

Specification for

ControlNet —

Part 3: Data link layer

ICS 25.040.40; 35.100.20

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW

BSI
Documents

AVA

Document Sharing Hub

Committees responsible for this Draft for Development

The preparation of this Draft for Development was entrusted to Technical Committee AMT/7, Monitoring and control aspects of AMT, upon which the following bodies were represented:

- ABB Kent plc
- Cegelec Industrial Controls Ltd.
- Department of Trade and Industry
- Fisher-Rosemount Ltd.
- Foxboro Great Britain
- Institution of Electrical Engineers
- Measurement Technology Ltd.
- Phoenix Contact Ltd.
- Process Automation and Computer Systems Ltd.
- Rockwell Automation Ltd.
- Schneider Ltd.
- Siemens Automation Ltd.
- University of Warwick
- Wago Ltd.

This Draft for Development, having been prepared under the direction of the Advanced Manufacturing Technology Assembly, was published under the authority of the Standards Board and comes into effect on 15 November 1997

© BSI 07-1999

The following BSI reference relates to the work on this publication:
Committee reference AMT/7

ISBN 0 580 28680 0

Amendments issued since publication

Amd. No.	Date	Comments

Contents

	Page
Committees responsible	Inside front cover
Foreword	iii
0 Introduction	1
1 Scope	5
2 MAC frame	6
2.1 Format	6
2.2 Preamble	6
2.3 Delimiters	7
2.4 Bytes and ordering	7
2.5 Total size of all Lpackets	7
2.6 Cyclic redundancy check	7
2.7 Abort MAC frame	7
3 Lpacket	7
3.1 Format	7
3.2 Size	8
3.3 Control	8
3.4 Tag	9
3.5 Link data	10
4 Modelling language (informative)	10
5 External interfaces	11
5.1 Transmit services	11
5.2 Queue maintenance services	12
5.3 Tag filter management	12
5.4 Receive services	13
5.5 Network synchronisation service	13
5.6 Synchronised parameter change	13
5.7 Event reports	14
5.8 Bad CRC indication	15
5.9 Current moderator indication	15
5.10 Power-up	16
5.11 Enable moderator	16
5.12 Listen only	16
6 Access Control Machine (ACM)	16
7 TxLLC	37
8 RxLLC	42
9 Transmit Machine (TxM)	45
10 Receive Machine (RxM)	49
11 Serialiser	56
12 Deserialiser	59
12.1 Byte construction	59
12.2 CRC checking	59
12.3 End of frame processing	59
13 DLL management	59
Figure 3-1 — NUT structure	1
Figure 3-2 — Media access during scheduled time	2
Figure 3-3 — Media access during unscheduled time	3
Figure 3-4 — Data Link Layer Internal Architecture	5
Figure 3-5 — Relationship to the ISO/OSI model	6

	Page
Figure 3-6 — MAC frame format	6
Figure 3-7 — Aborting a MAC frame during transmission	7
Figure 3-8 — Lpacket format	8
Figure 3-9 — Fixed tag Lpacket format	10
Figure 3-10 — Generic tag Lpacket format	10
Table 3-1 — Data Link Layer components	4
Table 3-2 — Link pad size	9
Table 3-3 — Fixed tag definitions	9
Table 3-4 — DLL events	15

Foreword

The DD 241 specification was developed by the Allen-Bradley Company, Inc. as “ControlNet”. Allen-Bradley has granted BSI a right to publish the specification as an open communications standard for digital communications networks in industrial control applications.

This part of DD 241 describes the General Description of the deterministic control network.

This publication consists of the following parts:

- *Part 1: General description;*
- *Part 2: Physical layer and media;*
- *Part 3: Data link layer;*
- *Part 4: Network and transport layer;*
- *Part 5: Data management;*
- *Part 6: Object model;*
- *Part 7: Device profiles;*
- *Part 8: Electronic data sheet;*
- *Part 9: Station management;*
- *Part 10: Object library.*

This Draft for Development is not to be regarded as a British Standard.

It is being issued in the Draft for Development series of publications and is of a provisional nature because the subject matter is still under development requiring wider exposure. It should be applied on this provisional basis so that information and experience of its practical application may be obtained.

A review of this Draft for Development will be carried out not later than 2 years after its Publication. Notification of the start of the review period, with a request for the submission of comments from users of this Draft for Development, will be made in an announcement in *Update Standards*.

According to the replies received the responsible BSI Committee will judge whether the Draft for Development can be converted into a British Standard or what other action should be taken.

Observations which it is felt should receive attention before the official call for comments will be welcomed.

These should be sent to the Secretary of BSI Technical Committee AMT/7 at BSI, 389 Chiswick High Road, London, W4 4AL, giving the document reference and clause number and proposing, where possible, an appropriate revision of the text.

Disclaimer of warranties

This document is provided on an “as is” basis without warranty by BSI and Allen-Bradley as to the accuracy or completeness of the information it contains or as to the results which may be achieved through the use of this information. Further, BSI and Allen-Bradley are not providing any warranties as to whether products produced based on DD 241 may infringe the intellectual property rights of any third parties except that, at the date of publication, to the best of BSI’s and Allen-Bradley’s knowledge, no written claims have been made against BSI and/or Allen-Bradley alleging that any such products would infringe the intellectual property rights of any third parties. In no event shall BSI and/or Allen-Bradley be responsible for any losses, damages, costs or expenses incurred as a result of the information contained in DD 241.

Summary of pages

This document comprises a front cover, an inside front cover, pages i to iv, pages 1 to 61 and a back cover.

This standard has been updated (see copyright date) and may have had amendments incorporated. This will be indicated in the amendment table on the inside front cover.

0 Introduction

The primary task of the Data Link Layer (DLL) is to determine, in co-operation with other Data Link Layers on the same link, the granting of permission to transmit on the medium. At its upper interface, the DLL provides services to receive and deliver service data units (SDUs) for the Network and Transport Layer entity.

The DLL protocol is based on a fixed, repetitive time cycle called a NUT (Network Update Time). The NUT is maintained in close synchronism among all nodes on the link. A node is not permitted access to transmit on the medium if its NUT does not agree with the NUT currently being used on a link wide basis. Different links may have different NUT duration. The NUT can be configured for a duration ranging from 2 to 100 milliseconds.

Each node contains its own timer synchronised to the NUT for the local link. Media access is determined by local sub-division of the NUT into access slots. Access to the medium is in sequential order based on the MAC ID of the node. Each node has a MAC ID in the range 1 to 99; however, specific behaviours have been incorporated into access protocol allowing a node which temporarily assumes a MAC ID of zero to perform link maintenance. The MAC ID numbers of all nodes on a link are unique. The DLL detects the presence of a duplicate MAC ID and immediately stops transmitting.

An implicit token passing mechanism is used to grant access to the medium. Each node monitors the source MAC ID of each MAC frame received. At the end of a MAC frame, each DLL sets an "implicit token register" to the received source MAC ID + 1. If the implicit token register is equal to the node's own MAC ID, then that node may now transmit one MAC frame. In all other cases, the node watches for either a new MAC frame from the node identified by the "implicit token register" or a time-out value if the identified node fails to transmit. In each case, the "implicit token" is automatically advanced to the next MAC ID. All nodes have the same value in their "implicit token register" preventing collisions on the medium.

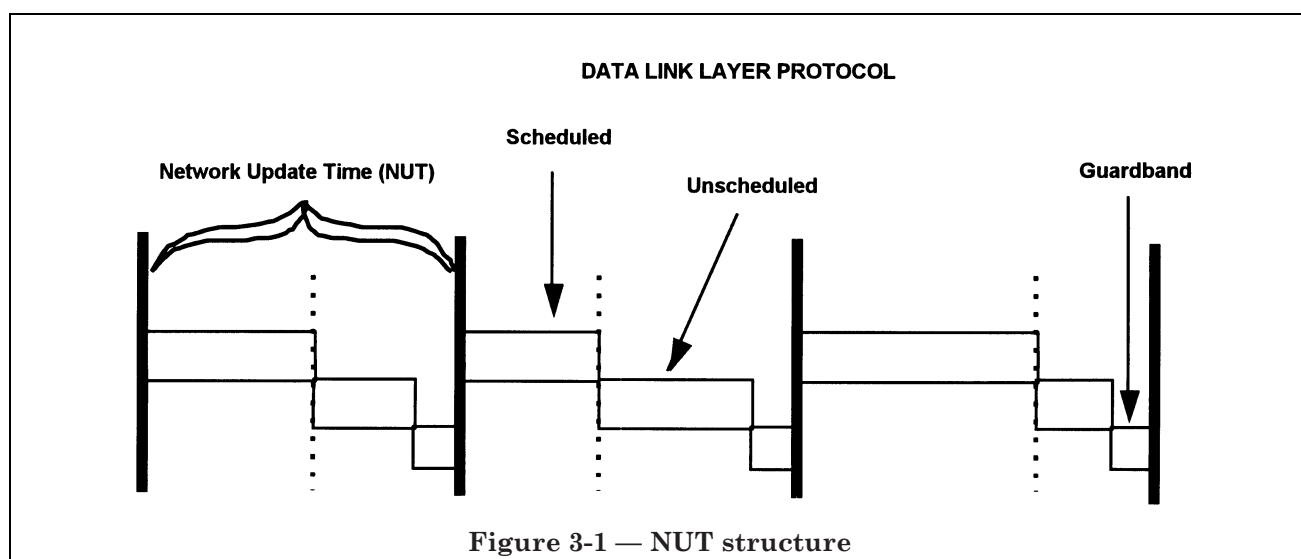
The time-out period (called the "slot time") is based on the amount of time required for

- the current node to hear the end of the transmission from the previous node;
- the current node to begin transmitting;
- the next node to hear the beginning of the transmission from the current node.

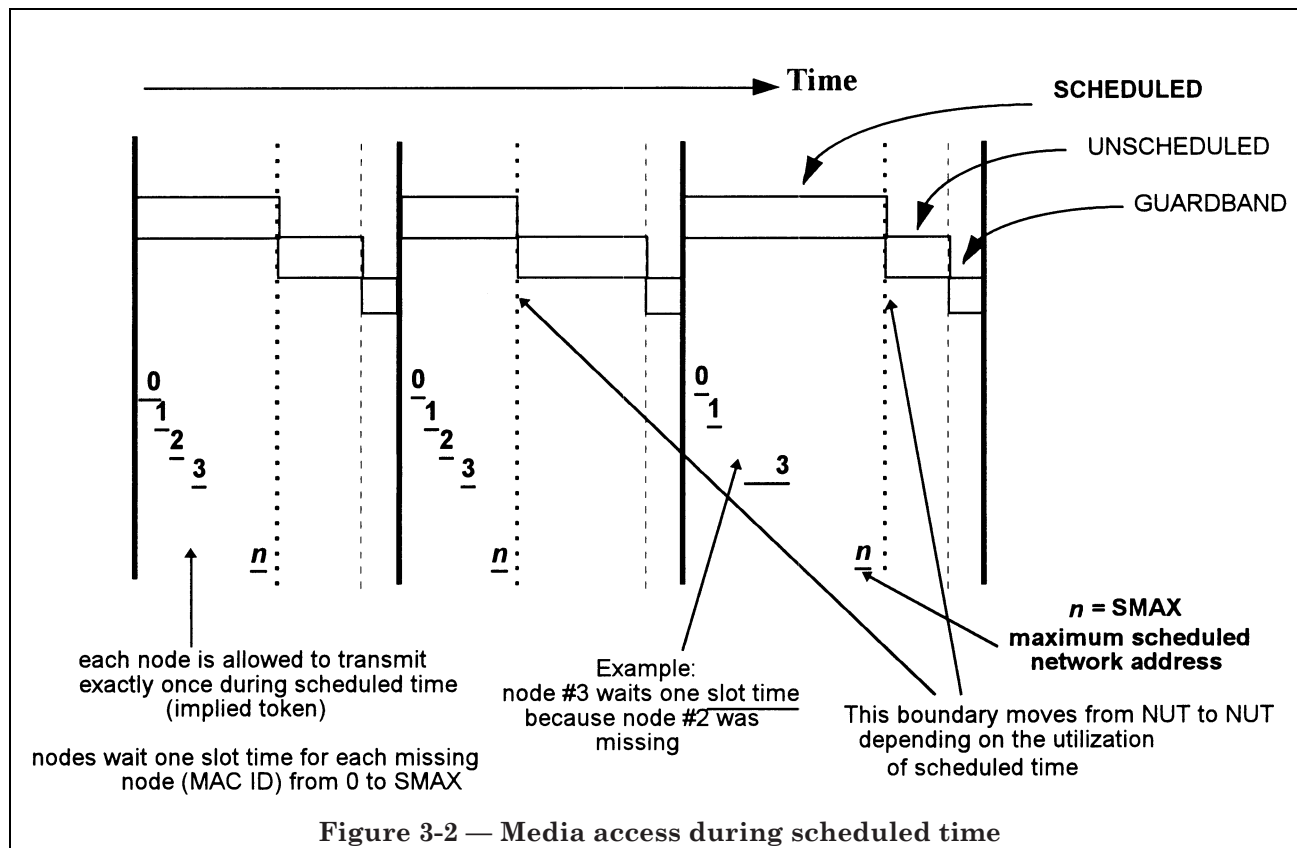
The slot time is adjusted to compensate for the total length of the medium since the propagation delay of the medium effects the first and last item on the previous list.

NOTE The calculation of slot time is specified in part 9, Station Management.

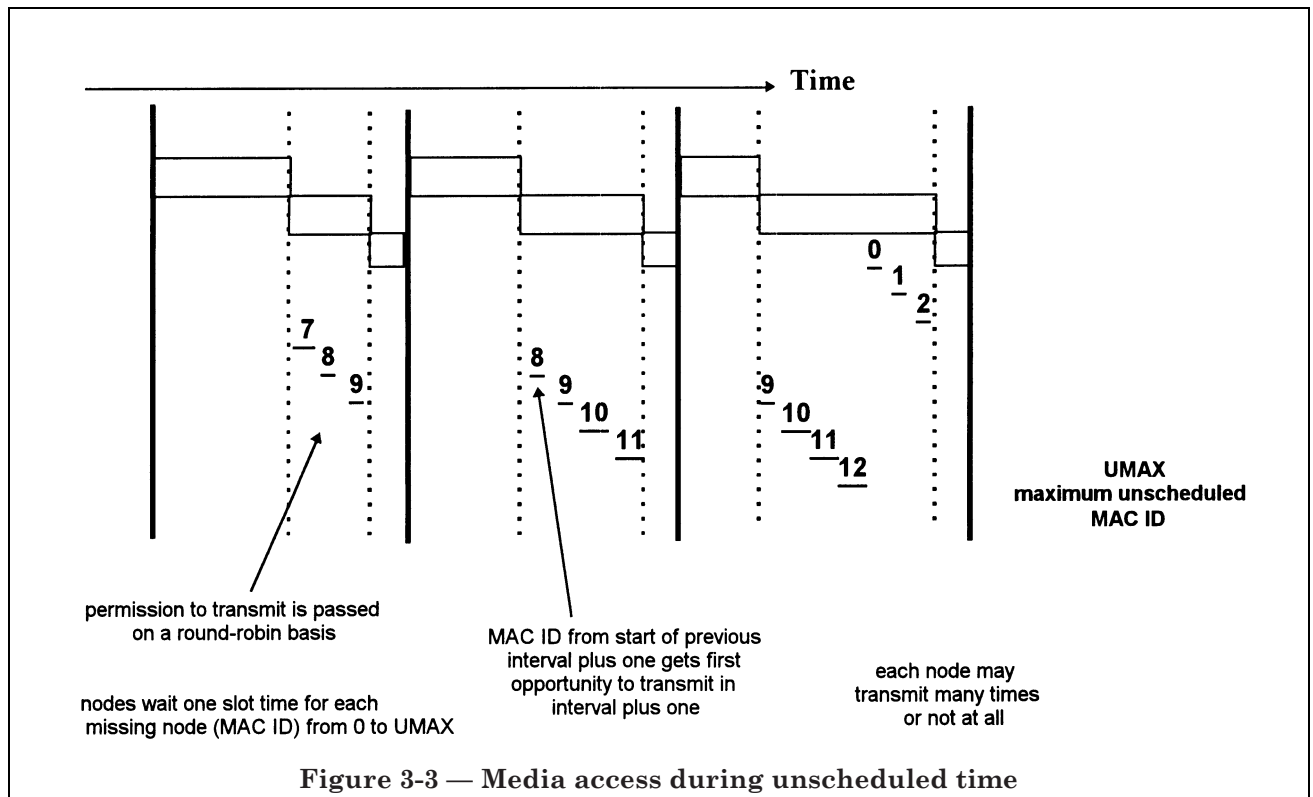
Each NUT is divided into three major parts: scheduled, unscheduled, and guardband as shown in Figure 3-1. This sequence is repeated in every NUT. The implicit token passing mechanism is used to grant access to the medium during both the unscheduled and scheduled intervals.



During the scheduled part of the NUT, each node, starting with node 0 and ending with node SMAX, gets a chance to transmit time-critical (scheduled) data. SMAX is the MAC ID of the highest numbered node that has access to the medium during the scheduled part of the NUT. Every node between 0 and SMAX has only one opportunity to send scheduled data in each NUT. The opportunity to access the medium during the scheduled time is the same for each node in every NUT. This allows data that is transmitted during the scheduled portion of the NUT to be sent in a predictable and deterministic manner. Figure 3-2 shows how the permission to transmit is granted during the scheduled time. The amount of data that each node may transmit during this first token pass is regulated by the Network and Transport Layer.



During the unscheduled part of the NUT, each node from 0 to UMAX shares the opportunity to transmit non-time critical data in a round robin fashion, until the allocated NUT duration is exhausted. UMAX is the MAC ID of the highest numbered node that has access to the medium during the unscheduled part of the NUT. Every node between 0 and UMAX may have zero, one or many opportunities to send unscheduled data depending on how much of the NUT remains after the completion of the scheduled time. The opportunity to access the medium during the unscheduled time may be different for each node in every NUT. Figure 3-3 shows how the permission to transmit is granted during the unscheduled time. The MAC ID of the node that goes first in the unscheduled part of the NUT is incremented by 1 for each NUT. The unscheduled token begins at the MAC ID specified in the unscheduled start register (USR) of the previous moderator frame. The USR increments by one modulo UMAX+1 each NUT. If the USR reaches UMAX before the guardband, it returns to zero and the token pass continues.



When the guardband time is reached, all nodes stop transmitting. A node is not allowed to start a transmission unless it can be completed before the beginning of the guardband. During the guardband time the node with the lowest MAC ID (called the “moderator”) transmits a maintenance message (called the “moderator frame”) that accomplishes two things:

- it keeps the NUT timers of all nodes synchronised;
- it publishes critical link parameters enabling all members of the local DLL group to share a common version of important DLL values such as NUT, slot time, SMAX, UMAX, etc.

The moderator transmits the moderator frame, which re-synchronises all nodes and restarts the NUT. Following the receipt of a valid moderator frame, each node compares its internal values with those transmitted in the moderator frame. A node using link parameters that disagree with the moderator disables itself. If the moderator frame is not heard for 2 consecutive NUTs, the node with the lowest MAC ID assumes the moderator role, and begins transmitting the moderator frame in the guardband of the third NUT. A moderator node that notices another node on-line and transmitting with a MAC ID lower than its own immediately cancels its moderator role.

Typical situations that may cause disruption of the DLL access protocol include

- induced noise on the link;
- poor cable or termination quality;
- physically connecting two links together while the network is operating.

One common consequence of such disruption is that nodes may be caused to disagree as to which node should be transmitting; this is called a network “non-concurrence”. Another potential problem occurs when the nodes do not agree to the same link configuration parameters. A node that disagrees with the link parameters as transmitted by the moderator is called a “rogue” and immediately stops transmitting. The DLL access protocol is designed to recover a rogue node and bring it back on-line.

Within the Data Link Layer, the Access Control Machine has the primary responsibility for detecting and recovering from network disruptions. The major objectives of the ACM are

- to make sure that the local node detects and fully utilises its assigned slots in the protocol;

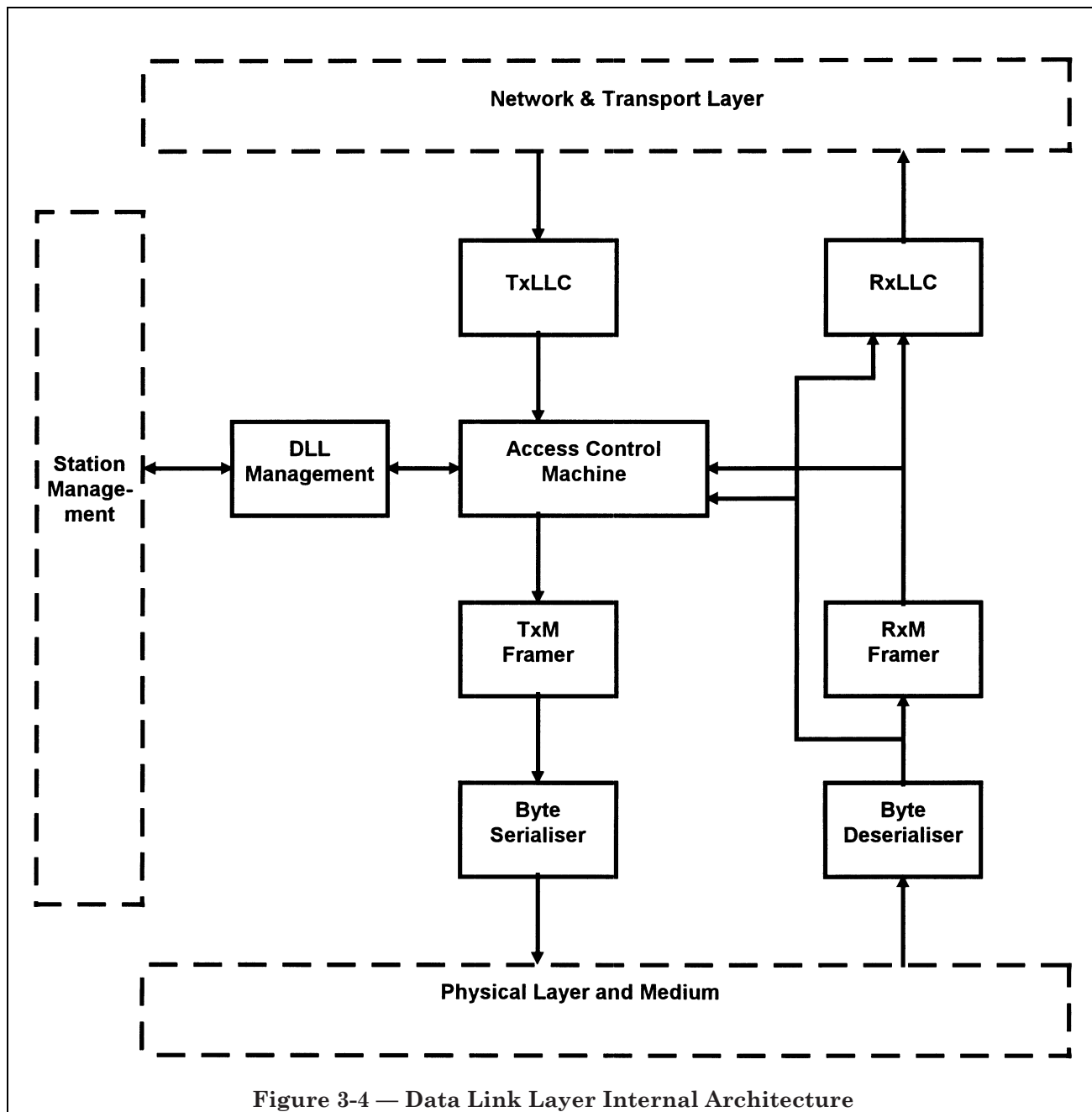
- to make sure that the local node does not interfere with the transmissions of other nodes, especially the moderator node;
- to start the next NUT on schedule, whether the moderator frame: is heard or not;
- if the local node is the moderator, to transmit each moderator frame strictly on schedule.

The Data Link Layer is comprised of the following components:

Table 3-1 — Data Link Layer components

Components	Description
Access Control Machine (ACM)	receives and transmits control frames and header information, and determines the timing and duration of transmissions.
Transmit LLC (TxLLC)	buffers SDUs received from the Network and Transport Layer, and determines which should be transmitted next.
Receive LLC (RxLLC)	performs the task of quarantining received link packets until they are validated by a good CRC.
Transmit Machine (TxM)	receives requests to send MAC frame headers and trailers, and Lpackets from the ACM, and breaks them down into byte symbol requests which are sent to the Serialiser.
Receive Machine (RxM)	assembles received Lpackets from byte symbols received from the Deserialiser, and submits them to the RxLLC.
Serialiser	receives byte symbols, encodes and serialises them, and sends them as MAC symbols to the Physical Layer. It is also responsible for generating the CRC.
Deserialiser	receives MAC symbols from the Physical Layer, converts MAC symbols into bytes and sends them to the receive machine. It is also responsible for checking the CRC.
DLL Management Interface	holds the Station Management variables that belong to the DLL, and helps manage synchronised change.

The internal arrangement of these components, and their interfaces, are shown in Figure 3-4. The arrow heads illustrate the primary direction of flow of data and control.

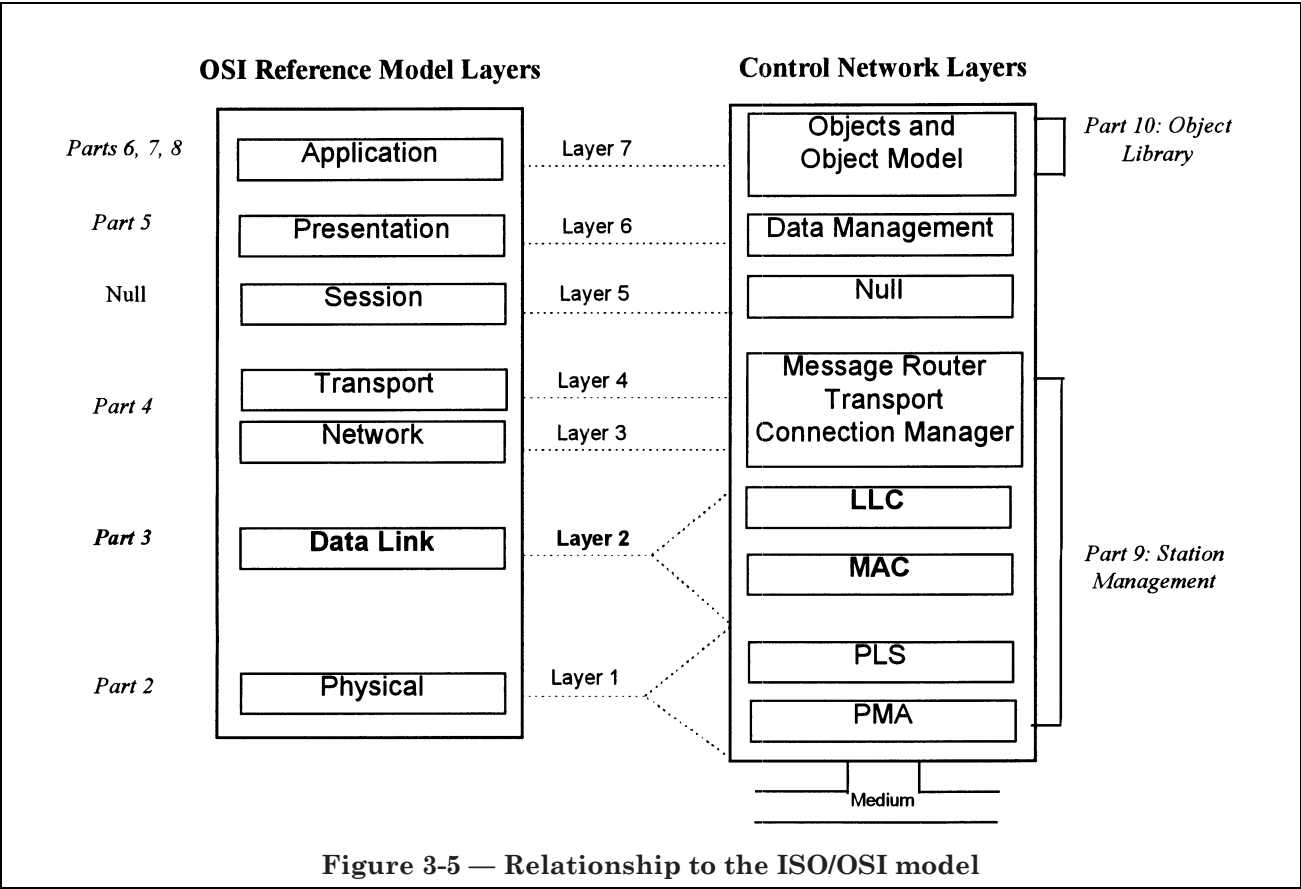


1 Scope

This part of the standard defines the requirements for the Data Link Layer of a node on the deterministic control network.

The Data Link Layer of the deterministic control network corresponds to the definition of Layer 2 of the seven layer OSI model in accordance with ISO 7498. Figure 3-5 shows the location of the Data Link Layer within the OSI model.

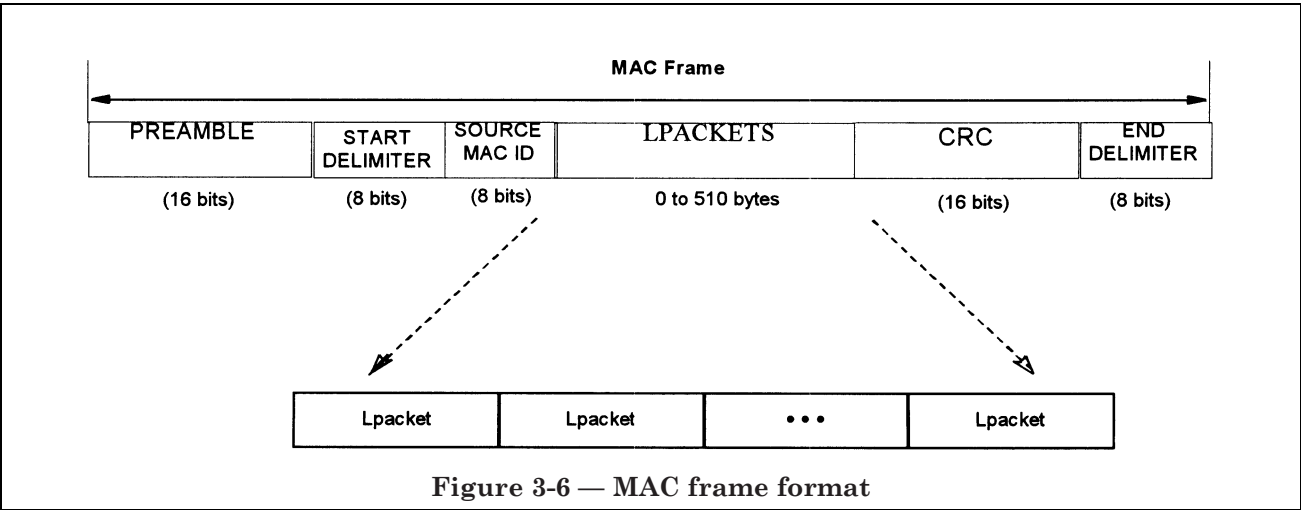
NOTE Most of the terms and models for the Data Link Layer were adopted from those in ISO/IEC 8802-4 or ISO 8802-3. The data types used are described in part 5, Data Management.



2 MAC frame

2.1 Format

The MAC frame, as shown in Figure 3-6, is the protocol data unit (PDU) with which peer Data Link Layers shall co-ordinate the permission to transmit. The components of the MAC frame shall be transmitted in the following order: preamble, start delimiter, source MAC ID, zero or more Lpackets, CRC and end delimiter.



2.2 Preamble

The preamble shall be composed of 16 consecutive {1} MAC_Symbols.

NOTE Part 2 (Physical Layer) defines MAC_Symbols.

2.3 Delimiters

The start delimiter shall consist of these MAC_Symbols, {+,0,-,+,-,1,0,1}, transmitted from left to right. The end delimiter shall consist of these MAC_Symbols, {1,0,0,1,+,-,+,-}, transmitted from left to right. The use of non-data MAC_Symbols shall be reserved for the start and end delimiters.

2.4 Bytes and ordering

The source MAC ID, Lpackets, and CRC shall be composed of bytes. A byte shall be composed of exactly eight MAC_Symbols each of which shall be either {0} or {1}. A byte shall be transferred to the Physical Layer low bit first. Multiple byte fields, such as the CRC, shall be transmitted low order byte first (little endian format).

The MAC ID shall be in the range 1 to 99; however, a node may temporarily assume a MAC ID of zero to perform link maintenance.

NOTE Lpackets are defined in clause 3.

2.5 Total size of all Lpackets

Zero or more Lpackets shall be concatenated into a single MAC frame subject to the constraint that the maximum number of bytes which compose all Lpackets in a MAC frame shall be 510. A MAC frame with an empty Lpacket field, called a null frame, shall be sent by a node to indicate that it has no data to send. A node shall always send a MAC frame at every opportunity it has to transmit.

2.6 Cyclic redundancy check

The CRC shall be a modified CCITT using the 16 bit polynomial: $X^{16}+X^{12}+X^5+1$. Two CRC bytes shall be included in each frame.

NOTE The generation of the CRC is detailed in clause 12.

2.7 Abort MAC frame

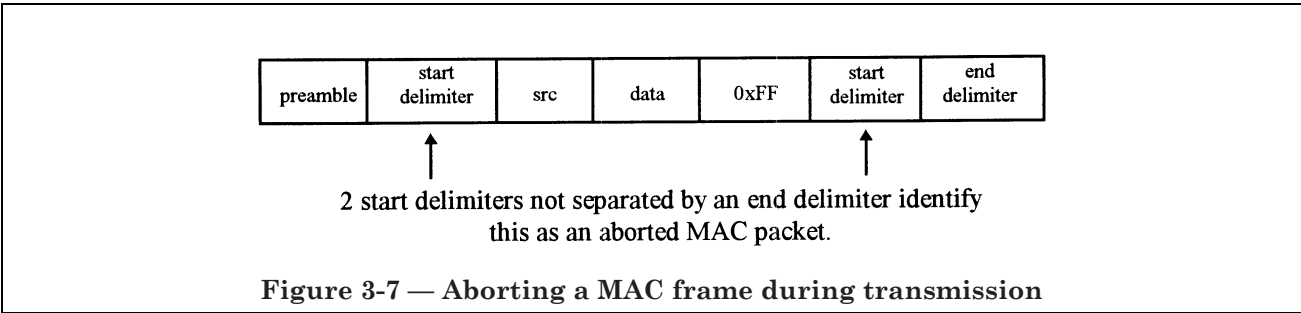
NOTE Some implementations of this standard may build the MAC frame as it is being transmitted. If a higher layer has provided only a partial frame to the Data Link Layer, and the DLL has begun transmission of the frame assuming that the rest will be provided shortly, a transmit underflow occurs if the rest of the frame is not provided to the DLL soon enough to complete the transmission.

To abort a partially transmitted MAC frame, the Data Link Layer shall transmit the following sequence:

- one 0xFF byte;
- start delimiter;
- end delimiter.

NOTE 1 Although the DLL assembles the entire MAC frame including delimiters, the Physical Layer performs start and end delimiter detection. The Physical Layer identifies a transmit abort as a MAC frame which includes two start delimiters which are not separated by an end delimiter, as shown in Figure 3-7.

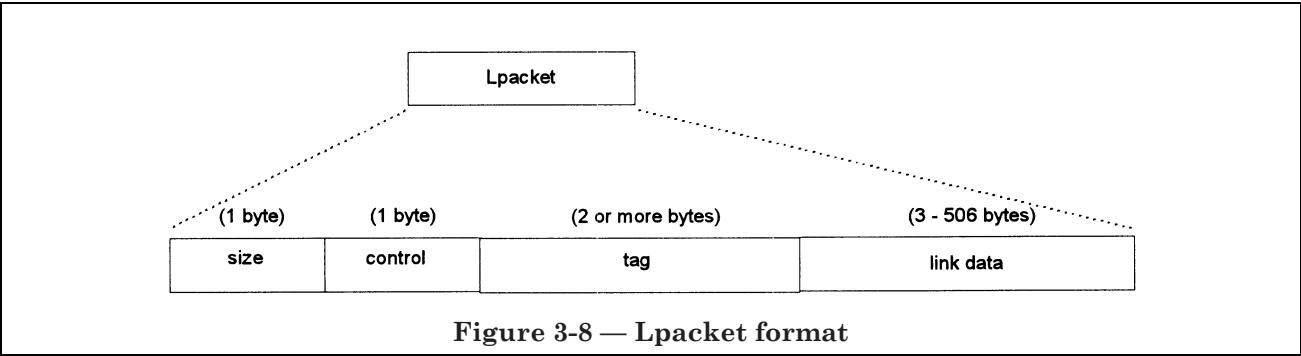
NOTE 2 The 0xFF byte is inserted before the start delimiter/end delimiter pair to avoid violating Physical Layer constraints governing consecutive physical symbol relationships on the media.



3 Lpacket

3.1 Format

The Lpacket (or link packet) is the PDU with which peer Data Link Layers shall pack and unpack transport SDUs from higher layers in the communication model. The components of the Lpacket shall be transmitted in the following order: size, control, tag, and link data as shown in Figure 3-8.



3.2 Size

The size field shall specify the number of byte pairs (from 4 to 255) contained in an individual Lpacket. The number of byte pairs shall include the size, control, tag, and link data fields counted in accordance with the following rules:

- Combined, the size and control shall count as a single byte pair;
- The number of byte pairs from the tag and link data fields shall be counted separately and odd byte counts shall be rounded up;
- Although the counting of byte pairs is done independently, the throat of an Lpacket as placed into a MAC frame shall not include the extra bytes caused by the round up.

NOTE An Lpacket with a 3 byte tag (2 byte pairs) and a 9 byte link data (5 byte pairs) has a size field of 1+2+5 = 8 byte pairs; however the Lpacket occupies 2+3+9 = 14 bytes in the MAC frame.

3.3 Control

3.3.1 Bit 0 and Bit 4 (type of Lpacket)

The bits of the control field shall be numbered 0 to 7, where bit 0 shall be the least significant bit of the control field. Bit 0 shall indicate the type of Lpacket. When set (bit 0 = 1), the Lpacket shall be a fixed tag Lpacket. Conversely, when clear (bit 0 = 0), the Lpacket shall be a generic tag Lpacket.

NOTE The terms fixed tag Lpacket and genetic tag Lpacket are defined in clauses 3.4.1 and 3.4.2.

Bit 4 of the control field shall be the inverse of bit 0. If bit 0 is clear, then bit 4 shall be set. If bit 0 is set, then bit 4 shall be clear.

3.3.2 Bit 1 (odd tag size)

Bit 1 of the control field shall indicate whether the tag field contains an even or odd number of bytes. When clear (bit 1 = 0), this bit shall indicate that the tag contains an even number of bytes. When set (bit 1 = 1), it shall indicate that the tag contains an odd number of bytes.

3.3.3 Bit 2 (odd link data size)

Bit 2 of the control field shall indicate whether the link data contains an even or odd number of bytes. When clear (bit 2 = 0), this bit shall indicate that the link data contains an even number of bytes. When set (bit 2 = 1), it shall indicate that the link data contains an odd number of bytes.

3.3.4 Bit 5 and Bit 6 (link pad size)

For generic tag Lpackets, bits 5 and 6 shall indicate the number of pad bytes contained within the link data as shown in Table 3-2.

Table 3-2 — Link pad size

Bit 6	Bit 5	Pad bytes
0	0	0
0	1	2
1	0	4
1	1	6

For fixed tag Lpackets, bits 5 and 6 shall both be zero.

3.3.5 Bit 3 and Bit 7 (reserved bits)

Bits 3 and bit 7 of the control field are reserved and shall be zero.

3.4 Tag

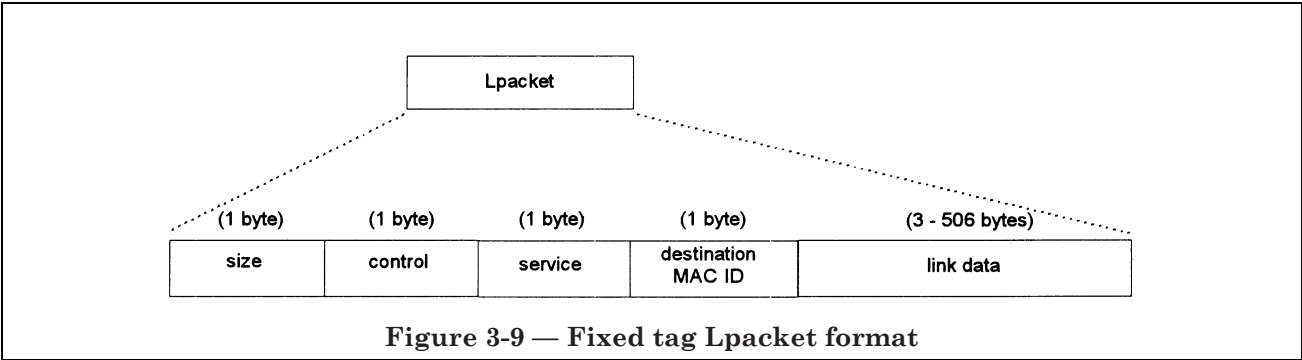
3.4.1 Fixed tag

Each fixed tag shall be at least two bytes long. The first byte of the fixed tag shall indicate the service as specified in Table 3-3.

Table 3-3 — Fixed tag definitions

Fixed tag service	Meaning	Location of definition
0x00	moderator	Data Link Layer (part 2)
0x01 – 0x14	vendor specific	
0x15	Tminus	Station Management (part 9)
0x16 – 0x3F	vendor specific	
0x40 – 0x7F	reserved	
0x80	I'm alive	Station Management (part 9)
0x81	link parameters	Station Management (part 9)
0x82	reserved	
0x83	UCMM	Network and Transport Layers (part 4)
0x84	TUI	Keeper Object (part 10, clause 241)
0x85	IP	
0x86	WAMI	Station Management (part 9)
0x87	reserved	
0x88	Keeper UCMM	Network and Transport Layers (part 4)
0x89 – 0x8B	reserved	
0x8C – 0x8F	Time distribution	Station Management (part 9)
0x8D – 0xEF	reserved	
0xF0 – 0xFF	vendor specific	

The second byte of the fixed tag shall be the destination address as shown in Figure 3-9. The destination address shall either be a MAC ID or 0xFF which indicates the broadcast address. If the fixed tag contains more than two bytes, the meaning of any other bytes shall be determined by the first byte.

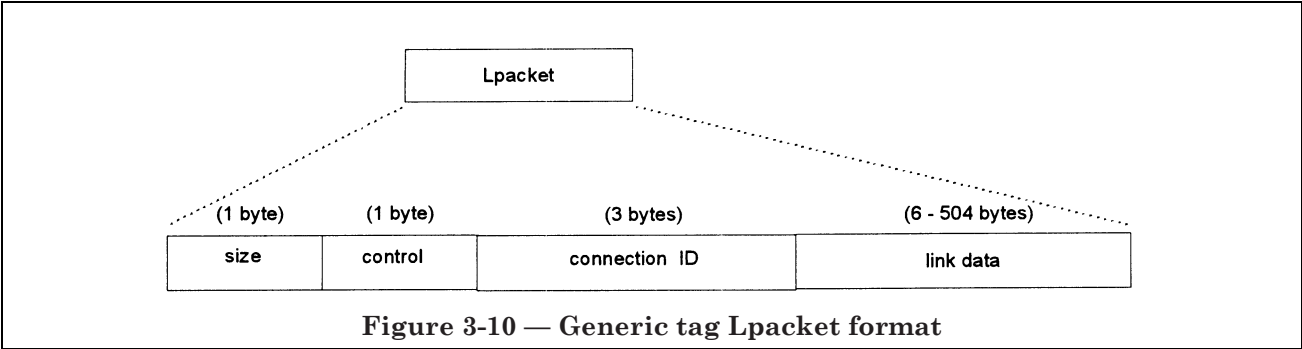


NOTE Fixed tag Lpackets are typically used for station management (part 9) and to establish connections (part 4).

3.4.2 Generic tag

The tag field shall identify the link data contained in the Lpacket. Each genetic tag shall contain the 3 byte connection identifier (CID) as shown as shown in Figure 3-10.

NOTE Genetic tags are used to send data on connections. These tags are managed by the Network and Transport Layer (part 4).



3.5 Link data

The link data field shall contain one transport SDU.
The minimum length fixed tag Lpacket shall be 7 bytes (minimum link data size of 3 bytes);
The minimum length generic tag Lpacket shall be 11 bytes (minimum link data size of 6 bytes).
If necessary to ensure a minimum number of bytes in a generic tag Lpacket, the link data shall be padded with up to three byte pairs. Fixed tag Lpackets shall not be padded; the definition of each Lpacket format shall comply with the minimum 7 byte rule.

4 Modelling language (informative)

Some of the components of the Data Link Layer are specified using a state machines description language. The action statement are expressed in C++. Other components do not require a state machine description and are specified using only C++.
A state machine is described using the following syntax, where * indicates repetition of one or more of the preceding component, and { } indicates an optional component:

```
State-machine :=  
    <header-statements>  
    state*  
  
state :=  
    "state:" <state-name>  
    transition*
```

```

transition :=
    "event:" <event-expression>
    {"condition:" <condition-expression>}
    "destination:" <state-name>
    {"action:" <action-statements>}

```

State-names are legal C++ identifiers, event-expressions and condition-expressions are C++ expressions, and header-statements and action-statements are lists of C++ statements.

Header-statements include

#include and #define;
variable and interface declarations;
subroutine definitions.

An event-expression indicates the specified event when the value of the expression is TRUE. An event is required to trigger a transition. A condition-expression qualifies an transition. A qualified transition occurs when its event occurs if the condition is true at the time of the event. In general, event-expressions are FALSE upon entry into a state. There are a few cases where an event-expression may be true upon entry to a state. In these cases, the transition takes place immediately.

Actions statements are executed when the transition is triggered.

All the event triggers and conditions of the current state are evaluated continuously and concurrently until a given transition is enabled. All expressions and statements execute instantaneously. The precedence of the various parts of a transition is illustrated by the following model:

```

if(the event has occurred && the condition is true)
{
    do the actions;
    go to the destination;
}

```

Within the state machine descriptions, BYTE is equivalent to USINT, and WORD is equivalent to UINT.

5 External interfaces

5.1 Transmit services

The request and confirmation services used to queue Lpackets with the Data Link Layer shall be of the form:

DLL_xmit_fixed_request (		DLL_xmit_fixed_confirm (
IDENTIFIER	id,	IDENTIFIER id,
USINT	packet [],	TXSTATUS status);
UINT	packet_size,	
PRIORITY	priority,	
USINT	service,	
USINT	dest ID);	
DLL_xmit_generic_request (		DLL_xmit_generic_confirm (
IDENTIFIER	id,	IDENTIFIER id,
USINT	packet [],	TXSTATUS status);
UINT	packet_size,	
PRIORITY	priority,	
USINT	gentag [3]);	

The packet parameter shall contain the transport SDU as an ordered sequence of bytes. The packet_size parameter shall specify the number of bytes contained in the packet array. The priority shall specify onto which queue the request shall be placed and shall be one of LOW, HIGH or SCHEDULED.

The IDENTIFIER, id, shall correlate a confirmation with a corresponding request. The status parameter shall return the status of the attempted transmit and shall be one of OK, TXABORT or FLUSHED.

The DLL_xmit_fixed request service queues an Lpacket with a two byte fixed tag for transmit. The service parameter shall specify the first byte of the fixed tag Lpacket. The destID parameter shall specify the second byte which shall determine the MAC ID of the destination node.

The remaining parameter (gentag) of the DLL_xmit_generic_request service shall specify which genetic tag shall be used to transmit the Lpacket.

5.2 Queue maintenance services

Once queued, an Lpacket shall remain on one of the three transmit queues until it is transmitted or de-queued. The request and confirmation services used to de-queue an Lpacket, before it is transmitted, shall be of the form:

DLL_flush_request(PRIORITY priority);

DLL_flush_confirm(PRIORITY priority);

DLL_flush_single_request(PRIORITY priority, IDENTIFIER id);

DLL_flush_request shall cancel all pending transmits at the specified priority. Each pending transmit shall immediately terminate with a confirmation containing the FLUSHED status.

DLL_flush_single_request shall cancel a specific Lpacket identified by the id used to transmit the packet. This service need not have a corresponding confirmation since the confirmation of the queued Lpacket shall serve this purpose.

5.3 Tag filter management

Only the moderator Lpacket (fixed tag 0x00) shall be processed by the Data Link Layer. By default, all other Lpackets shall be discarded. Other protocol layers may enable or disable reception of specific Lpackets using services of the form:

DLL_enable_generic_request (	DLL_enable_generic_confirm (
IDENTIFIER id,	IDENTIFIER id,
USINT gentag [3]);	BOOL result);
DLL_disable_generic_request (	DLL_disable_generic_confirm (
IDENTIFIER id,	IDENTIFIER id,
USINT gentag [3]);	BOOL result);
DLL_enable_fixed_request (	DLL_enable_fixed_confirm (
IDENTIFIER id,	IDENTIFIER id,
USINT service);	BOOL result);
DLL_disable_fixed_request (	DLL_disable_fixed_confirm (
IDENTIFIER id,	IDENTIFIER id,
USINT service);	BOOL result);

The IDENTIFIER, id, shall correlate a confirmation with a corresponding request. The parameter, result, shall return TRUE if the request was successful and FALSE if unsuccessful. The gentag parameter shall specify which generic tag to route to the higher layer that enabled the tag; likewise, the service parameter shall specify which fixed tag to route to the higher layer that enabled the tag.

NOTE Invoking the DLL_enable_generic_request service can return an error if the implementation of the Data Link Layer is unable to filter any more generic tags.

5.4 Receive services

The indications to higher protocol layers that an Lpacket has been received by the Data Link Layer shall be of the form:

DLL_rcv_fixed_indication (	DLL_rcv_generic_indication (
USINT packet [],	USINT packet []
UINT packet_size,	UINT packet_size,
USINT service,	USINT gentag [3]);
USINT sourceID);	

These indications shall signal the delivery of an Lpacket with a good CRC. Only fixed tag Lpackets which have had their service enabled by a successful call to DLL_enable_fixed_request shall cause a DLL_rcv_fixed_indication. Only generic tag Lpackets which have had their generic tag enabled by a successful call to DLL_enable_generic_request shall cause a DLL_rcv_generic_indication.

The array of bytes, packet, shall contain the transport SDU as an ordered sequence of bytes. The integer, packet_size, shall specify the number of bytes contained in the packet array.

5.5 Network synchronisation service

The indication that a new network update time (NU-T) has begun shall be of the form:

DLL_tone_indication(USINT cycle);

The cycle parameter shall return the interval counter from the moderator frame just received.

NOTE If a moderator frame is expected but does not arrive, the local node simulates the reception of the moderator frame based on an internal timer of the ACM as described in clause 6,

5.6 Synchronised parameter change

All nodes on a link shall maintain two copies of link parameters: current and pending. The current copy of link parameters shall be used for the on going operation of the Data Link Layer. The pending copy shall be maintained to allow a synchronised change of link parameters. The services that read and write these link parameters shall be of the form:

DLL_set_pending_request (	DLL_set_pending_confirm (
IDENTIFIER id,	IDENTIFIER id,
DLL_config_data params);	BOOL result);
DLL_set_current_request (	DLL_set_current_confirm (
IDENTIFIER id,	IDENTIFIER id,
DLL_config_data params);	BOOL result);
DLL_get_pending_request (	DLL_get_pending_confirm (
IDENTIFIER id);	IDENTIFIER id,
	DLL_config_data params);
DLL_get_current_request (	DLL_get_current_confirm (
IDENTIFIER id);	IDENTIFIER id,
	DLL_config_data params);

The IDENTIFIER, id, shall correlate a confirmation with a corresponding request. The result shall return TRUE if the request was successful and FALSE if unsuccessful.

The DLL_config_data shall contain the link parameters and be of the form:

```
class DLL_config_data
{
public:
    BYTE myaddr;          // the MAC ID of this node
    WORD NUT_length;      // the length of the NUT in 10 µs increments
    BYTE smax;            // highest MAC ID allowed to transmit scheduled
    BYTE umax;            // highest MAC ID allowed to transmit unscheduled
    BYTE slotTime;        // time allowed for line turnaround in 1 µs increments
    BYTE blanking;        // time to disable RX after MAC frame in 1.6 µs increments
    BYTE gb_start;        // 10 µs intervals from start of guardband to tone
    BYTE gb_center;       // 10 µs intervals from start of moderator to tone
    BYTE modulus;         // modulus of the interval, counter
    BYTE gb_prestart;     // transmit cut-off, 10 µs intervals before tone
};                        // may not transmit passed this limit
```

The moderator Lpacket contains a field, called Tminus, that shall be used to synchronise the update of the current link parameters. The DLL_tminus_request shall initialise the Tminus field of the moderator packet. This request shall only have an effect if this node is the moderator. The moderator shall decrement this field before transmitting each moderator until the field equals zero. When the Tminus field transitions from 1 to 0, the DLL in each node shall locally indicate the completion or the Tminus countdown and copy its pending link parameters into its current copy. The services which request, confirm, and indicate the completion of a Tminus countdown shall be of the form:

```
DLL_tminus_request (          DLL_tminus_confirm (
    IDENTIFIER id,            IDENTIFIER id,
    USINT      start_count. );    BOOL      result );
DLL_tminus_zero_indication ( );
```

The IDENTIFIER, id, shall correlate a confirmation with a corresponding request. The parameter, result, shall return TRUE if the request was successful and FALSE if unsuccessful.

5.7 Event reports

The indication used to alert the Station Management entity of various events internal to the Data Link Layer shall be of the form:

```
DLL_event_indication (
    DLL_event  event,
    USINT      mac_id [ optional ] );
```

The DLL_event, event, shall be the name of the event that occurred as shown in Table 3-4. The mac_id parameter shall be used in the case of (event = DLL_EV_badFrame); it shall indicate the source MAC ID of the node that transmitted the bad frame.

NOTE Since the frame was damaged, the source MAC ID may not be correct.

Table 3-4 — DLL events

DLL events	Description
DLL_EV_rxGoodFrame	A good MAC frame was received. This shall include frames that contain no data (null frames), but shall exclude moderators.
DLL_EV_txGoodFrame	A good MAC frame was transmitted. This shall include frames that contain no data (null frames), but shall exclude moderators.
DLL_EV_badFrame	A damaged MAC frame was received. The optional parameter, which is the apparent source MAC ID of the offending node, shall also be reported.
DLL_EV_errA	A bad frame was received on channel A of the physical medium, or a good frame was received on channel B and pls_frame_indication from channel A stayed false.
DLL_EV_errB	A bad frame was received on channel B of the physical medium, or a good frame was received on channel A and pls_frame_indication from channel B stayed false.
DLL_EV_txAbort	A transmit frame was terminated with an abort sequence
DLL_EV_NUT_overflow	NUT is not large enough to accommodate all the scheduled traffic.
DLL_EV_dribble	Scheduled Lpackets could not be sent during scheduled time.
DLL_EV_nonconcurrency	An event was detected that indicates that this node is out of step with the access control protocol.
DLL_EV_rxAbort	A MAC frame was received that was terminated with an abort sequence.
DLL_EV_lonely	Have not heard a MAC frame from another node on the link for 8 NUTs
DLL_EV_dupNode	Another node on the link is using this nodes MAC ID.
DLL_EV_noisePulse	pls_lock_indication went true then false before pls_frame_indication went true, nor was pls_lock_indication true long enough to indicate a possibly damaged MAC frame.
DLL_EV_collision	pls_frame_indication was true when this node was about to transmit.
DLL_EV_invalidModAddress	A moderator was received from a node that does not have the lowest MAC ID on the link.
DLL_EV_rogue	A moderator frame was received that does not match the link configuration information at this node.
DLL_EV_deafness	Cannot hear the moderator frame even though other link traffic is present.
DLL_EV_supernode	A moderator was received from MAC ID 0.

5.8 Bad CRC indication

The indication used to alert the Station Management entity that a received MAC frame had an invalid CRC shall be of the form:

DLL_badCRC_indication(CHANNEL channel);

The channel parameter shall indicate which Physical Layer entity provided the frame which was in error. This parameter shall be one of CHA or CHB, corresponding to Physical Layer channel A and Physical Layer channel B, respectively. This indication shall be provided, at most, once per error frame per channel.

5.9 Current moderator indication

The indication used to inform the Station Management entity which node is currently the moderator shall be of the form:

DLL_currentMod_indication (USINT macID);

The macID parameter shall indicate the MAC ID of the node that last transmitted a valid moderator frame.