*) allocated*	1 or 2	dec	"1".."5"for UEP *LO* for EEP, where *L* is protection level* "1".."4" *O* is protection option* "0".."7"
SubChId	Sub-channel identifier* allocated to the upstream entity	[1..2]	dec	"0".."63"
PA	Packet Address*	3	hex	"000".."3FF"

6.8.8 FIGBLCK messages

The FIGBLCK messages shall be used by the downstream entity to define restrictions on the upstream entity's entitlement for sending FIGs.

FIGBLCK messages shall only be used when a resource data exchange session is open. Only one FIGBLCK message per FIG type shall be used in a resource data exchange session.

The FIGBLCK messages are presented in table 33 and are defined in the following clauses.

FIGBLCK DEF shall be issued by the downstream entity to inform the upstream entity about a restricted set of FIGs which FIGs type and extension are blocked and which are transported from the time the message is issued.

In the case that the downstream entity does not issue FIGBLCK messages no FIGs shall be blocked.

6.8.8.1 FIGBLCK DEF

Only the downstream entity shall issue the FIGBLCK DEF message.

FIGBLCK DEF shall be issued by the downstream entity to inform the upstream entity about a restricted set of FIGs allocated to the upstream entity. FIGBLCK DEF defines which FIGs (identified by their type and extension) are blocked and which are transported by the downstream entity from the time the message is issued.

The FIGBLCK DEF message shall only be issued when a resource data exchange session is open.

Figure 51 presents the FIGBLCK DEF message structure.

FIGBLCK	**DEF**	*FIGType*	*FIGExt*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*

Figure 51: FIGBLCK DEF message structure

Type 7 (in-house) FIGs do not have an extension field defined (see EN 300 401 [1]). The FIGBLCK message only allows all type 7 FIGs to be blocked or no type 7 FIGs to be blocked. Therefore the FIGExt field shall be absent when the FIGType field has the value 7. However, even if the downstream entity blocks the use of type 7 FIGs, the special case of indicating an end-marker (see EN 300 401 [1]) is permitted.

Table 41 presents the definition of the FIGBLCK DEF message fields. Fields length is expressed as a number of characters.

The fields marked (*) contain parameters defined in EN 300 401 [1]. Their values shall be compliant with those defined in EN 300 401 [1].

Table 41: FIGBLCK DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
FIGType	FIG type* concerned	1	dec	"0", "1", "5", "6", "7"
FIGExt	Bit mask denoting entitlement for sending FIG of type *FIGType* and extension* corresponding to bit position if set, i.e. bit position b_n denotes extension n and val(b_n) = 1: extension blocked at the downstream entity val(b_n) = 0: extension not blocked at the downstream entity	[2..16]	hex	for *FIGType* = 0 (Length = 8): "00000000" ... "FFFFFFFF" for *FIGType* = 1 or 5 (Length = 2): "00" ... "FF" for *FIGType* = 6 (Length = 16): "0000000000000000" ... "FFFFFFFFFFFFFFF"
NOTE: The *FIGExt* field shall be absent when the *FIGType* field = 7.				

6.8.9 ANNSEND messages

The ANNSEND messages shall be used to define the entitlement for signalling announcements via a specified announcement cluster allocated to the upstream entity by the downstream entity.

ANNSEND messages shall only be used when a resource data exchange session is open.

The ANNSEND messages are presented in table 33 and are defined in the following clauses.

ANNSEND DEF shall be issued by the downstream entity to indicate a cluster identifier and related types of announcements which are allocated to the upstream entity at the time the message is issued.

6.8.9.1 ANNSEND DEF

Only the downstream entity shall issue the ANNSEND DEF message.

ANNSEND DEF shall be issued to indicate a cluster identifier and related types of announcements which are allocated to the upstream entity at the time the message is issued.

NOTE: The entitlement to send FIG of type = 0 and extension = 19 (Announcement switching) is a precondition for allocation of announcement cluster(s) by means of ANNSEND DEF. Therefore, ANNSEND DEF only provides useful information when FIG(0/19) is not blocked (see clause 6.8.8.1).

The ANNSEND DEF message shall only be issued when a resource data exchange session is open. Several ANNSEND DEF messages can be issued in a resource data exchange session, but no cluster identifier shall be addressed more than once.

Figure 52 presents the ANNSEND DEF message structure.

ANNSEND	**DEF**	*ClusterId*	*AnnType*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*

Figure 52: ANNSEND DEF message structure

Table 42 presents the definition of the ANNSEND DEF message fields. Fields length is expressed as a number of characters.

The fields marked (*) contain parameters defined in EN 300 401 [1]. Their values shall be compliant with those defined in EN 300 401 [1].

Table 42: ANNSEND DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
ClusterId	Announcement Cluster Identifier* allocated	2	hex	"01".."FF" Cluster Identifier* "00" and "FF" are predefined , see EN 300 401 [1]
AnnType	Announcement types* related to this cluster	4	hex	"0001" ... "FFFF" interpretation as for Asu flags in EN 300 401 [1]
NOTE: In general the ANNSEND DEF message permits a downstream entity to send announcements in FIG (0/19) only for those announcement types which are flagged in the field AnnType (together with the ClusterId indicated). However every downstream entity is free to send announcements with ClusterId '00' for all announcement types (alarm excluded). (ClusterId '00' is never included in the ANNSEND DEF message. It signals an announcement forming a programme item within the same service.)				

6.9 Information messages

The information messages shall be used to exchange information concerning capabilities and status of an STI connection.

For configurations and FIG files, information messages allow the upstream entity to monitor the maximum number available at the downstream entity as well as information concerning the ones currently defined.

A method is provided to allow the upstream entity to determine the signal path delay in the collection network.

The information messages are presented in table 43 and are defined in the following clauses.

Table 43: Information messages of the STI-C(LI)

INFORMATION MESSAGES				
CMD	***EXT***	***data fields***		
CONINFO	INF			
CONINFO	DEF	*MaxNum*	*UsedNum*	
CONNAME	INF			
CONNAME	DEF	*NRec*		
CONNAME	REC	*Name*		
CONNAME	END			
CONNAME	ERR	*Cause*		
FIGINFO	INF			
FIGINFO	DEF	*MaxNum*	*UsedNum*	
FIGNAME	INF			
FIGNAME	DEF	*NRec*		
FIGNAME	REC	*Name*		
FIGNAME	END			
FIGNAME	ERR	*Cause*		
COUNTER	INF			
COUNTER	DEF	*CIF count*	*UTC*	*DFCT*

6.9.1 General rules to use information messages

The information messages shall be used to exchange information concerning status and capabilities of an STI connection.

The CONINFO messages shall be used to determine how many configurations can be stored at the remote entity and how many are currently defined.

The CONNAME messages shall be used to determine the names of the configurations that are stored at the remote entity.

The FIGINFO messages shall be used to determine how many FIG files can be stored at the remote entity and how many are currently defined.

The FIGNAME messages shall be used to determine the names of the FIG files that are stored at the remote entity.

The COUNTER messages shall be used by the upstream entity to determine relationship between the CIF count, the DFCT and UTC.

6.9.2 CONINFO messages

The CONINFO messages shall be used to determine how many configurations can be stored at the remote entity and how many are currently defined.

The CONINFO messages are presented in table 43 and are defined in the following clauses.

CONINFO INF shall be issued to request the maximum number and used number of configurations available at the remote entity.

CONINFO DEF shall be issued to report the requested numbers.

6.9.2.1 CONINFO INF

CONINFO INF shall be issued to request the maximum number of configurations the remote entity is able to accept, and the number that are defined.

The entity receiving a CONINFO INF message shall reply with a CONINFO DEF message.

Figure 53 presents the CONINFO INF message structure.

CONINFO	**INF**
(CMD)	*(EXT)*

Figure 53: CONINFO INF message structure

6.9.2.2 CONINFO DEF

CONINFO DEF shall be issued to report the maximum number of configurations that the local entity is able to accept, and the number that are defined.

CONINFO DEF shall be issued as a response to a CONINFO INF request.

Figure 54 presents the CONINFO DEF message structure.

CONINFO	**DEF**	*MaxNum*	*UsedNum*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*

Figure 54: CONINFO DEF message structure

Table 44 presents the definition of the CONINFO DEF message fields. Fields length is expressed as a number of characters.

Table 44: CONINFO DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
MaxNum	Maximum number of configurations the entity is able to store	[1..2]	dec	"1".."99"
UsedNum	The number of configurations the entity currently has stored	[1..2]	dec	"0" ... *MaxNum*

6.9.3 CONNAME messages

The CONNAME messages shall be used to determine the names of the configurations that are stored at the remote entity.

The CONNAME messages are presented in table 43 and are defined in the following clauses.

CONNAME INF shall be issued to request the names of the configurations stored at the remote entity.

CONNAME DEF shall be issued to open a configuration name data exchange session to report the names of the configurations stored at the local entity.

CONNAME REC shall be issued to carry the name of one configuration.

CONNAME END shall be issued to close a configuration name data exchange session.

CONNAME ERR shall be issued to signal errors concerning a configuration name data exchange session.

6.9.3.1 CONNAME INF

CONNAME INF shall be issued to request the names of the configurations stored at the remote entity.

The entity receiving a CONNAME INF message shall open a configuration name data exchange session to transfer the names that are defined.

Figure 55 presents the CONNAME INF message structure.

CONNAME	**INF**
(CMD)	*(EXT)*

Figure 55: CONNAME INF message structure

6.9.3.2 CONNAME DEF

The CONNAME DEF message shall be issued to open a configuration name data exchange session.

The data field of the CONNAME DEF message shall provide the number of names to be transferred.

Following the issue of the CONNAME DEF message, CONNAME REC messages shall be issued to transfer the names and then close the data exchange session by issuing CONNAME END.

Figure 56 presents the CONNAME DEF message structure.

CONNAME	**DEF**	*NRec*
(CMD)	*(EXT)*	*(field 1)*

Figure 56: CONNAME DEF message structure

Table 45 presents the definition of the CONNAME DEF message fields. Fields length is expressed as a number of characters.

Table 45: CONNAME DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
NRec	number of configuration names to follow	[1..2]	dec	"0".."99"

6.9.3.3 CONNAME REC

The CONNAME REC message shall only be issued during an open configuration name data exchange session.

CONNAME REC messages shall be issued to transfer the names of the stored configurations.

Each CONNAME REC message shall carry one configuration name.

Figure 57 presents the CONNAME REC message structure.

CONNAME	**REC**	*Name*
(CMD)	*(EXT)*	*(field 1)*

Figure 57: CONNAME REC message structure

Table 46 presents the definition of the CONNAME REC message fields. Fields length is expressed as a number of characters.

Table 46: CONNAME REC message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Name	configuration name	[1..16]	string	the name of a configuration stored on the local entity

6.9.3.4 CONNAME END

CONNAME END message shall be issued to close a configuration name data exchange session.

Figure 58 presents the CONNAME END message structure.

CONNAME	**END**
(CMD)	*(EXT)*

Figure 58: CONNAME END message structure

6.9.3.5 CONNAME ERR

CONNAME ERR shall be issued to signal errors concerning a configuration name data exchange session.

Figure 59 presents the CONNAME ERR message structure.

CONNAME	**ERR**	*Cause*
(CMD)	*(EXT)*	*(field 1)*

Figure 59: CONNAME ERR message structure

Table 47 presents the definition of the CONNAME ERR message fields. Fields length is expressed as a number of characters.

Table 47: CONNAME ERR message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Cause	Type of error detected	4	string	"NREC" number of records is wrong

6.9.4 FIGINFO messages

The FIGINFO messages shall be used to determine how many FIG files can be stored at the remote entity and how many are currently defined.

The FIGINFO messages are presented in table 43 and are defined in the following clauses.

FIGINFO INF shall be issued to request the maximum number and used number of FIG files available at the remote entity.

FIGINFO DEF shall be issued to report the requested numbers.

6.9.4.1 FIGINFO INF

FIGINFO INF shall be issued to request the maximum number of FIG files the remote entity is able to accept, and the number that are defined.

The entity receiving a FIGINFO INF message shall reply with a FIGINFO DEF message.

Figure 60 presents the FIGINFO INF message structure.

FIGINFO	**INF**
(CMD)	*(EXT)*

Figure 60: FIGINFO INF message structure

6.9.4.2 FIGINFO DEF

FIGINFO DEF shall be issued to report the maximum number of FIG files that the local entity is able to accept, and the number that are defined.

FIGINFO DEF shall be issued as a response to a FIGINFO INF request.

Figure 61 presents the FIGINFO DEF message structure.

FIGINFO	**DEF**	*MaxNum*	*UsedNum*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*

Figure 61: FIGINFO DEF message structure

Table 48 presents the definition of the FIGINFO DEF message fields. Fields length is expressed as a number of characters.

Table 48: FIGINFO DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
MaxNum	Maximum number of FIG files the entity is able to store	[1..2]	dec	"1".."99"
UsedNum	The number of FIG files the entity currently has stored	[1..2]	dec	"0" ... *MaxNum*

6.9.5 FIGNAME messages

The FIGNAME messages shall be used to determine the names of the FIG files that are stored at the remote entity.

The FIGNAME messages are presented in table 43 and are defined in the following clauses.

FIGNAME INF shall be issued to request the names of the FIG files stored at the remote entity.

FIGNAME DEF shall be issued to open a FIG file name data exchange session to report the names of the FIG files stored at the local entity.

FIGNAME REC shall be issued to carry the name of one FIG file.

FIGNAME END shall be issued to close a FIG file name data exchange session.

FIGNAME ERR shall be issued to signal errors concerning FIG file name data exchange sessions.

6.9.5.1 FIGNAME INF

FIGNAME INF shall be issued to request the names of the FIG files stored at the remote entity.

The entity receiving a FIGNAME INF message shall open a FIG file name data exchange session to transfer the names that are defined.

Figure 62 presents the FIGNAME INF message structure.

FIGNAME	**INF**
(CMD)	*(EXT)*

Figure 62: FIGNAME INF message structure

6.9.5.2 FIGNAME DEF

The FIGNAME DEF message shall be issued to open a FIG file name data exchange session.

The data field of the FIGNAME DEF message shall provide the number of names to be transferred.

Following the issue of the FIGNAME DEF message, FIGNAME REC messages shall be issued to transfer the names and then close the data exchange session by issuing FIGNAME END.

Figure 63 presents the FIGNAME DEF message structure.

FIGNAME	**DEF**	*NRec*
(CMD)	*(EXT)*	*(field 1)*

Figure 63: FIGNAME DEF message structure

Table 49 presents the definition of the FIGNAME DEF message fields. Fields length is expressed as a number of characters.

Table 49: FIGNAME DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
NRec	number of FIG file names to follow	[1..2]	dec	"0".."99"

6.9.5.3 FIGNAME REC

The FIGNAME REC message shall only be issued during an open FIG file name data exchange session.

FIGNAME REC messages shall be issued to transfer the names of the stored FIG files.

Each FIGNAME REC message shall carry one FIG file name.

Figure 64 presents the FIGNAME REC message structure.

FIGNAME	**REC**	*Name*
(CMD)	*(EXT)*	*(field 1)*

Figure 64: FIGNAME REC message structure

Table 50 presents the definition of the FIGNAME REC message fields. Fields length is expressed as a number of characters.

Table 50: FIGNAME REC message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Name	FIG file name	[1..16]	string	the name of a FIG file stored on the local entity

6.9.5.4 FIGNAME END

FIGNAME END message shall be issued to close a FIG file name data exchange session.

Figure 65 presents the FIGNAME END message structure.

FIGNAME	**END**
(CMD)	*(EXT)*

Figure 65: FIGNAME END message structure

6.9.5.5 FIGNAME ERR

FIGNAME ERR shall be issued to signal errors concerning a FIG file name data exchange session.

Figure 66 presents the FIGNAME ERR message structure.

FIGNAME	**ERR**	*Cause*
(CMD)	*(EXT)*	*(field 1)*

Figure 66: FIGNAME ERR message structure

Table 51 presents the definition of the FIGNAME ERR message fields. Fields length is expressed as a number of characters.

Table 51: FIGNAME ERR message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Cause	Type of error detected	4	string	"NREC" number of records is wrong

6.9.6 COUNTER messages

The COUNTER messages shall be used by the upstream entity to determine the relationship between the CIF count, the DFCT and UTC.

NOTE: The relationship is only valid for synchronous data links.

The COUNTER messages are presented in table 43 and are defined in the following clauses.

COUNTER INF shall be issued by the upstream entity to request the counter relation.

COUNTER DEF shall be issued by the downstream entity to report the counter relation.

6.9.6.1 COUNTER INF

Only the upstream entity shall issue the COUNTER INF message.

COUNTER INF shall be issued to request the relationship between the CIF count, the DFCT and UTC.

Figure 67 presents the COUNTER INF message structure.

COUNTER	**INF**
(CMD)	*(EXT)*

Figure 67: COUNTER INF message structure

6.9.6.2 COUNTER DEF

Only the downstream entity shall issue the COUNTER DEF message.

COUNTER DEF shall be issued to return the relationship between the CIF count, the DFCT and UTC.

The reference data shall be one frame of either MSC sub-channel data or FIC FIB stream data for synchronous insertion carried in the STI-D(LI).

The frame reference given in the CIF count field shall be the CIF count for the transmission frame carrying the reference data. For transmission modes I and IV the reference data shall be carried in the first CIF of the transmission frame.

The time reference given in the UTC field shall be the time of transmission of the start of the null symbol in the transmission frame carrying the reference data.

The frame reference given in the DFCT field shall be the DFCT for the STI-D(LI) frame carrying the reference data.

For reconfigurations, COUNTER messages can be used to determine the delay between the output STI(PI, X) frame and the time its contents are part of the on-air DAB multiplex. In this case the *UTC* and *DFCT* fields shall provide the required information.

For FIB grids, COUNTER messages shall be used to determine the relationship between the Service providers DFCT and the CIF count for the transmission frame that shall carry the associated FIB data. In this case the *CIF count* and *DFCT* fields shall provide the required information.

Figure 68 presents the COUNTER DEF message structure.

COUNTER	**DEF**	*CIF count*	*UTC*	*DFCT*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*	*(field 3)*

Figure 68: COUNTER DEF message structure

Table 52 presents the definition of the COUNTER DEF message fields. Fields length is expressed as a number of characters.

Table 52: COUNTER DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
CIF count	CIF count carrying reference data	6	string	"UU:LLL" "UU" in "00".."19" for upper part, "LLL" in "000".."249" for lower part.
UTC	Long form time corresponding to the start of the null symbol of the transmission frame carrying the reference data	12	string	"HH:MM:SS:TTT" "HH" in "00".."23" for hours, "MM" in "00".."59" for minutes, "SS" in "00".."59" for seconds, "TTT" in "00".."999" for milliseconds.
DFCT	Data Frame Count carrying reference data	6	string	"UU:LLL" "UU" in "00".."19" for upper part, (DFCTH) "LLL" in "000".."249" for lower part. (DFCTL)

6.10 Supervision Messages

The supervision message class provides general error messages for signalling STI-C(LI) protocol errors and alarm messages to signal the current alarm status related to the STI-D(LI) information, PI errors and equipment errors.

The supervision messages are presented in table 53 and are defined in the following clauses.

Table 53: Supervision messages of the STI-C(LI)

SUPERVISION MESSAGES					
CMD	***EXT***	***data fields***			
PRERROR	GBG				
PRERROR	UKN	*Cmd*	*Ext*		
PRERROR	SYN	*Cmd*	*Ext*		
PRERROR	SEM	*Cmd*	*Ext*		
PRERROR	PRT	*Cmd*	*Ext*		
ALARMST	INF				
ALARMST	DEF	*State*	*UTC*		
STERROR	INF				
STERROR	DEF	*STID*	*TID*	*ErrType*	*DFCT*

6.10.1 General rules for the use of supervision messages

The supervision messages shall be used for supervision of the activity on the STI-C(LI) and for reporting the alarm status.

PRERROR messages shall be used to signal STI-C(LI) protocol errors.

ALARMST messages shall be used to request and signal the current alarm status.

STERROR messages shall be used to request and signal additional error information about individual streams.

6.10.2 PRERROR messages

The PRERROR messages shall be used to signal syntax, semantic or protocol errors in a received STI-C(LI) message, or to signal that a message contains garbage or an unknown command.

The PRERROR messages are presented in table 53 and are defined in the following clauses.

The PRERROR GBG message shall be issued to signal to the remote entity that a garbage message was received. A garbage message is defined as a message without the basic *CMD EXT* structure of the STI-C(LI).

The PRERROR UKN message shall be issued to signal to the remote entity that an unknown or unsupported command and/or extension was received.

The PRERROR SYN message shall be issued to signal a syntax error in a received message.

The PRERROR SEM message shall be issued to signal a semantic error in a received message.

The PRERROR PRT message shall be issued to signal a protocol violation caused by a received message.

6.10.2.1 PRERROR GBG

PRERROR GBG shall be issued to indicate that a garbage message was received.

A garbage message is defined as a message without the basic *CMD EXT* structure of the STI-C(LI).

Figure 69 presents the PRERROR GBG message structure.

PRERROR	**GBG**
(CMD)	*(EXT)*

Figure 69: PRERROR GBG message structure

6.10.2.2 PRERROR UKN

PRERROR UKN shall be issued to indicate the reception of a STI-C(LI) message where the command and/or command extension are unknown or unsupported.

Figure 70 presents the PRERROR UKN message structure.

PRERROR	**UKN**	*Cmd*	*Ext*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*

Figure 70: PRERROR UKN message structure

Table 54 presents the definition of the PRERROR UKN message fields. Fields length is expressed as a number of characters.

Table 54: PRERROR UKN message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Cmd	Command word of the unknown message	7	string	A string corresponding to a *CMD* field of a received message
Ext	Extension word of the unknown message	3	string	A string corresponding to an *EXT* field of a received message

6.10.2.3 PRERROR SYN

PRERROR SYN shall be issued to indicate reception of a STI-C(LI) message with one or more syntax errors.

A syntax error is defined as an error in the format of a *data field* in the received message such that it is not consistent with the definition provided in the present document (e.g. length or field type is wrong).

Figure 71 presents the PRERROR SYN message structure.

PRERROR	**SYN**	*Cmd*	*Ext*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*

Figure 71: PRERROR SYN message structure

Table 55 presents the definition of the PRERROR SYN message fields. Fields length is expressed as a number of characters.

Table 55: PRERROR SYN message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Cmd	Command word of the syntactically wrong message	7	string	A string corresponding to a *CMD* field of a received message
Ext	Extension word of the syntactically wrong message	3	string	A string corresponding to an *EXT* field of a received message

6.10.2.4 PRERROR SEM

PRERROR SEM shall be issued to indicate reception of a STI-C(LI) message with one or more semantic errors.

A semantic error is defined as an error in the value of a *data field* in the received message such that it is not consistent with the definition provided in the present document.

Figure 72 presents the PRERROR SEM message structure.

PRERROR	**SEM**	*Cmd*	*Ext*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*

Figure 72: PRERROR SEM message structure

Table 56 presents the definition of the PRERROR SEM message fields. Fields length is expressed as a number of characters.

Table 56: PRERROR SEM message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Cmd	Command word of the semantically wrong message	7	string	A string corresponding to a *CMD* field of a received message
Ext	Extension word of the semantically wrong message	3	string	A string corresponding to an *EXT* field of a received message

6.10.2.5 PRERROR PRT

PRERROR PRT shall be issued to indicate reception of a STI-C(LI) message that causes a protocol error.

A protocol error is defined as when a message is received out of the allowed sequence of commands (e.g. a SUBCHAN DEF message is received when there is no configuration data exchange session open).

Figure 73 presents the PRERROR PRT message structure.

PRERROR	PRT	*Cmd*	*Ext*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*

Figure 73: PRERROR PRT message structure

Table 57 presents the definition of the PRERROR PRT message fields. Fields length is expressed as a number of characters.

Table 57: PRERROR PRT message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
Cmd	Command word of the message causing the protocol error	7	string	A string corresponding to a *CMD* field of a received message
Ext	Extension word of the message causing the protocol error	3	string	A string corresponding to an *EXT* field of a received message

6.10.3 ALARMST messages

The ALARMST messages shall be used to request and signal the current alarm status related to the STI-D(LI). An alarm shall either be active, inactive or not available.

Five alarms are provided:

STI-D(LI) frame format errors: This alarm shall become active if frame format errors are detected in the STI-D(LI) frames (e.g. invalid CRCH, invalid CRCSTs, bad syntax in STC, bad increment of DFCT, etc.).

STI-D(LI) stream content errors: This alarm shall become active if invalid content of the streams carried in the STI-D(LI) is detected (e.g. audio stream invalid, the content of dynamic FIG stream invalid, etc.).

STI-D(LI) and STI-C(LI) inconsistency: This alarm shall become active if inconsistency between the configuration definition provided by the STI-C(LI) and the STI-D(LI) frame content is detected (e.g. stream missing in STI-D(LI) frame, stream TID is wrong, etc.).

Physical interface errors: This alarm shall become active if errors in the physical interface carrying the STI-D(LI) are detected (e.g. synchronization loss, bit/frame slip detected, etc.).

Equipment errors: This alarm shall become active if hardware or software malfunction is detected (e.g. the output interface in the downstream entity is broken).

NOTE: The conditions when to activate and deactivate each alarm are implementation dependent.

The ALARMST messages are presented in table 53 and are defined in the following clauses.

The ALARMST INF message shall be issued by the upstream entity to request the current alarm status.

The ALARMST DEF message shall be issued by the downstream entity to define the current alarm status if requested. The ALARMST DEF may also be issued by the downstream entity if the alarm status changes.

6.10.3.1 ALARMST INF

Only the upstream entity shall issue the ALARMST INF message.

The ALARMST INF message shall be issued to request the current alarm status.

The downstream entity shall respond with an ALARMST DEF message.

Figure 74 presents the ALARMST INF message structure.

ALARMST	**INF**
(CMD)	*(EXT)*

Figure 74: ALARMST INF message structure

6.10.3.2 ALARMST DEF

Only the downstream entity shall issue the ALARMST DEF message.

The ALARMST DEF message shall be issued in response to an ALARMST INF message.

The ALARMST DEF message may also be issued by the downstream entity to indicate that the alarm status has changed.

Figure 75 presents the ALARMST DEF message structure.

ALARMST	**DEF**	*State*	*UTC*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*

Figure 75: ALARMST DEF message structure

Table 58 presents the definition of the ALARMST DEF message fields. Field lengths are expressed as a number of characters.

Table 58: ALARMST DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
State	Current alarm status	5	string	coded as *FCIPE* where: *F* ("0","1" or "X") STI-D(LI) frame format errors *C* ("0","1" or "X") STI-D(LI) stream content errors *I* ("0","1" or "X") STI-D(LI) / STI-C(LI) inconsistency errors *P* ("0","1" or "X") physical interface errors *E* ("0","1" or "X") equipment malfunctions where: "0" signals that the alarm is inactive "1" signals that the alarm is active "X" signals that the alarm is not available
UTC	time that current alarm status began	8	string	"HH:MM:SS" "HH" in "00".."23" for hours, "MM" in "00".."59" for minutes, "SS" in "00".."59" for seconds.

6.10.4 STERROR messages

The STERROR messages shall be used to request and signal the current error status related to individual streams of the STI-D(LI).

The STERROR messages are presented in table 53 and are defined in the following clauses.

The STERROR INF message shall be issued by the upstream entity to request the current stream error status.

The STERROR DEF message shall be issued by the downstream entity to define the current stream error status if requested. The STERROR DEF may also be issued by the downstream entity if the stream error status changes.

6.10.4.1 STERROR INF

Only the upstream entity shall issue the STERROR INF message.

The STERROR INF message shall be issued to request the current stream error status.

The downstream entity shall respond by sending STERROR DEF messages.

Figure 76 presents the STERROR INF message structure.

STERROR	**INF**
(CMD)	*(EXT)*

Figure 76: STERROR INF message structure

6.10.4.2 STERROR DEF

Only the downstream entity shall issue the STERROR DEF message.

The STERROR DEF message shall be issued in response to a STERROR INF message.

The STERROR DEF message may also be issued by the downstream entity to indicate that a stream error status has changed.

NOTE: More than one STERROR DEF message may be required to fully describe the error status of a stream.

Figure 77 presents the STERROR DEF message structure.

STERROR	**DEF**	*STID*	*TID*	*ErrType*	*DFCT*
(CMD)	*(EXT)*	*(field 1)*	*(field 2)*	*(field 3)*	*(field 4)*

Figure 77: STERROR DEF message structure

Table 59 presents the definition of the STERROR DEF message fields. Field lengths are expressed as a number of characters.

Table 59: STERROR DEF message fields definition

Field Name	Meaning	Length	Field Type	Possible Value
STID	Identifier of the stream	3	hex	"000".."FFF"
TID	Type of data stream received	1	dec	"0".."7" (note: for a missing stream use 0)
ErrType	Type of error detected	1	char	"F" indicates a stream format error (i.e. CRC error) "C" indicates a stream contents error (i.e. bad FIG) "I" indicates a stream inconsistency error (i.e. stream missing, wrong bit rate) "N" indicates a stream without error
DFCT	Data frame count corresponding to change of error status for the stream.	6	string	"UU:LLL" "UU" in "00".."19" for upper part, "LLL" in "000".."249" for lower part

7 Transport Adaptation for the STI control part STI-C(TA)

This clause describes a transport adaptation for the STI control part to provide safe and reliable transportation of the STI-C(LI) between upstream and downstream entities. It is applicable to both the synchronous and asynchronous physical interfaces defined in clauses 9 and 10 respectively of the present document.

7.1 General structure

The STI-C(TA) shall be composed of a number of layers.

The layers of the STI-C(TA) shall be the data link layer, the network layer, the transport layer and the logical layer. The data link layer provides framing information and error detection. The network layer provides message routeing to permit control messages from different sources and destinations to be concentrated onto a single physical interface. The transport layer provides the means to repeat data or to resend data. The logical layer contains characters from the STI-C(LI).

Figure 78 shows the layered structure of the STI-C(TA).

Each STI-C(TA) frame shall contain a single C-TADATA field, which comprises a data link packet and/or padding. The total length of the C-TADATA field shall be CTL characters.

7.1.1 STI-C(TA) on synchronous physical links

For synchronous physical links the length of the C-TADATA field should be constant from frame to frame. The actual length chosen will depend on the error characteristics of the link and the amount of bandwidth available for carrying control messages. The data link packet has an overhead of 32 characters. Therefore the length of the C-TADATA field shall exceed 32 characters. The maximum length of the C-TADATA field shall be 256 characters. If there is no data link packet to send on a particular frame, or the length of the data link packet is less than the length of the C-TADATA field, then padding characters shall be used.

7.1.2 STI-C(TA) on asynchronous physical links

For asynchronous physical links the length of the C-TADATA field may vary. The actual length chosen will depend on the error characteristics of the link. The data link packet has an overhead of 32 characters. Therefore the length of each C-TADATA field shall exceed 32 characters. The maximum length of the C-TADATA field shall be 256 characters. Data link packets shall be sent whenever required. Padding characters should not be inserted.

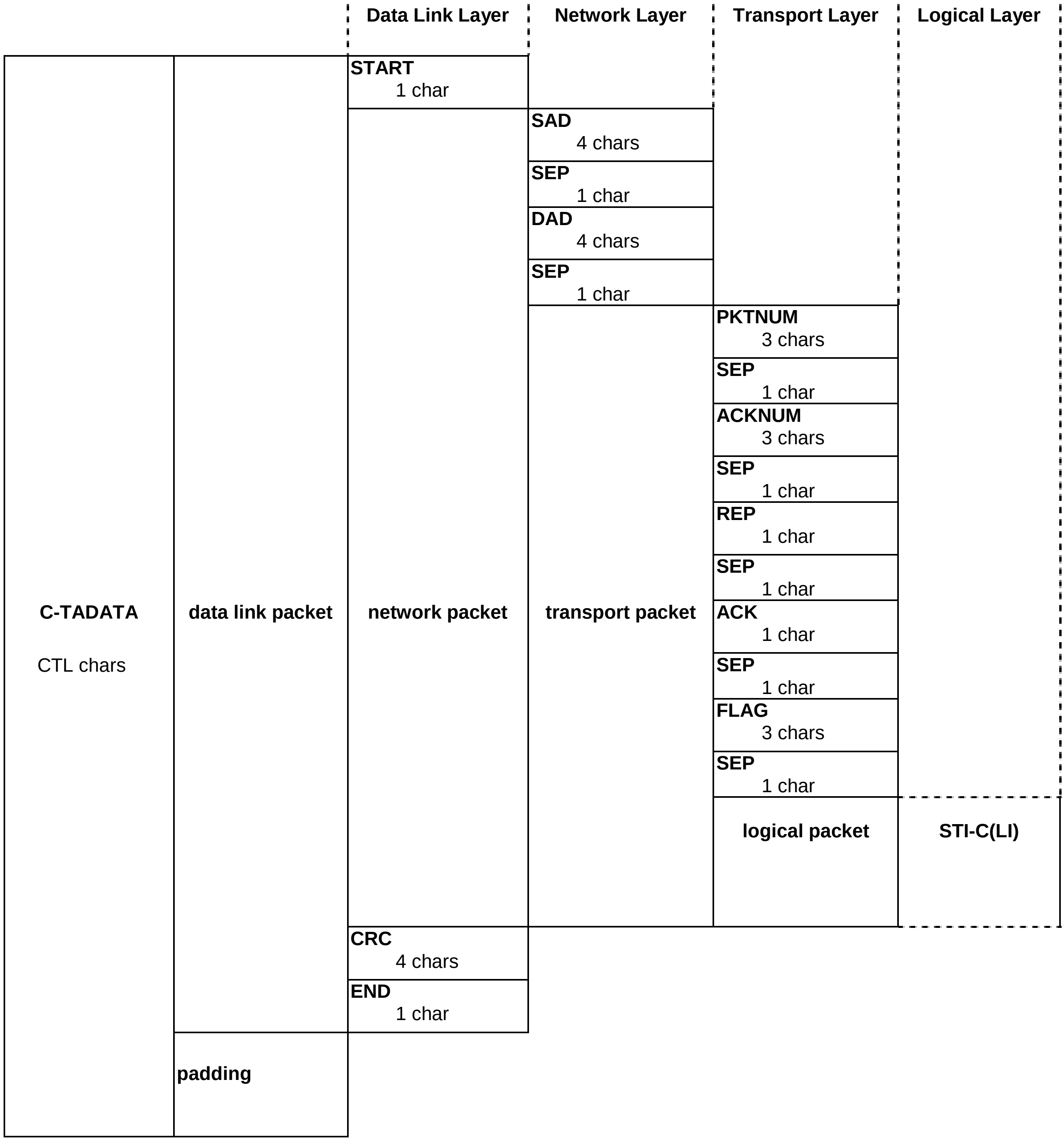

Figure 78: Structure of STI-C(TA)

7.2 The data link layer

The data link layer shall provide framing and error detection. The format of the data link packet shall be as given in figure 78.

There shall be no separator between the fields of the data link packet.

The length of the data link packet shall be in the range 32 to 256 characters.

7.2.1 Start field (START)

The START field shall indicate the start of a data link packet. It shall be coded as a single line feed character (bit combination 0/10).

7.2.2 Network packet

The network packet field shall contain the network packet as defined in clause 7.4.

7.2.3 Cyclic redundancy checksum field (CRC)

The CRC field shall carry the cyclic redundancy checksum calculated for the characters in the network packet. The CRC field shall contain four hexadecimal characters which represent the value of the calculated CRC.

The CRC calculation shall use the polynomial given in annex A.

7.2.4 End field (END)

The END field shall indicate the end of a data link packet. It shall be coded as a single carriage return character (bit combination 0/13).

7.2.5 Data link packet handling

7.2.5.1 Packet transmission

The data link layer shall calculate the CRC for the network packet using the polynomial given in annex A. The data link packet shall then be assembled as described in figure 78.

7.2.5.2 Packet reception

When a data link packet is detected, by receipt of a start field, the length of the packet shall be determined by counting characters until the end field is received. If no end field is received before the end of the STI-C(TA) frame, then the packet shall be discarded. The CRC shall be calculated for the network packet. The network packet shall be valid if the CRC matches that contained in the CRC field and the network packet shall be passed to the network layer. The packet shall be discarded by the data link layer if the CRC does not match.

7.3 Padding character

The padding character shall be the carriage return character (bit combination 0/13).

7.4 The network layer

The network layer shall permit the source and destination addresses to be indicated. The format of the network packet shall be as given in figure 78.

7.4.1 Source address field (SAD)

The SAD field shall contain a hexadecimal number, four characters in length.

The SAD field shall contain an identifier that uniquely defines the source of the network packet. For a packet originated by a Service provider it shall be the SPID and for a packet originated by an Ensemble provider it shall be the EPID.

7.4.2 Destination address field (DAD)

The DAD field shall contain a hexadecimal number, four characters in length.

The DAD field shall contain an identifier that uniquely defines the destination of the network packet. For a packet destined for a Service provider it shall be the SPID and for a packet destined for an Ensemble provider it shall be the EPID.

7.4.3 Transport packet

The transport packet field shall contain the transport packet as defined in clause 7.5.

7.4.4 Separator fields (SEP)

The SEP field shall contain one space character (bit combination 2/0).

It shall inserted between the fields of the network packet as given in figure 78.

7.4.5 Network packet handling

7.4.5.1 Packet transmission

The network layer shall assemble the network packet as described in figure 78 and pass it to the data link layer.

7.4.5.2 Packet reception

On receipt of a network packet from the data link layer the DAD field shall be examined. If the DAD field matches the recipient's unique identity (either SPID or EPID) then the network packet shall be passed on to the transport layer. The message shall be discarded by the network layer if the DAD field does not match. The SAD field provides the return address.

7.5 The transport layer

The transport layer provides safe and reliable transportation of the STI-C(LI) messages. The STI-C(LI) messages shall be divided into logical packets and each packet shall be numbered. The transport layer may repeat each logical packet a number of times, or may retransmit a packet which has not been acknowledged by the remote entity.

A transport packet shall carry a **payload** when it carries a logical packet or has its flag field set to "SYN" or "END".

7.5.1 Packet number (PKTNUM)

The PKTNUM field shall contain the logical packet number. The packet number shall be a modulo-1000 counter, coded as three decimal characters in the range "000" to "999".

The packet number shall be incremented by one for each new transport packet carrying a payload.

When the transport packet carries no payload, then the packet number shall be set to be equal to the packet number of the previous packet.

7.5.2 Acknowledge Number (ACKNUM)

The ACKNUM field shall contain the logical packet number that is next expected from the remote entity. It shall acknowledge that a continuous sequence of packets has been received of packet number less than acknowledge number. It shall be coded as three decimal characters in the range "000" to "999".

NOTE 1: The validity of the acknowledge number field depends on the value of the acknowledge field, see clause 7.5.4.

NOTE 2: Packets of higher value than acknowledge number may have been received, but they shall not be acknowledged unless in a continuous sequence. This is the mechanism that indicates packet loss to the remote entity.

7.5.3 Repetition Index (REP)

The REP field shall contain the packet repetition index. It shall be coded as a single decimal character in the range "0" to "9". The repetition index indicates the number of repetitions that shall follow.

7.5.4 Acknowledge field (ACK)

The ACK field shall carry an indication of whether the acknowledge number is valid or not. It shall be coded as a single character, "S" or "X".

The acknowledge field shall be set to "S" if the acknowledge number is valid. It shall be set to "X" if the acknowledge number is not valid.

7.5.5 Flag field (FLAG)

The FLAG field shall be used to open, continue or close a connection between upstream and downstream entities. It shall be coded as a string of three characters with the following possible values "SYN", "XXX" or "END".

The string "SYN" shall be used to open or re-synchronize a connection.

The string "END" shall be used to close a connection.

The string "XXX" shall be used in all other cases.

7.5.6 Logical packet

The logical packet field shall contain the logical packet from the logical layer, see clause 7.6. If there is no logical packet to send, then the transport layer serves to acknowledge the receipt of logical packets from the remote entity.

7.5.7 Separator fields (SEP)

The SEP field shall contain one space character (bit combination 2/0).

It shall inserted between the fields of the transport packet as given in figure 78.

7.5.8 Transport packet handling

The transport layer provides reliable data transfer by means of repeating and resending transport packets. To achieve this, each entity shall acknowledge transport packets carrying a payload, it receives from the remote entity. Transport packets which do not carry a payload shall not be acknowledged.

Each entity shall create a stack of transmitted transport packets such that they may be repeated or retransmitted as required. Packets containing a logical packet shall only be removed from the stack when they have been acknowledged or when the connection is closed or lost. Packets which do not contain a logical packet shall be removed when the repetition field reaches zero or when the connection is closed or lost.

The number of repeats should be determined according to the error performance of the physical link in use. The transport layer allows each transport packet to be transmitted up to ten times by use of the REP field.

Packet loss shall be detected in two ways. Firstly, by receipt of a packet in which the packet number is greater than that expected (i.e. that the packet number is not in sequence), and secondly, by no packet *that would acknowledge a previously sent packet* arriving within a time-out period determined by the implementation.

When packet loss is detected, the transmit packet stack shall be examined and the lowest numbered transport packet shall be removed from the stack and resent with the same packet number. The acknowledge number, acknowledge field and flag field shall be set according to their value at the time of re-transmission. The repetition field shall be used in order to resend the packet more than once if desired. This new transport packet shall be replaced onto the packet stack.

Annex G provides some examples of using the transport layer.

7.5.8.1 Opening a connection

To open a connection between entities, either at the start or to re-open a connection lost due to major errors, the three stage process defined in this clause shall be used.

Firstly, the entity wishing to open the connection sends a transport packet in which the flag field shall be set to "SYN". The acknowledge number is invalid and shall be set to an arbitrary value and the acknowledge field shall be set to "X". The packet number shall also be set to an arbitrary value. The REP is set to a suitable value for the link. No logical packet shall be present.

Secondly, the receiving entity shall provide an acknowledgement by sending a transport packet in return. The flag field shall be set to "SYN". The acknowledge number is valid and shall be set according to clause 7.5.2 and the acknowledge field shall be set to "S". The packet number shall be set to an arbitrary value. The REP is set to a suitable value for the link. No logical packet shall be present.

Finally, the connection is opened when the first entity (i.e. the one who originated the opening sequence) acknowledges the transport packet from the remote entity. This transport packet shall have the flag field set to "XXX". The acknowledge number shall be set as defined in clause 7.5.2 and the acknowledge field shall be set to "S". The packet number shall be set as defined in clause 7.5.1. The REP is set to a suitable value for the link. No logical packet shall be present.

The next transport packets from each entity shall use the respective packet numbers and acknowledge numbers which are now agreed and synchronized.

7.5.8.2 Closing a connection

To close a connection between entities, the three stage process defined in this clause shall be used.

Firstly, the entity wishing to close the connection sends a transport packet in which the flag field shall be set to "END". No logical packet shall be present. All other fields shall be coded in accordance with an established connection.

Secondly, the receiving entity shall provide an acknowledgement by sending a transport packet in return. If there is more data to send then closure shall be denied by returning a transport packet coded in accordance with an open connection, i.e. with the flag field set to "XXX". If the first entity still wishes to close the connection, the closure sequence shall be restarted.

If the receiving entity has no more data to send and also wishes to close the connection, the return packet will have the flag field set to "END". No logical packet shall be present. All other fields shall be coded in accordance with an open connection.

Finally, the connection is closed when the first entity (i.e. the one who originated the closure sequence) acknowledges the transport packet from the remote entity. This transport packet shall have the flag field set to "XXX". No logical packet shall be present. All other fields shall be coded in accordance with an open connection.

7.5.8.3 Transmission on an open connection

The transport layer shall assemble the transport packet as described in figure 78 and pass it to the network layer.

The acknowledge number shall be set equal to the highest received packet number in a continuous sequence incremented by one modulo-1000.

NOTE: If the sequence of received packet numbers is broken it indicates packet loss. The acknowledge number is an indication of the next new packet number expected to be received. It acknowledges all received packets with packet numbers less than acknowledge number modulo-1000.

The REP shall be set equal to the number of repeats which will follow. On each subsequent sending of the transport packet, the repetition field shall decrement by one until it is zero.

The acknowledge field shall be set to "S".

The flag field shall be set to "XXX".

7.5.8.4 Reception on an open connection

On receipt of a transport packet from the network layer, the transport layer shall examine the flag field. If it is set to "SYN" then the connection shall have been lost and the procedure described in clause 7.5.8.1 shall be followed to re-open the connection. If it is set to "END" then the procedure described in clause 7.5.8.2 shall be followed to close the connection. If the flag field is set to "XXX" then the following procedure shall be followed.

The acknowledge number and acknowledge field shall be examined, and all acknowledged packets shall be removed from the transmit packet stack. An acknowledged packet shall be removed even if its REP is non-zero.

The packet number shall be examined to determine if packet loss has occurred.

If the transport packet carries a payload, then the transport layer shall examine the packet number to determine if it is a new one or if it has been received already.

Logical packets shall be passed only once to the logical layer and the logical packet sequence shall be maintained.

If the logical packet has been received already it shall be discarded.

If it is the expected logical packet (i.e. the packet number is the next in the sequence) then it shall be passed to the logical layer.

If it is a new packet, but the packet number indicates that packet loss has occurred, then the new packet should be placed on a receive packet stack with its packet number but shall not be passed to the logical layer until the correct sequence of packets is available in the received packet stack.

7.6 The logical layer

The logical layer manages the transfer of messages between the STI-C(LI) and the transport layer.

7.6.1 STI-C(LI)

The messages of the STI-C(LI) shall be placed in sequence thus forming a character stream. The character stream shall be divided into logical packets. The number of characters in each logical packet may vary up to a maximum of 224 characters. The logical packets shall be transmitted in sequence by passing them to the transport layer. The transport layer will pass received logical packets to the logical layer in sequence.

7.6.2 Logical packet handling

7.6.2.1 Packet transmission

The logical layer shall assemble the logical packet by taking a number of characters in sequence from the character stream formed from the STI-C(LI) messages. The logical packet shall be passed to the transport layer.

7.6.2.2 Packet reception

On receipt of a logical packet from the transport layer, the logical layer shall assemble a character stream by appending the received logical packet to the previously received logical packet. The character stream so formed shall contain the STI-C(LI) messages.

8 Generic transport frame STI(PI, X)

This clause defines a generic transport frame structure to allow the transport of either STI-D(LI) or STI-C(TA) or both over synchronous and asynchronous links.

8.1 General

The generic STI(PI, X) frame structure is 24 ms based and provides synchronization and two containers that allow STI-D(LI) data and STI-C(TA) data to be carried. The generic transport frame structure, defined in this clause, is used by the physical interfaces defined for synchronous links in clause 9, and asynchronous links in clause 10.

The STI(PI, X) adaptation is presented in figure 79.

8.2 Adaptation of the logical layer

STI(PI, X) shall be organized as a uniform stream of bytes every 24 ms. The TFL, shall be the total number of bytes in the STI(PI, X) frame and shall be constant. The first STI(PI, X) byte of frame *p* shall be denoted as $B_{-8,p}$ and the last transmitted byte shall be $B_{TFL-9,p}$.

The STI(PI, X) shall consist of 5 fields: SYNC, TFH, DF, CF and FRPD.

The SYNC field shall carry frame synchronization and status information. The status information shall be derived from the STI-D(LI) ERR field.

The TFH field shall carry two fields, DFS and CFS indicating the allocated size for the DF and the CF.

NOTE: DFS and CFS shall be chosen to provide sufficient space for the intended use of the physical link. The values should not be changed since no additional check bits are provided to detect errors in these fields. Error detection may be implemented by checking that the values have been received consistently over a number of STI(PI, X) frames.

The DF field shall be inserted after the TFH field if the allocated size is greater than zero (i.e. DFS ≠ 0) and carries STI-D(LI) data. The DF field shall contain the D-LIDATA field and optional padding bytes (DFPD). The first bit of the D-LIDATA field shall be inserted at $B_{0,p}(b_0)$.

The CF shall be inserted at $B_{DFS,p}(b_0)$ when the allocated size is greater than zero (i.e. CFS ≠ 0). The CF field shall contain one or more C-TADATA fields, see clause 7.

Any unused tail bytes of the STI(PI, X) frame shall be filled with padding bytes (FRPD), see clause 8.2.5.

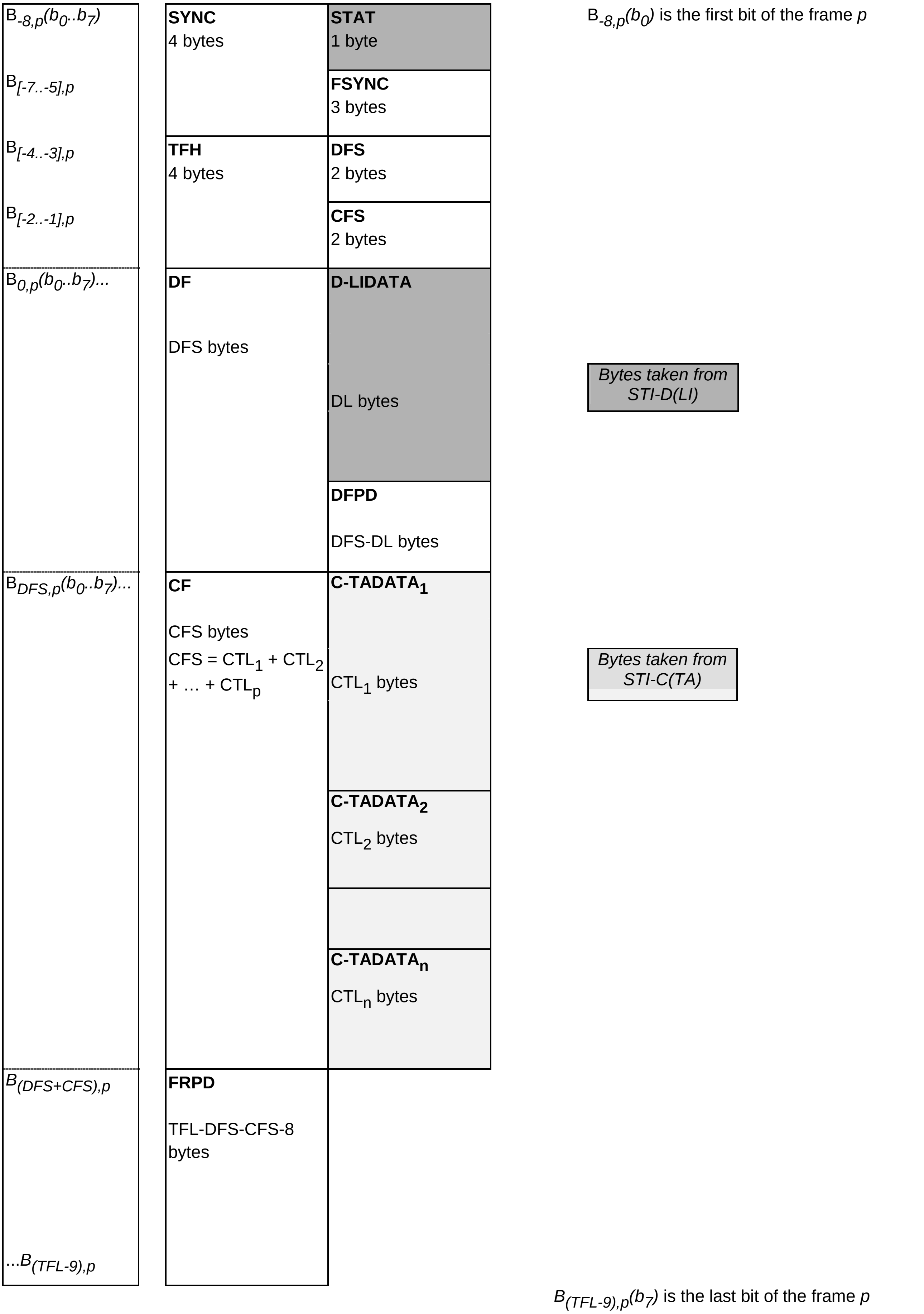

Figure 79: Mapping of STI-D(LI) and STI-C(TA) to the STI(PI, X)

8.2.1 Synchronization field (SYNC)

The SYNC field shall contain two data fields, an eight-bit STI-D(LI) ERR field and a 24-bit field for frame synchronization.

8.2.1.1 Error field (ERR)

Byte $B_{-8,p}$ shall carry the STI-D(LI) ERR field.

The STI(PI, X) receiver can modify the ERR field to indicate the error status associated with the use of the STI(PI, X) layer.

The ERR field shall take one of four levels, 0 to 3, as defined in clause 5.2.

In addition, the level of ERR can be increased by the STI(PI, X) receiver when CRC violations are detected in the D-LIDATA according to the rules given in table 60.

Table 60: Setting of error levels at the PI Layer

CRC violated	Action
None	Current error level retained
MST only	Error level may be increased to 1
EOH only	Error level may be increased to 2
MST and EOH	Error level may be increased to 3

The STI(PI, X) receiving equipment shall not decrease the error level of the ERR field.

When no DF field is present in the STI(PI, X), i.e. DFS = 0, the ERR field has no significance and shall be ignored by the receiving equipment.

8.2.1.2 Frame synchronization field (FSYNC)

Bytes $B_{[-7..-5],p}$ shall carry 24 ms FSYNC bits. FSYNC shall be one's complemented on successive frames between the two patterns $1F90CA_{16}$ and $E06F35_{16}$. The byte values for FSYNC are given in table 61.

Table 61: FSYNC definition for STI(PI, X)

FSYNC bytes	FSYNC0	FSYNC1
$B_{-7,p}(b_0..b_7)$	$1F_{16}$	$E0_{16}$
$B_{-6,p}(b_0..b_7)$	90_{16}	$6F_{16}$
$B_{-5,p}(b_0..b_7)$	CA_{16}	35_{16}

With FSYNC0 set to $1F90CA_{16}$ and FSYNC1 set to $E06F35_{16}$, STI(PI, X) synchronization should be obtained when either:

FSYNC0 is present in frame p;

AND FSYNC1 is present in frame p + 1;

AND FSYNC0 is present in frame p + 2;

or:

FSYNC1 is present in frame p;

AND FSYNC0 is present in frame p + 1;

AND FSYNC1 is present in frame p + 2.

Synchronization should be lost if two consecutive synchronization words are incorrectly received.

8.2.2 Transport frame header field (TFH)

The TFH field shall contain two data fields, a sixteen bit data frame size field and a sixteen bit control frame size field, DFS and CFS. These two fields shall be used to indicate the length in bytes of the following DF and CF fields. The following equation shall always be fulfilled:

$$8 + DFS + CFS \leq TFL$$

where TFL is the length of the transport frame, and is specific to each STI(PI, X) interface.

8.2.2.1 Data frame size field (DFS)

The DFS field shall indicate the length of the DF field. The length shall be indicated by a 16-bit number which shall give the total number of bytes carried in the DF field. When the DFS field is set to zero, it shall indicate that no DF field is present in the STI(PI, X), i.e. no D-LIDATA field is carried in the interface.

The DFS field shall be carried in byte $B_{-4,p}(b_0.. b_7)$ and $B_{-3,p}(b_0.. b_7)$.

8.2.2.2 Control frame size field (CFS)

The CFS field shall indicate the length of the CF field. The length shall be indicated by a 16-bit number which shall give the total number of bytes carried in the CF field. When the CFS field is set to zero, it shall indicate that no CF field is present in the STI(PI, X), i.e. no C-TADATA is carried in the interface.

The CFS field shall be carried in byte $B_{-2,p}(b_0.. b_7)$ and $B_{-1,p}(b_0.. b_7)$.

8.2.3 Data frame field (DF)

When present, the DF field shall contain two fields, the D-LIDATA field and an optional DFPD field.

8.2.3.1 STI-D(LI) data field (D-LIDATA)

The D-LIDATA field of the STI-D(LI) frame *p*, from $B_{0,p}$ to $B_{(DL-1),p}$, shall be inserted in the STI(PI, X) frame p into the bytes $B_{0,p}$ to $B_{(DL-1),p}$.

8.2.3.2 Data frame padding field (DFPD)

The padding information should be inserted after the D-LIDATA field if required to fill up the allocated size of the DF field as given by DFS. The value of the padding bytes shall be 55_{16}.

The DFPD field shall be inserted into bytes $B_{DL,p}$ to $B_{(DFS-1),p}$.

8.2.4 Control frame field (CF)

When present, the CF field carries one or more C-TADATA fields as defined in clause 7. The C-TADATA fields shall be carried in bytes $B_{DFS,p}$ to $B_{(DFS+CFS-1),p}$.

8.2.5 Frame padding field (FRPD)

The padding information should be inserted at the end of the STI(PI, X) frame. The value of the padding bytes shall be 55_{16}.

The FRPD field shall be inserted into bytes $B_{(DFS+CFS),p}$ to $B_{(TFL-9),p}$.

FRPD may be used to carry user specific data. The format and protocol used in this case are not subject to standardization. Moreover, the FRPD field can be modified in case of further adaptation or cascading of equipment.

9 Physical Interfaces for synchronous links

9.1 G.703 interfaces, STI(PI, G.703)

9.1.1 General description

The purpose of the STI(PI, G.703) is to provide a physical form to the STI-D(LI) and STI-C(TA) for local connections and test purposes.

STI(PI, G.703) uses G.703-HDB3 line coding (see ITU-T Recommendation G.703 [4]), carrying data and clock on the same 2 048 kbit/s serial connection. No additional protection is provided by this STI(PI, G.703) physical adaptation.

STI(PI, G.703) should not be applied directly to standard telecommunication networks. STI(PI, G.704/1) or STI(PI, G.704/2) described later should be used instead. The principal difficulty, apart from the lack of error protection, is that network monitoring equipment may interpret any data field containing long strings of ones as an Alarm Indication Signal (AIS). The network monitoring may require disabling if the STI(PI, G.703) is to be passed.

9.1.2 Adaptation of the STI(PI, X) to the STI(PI, G.703)

STI(PI, G.703) shall use the generic transport frame structure defined in clause 8. Each STI(PI, G.703) frame shall carry 6 144 bytes of STI(PI, X) data, $B_{[-8..6\ 135]}$, which represents a net bitrate of 2 048 kbit/s. The length of the STI(PI, X) frame, TFL, shall be 6 144 bytes.

9.1.3 Physical interface

The physical characteristics of STI(PI, G.703) shall conform to the requirement of the ITU-T Recommendation G.703 [4] for 2 048 kbit/s interfaces. The minimum requirement shall be a 75 Ω female BNC connector fitted to the equipment.

9.2 V.11 interface, STI(PI, V.11)

9.2.1 General description

The purpose of STI(PI, V.11) is to provide a physical interface suitable for local connections, or distant connections via telecommunication networks or through modems using ITU-T Recommendations X.24 [5] and V.11 [6] interfaces.

The STI(PI, V.11) interface offers a junction having a net bitrate of N × 8 kbit/s. Clock and data signals are produced separately.

No specific protection shall be provided by the STI(PI, V.11) adaptation layer.

9.2.2 Adaptation of the STI(PI, X) to the STI(PI, V.11)

STI(PI, V.11) shall use the generic transport frame structure defined in clause 8. Each STI(PI, V.11) frame shall contain N × 24 bytes of STI(PI, X) data, $B_{[-8..N \times 24-9]}$, which represents a net bitrate of N × 8 kbit/s. For example, for N = 32, 768 bytes per 24 ms (or 256 kbit/s) are available. For a given N, the length of the STI(PI, X) frame, TFL, shall be N × 24 bytes.

9.2.3 Physical interface

The physical characteristics of STI(PI, V.11) shall comply with the general requirements of ITU-T Recommendations V.11 [6] and X.24 [5] and shall have the following specific attributes:

- equipment connector: D-Sub 15, female, connections as defined in table 62;
- circuit connections: X.24, clock and data only;
- clock/data timing: as defined in ITU-T Recommendation X.24 [5];
- electrical levels: as defined in ITU-T Recommendation V.11 [6];
- bitrate: N × 8 kbit/s, N shall be chosen to provide sufficient network capacity to exceed the maximum envisaged size of the STI-D(LI) and/or STI-C(TA).

Table 62: Pin allocations for STI(PI, V.11)

Pin number	Signal			Pin number	Signal		
1	Frame ground	FG					
2	Transmit data +	T+	(output)	9	Transmit data -	T-	(output)
3	Transmit clock +	X+	(output)	10	Transmit clock -	X-	(output)
4	Receive data +	R+	(input)	11	Receive data -	R-	(input)
5	n/c			12	n/c		
6	Receive clock +	S+	(input)	13	Receive clock -	S-	(input)
7	n/c			14	n/c		
8	Signal ground	SG		15	n/c		
NOTE: n/c = no connection to pin.							

The Transmit clock circuit should be the exact echo of the Receive clock circuit.

9.3 WG1/WG2 interface, STI(PI, WG1/2)

9.3.1 General description

The purpose of STI(PI, WG1/2) is to provide a physical interface suitable for local connections.

STI(PI, WG1/2) shall use one slot of the WG1/2 interface. It has a fixed bitrate of 384 kbit/s. Clock and data signals are produced separately. The WG1/2 interface can carry up to 16 STI(PI, WG1/2) frames simultaneously.

No specific protection shall be provided by the STI(PI, WG1/2) adaptation layer.

9.3.2 Adaptation of the STI(PI, X) to the STI(PI, WG1/2)

STI(PI, WG1/2) shall use the generic transport frame structure defined in clause 8. Each STI(PI, WG1/2) frame shall carry 1 152 bytes of STI(PI, X) data, $B_{[-8..1\ 143]}$, which represents a net bitrate of 384 kbit/s. The length of the STI(PI, X) frame, TFL, shall be 1 152 bytes.

If STI(PI, WG1/2) carries STI-D(LI) data, the STI-D(LI) shall always carry eight individual data streams, although all may be empty. Therefore the NST field shall contain the value 8.

9.3.3 Adaptation to the WG1/2 frame structure

Bit $B_{44,p}(b_0)$ of the STI(PI, WG1/2) frame shall be inserted as the first bit of the WG1/2 slot in use. This bit is identified by the hardware synchronization signal RCCSY. This bit may be the first bit of the first data stream of the STI-D(LI) frame (i.e. when DFS ≠ 0), as shown in figure 80.

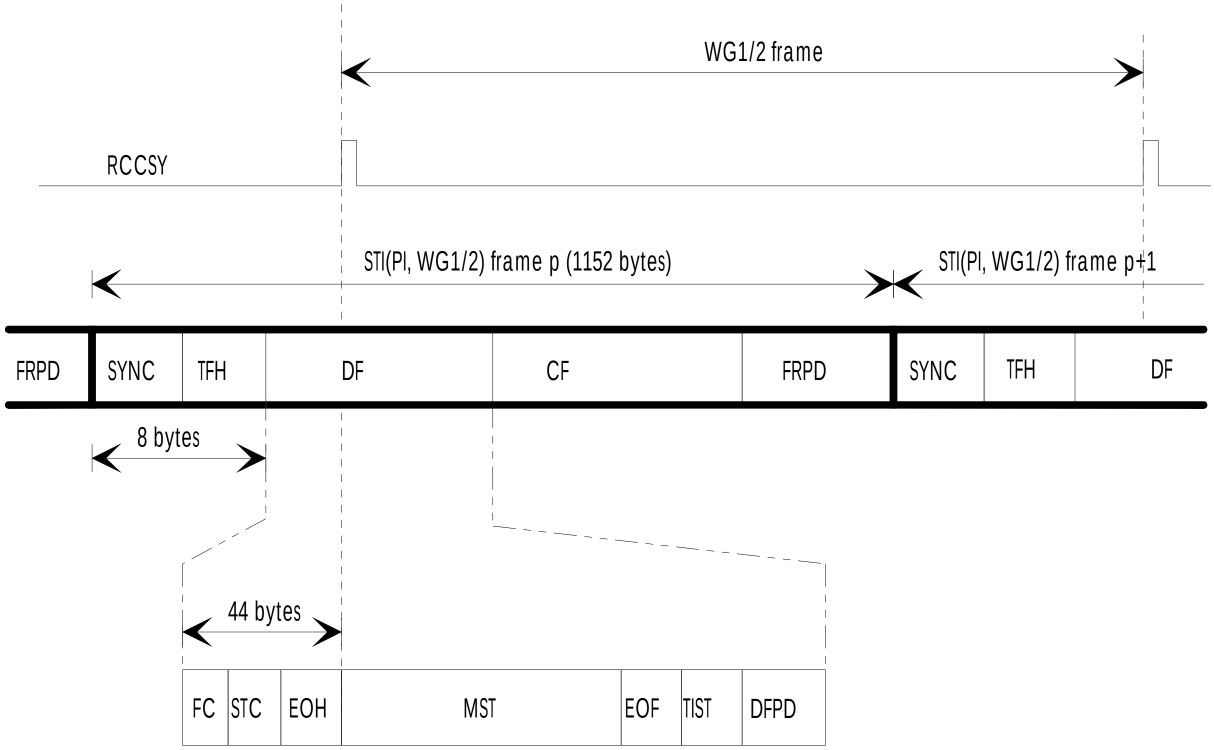

Figure 80: Adaptation of the STI(PI, WG1/2) frame to the WG1/2 frame

9.3.4 Physical interface

The physical characteristics of STI(PI, WG1/2) shall conform to the definition given in annex C.

9.4 IEC 60958 interface, STI(PI, IEC 60958)

9.4.1 General description

The purpose of STI(PI, IEC 60958) is to provide a physical interface suitable for local connections.

STI(PI, IEC 60958) shall use one complete IEC 60958 audio frame which has a gross bitrate of 3 072 kbit/s.

No specific protection shall be provided by the STI(PI, IEC 60958) adaptation layer.

9.4.2 Adaptation of the STI(PI, X) to the STI(PI, IEC 60958)

STI(PI, IEC 60958) shall use the generic transport frame structure defined in clause 8. Each STI(PI, IEC 60958) frame shall carry 4 608 bytes of STI(PI, X) data, $B_{[-8..4\,599]}$, which represents a net bitrate of 1 536 kbit/s. The length of the STI(PI, X) frame, TFL, shall be 4 608 bytes.

9.4.3 Adaptation to the IEC 60958 frame structure

The physical adaptation of the STI(PI, X) frame structure into the IEC 60958 interface is shown in figure 81. Timeslots 12 to 27 in the IEC 60958 sub frame are used to carry the STI(PI, X). There is no specific relationship between the IEC 60958 block, frame and sub frame and the STI(PI, IEC 60958) 24 ms frame.

The Validity bit (V) shall be set to 1.

The Channel Status bits shall be used as summarized in table 63.

Table 63: Setting of channel status bits

Byte	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7
0	1	1	0	0	0	0	0	1
1	1	0	0	0	x	x	x	x
2	0	0	0	1	0	0	0	0
NOTE 1: The value x indicates that these bits are not relevant for this interface and are defined in the IEC 60958 [13]. NOTE 2: Byte 1, bit 0..3 indicates DAB/STI mode.								

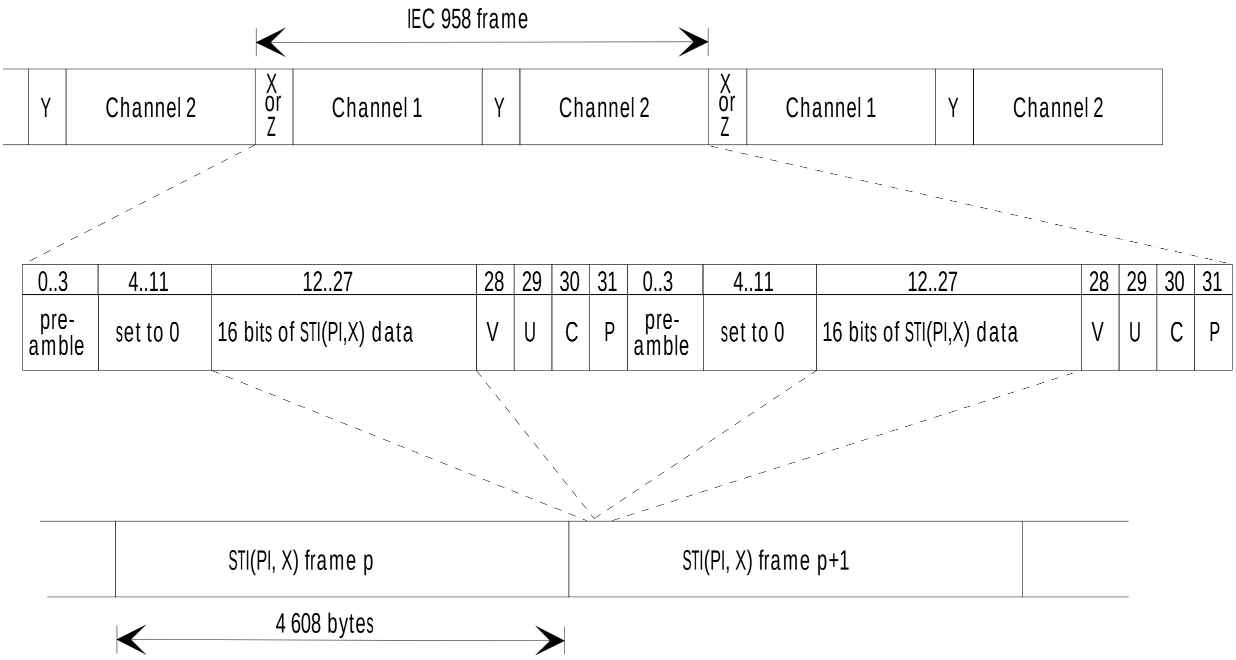

Figure 81: Physical adaptation of STI in the IEC 60958 frame

9.4.4 Physical interface

The physical characteristics of STI(PI, IEC 60958) shall comply with the general requirements of IEC 60958-2 [13].

9.5 G.704 interface with error protection, STI(PI, G.704/1)

9.5.1 General description

The STI(PI, G.704/1) is suitable for use on networks based on the first level of the PDH (2 048 kbit/s) as defined in ITU-T Recommendation G.704 [7]. As well as catering for the reserved G.704 signalling and synchronization bytes, STI(PI, G.704/1) includes time stamps to permit compensation for the effect of network delays. It also uses Reed-Solomon forward error coding to allow transport network errors to be corrected.

NOTE 1: The STI(PI, G.704/1) uses the same adaptation as ETI(NA, G.704), defined in ETS 300 799 [3].

NOTE 2: Annex H outlines a method of adapting STI(PI, G.704/1) for use in countries whose PDH networks use a first level of 1 544 kbit/s.

STI(PI, G.704/1) has two variants: STI(PI, G.704/1)$_{5\,592}$ and STI(PI, G.704/1)$_{5\,376}$. The two differ in the balance between their data capacity and the number of bytes which are reserved for forward error correction codes.

In the G.704 frame structure, 1 920 kbit/s are available to carry user data, the remaining 128 kbit/s are reserved by the G.704 network for its framing, signalling and monitoring bytes.

In each 24 ms multiframe, STI(PI, G.704/1) $_{5\,592}$, has the capacity to carry 5 760 bytes allocated as follows:

- 5 592 bytes of STI(PI, X) data (a data rate of 1 864 kbit/s);
- 120 bytes (40 kbit/s) for forward error correction;
- 48 bytes (16 kbit/s) for management and signalling.

In each 24 ms multiframe, STI(PI, G.704/1) $_{5\,376}$, has the capacity to carry 5 760 bytes allocated as follows:

- 5 376 bytes of STI(PI, X) data (a data rate of 1 792 kbit/s);
- 336 bytes (112 kbit/s) for forward error correction;
- 48 bytes (16 kbit/s) for management and signalling.

9.5.2 Transparency of STI(PI, G.704/1) layer to STI-D(LI)

9.5.2.1 Transparency of STI(PI, G.704/1) $_{5\,592}$ layer to STI(PI, X)

STI(PI, G.704/1) $_{5\,592}$ is able to carry up to 5 592 bytes of STI(PI, X) data, $B_{[-4..5\,587]}$. The length of the STI(PI, X) frame, TFL, shall be 5 596 bytes.

NOTE: The SYNC field is not carried by this physical interface.

STI(PI, G.704/1)$_{5\,592}$ is also able to carry the STI-D(LI) ERR field but the content of this field may be amended by STI(PI, G.704/1) receiving equipment.

9.5.2.2 Transparency of STI(PI, G.704/1)$_{5\,376}$ layer to STI(PI, X)

STI(PI, G.704/1)$_{5\,376}$ is able to carry up to 5 376 bytes of STI(PI, X) data, $B_{[-4..5\,371]}$. The length of the STI(PI, X) frame, TFL, shall be 5 380 bytes.

NOTE: The SYNC field is not carried by this physical interface.

STI(PI, G.704/1)$_{5\,376}$ is also able to carry the STI-D(LI) ERR field but the content of this field may be amended by STI(PI, G.704/1) receiving equipment.

9.5.3 STI(PI, G.704/1) structure

The purpose of the STI(PI, G.704/1) multiframe structure is to map the STI(PI, X) frame onto the G.704 frame structure. In the ITU-T Recommendation G.704 [7], the 2 048 kbit/s data stream is organized into frames. Each frame has a nominal duration of 125 µs and is made up of 256 bits organized into 32 timeslots. Each timeslot carries one byte of data or frame management information. Two of the timeslots are reserved for G.704 synchronization and signalling. The remaining 30 timeslots are available to carry user data.

The STI(PI, G.704/1) multiframe shall have a FL of 24 ms. It shall consist of 192 G.704 frames, equivalent to 6 144 timeslots, or bytes.

NOTE: This text uses the word "byte" where a strict adherence to G.704 terminology would require the use of "timeslot".

The multiframe structure is illustrated in figure 82 and shall consist of:

- 3 superblocks ($s_{[0..2]}$) of 2 048 bytes each;
- each superblock shall consist of 8 blocks ($bl_{[0..7]}$) of 256 bytes each;
- each block shall consist of 8 G.704 frames ($f_{[0..7]}$) of 32 bytes each;
- each G.704 frame shall consist of 32 timeslots ($ts_{[0..31]}$) of 1 byte each;
- each timeslot shall consist of 8 bits ($b_{[0..7]}$).

Table 64 summarizes the relation of elements within a multiframe.

Table 64: Relation of elements within a multiframe

	Superblocks	Blocks	G.704 frames	Timeslots	Bits
Multiframe	3	24	192	6 144	49 152
Superblock	1	8	64	2 048	16 384
Block		1	8	256	2 048
G.704 frame			1	32	256
Timeslot				1	8

9.5.3.1 G.704 reserved bytes

The following bytes of an STI(PI, G.704/1) multiframe shall be reserved for G.704 frame control:

- ts_0 in each G.704 frame for the G.704 synchronization byte, G_0;
- ts_{16} in each G.704 frame for the G.704 supervision byte, G_1.

9.5.3.2 STI(PI, G.704/1) reserved bytes

The following bytes of each multiframe are reserved by the STI(PI, G.704/1) layer:

- ts_1 of f_0 in each block for a multiframe management byte, $M_{bl,s}$;
- ts_2 of f_0 in each block for a multiframe supervision byte, $S_{bl,s}$;

where *bl* and *s* shall be the block and superblock numbers of the multiframe. Each multiframe shall have 24 management and 24 supervision bytes.

9.5.3.2.1 Multiframe management byte, $M_{bl,s}$

The bits of $M_{bl,s}$ shall be assigned as follows:

- $M_{bl,s}(b_{0..2})$: a block counter containing the binary value *bl* (MSb in b_0 and LSb in b_2);
- $M_{bl,s}(b_{3,4})$: a superblock counter containing the binary value *s* (MSb in b_3 and LSb in b_4);
- $M_{bl,s}(b_5)$: a timestamp bit;
- $M_{bl,s}(b_6)$: a frame signalling bit;
- $M_{bl,s}(b_7)$: Rfa.

The block and superblock counters should be used for synchronization of the STI(PI, G.704/1) multiframe. The synchronization method shall not assume that there is any fixed relationship between the G.704 synchronization byte, G_0, and the multiframe synchronization counters.

During the course of a multiframe, the 24-bit word formed by the timestamp bits carried in $M_{bl,s}(b_5)$ shall form a word carrying time information relevant to that frame. The MSb of the word shall be carried in $M_{0,0}(b_5)$ whilst the LSb shall be found in $M_{7,2}(b_5)$.

Information on the coding and use of timestamp data is given in annex B.

The 24-bit word formed by the signalling bits in each frame, shall carry signalling information. The MSb of the signalling word shall be carried in $M_{0,0}(b_6)$ whilst the LSb shall be found in $M_{7,2}(b_6)$. Table 65 summarizes the use of signalling functions.

Table 65: Use of multiframe signalling bits

Bytes $M_{x,0}$	**b_6**	**Signalled information**
$M_{0,0}$	1	STI-D(LI) layer input contains CRC violations
	0	No CRC violations in the STI-D(LI) layer input
$M_{1,0}$	1	STI(PI, G.704/1)$_{5\ 376}$ in use
	0	STI(PI, G.704/1)$_{5\ 592}$ in use
$M_{[2..7],0}$	0	Rfa
Bytes $M_{x,1}$		
$M_{0,1}$	x	First bit, b_0, of STI-D(LI) ERR field
$M_{[1..6],1}$	x	Bits $b_1..b_6$ of STI-D(LI) ERR field
$M_{7,1}$	x	Last bit, b_7, of STI-D(LI) ERR field
Bytes $M_{x,2}$		
$M_{[0..7],2}$	0	Rfa
NOTE: $M_{0,0}$ and $M_{[0..7],1}$ has no significance when DFS = 0 (i.e. no DF field present) in the STI(PI, X) frame and shall be ignored.		

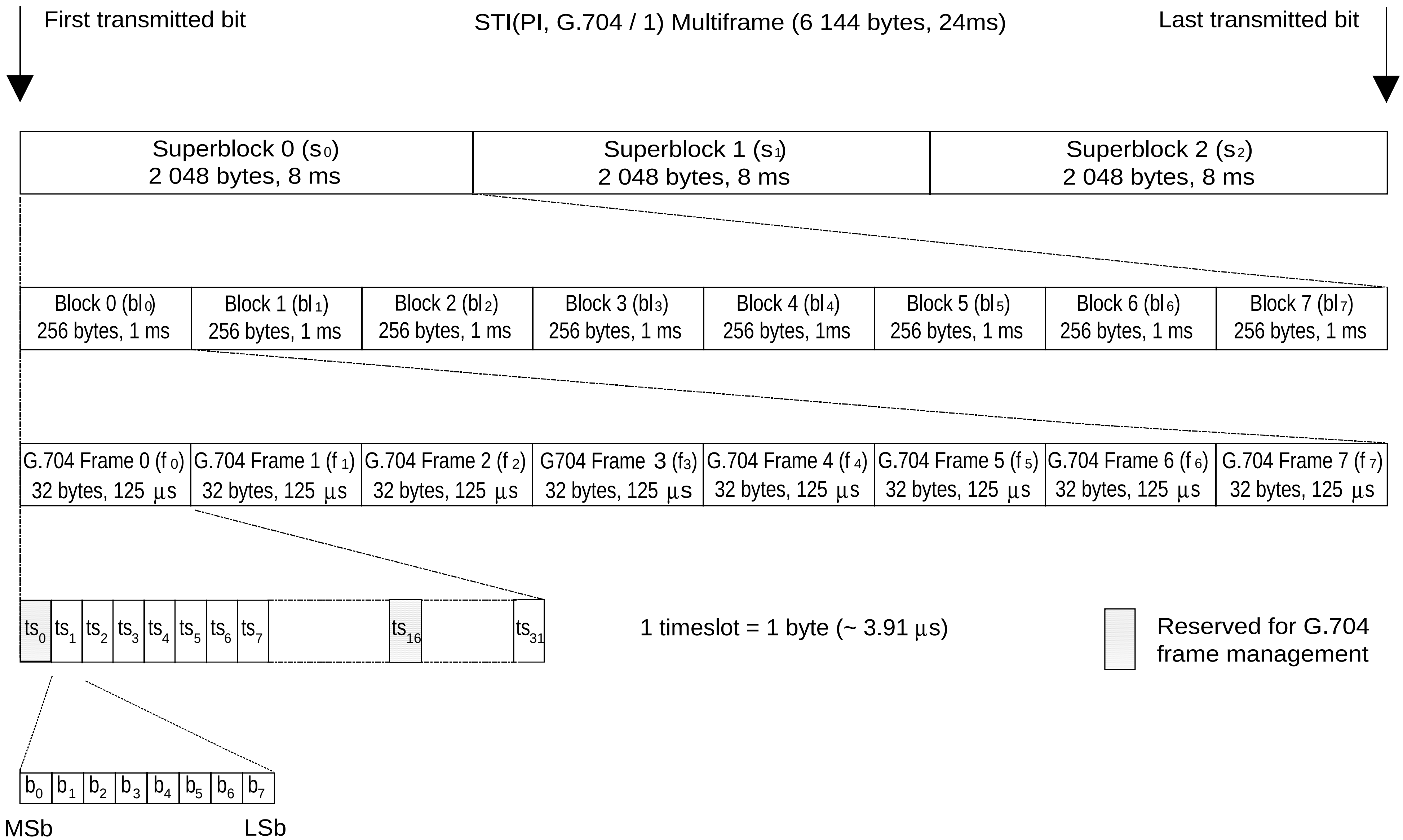

Figure 82: STI(PI, G.704/1) multiframe structure

9.5.3.2.2 Multiframe supervision byte, $S_{bl,s}$

The $S_{bl,s}$ byte shall be used to provide a signalling channel in the STI(PI, G.704/1), called the NASC (Network Adapted Signalling Channel).

24 bytes per multiframe are available: $S_{[0..7],[0..2]}$. They shall be arranged as three eight-byte signalling groups. Each group shall start with byte $S_{0,s}$ followed by $S_{1,s}$ etc., ending with $S_{7,s}$.

Table 66 gives details of the formatting of the NASC channel, the coding for the NASC FSS type or ASS field identifier is given in table 67. Annex D contains additional information.

Table 66: General formatting of NASC signalling bytes

$S_{0,s}$			$S_{1,s}$	$S_{2,s}$	$S_{[3..6],s}$	$S_{7,s}$
(b_0,b_1)	(b_2,b_3)	$(b_4..b_7)$	$(b_0..b_7)$	$(b_0..b_7)$	$(b_0..b_7)$	$(b_0..b_7)$
00	Rfu	FSS type identifier	Rfu	FSS (MSb)	..	FSS (LSb)
01	Rfa	Rfa	Rfa	Rfa	Rfa	Rfa
10	Rfa	Rfa	Rfa	Rfa	Rfa	Rfa
11	Rfu	ASS field identifier	Rfu	ASS (MSb)	..	ASS (LSb)

Table 67: Coding of NASC FSS type and ASS field identifier

$S_{0,s(b4..b7)}$ identifier	$S_{0,s(b0,b1)}$ = 00 FSS type	$S_{0,s(b0,b1)}$ = 11 ASS field
0000	Rfa	Start group
0001	Rfa	Rfa
0010	Rfa	Rfa
0011	Rfa	Continuation group
0100	Rfa	Rfa
0101	Rfa	Rfa
0110	Rfa	Rfa
0111	Rfa	Rfa
1000	User definable group	Rfa
1001	User definable group	Rfa
1010	User definable group	Rfa
1011	User definable group	Rfa
1100	User definable group	End group
1101	User definable group	Rfa
1110	User definable group	Rfa
1111	User definable group	Padding group

9.5.4 STI(PI, G.704/1) multiframe generation

9.5.4.1 General description

The specific details of the multiframe generation vary with the particular variant of the STI(PI, G.704/1) layer in use. The general principles are described in this clause. Detailed information appears in the following clauses and figures 83 and 84.

The first stage in producing the STI(PI, G.704/1) multiframe is to copy the bytes of the STI(PI, X) frame, excluding the SYNC field, into a coding array, C. The rows of the array are only partially filled by STI(PI, X) data.

The next step is to form the multiframe management and signalling bytes from the appropriate information source which includes the STI-D(LI) ERR field of the STI(PI, X) frame, carried in byte $B_{-8,p}$. These are then written into the appropriate positions in the coding array.

The array is filled by adding error-protection bytes. The bytes are calculated so that each row in the array forms a complete Reed-Solomon codeword.

The coding array, C, is then mapped into a single row interleaving array I, with elements $I_{[0..5\,759]}$. The depth of the interleaving is eight rows of the coding array which means that the interleaving is confined within one superblock.

Finally, the interleaved data, together with the G.704 reserved bytes, are mapped into another single row array O, with elements $O_{[0..6\,143]}$. O contains a full set of data corresponding to a 24 ms STI(PI, G.704/1) multiframe.

9.5.4.2 Error coding and interleaving for STI(PI, G.704/1)$_{5\,592}$

9.5.4.2.1 Coding array formation

The elements of the coding array, $C_{[0..23],[0..239]}$, shall be filled by the information bytes, $B_{[-4..5\,587]}$, together with 24 multiframe management bytes, $M_{[0..7],[0..2]}$, 24 multiframe supervision bytes, $S_{[0..7],[0..2]}$, and the error protection bytes $R_{[0..23],[235..239]}$.

$B_{[-4..5\,587]}$ are copied from the correspondingly numbered bytes of the STI(PI, X) frame p.

The position of bytes in the coding array, C, is defined by:

$$C_{i,j} = \begin{cases} M_{k,l} & (i \text{ MOD } 8) = 0 \text{ AND } (j \text{ MOD } 30) = 0 \\ S_{k,l} & (i \text{ MOD } 8) = 1 \text{ AND } (j \text{ MOD } 30) = 0 \\ B_r & (i \text{ MOD } 8) < 2 \text{ AND } (j \text{ MOD } 30) \neq 0 \text{ AND } j \leq 234 \\ B_s & (i \text{ MOD } 8) \geq 2 \text{ AND } j \leq 234 \\ R_{i,j} & 235 \leq j \leq 239 \end{cases}$$

where:

- $C_{i,j}$ shall be the elements of C with row index *i [*0..23*]* and column index *j [*0..239*]*;
- $M_{k,l}$ shall be the multiframe management byte of bl_k of s_l with $k = j$ DIV 30 and $l = i$ DIV 8;
- *l* shall be the superblock index;
- $S_{k,l}$ shall be the multiframe supervision byte of bl_k of s_l, the values of *k* and *l* are as given for $M_{k,l}$;
- B_r shall be the *r*th information byte with:

  $r = (i \text{ DIV } 8) \times 1\,864 + (i \text{ MOD } 2) \times 227 - (j \text{ DIV } 30) + (j - 1) - 4;$
- B_s shall be the *s*th information byte with:

  $s = (i \text{ DIV } 8) \times 1\,864 + (i \text{ MOD } 8) \times 235 + (j - 16) - 4;$
- $R_{i,[235..239]}$ shall be the 5 parity check bytes calculated for the information bytes $C_{i,[0..234]}$.The check bytes shall be calculated using the RS(235, 240) code described in clause 9.5.6.

9.5.4.2.2 Interleaving

The interleaving shall be performed by mapping the elements of the coding array, $C_{[0..23],[0..239]}$, into the single row array I. The mapping shall be given by:

$$I_p = C_{i,j}$$

where: I_p shall be the elements of the Interleaving array with index *p* given by:

$$p = 1\,920 \times (i \text{ DIV } 8) + 8 \times j + (i \text{ MOD } 8).$$

9.5.4.2.3 Output array formation

The elements of the output array, $O_{[0..6\,143]}$, shall be mapped from the interleaving array elements, $I_{[0..5\,759]}$, as follows:

$$O_q = \begin{cases} G_0 & (q \text{ MOD } 32) = 0 \\ G_1 & (q \text{ MOD } 32) = 16 \\ I_p & (q \text{ MOD } 32) \neq 0 \text{ AND } (q \text{ MOD } 32) \neq 16 \end{cases}$$

where:

- G_0 shall be a G.704 synchronizing byte;
- G_1 shall be a G.704 signalling byte;
- q shall be the index [0..6 143] of the output array given by:

  $q = p + (p \text{ DIV } 15) + 1$.

The formation of the output data shall be as illustrated in figure 83.

9.5.4.3 Error coding and interleaving for STI(PI, G.704/1)$_{5\,376}$

9.5.4.3.1 Coding array formation

The elements of the coding array, $C_{[0..23],[0..239]}$, shall be filled by the information bytes, $B_{[-4..5\,371]}$, together with 24 multiframe management bytes, $M_{[0..7],[0..2]}$, 24 multiframe supervision bytes, $S_{[0..7],[0..2]}$, and the error protection bytes $R_{[0..23],[226..239]}$.

$B_{[-4..5\,371]}$ are copied from the correspondingly numbered bytes of the STI(PI, X) frame p.

The position of bytes in the coding array, C, shall be defined by:

$$C_{i,j} = \begin{cases} M_{k,l} & (i \text{ MOD } 8) = 0 \text{ AND } (j \text{ MOD } 30) = 0 \\ S_{k,l} & (i \text{ MOD } 8) = 1 \text{ AND } (j \text{ MOD } 30) = 0 \\ B_r & (i \text{ MOD } 8) < 2 \text{ AND } (j \text{ MOD } 30) \neq 0 \text{ AND } j \leq 225 \\ B_s & (i \text{ MOD } 8) \geq 2 \text{ AND } j \leq 225 \\ R_{i,j} & 226 \leq j \leq 239 \end{cases}$$

where:

- $C_{i,j}$ shall be the elements of C with row index i [0..23] and column index j [0..239];
- $M_{k,l}$ shall be the multiframe management byte of bl_k of s_l with $k = j$ DIV 30 and $l = i$ DIV 8;
- l is the superblock index;
- $S_{k,l}$ shall be the multiframe supervision byte of bl_k of s_l, the values of k and l are as given for $M_{k,l}$;
- B_r shall be the rth information byte with:

  $r = (i \text{ DIV } 8) \times 1\,792 + (i \text{ MOD } 2) \times 218 - (j \text{ DIV } 30) + (j - 1) - 4$;

- B_s shall be the sth information byte with:

  $s = (i \text{ DIV } 8) \times 1\,792 + (i \text{ MOD } 8) \times 226 + (j - 16) - 4$;

- $R_{i,[226..239]}$ shall be the 14 parity check bytes calculated for the information bytes $C_{i,[0..225]}$. The check bytes shall be calculated using the RS(226, 240) code described in clause 9.5.6.

9.5.4.3.2 Interleaving

This process shall be exactly the same as that for STI(PI, G.704/1)$_{5\ 592}$ as presented in clause 9.5.4.2.2.

9.5.4.3.3 Output array formation

This process shall be exactly the same as that for STI(PI, G.704/1)$_{5\ 592}$ as presented in clause 9.5.4.2.3. The formation of the output data is illustrated in figure 84.

9.5.5 Order of data transmission

Data shall be transmitted by reading out the elements of the output array $O_{[0..6\ 143]}$. The bytes shall be read in sequence starting with O_0. The lowest numbered bit of the byte with the lowest address shall come first.

9.5.6 Error protection code

In both variants of STI(PI, G.704/1), Reed-Solomon coding shall be used to provide data which may be analysed by the interface receiving equipment to detect and correct errors occurring on the Transport Network.

The Reed-Solomon code uses symbols from GF(2^8), and shall be generated by the polynomial:

$$P(x) = x^8 + x^7 + x^2 + x + 1$$

The polynomial generator of the code shall be:

$$G(x) = \prod_{i=120}^{119+R} \left(x - \alpha^i \right)$$

For STI(PI, G.704/1)$_{5\ 592}$, R = 5 resulting in an RS(235,240) code.

For STI(PI, G.704/1)$_{5\ 376}$, R = 14 resulting in an RS(226,240) code.

9.5.7 Synchronization

9.5.7.1 Synchronization of G.704 frames

The synchronization of G.704 frames is described in ITU-T Recommendation G.706 [8]. Frame synchronization should be achieved in f_n if:

G_0 is present in f_n;

AND it is absent in f_{n+1};

AND b_1 of ts_o in f_{n+1} is set to 1;

AND G_0 is present in f_{n+2}.

Synchronization should be lost if three consecutive synchronization bytes are incorrectly received.

NOTE: There is no fixed phase relationship specified between the G.704 framing bytes, G_0 and G_1, and the STI(PI, G.704/1) multiframe framing bytes.

9.5.7.2 Synchronization of STI(PI, G.704/1) multiframes

The synchronization of the STI(PI, G.704/1) multiframe shall be achieved using $b_{0..4}$ of the multiframe management byte.

Multiframe synchronization should be achieved in multiframe m when:

G.704 synchronization is achieved in f_n;

AND the block count in $M_{bl,s}(b_{0..2})$ in f_{n+a+8} is an increment of 1 (modulo 8) above the count in f_{n+a} (a is incremented from 0 to 7);

AND the block count in $M_{bl,s}(b_{0..2})$ in f_{n+a+16} is an increment of 1 (modulo 8) above the count in f_{n+a+8}.

Data decoding should commence with the beginning of the next complete multiframe, m+1.

STI(PI, G.704/1) multiframe synchronization should be lost if the count sequence of the block and superblock counter is lost in three consecutive positions.

9.5.8 Physical interface

In physical characteristics of both variants of STI(PI, G.704/1) shall conform to the requirements of ITU-T Recommendation G.703 [4] for a 2 048 kbit/s interface.

There shall be a 75 Ω female BNC connector fitted to the equipment.

9.5.9 Modifying the STI-D(LI) ERR field

The STI(PI, G.704/1) layer can modify the STI-D(LI) ERR field to indicate the error status associated with the use of the STI(PI, G.704/1) layer.

The ERR field shall take one of four levels, 0 to 3, as defined in clause 5.2. The ERR field shall be carried through the physical interface by mapping the ERR field onto eight bits of the multiframe management byte as described in table 65. At the receiving equipment the appropriate bits of the multiframe management byte shall be re-mapped onto the output STI-D(LI) logical frame.

In addition, the level of ERR can be increased during the re-mapping in the receiver according to the following rules:

The Error Level may be increased to 1 if the receiving equipment detects:

either: a CRC violation of any CRCST field of the recovered D-LIDATA;

or: D-LIDATA contains errors which have been corrected by the Reed-Solomon decoder.

The Error Level may be increased to 2 if the NA equipment detects:

either: a CRC violation of the EOH field of the recovered D-LIDATA;

or: D-LIDATA contains uncorrected errors.

The Error Level may be increased to 3 if the receiving equipment is unable to synchronize to the incoming data.

The receiving equipment shall not decrease the error level of the ERR field.

When no DF field is present in the STI(PI, X), i.e. DFS = 0, the ERR field has no significance and shall be ignored by the receiving equipment.

Index																		
		0	1..29	30	31..59	60	61..89	90	91..119	120	121..149	150	151..179	180	181..209	210	211..234	235..239
	0	$M_{0,0}$	$B_{-4}..B_{24}$	$M_{1,0}$	$B_{25}..B_{53}$	$M_{2,0}$	$B_{54}..B_{82}$	$M_{3,0}$	$B_{83}..B_{111}$	$M_{4,0}$	$B_{112}..B_{140}$	$M_{5,0}$	$B_{141}..B_{169}$	$M_{6,0}$	$B_{170}..B_{198}$	$M_{7,0}$	$B_{199}..B_{222}$	$R_{0,235..239}$
	1	$S_{0,0}$	$B_{223}..B_{251}$	$S_{1,0}$	$B_{252}..B_{280}$	$S_{2,0}$	$B_{281}..B_{309}$	$S_{3,0}$	$B_{310}..B_{338}$	$S_{4,0}$	$B_{339}..B_{367}$	$S_{5,0}$	$B_{368}..B_{396}$	$S_{6,0}$	$B_{397}..B_{425}$	$S_{7,0}$	$B_{426}..B_{449}$	$R_{1,235..239}$
	2	B_{450}	$B_{451}..$	B_{480}	$B_{481}..$	B_{510}	$B_{511}..$	B_{540}	$B_{541}..$	B_{570}	$B_{571}..$	B_{600}	$B_{601}..$	B_{630}	$B_{631}..$	B_{660}	$..B_{684}$	$R_{2,235..239}$
i	3	B_{685}	$B_{686}..$	B_{715}	$B_{716}..$	B_{745}	$B_{746}..$	B_{775}	$B_{776}..$	B_{805}	$B_{806}..$	B_{835}	$B_{836}..$	B_{865}	$B_{866}..$	B_{895}	$..B_{919}$	$R_{3,235..239}$
	4	B_{920}	$B_{921}..$	B_{950}	$B_{951}..$	B_{980}	$B_{981}..$	$B_{1\,010}$	$B_{1\,011}..$	$B_{1\,040}$	$B_{1\,041}..$	$B_{1\,070}$	$B_{1\,071}..$	$B_{1\,100}$	$B_{1\,101}..$	$B_{1\,130}$	$..B_{1\,154}$	$R_{4,235..239}$
	5	$B_{1\,155}$	$B_{1\,156}..$	$B_{1\,185}$	$B_{1\,186}..$	$B_{1\,215}$	$B_{1\,216}..$	$B_{1\,245}$	$B_{1\,246}..$	$B_{1\,275}$	$B_{1\,276}..$	$B_{1\,305}$	$B_{1\,306}..$	$B_{1\,335}$	$B_{1\,336}..$	$B_{1\,365}$	$..B_{1\,389}$	$R_{5,235..239}$
	6	$B_{1\,390}$	$B_{1\,391}..$	$B_{1\,420}$	$B_{1\,421}..$	$B_{1\,450}$	$B_{1\,451}..$	$B_{1\,480}$	$B_{1\,481}..$	$B_{1\,510}$	$B_{1\,511}..$	$B_{1\,540}$	$B_{1\,541}..$	$B_{1\,570}$	$B_{1\,571}..$	$B_{1\,600}$	$..B_{1\,624}$	$R_{6,235..239}$
	7	$B_{1\,625}$	$B_{1\,626}..$	$B_{1\,655}$	$B_{1\,656}..$	$B_{1\,685}$	$B_{1\,686}..$	$B_{1\,715}$	$B_{1\,716}..$	$B_{1\,745}$	$B_{1\,746}..$	$B_{1\,775}$	$B_{1\,776}..$	$B_{1\,805}$	$B_{1\,806}..$	$B_{1\,835}$	$..B_{1\,859}$	$R_{7,235..239}$
	8	$M_{0,1}$	$B_{1\,860}..$	$M_{1,1}$	$B_{1\,889}..$	$M_{2,1}$	$B_{1\,918}..$	$M_{3,1}$	$B_{1\,947}..$	$M_{4,1}$	$B_{1\,976}..$	$M_{5,1}$	$B_{2\,005}..$	$M_{6,1}$	$B_{2\,034}..$	$M_{7,1}$	$..B_{2\,086}$	$R_{8,235..239}$

p	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**..**	**1 919**	**..**
	$M_{0,0}$	$S_{0,0}$	B_{450}	B_{685}	B_{920}	$B_{1\,155}$	$B_{1\,390}$	$B_{1\,625}$	B_{-4}	B_{223}	B_{451}	B_{686}	B_{921}	$B_{1\,156}$	$B_{1\,391}$	$B_{1\,626}$	B_{-3}	B_{224}	..	$R_{7,239}$	..

q	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**..**	**2 047**	**..**
	G_0	$M_{0,0}$	$S_{0,0}$	B_{450}	B_{685}	B_{920}	$B_{1\,155}$	$B_{1\,390}$	$B_{1\,625}$	B_{-4}	B_{223}	B_{451}	B_{686}	B_{921}	$B_{1\,156}$	$B_{1\,391}$	G_1	$B_{1\,626}$	..	$R_{7,239}$	..

Figure 83: Steps in the formation of STI(PI, G.704/1)$_{5\,592}$

Index																		
		0	**1..29**	**30**	**31..59**	**60**	**61..89**	**90**	**91..119**	**120**	**121..149**	**150**	**151..179**	**180**	**181..209**	**210**	**211..225**	**226..239**
	0	$M_{0,0}$	$B_{-4}..B_{24}$	$M_{1,0}$	$B_{25}..B_{53}$	$M_{2,0}$	$B_{57}..B_{82}$	$M_{3,0}$	$B_{83}..B_{111}$	$M_{4,0}$	$B_{112}..B_{140}$	$M_{5,0}$	$B_{141}..B_{169}$	$M_{6,0}$	$B_{170}..B_{198}$	$M_{7,0}$	$B_{199}..B_{213}$	$R_{0,226..239}$
	1	$S_{0,0}$	$B_{214}..B_{242}$	$S_{1,0}$	$B_{243}..B_{271}$	$S_{2,0}$	$B_{272}..B_{300}$	$S_{3,0}$	$B_{301}..B_{329}$	$S_{4,0}$	$B_{330}..B_{358}$	$S_{5,0}$	$B_{359}..B_{387}$	$S_{6,0}$	$B_{388}..B_{416}$	$S_{7,0}$	$B_{417}..B_{431}$	$R_{1,226..239}$
	2	B_{432}	B_{433}..	B_{462}	B_{463}..	B_{492}	B_{493}..	B_{522}	B_{523}..	B_{552}	B_{553}..	B_{582}	B_{583}..	B_{612}	B_{613}..	B_{642}	..B_{657}	$R_{2,226..239}$
i	**3**	B_{658}	B_{659}..	B_{688}	B_{689}..	B_{718}	B_{719}..	B_{748}	B_{749}..	B_{778}	B_{779}..	B_{808}	B_{809}..	B_{838}	B_{839}..	B_{868}	..B_{883}	$R_{3,2\ 326..239}$
	4	B_{884}	B_{885}..	B_{914}	B_{915}..	B_{944}	B_{945}..	B_{974}	B_{975}..	$B_{1\ 004}$	$B_{1\ 005}$..	$B_{1\ 034}$	$B_{1\ 035}$..	$B_{1\ 064}$	$B_{1\ 065}$..	$B_{1\ 094}$	..$B_{1\ 109}$	$R_{4,226..239}$
	5	$B_{1\ 110}$	$B_{1\ 111}$..	$B_{1\ 140}$	$B_{1\ 141}$..	$B_{1\ 170}$	$B_{1\ 171}$..	$B_{1\ 200}$	$B_{1\ 201}$..	$B_{1\ 230}$	$B_{1\ 231}$..	$B_{1\ 260}$	$B_{1\ 261}$..	$B_{1\ 290}$	$B_{1\ 291}$..	$B_{1\ 320}$	..$B_{1\ 335}$	$R_{5,226..239}$
	6	$B_{1\ 336}$	$B_{1\ 337}$..	$B_{1\ 366}$	$B_{1\ 367}$..	$B_{1\ 396}$	$B_{1\ 397}$..	$B_{1\ 426}$	$B_{1\ 427}$..	$B_{1\ 456}$	$B_{1\ 457}$..	$B_{1\ 486}$	$B_{1\ 487}$..	$B_{1\ 516}$	$B_{1\ 517}$..	$B_{1\ 546}$	..$B_{1\ 561}$	$R_{6,226..239}$
	7	$B_{1\ 562}$	$B_{1\ 563}$..	$B_{1\ 592}$	$B_{1\ 593}$..	$B_{1\ 622}$	$B_{1\ 623}$..	$B_{1\ 652}$	$B_{1\ 653}$..	$B_{1\ 682}$	$B_{1\ 683}$..	$B_{1\ 712}$	$B_{1\ 713}$..	$B_{1\ 742}$	$B_{1\ 743}$..	$B_{1\ 772}$	..$B_{1\ 787}$	$R_{7,226..239}$
	8	$M_{0,1}$	$B_{1\ 788}$..	$M_{1,1}$	$B_{1\ 817}$..	$M_{2,1}$	$B_{1\ 846}$..	$M_{3,1}$	$B_{1\ 875}$..	$M_{4,1}$	$B_{1\ 904}$..	$M_{5,1}$	$B_{1\ 933}$..	$M_{6,1}$	$B_{1\ 962}$..	$M_{7,1}$	..$B_{2\ 005}$	$R_{8,226..239}$

p	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**..**	**1 919**	**..**
	$M_{0,0}$	$S_{0,0}$	B_{432}	B_{658}	B_{884}	$B_{1\ 110}$	$B_{1\ 336}$	$B_{1\ 562}$	B_{-4}	B_{214}	B_{433}	B_{659}	B_{885}	$B_{1\ 111}$	$B_{1\ 337}$	$B_{1\ 563}$	B_{-3}	B_{215}	..	$R_{7,239}$	..

q	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**..**	**2 047**	**..**
	G_0	$M_{0,0}$	$S_{0,0}$	B_{432}	B_{658}	B_{884}	$B_{1\ 110}$	$B_{1\ 336}$	$B_{1\ 562}$	B_{-4}	B_{214}	B_{433}	B_{659}	B_{885}	$B_{1\ 111}$	$B_{1\ 337}$	G_1	$B_{1\ 563}$	..	$R_{7,239}$	..

Figure 84: Steps in the formation of STI(PI, G.704/1)$_{5\ 376}$

9.6 G.704 interface without error protection, STI(PI, G.704/2)

9.6.1 General description

The purpose of the STI(PI, G.704/2) is to provide a physical form to the STI-D(LI) and STI-C(TA) for use on networks based on the first level of the PDH (2 048 kbit/s) as defined in ITU-T Recommendation G.704 [7]. This network adaptation layer only provides adaptation to the G.704 frame structure and does not provide any forward error coding or network delay compensation. If this is required STI(PI, G.704/1) should be used.

9.6.2 Adaptation of the STI(PI, X) to the STI(PI, G.704/2)

STI(PI, G.704/2) shall use the generic transport frame structure defined in clause 8. Each STI(PI, G.704/2) frame shall carry 5 760 bytes of STI(PI, X) data, $B_{[-8..5\ 751]}$, which represents a net bitrate of 1 920 kbit/s. The length of the STI(PI, X) frame, TFL, shall be 5 760 bytes.

9.6.3 Adaptation to the G.704 frame structure

There is no specific alignment of the STI(PI, X) transport frame to the G.704 frames, other than that there shall be byte alignment.

9.6.3.1 G.704 reserved bytes

The following bytes of each G.704 frame carrying STI(PI, G.704/2) shall be reserved for G.704 frame control:

- ts_0 in each G.704 frame for the G.704 synchronization byte, G_0;
- ts_{16} in each G.704 frame for the G.704 supervision byte, G_1.

9.6.3.2 STI(PI, G.704/2) generation

The STI(PI, G.704/2) frame shall have a FL of 24 ms equivalent to 6 144 timeslots, or bytes.

The STI(PI, G.704/2) structure is built using the generic STI(PI, X) transport frame structure mapped into an Output array, O, with elements $O_{[0..6\ 143]}$. O contains a full set of data corresponding to a 24 ms STI(PI, G.704/2) frame.

NOTE: There is no specific relationship between the first byte of the Output array (O_0) and the G.704 timeslot.

9.6.3.2.1 Output array formation

The elements of the output array, $O_{[0..6\ 143],p}$, shall be filled by the information bytes, $B_{[-8..5\ 751],p}$, together with G.704 synchronizing and signalling bytes as follows:

$$O_{q,p} = \begin{cases} G_0 & \left((q+\varphi) \text{ MOD } 32\right) = 0 \\ G_1 & \left((q+\varphi) \text{ MOD } 32\right) = 16 \\ B_{q-8-((q+\varphi) \text{ DIV } 16),p} & \left((q+\varphi) \text{ MOD } 32\right) \neq 0 \text{ AND } \left((q+\varphi) \text{ MOD } 32\right) \neq 16 \end{cases}$$

where:

- G_0 shall be a G.704 synchronizing byte;
- G_1 shall be a G.704 signalling byte;
- $B_{[-8..5\ 751],p}$ shall be the complete generic STI(PI, X) transport frame *p*;
- *q* shall be the index [0..6 143] of the output array;
- φ shall be a constant value in the range 0..31.

The formation of the output data is illustrated in figure 85 where φ is chosen to be 4.

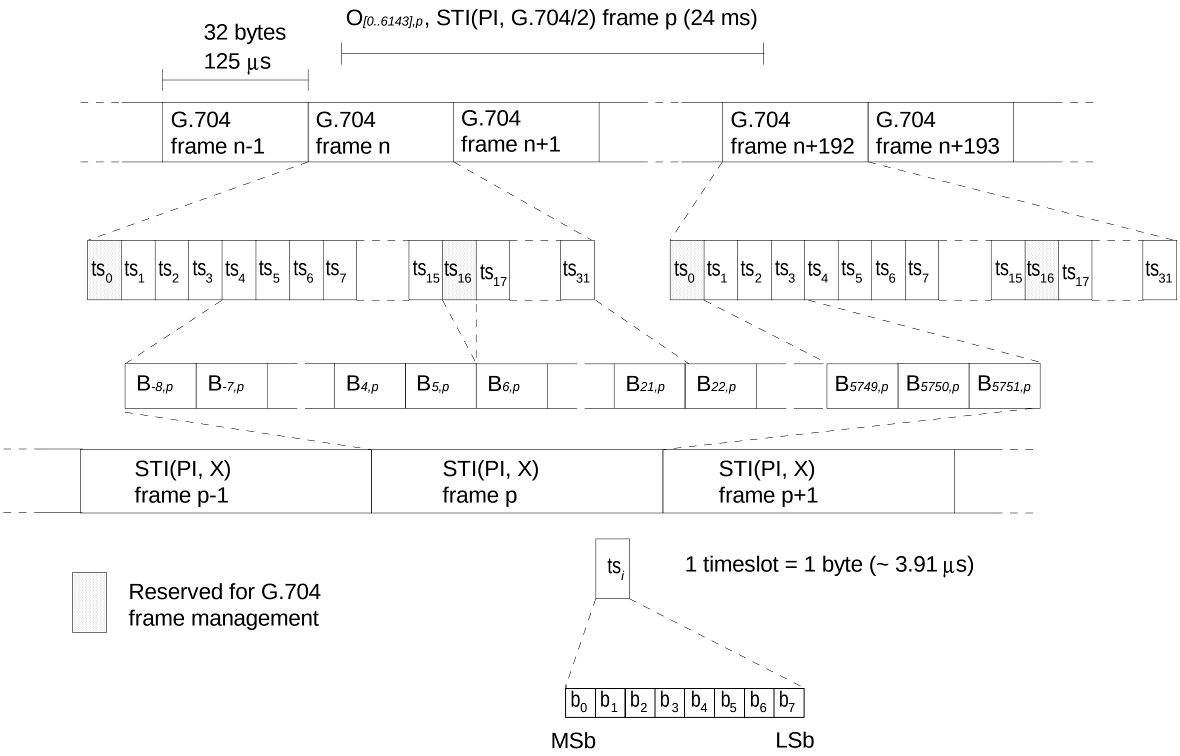

Figure 85: Mapping of the Generic STI(PI, X) transport frame into G.704 frames

9.6.3.2.2 Order of data transmission

Data shall be transmitted by reading out the elements of the output array $O_{[0..6\ 143],p}$. The bytes shall be read in sequence starting with $O_{0,p}$. The lowest numbered bit of the byte with the lowest address shall come first.

9.6.3.2.3 Synchronization of G.704 frames

The synchronization of G.704 frames is described in ITU-T Recommendation G.706 [8]. Frame synchronization should be achieved in f_n if:

G_0 is present in f_n;

AND it is absent in f_{n+1};

AND b_1 of ts_o in f_{n+1} is set to 1;

AND G_0 is present in f_{n+2}.

Synchronization should be lost if three consecutive synchronization bytes are incorrectly received.

9.6.4 Physical interface

The physical characteristics of STI(PI, G.704/2) shall conform to the requirements of ITU-T Recommendation G.703 [4] for a 2 048 kbit/s interface.

There shall be a 75 Ω female BNC connector fitted to the equipment.

9.7 H.221 interfaces, STI(PI, H.221)

9.7.1 General description

The purpose of STI(PI, H.221) is to provide a physical interface which may typically be used in ISDN networks or on leased lines or permanent connections.

ITU-T Recommendation H.221 [11] provides for dynamically subdividing a transmission channel of 64 kbit/s to 1 920 kbit/s into lower rates suitable for audio, video and telematics purposes. It provides for synchronizing a number of 64 kbit/s channels by adding signalling bits. ITU-T Recommendation H.242 [12] provides the protocol to use the signalling of H.221.

All possible data rates identified in ITU-T Recommendation H.221 [11] as applicable to the STI shall be available for use. The protocols specified by ITU-T Recommendation H.242 [12] shall be used for the management of the interface.

9.7.2 Adaptation of the STI(PI, X) to the STI(PI, H.221)

STI(PI, H.221) shall use the generic transport frame structure defined in clause 8. Each STI(PI, H.221) frame shall contain TFL bytes of STI(PI, X) data, $B_{[-8..TFL-9]}$, which represents a net bitrate of (TFL / 3) kbit/s.

9.7.3 Adaptation to the H.221 frame structure

There is no specific alignment of the bits and bytes of the STI(PI, X) transport frame to the H.221 frames.

9.7.3.1 H.221 reserved bits

The Service Channel, defined in ITU-T Recommendation H.221 [11], in addition to the frame alignment signal and bitrate allocation signal, shall have octets 17 to 20 reserved and set to zero. They shall not be used to carry STI(PI, X) data.

NOTE: This provides for an integer number of bytes to be available in each 24 ms period.

9.7.3.2 STI(PI, H.221) generation

The STI(PI, H.221) frame shall have a FL of 24 ms.

The STI(PI, X) shall be transferred sequentially bit by bit into the H.221 frames. The lowest numbered bit of the byte with the lowest address shall come first.

9.7.4 Physical interface

Any suitable physical connector may be fitted to the equipment.

10 Physical Interfaces for asynchronous links

This clause defines a number of transport adaptations in combination with physical implementation to allow the transport of either STI-D(LI) or STI-C(TA) or both over interfaces with an asynchronous nature.

10.1 V.24 interface, STI(PI, V.24)

10.1.1 General

The purpose of STI(PI, V.24) is to provide a physical form to the STI-D(LI) and STI-C(TA) suitable for local connections, or distant connections via telecommunication networks.

STI(PI, V.24) may be used at any baud rate.

No specific protection shall be provided by the STI(PI, V.24) adaptation layer.

10.1.2 Adaptation of the STI(PI, X) to the STI(PI, V.24)

STI(PI, V.24) shall use the generic transport frame structure defined in clause 8. The size of each STI(PI, V.24) frame may vary.

NOTE: The DFCT field of the STI-D(LI) frame (if present) may have no significance other than as a sequence counter.

10.1.3 Physical interface

The STI(PI, V24) shall be in compliance with ITU-T Recommendation V.24 [9] and ITU-T Recommendation V.28 [10] and shall have the following specific attributes:

- Sub-D connector, type DTE, 9 pins (male);
- unbalanced electrical interface in accordance with ITU-T Recommendation V.28 [10]
- pin allocation as defined in table 68;
- hardware handshake: using the handshake signals CTS (by DCE) or RTS (by DTE) data transfer can be interrupted;
- 1 start bit;
- 8 data bits, LSB first;
- no parity;
- 1 stop bit.

Table 68: Pin allocations for STI(PI, V.24)

	Input/ Output	Pin	Function
1		*(DCD)*	Data Carrier Detect (see note)
2	I	*RXD*	Received Data
3	O	TXD	Transmitted Data
4		*(DTR)*	Data Terminal Ready (see note)
5		GND	Signal Ground
6		*(DSR)*	Data Set Ready (see note)
7	O	*RTS*	Request To Send
8	I	CTS	Clear To Send
9		*(RI)*	Ring Indicator (see note)
NOTE: Signal is not required but may be optionally used.			

Annex A (normative): Calculation of CRC words in the STI

A CRC code is defined by its generator polynomial of degree *n*:

$$G(x) = \sum_{i=0}^{n} g_i x^i$$

with $n \geq 1$
and $g_j = 0$ or 1 except that $g_0 = 1$, $g_n = 1$.

The CRC calculation may be performed by means of a shift register containing *n* register stages, equivalent to the degree of the polynomial (see figure A.1). The stages are denoted by $b_0 \ldots b_{n-1}$, where b_0 corresponds to x^0, b_1 to x^1, b_2 to x^2, ... , b_{n-1} to x^{n-1}. The shift register is tapped by inserting XORs at the input of those stages, where the corresponding coefficients g_i of the polynomial are "1".

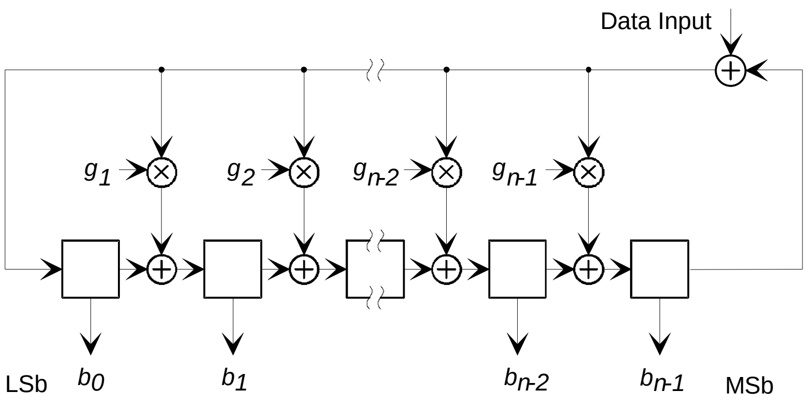

Figure A.1: General CRC block diagram

At the beginning of the CRC calculation, the register stage contents are initialized to all ones. After applying the first bit of the data block (MSb first) to the input, the shift clock causes the register to shift its content by one stage towards the MSb stage (b_{n-1}), while loading the tapped stages with the result of the appropriate XOR operations. The procedure is then repeated for each data bit. Following the shift after applying the last bit (LSb) of the data block to the input, the shift register contains the CRC word which is then read out.

The CRC word shall be inverted prior to transmission. At the receiving end, an error free transmission shall give a CRC result of $1D0F_{16}$.

The CRC code used in the STI shall use the following polynomial:

$$G(x) = x^{16} + x^{12} + x^5 + 1 .$$

Annex B (normative): Coding of timestamps in STI

B.1 General

This annex gives detailed information about the coding of timestamp information carried in STI-D(LI) and STI(PI, G.704/1).

B.2 Timestamp coding

The timestamp shall be coded as a 24-bit unsigned binary integer giving the timestamp value as a multiple of 16,384 MHz clock periods (approximately 61 ns). The coding of the 24 bits shall be as given in table B.1.

Table B.1: Coding of timestamps

Bit number	b_0(MSb)..b_6	b_7..b_9	b_{10}..b_{17}	b_{18}..b_{20}	b_{21}..b_{23}(LSb)
Timestamp level	1	2	3	4	5
Valid range	[0..124], 127	[0..7]	[0..255]	[0..7]	[0..7]
Approximate time resolution	8 ms	1 ms	3,91 μs	488 ns	61 ns

B.2.1 Expected range of timestamp values

The timestamp value, expressed as a hexadecimal number shall lie between 0 and F9 $FFFF_{16}$ (giving time values between 0 and 999,999 939 ms).

B.2.2 Null timestamp

The timestamp value FF $FFFF_{16}$ may also be allowed and shall be used as a null timestamp value. This shall be interpreted as meaning that there is no timestamp information available.

B.2.3 Reserved timestamp values

Timestamp values between FA 0000_{16} and FF $FFFE_{16}$ shall be reserved for future use.

B.2.4 Timestamp levels

Not all applications will require a timestamp value to be specified to a resolution of 61 ns. For this reason a number of different timestamp levels are specified (1 to 5). The different levels allow timestamps to be specified with different resolutions e.g. use of timestamps to level 2 allow a time resolution of 1 ms which should suffice in many applications. The timestamp values at levels higher than those used shall be set to 0, except in the specific case of transmission of a null timestamp.

B.3 Mapping to STI-D(LI) timestamp bits

The mapping of the timestamp into the STI-D(LI) layer shall be as given in table B.2.

Table B.2: Mapping of the timestamp into the STI-D(LI) layer

STI-D(LI) byte	$B_{(DL-4),p}$	$B_{(DL-4)+1,p}$	$B_{(DL-4)+2,p}$	$B_{(DL-4)+3,p}$
Bit index	0..7	0..7	0..7	0..7
Timestamp bit	Rfa, set to FF_{16}.	$b_0..b_7$	$b_8..b_{15}$	$b_{16}..b_{23}$

B.4 Mapping to STI-D(PI, G.704/1) timestamp bits

The mapping of the timestamp into $M_{bl,s}(b_5)$ of the STI(PI, G.704/1) layer shall be as given in table B.3.

Table B.3: Mapping of the timestamp into $M_{bl,s}(b_5)$ of the STI(PI, G.704/1) layer

$M_{bl,s}(b_5)$		*bl*							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
s	**0**	b_0(MSb)	b_1	b_2	b_3	b_4	b_5	b_6	b_7
	1	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}
	2	b_{16}	b_{17}	b_{18}	b_{19}	b_{20}	b_{21}	b_{22}	b_{23}(LSb)

B.5 Interpretation of timestamp value

The timestamp carried within a 24 ms STI-D frame defines the transmission time of the first bit, $B_{0,p}(b_0)$, of the STI-D(LI) frame associated with that frame. The start time is specified as an offset value (always positive) from a time reference point derived from a time reference signal.

The time reference signal, from which the timestamp offset is measured, shall be available at both the originating point of the timestamp in the STI frame and at the intermediate and final "delivery points" of the data streams included in the frame.

Suitable time references may be derived from the Global Positioning Satellite system (as a 1 pulse per second reference).

NOTE: Although the timestamp allows time resolutions down to 61 ns, the accuracy of signal timing is likely to be determined largely by the accuracy of the time reference.

B.6 Use of timestamps in LI and PI layers

The use of the timestamp is largely system dependant and depends on the network configuration, transport mechanism, etc.

The time given by the timestamp in the LI layer should define the "notional delivery time" of the first bit $B_{0,p}(b_0)$ of the STI-D(LI) frame at the input of the channel encoder at the transmitters. The Ensemble provider (and intermediate Service providers, if any) shall use the timestamp to compensate for the various transit delays in the DAB network.

The time given by the timestamp in the PI layer should define the "notional delivery time" of the first bit $B_{0,p}(b_0)$ of STI-D(LI) frame at the output of an STI(PI, X) to STI-D(LI) converter. This timestamp can be used, for example, to re-time the STI(PI, X) sections in a cascaded collection network.

Annex C (normative): Definition of the WG1/2 Interface

The WG1/2 interface was developed in the early days of the Eureka 147 project to allow the connection of a number of audio encoders to prototype channel encoder equipment.

The WG1/2 interface allows:

- to link up to sixteen separate audio encoders (or data source equipment);
- to distribute and to manage clock and synchronization reference signal.

Because of its high speed and parallel nature, the WG1/2 interface is only appropriate for the connection of equipment located in relatively close physical proximity. Despite this, the WG1/2 interface is still in common use in many DAB networks.

The STI(PI, WG1/2) interface defined in the present document (see clause 9), specify the way to carry STI(LI) onto one WG1/2 interface slot. This annex describes the basic characteristics of the WG1/2 interface.

C.1 WG1/2 interface overview

The WG1/2 interface is basically a high bitrate point-to-point interface, between a data producer and a data receiver. It is able to carry two groups of unidirectional signals:

- a downstream group: made of data, clock and synchronization lines, it allows a data producer to send synchronous serial data;
- an upstream group: made of reference clock and reference synchronization lines, it allows a data receiver to send its reference signals to the data producer equipment.

Several point-to-point WG1/2 interfaces can be used to connect several pieces of equipment in order to constitute a ring on which each piece of equipment is able:

- on the upstream group: to route the received reference signals or to replace them by its own ones;
- on the downstream group: to route the received data signals, to add its own data signal to the received ones or to replace one received data signal with its own data signal.

An example of a ring connection is shown in figure C.1. It highlights the possibility to implement an add / drop multiplexer both on the upstream and downstream signal groups.

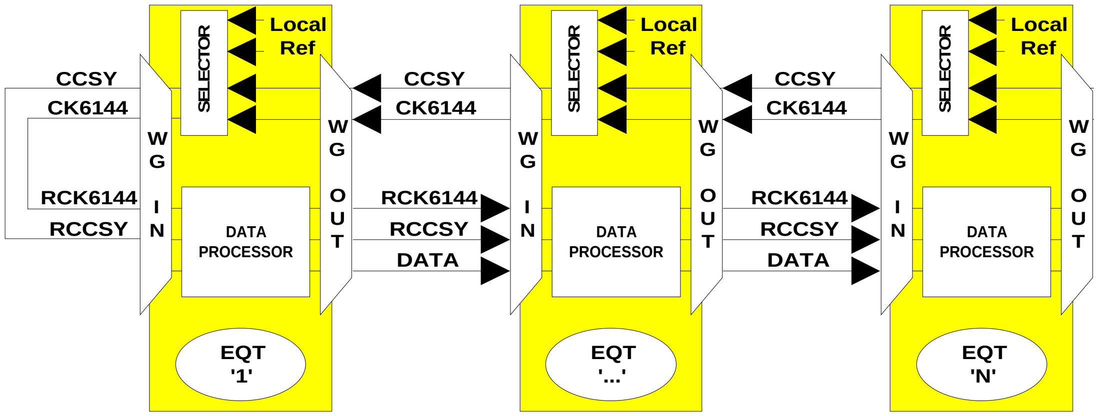

Figure C.1: Example of a ring connection using WG1/2 interfaces

The point-to-point WG1/2 interface includes two ports referenced as input or output ports:

- an input port receives serial data with the associated clock and synchronization signals (downstream group) and provides a source for the reference clock and synchronization signals (upstream group);
- an output port provides serial data with the associated clock and synchronization signals (downstream group) and expects to receive the reference clock and synchronization signals (upstream group).

In general, an equipment which produces data (e.g.: audio encoder) will provide both WG1/2 input and WG1/2 output ports. An equipment which uses data (e.g.: multiplexers) will implement only one WG1/2 input port.

C.2 WG1/2 interface signals definition

WG1/2 interface shall be made of the five following signals:

- CK6144 and CCSY shall constitute the upstream signals;
- RCK6144, RCCSY and DATA shall constitute the downstream signals.

A WG1/2 input port shall provide upstream signals and shall receive downstream signals.

A WG1/2 output port shall provide downstream signals and shall receive upstream signals.

The five signals implementing the WG1/2 interface shall have the following definition:

- CK6144

  shall be a 6 144 kHz reference clock signal. CK6144 shall be mandatory produced on a WG1/2 input port.

- CCSY

  shall be a 24 ms reference synchronization signal. CCSY shall be mandatory on a WG1/2 input port. Rising edge of CCSY shall occur every 147 456 periods of CK6144 (24 ms). The duration of CCSY shall be 32 periods of CK6144. Rising and falling edges of CCSY shall occur during the falling edge of CK6144.

- RCK6144

  shall be a 6 144 kHz clock signal. RCK6144 shall be mandatory on a WG1/2 output port.

- RCCSY

  shall be a 24 ms synchronization signal. RCCSY shall be mandatory on a WG1/2 output port. Rising edge of RCCSY shall occur every 147 456 periods of RCK6144 (24 ms). The duration of RCCSY shall be 32 periods of RCK6144. Rising and falling edges of RCCSY shall occur during falling edge of RCK6144.

- DATA

  shall be made of 24 ms DATA-frames comprising 147 456 data-bit periods (or data-phases) occurring during one RCCSY period. DATA shall be mandatory on a WG1/2 output port. Data-bits shall be stable during the rising edge of RCK6144.
  Each DATA-frame shall include 9 216 DATA-blocks. Each DATA-block shall be made of sixteen consecutive data-phase. The first DATA-phase of the first DATA-block shall occur on the rising edge of RCCSY.

Figure C.2 shows the phase relationship between WG1/2 interface signals.

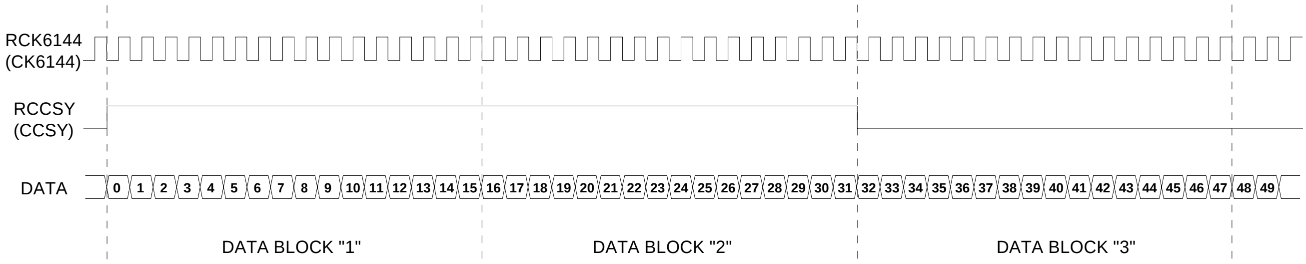

Figure C.2: Phase relationship of WG1/2 interface signals

C.3 WG1/2 interface data-frame syntax

The WG1/2 serial data connection have a payload capacity of 6 144 Mbit/s.

This overall payload shall be organized in 16 slots, each having a payload capacity of 384 kbit/s. Each slot shall occupy a dedicated DATA-phase of each DATA-block.

The correspondence between the WG1/2 DATA-phase and the WG1/2 slot shall be compliant with figure C.3.

DATA phases	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
WG1/2 Slots	1	9	5	13	3	11	7	15	2	10	6	14	4	12	8	16

Figure C.3: Correspondence between DATA-phases and WG1/2 slots

C.4 WG1/2 physical interface

All the signals of the WG1/2 interface shall comply with the general requirements of ITU-T Recommendations V.11 [6] and X.24 [5] and shall have the following specific attributes:

- clock/data timing: as defined in ITU-T Recommendation X.24 [5];
- electrical levels: balanced electrical interface as defined in ITU-T Recommendation V.11 [6].

Equipment implementing WG1/2 interface shall use, both for input or output ports, Sub-D connector, 25 pins (female, all devices). Pin assignment shall be in compliance with table C.1.

Table C.1: Pin allocations for STI-D(PI, WG1/2)

Pin	Signal	Pin	Signal
1	GND		
2	CK6144 (+)	14	CK6144 (-)
3	GND	15	GND
4	CCSY (+)	16	CCSY (-)
5	GND	17	GND
6	RCCSY (+)	18	RCCSY (-)
7	GND	19	GND
8	RCK6144 (+)	20	RCK6144 (-)
9	GND	21	GND
10	DATA (+)	22	DATA (-)
11	GND	23	GND
12	rfu	24	rfu
13	GND	25	GND

Annex D (normative): Coding of NASC data

D.1 General

This annex gives detailed information about the coding of the signalling data groups used into the Network Adapted Signalling Channel (NASC), provided by the physical interface STI(PI, G.704/1), described in clause 9.5.3.2.2.

NASC is able to carry two categories of messages: Frame Synchronous Signalling (FSS) and Frame Asynchronous Signalling (ASS).

D.2 Frame Synchronous Signalling (FSS)

D.2.1 FSS messages structure

For FSS, only one group per message shall be used. The FSS group structure shall be as given in table D.1.

Table D.1: Coding of FSS group types

$S_{0,s}$			$S_{1,s}$	$S_{2,s}$	$S_{3,s}$	$S_{4,s}$	$S_{5,s}$	$S_{6,s}$	$S_{7,s}$
(b_0,b_1)	(b_2,b_3)	$(b_4..b_7)$	$(b_0..b_7)$	$(b_0..b_7)$	$(b_0..b_7)$	$(b_0..b_7)$	$(b_0..b_7)$	$(b_0..b_7)$	$(b_0..b_7)$
00	Rfu	FSS type identifier	Rfu	FSS byte (MSb)	FSS byte	FSS byte	FSS byte	FSS byte	FSS byte (LSb)

The FSS type identifier is given by $S_{0,q}(b_4..b_7)$ as defined in clause 9.5.3.2.2. FSS message types may be pre-assigned (with $SB_{0,q}(b_4) = 0$) or user defined (with $SB_{0,q}(b_4) = 1$) as specified in clause D.2.2.

D.2.2 Pre-assigned FSS message types

No pre-assigned FSS message is defined by the present document.

D.3 Asynchronous Signalling (ASS)

D.3.1 ASS messages structure

ASS messages shall consist of a sequence of signalling groups.

Groups shall be defined as start, continuation, end or padding groups:

- each ASS message shall commence with a single start group which shall define the message type and the number of continuation groups which are present in the ASS message;
- continuation groups shall carry ASS message data;
- padding groups shall carry null padding information;
- each ASS message shall end with a single end group which shall contain CRC check fields.

The coding of the different group types shall be as given in table D.2.

Table D.2: Coding of different ASS group types

ASS group type	$S_{0,s}$ $(b_0..b_7)$	$S_{1,s}$ $(b_0..b_7)$	$S_{2,s}$ $(b_0..b_7)$	$S_{3,s}$ $(b_0..b_7)$	$S_{4,s}$ $(b_0..b_7)$	$S_{5,s}$ $(b_0..b_7)$	$S_{6,s}$ $(b_0..b_7)$	$S_{7,s}$ $(b_0..b_7)$
Start	11xx0000	Rfu	ASS Message type	*GrNb*	Message byte	Message byte	Message byte	Message byte
Continuation	11xx0011	Rfu	Message byte	Message byte	Message byte	Message byte	Message byte	Message byte
End	11xx1100	Rfu	Message byte	Message byte	Message byte	Message byte	CRC byte (MSB)	CRC byte (LSB)
Padding	11xx1111	FF_{16}	FF_{16}	FF_{16}	FF_{16}	FF_{16}	FF_{16}	FF_{16}

$S_{2,s}$ of the start group shall define the message type. Pre-assigned types shall have $S_{2,s}(b_0)$ set to 0. User definable types shall have $S_{2,s}(b_0)$ set to 1.

The general coding of ASS messages types shall be as defined in clause 9.5.3.2.2. $S_{2,s}$ of the start group shall define the ASS message type. Pre-assigned types shall have $S_{2,s}(b_0)$ set to 0, $S_{2,s}(b_0)$ shall be set to 1 for user definable types.

GrNb shall give the number of continuation groups in the ASS message. Padding groups shall not be included in this count. The overall total number of groups which constitutes the ASS message shall be given by:

Number of groups in message = *GrNb* + 2.

The CRC bytes in the end group shall contain the product of a cyclic redundancy checksum performed on the ASS Message content from the first byte of the start group ($S_{0,s}$) to the last message byte in the end group ($S_{5,s+x}$), excluding any padding groups.

CRC calculations shall use the method defined in annex A.

D.3.2 Pre-assigned ASS message types

No pre-assigned ASS messages are defined in the present document.

Annex E (normative): Behaviour of the STI during reconfiguration

The DAB multiplex is very flexible, allowing the multiplex to change dynamically in the following ways:

- a service may be added to or removed from the multiplex;
- a service component may be added to or removed from a service;
- the bit-rate allocated to a service component may vary.

These changes affect both the Multiplex Configuration Information signalled in the FIC and the organization of the MSC. The effects of these changes on the MSC are:

- a sub-channel may be added or removed from the transmission frame;
- the size of the sub-channel may be modified due to a change of data rate or protection level;
- the position of the sub-channel may change.

Multiplex re-configuration is defined as taking place at a specific instant in time. The behaviour of the transmitted ensemble during a re-configuration time, and its relation to the notional re-configuration time, is defined in EN 300 401 [1]. This annex defines how this behaviour is mirrored by the STI.

E.1 DAB multiplex configuration management

The DAB multiplex is made of a collection of services, each being composed of service components carried in MSC sub-channels or the FIDC. The service components are generated by one or more Service providers.

DAB multiplex reconfiguration shall be performed if any modification affects the Multiplex Configuration Information (MCI) as defined in EN 300 401 [1]. Accordingly any Service provider intending to modify its own service organization will be the originator of a DAB multiplex reconfiguration.

Due to the constraints imposed by the DAB multiplex reconfiguration procedure, the management of the reconfiguration operation is the responsibility of the Ensemble provider. Specifically, the Ensemble provider shall ensure that:

- the new configuration of any Service provider shall not adversely affect the other Service providers which share the DAB multiplex;
- the new configuration shall be compliant with EN 300 401 [1];
- the MCI for the new configuration shall be transmitted in advance of the reconfiguration instant as defined in EN 300 401 [1];
- the reconfiguration shall be performed at the correct instant, even if the Service provider fails to modify its STI streams correctly.

E.2 STI reconfiguration procedure

Performing a DAB multiplex reconfiguration involves several steps which shall be followed by the Service providers and managed by the Ensemble provider.

Those steps are described in the following clauses and are related to:

- defining the new configuration;
- choosing the reconfiguration instant;
- making the reconfiguration request;
- implementing the reconfiguration.

E.2.1 Service configuration definition

The new configuration shall be made known to the Ensemble provider in advance.

The configuration can be defined by using STI-C(LI) messages or by any method agreed between Service provider and Ensemble provider.

When STI-C(LI) messages are used, the resource category allows a Service provider to know the DAB resources that have been reserved by the Ensemble provider to broadcast its services and the configuration category allows the details of a Service provider configuration to be defined or monitored (see clause 6).

A Service provider configuration can be transferred to the Ensemble provider at any time. Therefore, the Ensemble provider is only able to check the consistency of the definition of the configuration.

This checking does not guarantee that a reconfiguration will be possible because the resources required may not be available at the time the reconfiguration is required.

E.2.2 Choosing the reconfiguration instant

In many cases the Service provider can reconfigure without needing to know the precise time that the on-air multiplex will change. However, in some cases the Service provider will want to reconfigure at the frame closest to a particular moment in time related to the content of a service component. The COUNTER message (see clause 6) can be used to obtain the relationship between reference data carried in the STI-D(LI) and the reference time that it is actually transmitted. Since the Service provider knows the time at which the reference data entered the collection network, it is able to calculate the total transit time through the DAB network. Alternatively, the Service provider can set the total transit time by use of the STI-D(LI) timestamp in networks that support this feature. The transit time can be used to calculate the reconfiguration instant required.

The Service provider shall set the two fields *DFCT* and *UTC* in the RCONFIG message (see clause 6) as follows:

- *DFCT* shall give the data frame count at which the new configuration shall begin;
- *UTC* shall give the time, to the nearest second, at which the new configuration shall begin.

The Ensemble provider shall interpret the two fields as follows:

- *DFCT* shall give the data frame count at which the new configuration shall begin;
- *UTC* shall give the time, plus or minus one second, at which the new configuration shall begin.

E.2.3 Requesting a reconfiguration

Reconfiguration can be requested by any Service provider (in regard to its own configuration) or by the Ensemble provider.

When a Service provider requests a reconfiguration, the Ensemble provider shall first check that the configuration requested has been defined and if so shall translate the Service provider's DFCT to the CIF count at which the reconfiguration shall occur. If a reconfiguration is permitted at this CIF count then the Ensemble provider shall verify that the new configuration is possible without disturbing any other Service provider's services (i.e. that all the required resources are available, including capacity in the MSC and FIC).

If all these criteria are fulfilled, the Ensemble provider shall accept the reconfiguration. Only the Service provider requesting the reconfiguration shall be aware that it will take place.

E.2.4 Implementing the reconfiguration

The implementation of a new configuration at the level of the STI-D(LI) involves one or more of the following:

- modification of the number of streams;
- modification of the size of existing streams.

The number of streams should only be modified at the reconfiguration instant. In this case, the length of the STI-D(LI) STC field shall be constant during the lifetime of a configuration.

The timing of the modification of existing streams shall vary according to the stream type, as follows:

- MSC sub-channel streams carry data in the form of complete sub-channels. Only changes to the length of the stream require frame accurate changes in the STI-D(LI). Reconfigurations that change the protection applied to the sub-channel without changing the STL have no effect on the STI-D(LI). Reconfigurations that vary the number of service components within an MSC packet mode stream but do not change the STL have no effect on the STI-D(LI).
- MSC sub-channel contribution streams are generally combined with other such streams from other service providers by the downstream entity to form a complete sub-channel. Changes to the length of the stream can take place at any time. The content should reflect the composition of the current configuration.
- FIC FIG streams are generally combined with other such streams from other service providers by the downstream entity to form the FIC. Changes to the length of the stream can take place at any time. The content should reflect the composition of the current configuration.
- FIC FIB streams using asynchronous insertion are generally combined with other such streams from other service providers by the downstream entity to form the FIC. Changes to the length of the stream can take place at any particular time. The content should reflect the composition of the current configuration.
- FIC FIB streams using synchronous insertion are governed by the FIB grid, and are not affected by reconfiguration. The content should reflect the composition of the current configuration.

Therefore only MSC sub-channel streams require co-ordination of their size and the reconfiguration instant. The reconfiguration timing is governed by the data frame count (*DFCT*). If the first frame of the new configuration occurs at frame *DFCT = R*, then the following rules shall be observed:

- MSC sub-channel stream joins the STI-D(LI) at frame *DFCT = R*;
- MSC sub-channel stream leaves the STI-D(LI) at frame *DFCT = R - 15*;
- MSC sub-channel stream increases in size at frame *DFCT = R*;
- MSC sub-channel stream decreases in size at frame *DFCT = R - 15*.

The STL field shall be modified at the frame indicated. The ISTC should change at frame *DFCT = R* (i.e. for an MSC sub-channel stream leaving the multiplex, the stream remains open until *DFCT = R - 1* with STL = 0 for frames *DFCT = R - 15* to *DFCT = R - 1*).

Annex F (informative): Use of the STI timestamp

The STI includes a timestamp field carried in either the STI-D(LI) or the STI(PI, G.704/1) which gives an indication of the time at which the STI data contained in each frame shall be processed by the channel encoder (at the transmitter).

Delay management is not essential in the collection network, but there are situations where it improves the quality of service delivered to the user. This annex gives some examples of the way to use the STI timestamp.

F.1 Delay between Service provider and users

A typical DAB network will be organized as shown in figure F.1.

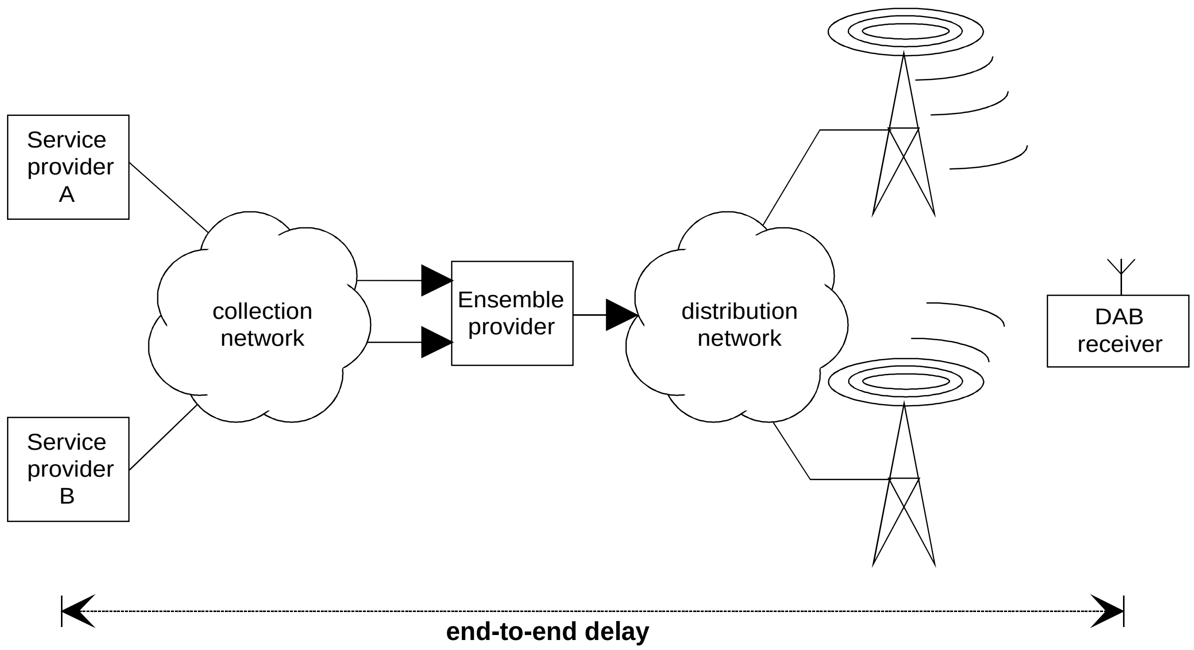

Figure F.1: Example of a typical DAB network

In such a network, several kinds of delay may be identified and these shall be managed so that the end-to-end delay remains constant and known. Figure F.2 identifies the different delay blocks.

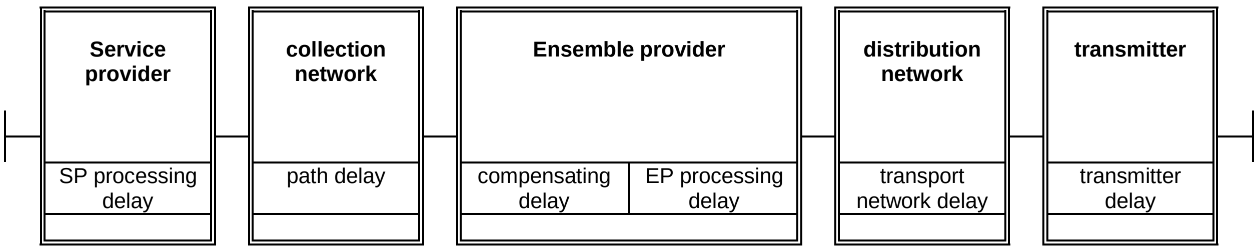

Figure F.2: Delay blocks in typical DAB network

The different delays are:

Service provider (SP) processing delay: This delay represents the delay to the input service component incurred in building the STI(PI, X) frame. It may include the audio encoding delay.

path delay: This is the delay to which the STI-D frame is subject to on its way through the collection network.

compensating delay: This delay is applied by the Ensemble provider to ensure that the service component is transmitted at the desired time. The value of the compensating delay is set by examining the timestamp in each frame.

Ensemble provider (EP) processing delay: This delay represents the delay to the service component incurred in building the ETI frame.

transport network delay: This delay is the sum of all the various delays in the distribution network. It is described in ETS 300 799 [3].

transmitter delay: This delay is the sum of all the various delays in the transmitter. It is described in ETS 300 799 [3].

F.2 Setting the timestamp value

The STI timestamp can be used by a Service provider to define the end-to-end delay that it requires. This may be done to improve the quality of the service delivered to the users. For example, if an audio programme makes reference to the time, the information is most useful if the time is correct. By managing the end-to-end delay the Service provider knows that studio time and on-air time have a constant and known relation.

The timestamp source marks all its generated STI frames with a timestamp which defines the instant at which the frames should be delivered to the channel encoder at the transmitters.

The STI receiving equipment applies the same timestamp to its output frames, either STI frames or in this case ETI frames. It may need to apply a compensating delay to bring the received STI frames into alignment with others from other Service providers, because the output timestamp applies to the whole frame. Note that the STI timestamp carries an uncertainty of 24 ms (the frame period) because the Service provider does not necessarily know the DAB time reference (although this can be obtained by using the STI-C(LI) COUNTER messages).

In order for the delay compensation to function correctly, the period of the time reference signal has to be greater than the delay through the DAB network.

F.3 Using the timestamp in multicasting

In some circumstances a Service provider may wish to send the same STI-D(LI) frame to more than one Ensemble provider. This is known as STI multicasting (see clause 4).

The timestamp allows the Service provider to know that any service components carried on all Ensembles will be broadcast at the same time (to within 24 ms), even if the distribution network delays are different for each Ensemble.

Annex G (informative):
Examples of STI-C(TA) protocol

This informative annex provides some examples and guidance on how the transport layer in the STI-C(TA) shall be used. It describes some mechanisms such as acknowledgement packets, loss of packets, opening/closing a connection etc.

In the following clauses, sequence numbers (SEQXX) are used only to reference the exchange in the text. Moreover, the repetition field of each packet is set to "0".

G.1 Opening and closing of an STI-C(TA) connection

Table G.1 shows the basic sequence used to open and close a connection (as defined in clauses 7.5.8.1 and 7.5.8.2) between entity A and B.

In SEQ01, entity A requests a connection to be opened by setting the FLAG field to "SYN". The PKTNUM field contains an arbitrary number. The ACKNUM field also contains an arbitrary number, since there is no previous exchange, and it is shown to be invalid by setting the ACK field to "X". No logical packet is carried.

In SEQ02, entity B accepts the request to open the connection by setting the FLAG field to "SYN". The PKTNUM field contains an arbitrary number. The ACKNUM field is set to acknowledge the received packet, and it is shown to be valid by setting the ACK field to "S". No logical packet is carried.

In SEQ03, entity A acknowledges the receipt of the transport packet from B and the connection is open.

Table G.1: Opening and closing an STI-C(TA) connection

	Direction	Description	Transport Packet header					Logical Packet
			PKTNUM	ACKNUM	REP	ACK	FLAG	
SEQ01	A → B	Request to open connection	789	999	0	X	SYN	no
SEQ02	A ← B	Accept the request	456	790	0	S	SYN	no
SEQ03	A → B	Connection opened	789	457	0	S	XXX	no
….	…..	…..	….	….	….	….	….	….
SEQ04	A ← B	Request to close connection	512	845	0	S	END	no
SEQ05	A → B	Accept the request	845	513	0	S	END	no
SEQ06	A ← B	Connection closed	512	846	0	S	XXX	no
….	…..	…..	….	….	….	….	….	….
SEQ07	A ← B	Request to close the connection	512	845	0	S	END	no
SEQ08	A → B	A sends its remaining message	845	513	0	S	XXX	yes
SEQ09	A ← B	Acknowledge of packet 845 and request to close the connection	513	846	0	S	END	no
SEQ10	A → B	Accept the request	846	514	0	S	END	no
SEQ11	A ← B	Connection closed	513	847	0	S	XXX	no

The request to close the connection can be initiated by any of the two entities. In the example given in table G.1 two possible scenarios are given.

In SEQ04, entity B requests to close the connection by setting the FLAG field to "END". Although no logical packet is carried, the transport packet is carrying a payload because of the flag field setting, and so the PKTNUM is incremented. All the other fields are coded as if for an open connection.

In SEQ05, entity A accepts the request by setting the FLAG field to "END". Again no logical packet is carried, but the transport packet is carrying a payload and so the PKTNUM is incremented. All the other fields are coded as if for an open connection.

In SEQ06, entity B acknowledges the receipt of the transport packet from B and the connection is closed.

In SEQ07, entity B requests to close the connection by setting the FLAG field to "END".

In SEQ08, entity A does not accept the request because it still has data to send. The transport packet is coded as for an open connection.

In SEQ09, entity B acknowledges the receipt of the transport packet from B but still wants to close the connection and so restarts the closure sequence by again setting the FLAG field to "END".

In SEQ10, entity A accepts the request by setting the FLAG field to "END".

In SEQ11, entity B acknowledges the receipt of the transport packet from B and the connection is closed.

G.2 Transmission on an open connection

As long as there is a connection open between entities, they shall be allowed to exchange transport layer packets. Each transport packet may contain a logical packet and if it does the transport packet shall be acknowledged by the remote entity. The remote entity may carry a logical packet in the transport packet used in the acknowledgement of a received transport packet. This is illustrated in the example given in table G.2.

Table G.2: Transmission on an open connection

	Direction	Description	Transport Packet header					Logical Packet
			PKTNUM	ACKNUM	REP	ACK	FLAG	
….	…..	…..	….	….	….	….	….	….
SEQ01	A → B	A sends logical packet	102	502	0	S	XXX	yes
SEQ02	A ← B	B acknowledges packet 102 and sends logical packet	502	103	0	S	XXX	yes
SEQ03	A → B	A acknowledges packet 502 and sends logical packet	103	503	0	S	XXX	yes
SEQ04	A ← B	B acknowledges packet 103	502	104	0	S	XXX	no
….	…..	…..	….	….	….	….	….	….

In SEQ01, entity A sends a transport packet carrying a logical packet to entity B.

In SEQ02, entity B acknowledges the received transport packet and at the same time sends a logical packet to entity A.

In SEQ03, this process is reversed.

In SEQ04, entity B acknowledges the received transport packet, but does not send a logical packet to entity A.

The transport layer also offers the possibility of using a single transport packet to acknowledge the reception of several transport packets carrying logical data packets. This is illustrated in table G.3.

Table G.3: Acknowledge of multiple logical data packets

	Direction	Description	Transport Packet header					Logical Packet
			PKTNUM	ACKNUM	REP	ACK	FLAG	
….	…..	…..	….	….	….	….	….	….
SEQ01	A → B	A sends logical packet	987	513	0	S	XXX	yes
SEQ02	A → B	A sends logical packet	988	513	0	S	XXX	yes
SEQ03	A → B	A sends logical packet	989	513	0	S	XXX	yes
SEQ04	A ← B	B acknowledges packets 987, 988, 989	512	990	0	S	XXX	no
SEQ05	A → B	A sends logical packet	990	513	0	S	XXX	yes
SEQ06	A → B	A sends logical packet	991	513	0	S	XXX	yes
SEQ07	A → B	A sends logical packet	992	513	0	S	XXX	yes
SEQ08	A ← B	B acknowledges packets 990, 991, 992 and sends logical packet	513	993	0	S	XXX	yes
SEQ09	A ← B	B sends logical packet	514	993	0	S	XXX	yes
SEQ10	A → B	A acknowledges packet 514	992	515	0	S	XXX	no
….	…..	…..	….	….	….	….	….	….

In SEQ04, entity B acknowledges all three received transport packets by setting the ACKNUM field to the packet number that is next expected from entity A.

In SEQ08, entity B acknowledges all three received transport packets and at the same time sends a logical packet.

In SEQ09, entity B sends another transport packet carrying a logical packet.

In SEQ10, entity A acknowledges both these packets.

G.3 Handling loss of packets

The sequence numbers embedded in PKTNUM and ACKNUM fields shall be used to detect the loss of packets during transmission and to perform automatic retransmission of the interrupted sequence of packets.

In table G.4, two situations are shown.

Table G.4: Automatic retransmission of packet loss

	Direction	Description	Transport Packet header					Logical Packet
			PKTNUM	ACKNUM	REP	ACK	FLAG	
....	.....	.....	....	....	....	....	....	....
SEQ01	A → B	A sends logical packet	998	513	0	S	XXX	yes
SEQ02	A → ⊠	A sends logical packet, but it is never received by B	999	513	0	S	XXX	yes
SEQ03	A → B	A sends logical packet	000	513	0	S	XXX	yes
SEQ04	A ← B	B acknowledges packet 998	513	999	0	S	XXX	yes
SEQ05	A → B	A sends logical packet again	999	514	0	S	XXX	yes
SEQ06	A → B	A sends logical packet	001	514	0	S	XXX	yes
SEQ07	A ← B	B acknowledges packets 999, 000, 001.	513	002	0	S	XXX	no
SEQ08	A → B	A sends logical packet	002	514	0	S	XXX	yes
SEQ09	A → B	A sends logical packet	003	514	0	S	XXX	yes
SEQ10	⊠ ← B	B acknowledges packets 002 and 003. This packet is never received by A	513	004	0	S	XXX	no
		A time out occurs at A since no acknowledge has been received of 002 and 003						
SEQ11	A → B	A sends logical packet again	002	514	0	S	XXX	yes
SEQ12	A ← B	B acknowledges packets 002, 003.	514	004	0	S	XXX	yes
....	.....	.....	....	....	....	....	....	....

In SEQ01, entity A sends a transport packet containing a logical packet, which is received by entity B.

In SEQ02, entity A sends a transport packet containing a logical packet, but the packet is lost and so is never received by entity B.

In SEQ03, entity A sends a transport packet containing a logical packet, which is received by entity B. Entity B is able to detect an interruption in the PKTNUM sequence (i.e. "998" then "000"). Entity B may choose to store packet "000" on a receive packet stack, or may choose to discard the packet.

In SEQ04, entity B signals the packet loss by acknowledging packet "998" as this is the last one received in the correct sequence. Entity A receives the packet with the ACKNUM value less than its current PKTNUM value and therefore knows that packet loss has occurred.

In SEQ05, entity A retransmits packet "999" and acknowledges receipt of packet "513" from entity B. Note that the ACKNUM is different to that in SEQ02.

In SEQ06, entity A sends the next logical packet.

In SEQ07, entity B acknowledges packets "999", "000" and "001". In this case packet "000" was retrieved from a receive packet stack. If entity B has discarded packet "000" then the acknowledgement would be for packet "999" and entity A would know to retransmit all the packets.

In SEQ08 and SEQ09 entity A sends more logical packets.

In SEQ10, entity B acknowledges packets "002" and "003" but this packet is lost and entity A does not receive it.

In SEQ11, entity A retransmits logical packet "002" because no acknowledgement has been received within the time-out period. Entity B discards the received logical packet since he has already received it.

In SEQ12, entity B acknowledges packets "002" and "003" and also sends a logical packet to entity A.

Annex H (informative): Use of STI(PI,G.704/1) on T1 networks

H.1 Introduction

STI(PI, G.704/1), see clause 9.5, has been designed for use with telecommunication network hierarchies which are based on 2 048 kbit/s networks. However, the DAB system will also be used in countries which have a hierarchy based on 1 544 kbit/s (often referred to as G.704-T1).

This annex gives some guidelines for transporting an STI(LI) on such telecommunication networks and includes a suggestion for a suitable G.704 adaptation, STI(PI, G.704/1-T1).

H.2 General outline of STI(PI, G.704/1-T1)

The STI(PI, G.704/1-T1) layer described in this clause is suitable for use on networks based on the first level of the PDH (1 544 kbit/s) as defined by the ITU-T in Recommendation G.704 [7]. As well as the G.704 signalling and synchronization bytes, STI(PI, G.704/1-T1) includes timestamps to permit compensation for the effect of differential network delays. It also uses Reed-Solomon forward error coding to allow the correction of transport network errors.

STI(PI, G.704/1-T1) has two variants: STI(PI, G.704/1-T1)$_{4\,464}$ and STI(PI, G.704/1-T1)$_{4\,320}$. The two differ in the balance between their data capacity and the number of bytes which are reserved for forward error correction.

In the G.704-T1 frame structure, 1 536 kbit/s are available to carry user data, the remaining 8 kbit/s are reserved by the G.704-T1 network.

In each 24 ms multiframe, STI(PI, G.704/1-T1)$_{4\,464}$, has the capacity to carry 4 608 bytes allocated as follows:

- 4 464 bytes of STI(PI, X) data (a data capacity of 1 488 kbit/s);
- 96 bytes (32 kbit/s) for forward error correction;
- 48 bytes (16 kbit/s) for management and signalling.

In each 24 ms multiframe, STI(PI, G.704/1-T1)$_{4\,320}$, has the capacity to carry 4 608 bytes allocated as follows:

- 4 320 bytes of STI(PI, X) data (a data capacity of 1 440 kbit/s);
- 240 bytes (80 kbit/s) for forward error correction;
- 48 bytes (16 kbit/s) for management and signalling.

H.3 Transparency of STI(PI, G.704/1-T1) layer to STI(PI, X)

H.3.1 Transparency of STI(PI, G.704/1-T1)$_{4\,464}$ layer to STI(PI, X)

STI(PI, G.704/1-T1)$_{4\,464}$ can carry up to 4 464 bytes of the STI(PI, X) data frame, from $B_{[-4..4\,459]}$. The length of the STI(PI, X) frame, TFL, shall be 4 468.

NOTE: The SYNC field is not carried by this physical interface.

STI(PI, G.704/1-T1)$_{4\,464}$ is also able to carry the STI-D(LI) STAT field but the content of this field may be amended by the STI(PI, G.704/1-T1) receiving equipment.

H.3.2 Transparency of STI(PI, G.704/1-T1)$_{4\,320}$ layer to STI(PI, X)

STI(PI, G.704/1-T1)$_{4\,320}$ can carry up to 4 320 bytes of the STI(PI, X) data frame, from $B_{[-4..4\,315]}$. The length of the STI(PI, X) frame, TFL, shall be 4 324.

NOTE: The SYNC field is not carried by this physical interface.

STI(PI, G.704/1-T1)$_{4\,320}$ is also able to carry the STI-D(LI) STAT field but the content of this field may be amended by the STI(PI, G.704/1-T1) receiving equipment.

H.4 STI(PI, G.704/1-T1) structure

The purpose of the STI(PI, G.704/1-T1) multiframe structure is to map the STI(PI, X) frame onto the G.704 frame structure. In the ITU-T Recommendation G.704 [7], the 1 544 kbit/s data stream is organized into frames. Each frame has a nominal duration of 125 µs and is made up of 193 bits organized into 24 1-byte timeslots plus 1 signalling bit which is used by G.704. All 24 timeslots per 125 µs frame are available to carry user data.

The STI(PI, G.704/1-T1) multiframe has a FL of 24 ms. It consists of 192 G.704 frames, equivalent to 4 608 timeslots, or bytes. The multiframe structure is illustrated in figure H.1 and consists of:

- 3 superblocks ($s_{[0..2]}$) of 1 536 bytes each;
- each superblock consists of 8 blocks ($bl_{[0..7]}$) of 192 bytes each;
- each block consists of 8 G.704 frames ($f_{[0..7]}$) of 24 bytes each;
- each G.704 frame consists of 24 timeslots ($ts_{[0..23]}$) of 1 byte each;
- each timeslot consists of 8 bits ($b_{[0..7]}$).

Table H.1 summarizes the relation of elements within a multiframe.

Table H.1: Relation of elements within an STI(PI, G.704/1-T1) multiframe

	Superblocks	Blocks	G.704 frames	Timeslots	Bits
Multiframe	3	24	192	4 608	36 864
Superblock	1	8	64	1 536	12 288
Block		1	8	192	1 536
G.704 frame			1	24	192
Timeslot				1	8

Figures H.2 and H.3 show the method used to build the coding array. The general method follows the outline given in clause 9.5.

H.5 Error protection for STI(PI, G.704/1-T1)

In both variants of STI(PI, G.704/1-T1), Reed-Solomon coding may be used to provide data which may be analysed by the interface receiving equipment to detect and correct errors occurring on the Transport Network.

The Reed-Solomon code should use the same general scheme described in clause 9.5.6 but with the codeword length shortened from 240 bytes to 192 bytes.

Using the terminology of clause 9.5.6:

For STI(PI, G.704/1-T1)$_{4\,464}$, R = 4 resulting in an RS(188,192) code.

For STI(PI, G.704/1-T1)$_{4\,320}$, R = 10 resulting in an RS(182,192) code.

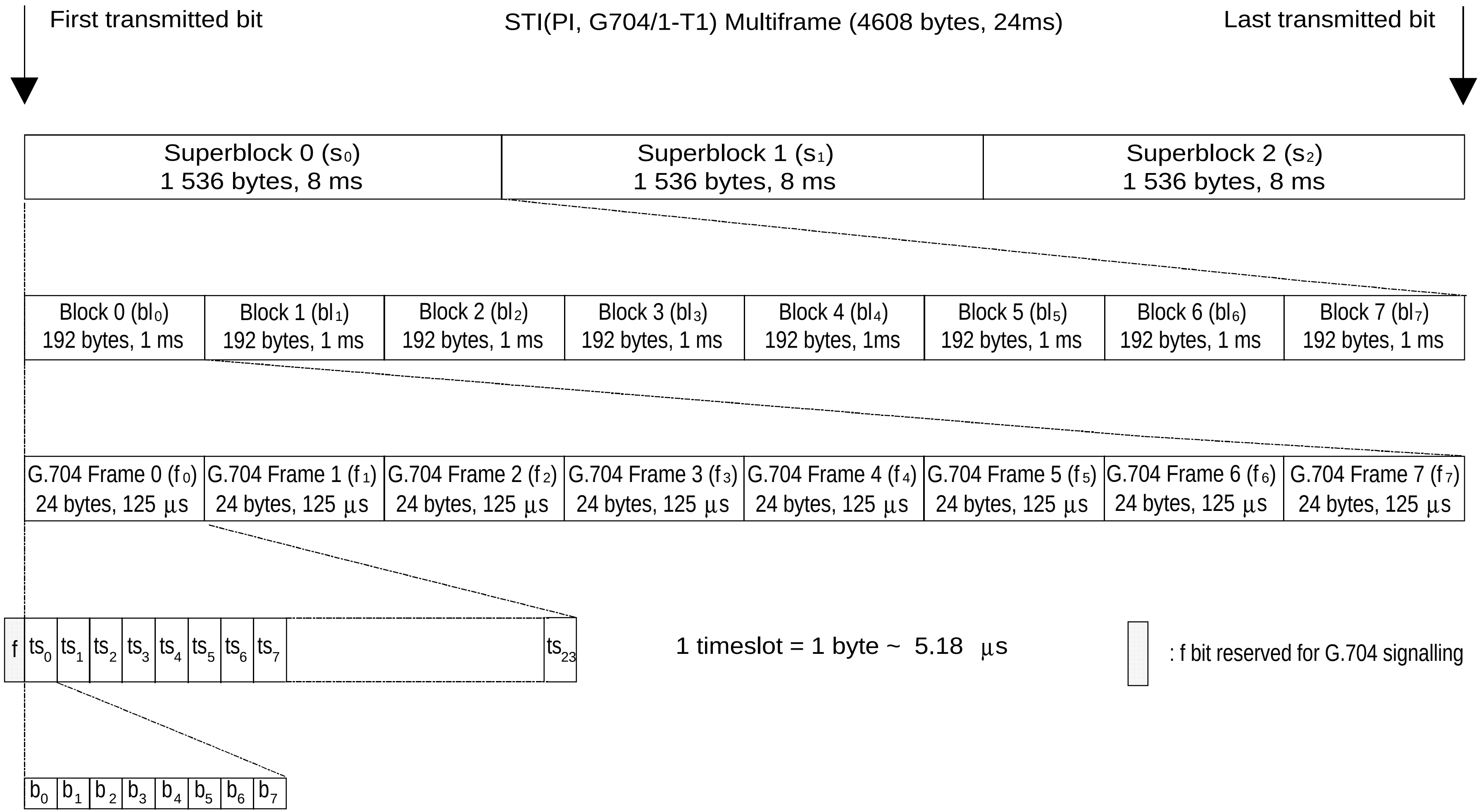

Figure H.1: STI(PI, G.704/1-T1) multiframe structure

Index		**0**	**1..23**	**24**	**25..47**	**48**	**49..71**	**72**	**73..95**	**96**	**97..119**	**120**	**121..143**	**144**	**145..167**	**168**	**169..187**	**188..191**
i	**0**	$M_{0,0}$	$B_{-4}..B_{18}$	$M_{1,0}$	$B_{19}..B_{41}$	$M_{2,0}$	$B_{42}..B_{64}$	$M_{3,0}$	$B_{65}..B_{87}$	$M_{4,0}$	$B_{88}..B_{110}$	$M_{5,0}$	$B_{111}..B_{133}$	$M_{6,0}$	$B_{134}..B_{156}$	$M_{7,0}$	$B_{157}..B_{175}$	$R_{0,188..191}$
	1	$S_{0,0}$	$B_{176}..B_{198}$	$S_{1,0}$	$B_{199}..B_{221}$	$S_{2,0}$	$B_{222}..B_{244}$	$S_{3,0}$	$B_{245}..B_{267}$	$S_{4,0}$	$B_{268}..B_{290}$	$S_{5,0}$	$B_{291}..B_{313}$	$S_{6,0}$	$B_{314}..B_{336}$	$S_{7,0}$	$B_{337}..B_{355}$	$R_{1,188..191}$
	2	B_{356}	$B_{357}..$	B_{380}	$B_{381}..$	B_{404}	$B_{405}..$	B_{428}	$B_{429}..$	B_{452}	$B_{453}..$	B_{476}	$B_{477}..$	B_{500}	$B_{501}..$	B_{524}	$..B_{543}$	$R_{2,188..191}$
	3	B_{544}	$B_{545}..$	B_{568}	$B_{569}..$	B_{592}	$B_{593}..$	B_{616}	$B_{617}..$	B_{640}	$B_{641}..$	B_{664}	$B_{665}..$	B_{688}	$B_{689}..$	B_{712}	$..B_{731}$	$R_{3,188..191}$
	4	B_{732}	$B_{733}..$	B_{756}	$B_{757}..$	B_{780}	$B_{781}..$	B_{804}	$B_{805}..$	B_{828}	$B_{829}..$	B_{852}	$B_{853}..$	B_{876}	$B_{877}..$	B_{900}	$..B_{919}$	$R_{4,188..191}$
	5	B_{920}	$B_{921}..$	B_{944}	$B_{945}..$	B_{968}	$B_{969}..$	B_{992}	$B_{993}..$	$B_{1\,016}$	$B_{1\,017}..$	$B_{1\,040}$	$B_{1\,041}..$	$B_{1\,064}$	$B_{1\,065}..$	$B_{1\,088}$	$..B_{1\,107}$	$R_{5,188..191}$
	6	$B_{1\,108}$	$B_{1\,109}..$	$B_{1\,132}$	$B_{1\,133}..$	$B_{1\,156}$	$B_{1\,157}..$	$B_{1\,180}$	$B_{1\,181}..$	$B_{1\,204}$	$B_{1\,205}..$	$B_{1\,228}$	$B_{1\,229}..$	$B_{1\,252}$	$B_{1\,253}..$	$B_{1\,276}$	$..B_{1\,295}$	$R_{6,188..191}$
	7	$B_{1\,296}$	$B_{1\,297}..$	$B_{1\,320}$	$B_{1\,321}..$	$B_{1\,344}$	$B_{1\,345}..$	$B_{1\,368}$	$B_{1\,369}..$	$B_{1\,392}$	$B_{1\,393}..$	$B_{1\,416}$	$B_{1\,417}..$	$B_{1\,440}$	$B_{1\,441}..$	$B_{1\,464}$	$..B_{1\,483}$	$R_{7,188..191}$
	8	$M_{0,1}$	$B_{1\,484}..$	$M_{1,1}$	$B_{1\,507}..$	$M_{2,1}$	$B_{1\,530}..$	$M_{3,1}$	$B_{1\,553}..$	$M_{4,1}$	$B_{1\,576}..$	$M_{5,1}$	$B_{1\,599}..$	$M_{6,1}$	$B_{1\,622}..$	$M_{7,1}$	$..B_{1\,663}$	$R_{0,188..191}$

p	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**..**	**1 535**	**..**
	$M_{0,0}$	$S_{0,0}$	B_{356}	B_{544}	B_{732}	B_{920}	$B_{1\,108}$	$B_{1\,296}$	B_{-4}	B_{176}	B_{357}	B_{545}	B_{733}	B_{921}	$B_{1\,109}$	$B_{1\,297}$	B_{-3}	B_{177}	..	$R_{7,191}$	..

Figure H.2: Steps in the formation of STI(PI, G.704/1-T1)$_{4\,464}$

Index																		
		0	**1..23**	**24**	**25..47**	**48**	**49..71**	**72**	**73..95**	**96**	**97..119**	**120**	**121..143**	**144**	**145..167**	**168**	**169..181**	**182..191**
i	**0**	$M_{0,0}$	B_{-4}..B_{16}	$M_{1,0}$	B_{19}..B_{41}	$M_{2,0}$	B_{42}..B_{64}	$M_{3,0}$	B_{65}..B_{87}	$M_{4,0}$	B_{88}..B_{110}	$M_{5,0}$	B_{111}..B_{133}	$M_{6,0}$	B_{134}..B_{156}	$M_{7,0}$	B_{157}..B_{169}	$R_{0,188..191}$
	1	$S_{0,0}$	B_{170}..B_{192}	$S_{1,0}$	B_{193}..B_{215}	$S_{2,0}$	B_{216}..B_{238}	$S_{3,0}$	B_{239}..B_{261}	$S_{4,0}$	B_{262}..B_{284}	$S_{5,0}$	B_{285}..B_{307}	$S_{6,0}$	B_{308}..B_{330}	$S_{7,0}$	B_{331}..B_{343}	$R_{1,188..191}$
	2	B_{344}	B_{345}..	B_{368}	B_{369}..	B_{392}	B_{393}..	B_{416}	B_{417}..	B_{440}	B_{441}..	B_{464}	B_{465}..	B_{488}	B_{489}..	B_{512}	..B_{525}	$R_{2,188..191}$
	3	B_{526}	B_{527}..	B_{550}	B_{551}..	B_{574}	B_{575}..	B_{598}	B_{599}..	B_{622}	B_{623}..	B_{646}	B_{647}..	B_{670}	B_{671}..	B_{694}	..B_{707}	$R_{3,188..191}$
	4	B_{708}	B_{709}..	B_{732}	B_{733}..	B_{756}	B_{757}..	B_{780}	B_{781}..	B_{804}	B_{805}..	B_{828}	B_{829}..	B_{852}	B_{853}..	B_{876}	..B_{889}	$R_{4,188..191}$
	5	B_{890}	B_{891}..	B_{914}	B_{915}..	B_{938}	B_{939}..	B_{962}	B_{963}..	B_{986}	B_{987}..	$B_{1\,010}$	$B_{1\,011}$..	$B_{1\,034}$	$B_{1\,035}$..	$B_{1\,058}$	..$B_{1\,071}$	$R_{5,188..191}$
	6	$B_{1\,072}$	$B_{1\,073}$..	$B_{1\,096}$	$B_{1\,097}$..	$B_{1\,120}$	$B_{1\,121}$..	$B_{1\,144}$	$B_{1\,145}$..	$B_{1\,168}$	$B_{1\,169}$..	$B_{1\,192}$	$B_{1\,193}$..	$B_{1\,216}$	$B_{1\,217}$..	$B_{1\,240}$	..$B_{1\,253}$	$R_{6,188..191}$
	7	$B_{1\,254}$	$B_{1\,255}$..	$B_{1\,278}$	$B_{1\,279}$..	$B_{1\,302}$	$B_{1\,303}$..	$B_{1\,326}$	$B_{1\,327}$..	$B_{1\,350}$	$B_{1\,351}$..	$B_{1\,374}$	$B_{1\,375}$..	$B_{1\,398}$	$B_{1\,399}$..	$B_{1\,422}$	..$B_{1\,435}$	$R_{7,188..191}$
	8	$M_{0,1}$	$B_{1\,436}$..	$M_{1,1}$	$B_{1\,459}$..	$M_{2,1}$	$B_{1\,482}$..	$M_{3,1}$	$B_{1\,505}$..	$M_{4,1}$	$B_{1\,528}$..	$M_{5,1}$	$B_{1\,551}$..	$M_{6,1}$	$B_{1\,574}$..	$M_{7,1}$	..$B_{1\,609}$	$R_{0,188..191}$

p	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**..**	**1 535**	**..**
	$M_{0,0}$	$S_{0,0}$	B_{344}	B_{526}	B_{708}	B_{890}	$B_{1\,072}$	$B_{1\,254}$	B_{-4}	B_{170}	B_{345}	B_{527}	B_{709}	B_{891}	$B_{1\,073}$	$B_{1\,255}$	B_{-3}	B_{171}	..	$R_{7,191}$	..

Figure H.3: Steps in the formation of STI(PI, G.704/1-T1)$_{4\,320}$

Annex I (informative): Bibliography

EUREKA Project 147 (1997): "Digital Audio Broadcasting System: Definition of the Service Transport Interface".

EBU Recommendation R79 (1994): "Recommended system for digital sound broadcasting to mobile, portable and fixed receivers in the appropriate frequency bands in the range 30 MHz to 3 GHz".

ITU-R Recommendation BS.774 (1994): "Digital sound broadcasting to vehicular, portable and fixed receivers using terrestrial transmitters in the VHF/UHF bands".

ITU-R Recommendation BO.789 (1994): "Digital sound broadcasting to vehicular, portable and fixed receivers for BSS (sound) bands in the frequency range 500 - 3 000 MHz".

History

Document history			
V1.1.1	February 1999	Publication	
V1.2.1	January 2005	One-step Approval Procedure	OAP 20050527: 2005-01-26 to 2005-05-27
V1.2.1	May 2005	Publication	