INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 2017-2020

SG15-TD156R1/PLEN

STUDY GROUP 15

Original: English

Question(s): 10/15 Geneva, 29 Jan – 9 Feb 2018

TD

Source: Editor G.8021/Y.1341

Title: Draft revised recommendation ITU-T G.8021/Y.1341 (for consent)

Purpose: Discussion

Contact: Akira SAKURAI
NEC Corporation
Japan
Tel: +81-4-7185-7652
Fax: +81-4-7185-6856
E-mail: a-sakurai@da.jp.nec.com

Keywords: G.8021; Atomic functions; equipment functional blocks; Ethernet transport network.

Abstract: This document is the latest draft of revised Recommendation G.8021/Y.134 “Characteristics of Ethernet transport network equipment functional blocks”. It is based on the published version of G.8021/Y.1341 (11/2016) and it covers some changes agreed in the June 2017 plenary meeting.

Introduction

This document is the latest draft of G.8021/Y.1341 and based on the 2016 edition of G.8021/Y.1341. All editable figures are available at:

http://ifa-int.itu.int/t/2017/sg15/exchange/wp3/q10/G.8021/figures/G.8021_figures_2018-02.pptx

Revision history

Rev	Document number	Date	Description
0.0	-	April 2017	• Initial text – 2016 edition of G.8021/Y.1341
0.1	Wd10-05	June 2017	• Updates Tabale 9.1 and clause 9.1.2 per C142 (added three ETH_C_MI_PS signals) • Removed the desctiption for MI_Active signals in the NOTE of clause 9.3 • Removed MI_Active signals in interface signals for all adaptation functions (20x2=40 signals) • Remove the expranation for MI_Active signals in clauses 10.7.1 and 10.7.2
0.2	Wd10-05r1	June 2017	• Corrected the result description for C142

Rev	**Document number**	**Date**	**Description**
0.3	Wd10-05r2 TD77r1/3	June 2017	• Removed clause 10
0.4	TDxxx/P	December 2017	• Highlighted the ETYn related descriptions as EDITOR'S NOTE (Fig 1-1, Clause 8, Clause 9.7, and Fig I-1)
0.5	Wd10-12	February 2018	• Incorporated the result of drafting on 2/1, 2/2 and 2/5 (wd10-11r4) ➢ Clause 10 and Appendix X (moved to G.8023) ➢ Clause 8.5 to 8.8.4 (Srv/ETH internal process) ➢ Clause 9.7 (LAG) ➢ Clause 3 and 4, Figure 1-1 and Figure I.1 ➢ Clause 11.5 (ODUkP/ETH; moved to G.798)
0.6	Wd10-12r1	February 2018	• Updated Figure 1-1 • Added the terminology "MAC frame" in clause 3.1 • Updated clause 8.2 to 8.8.4 for the proper description of the ETH_CI and MAC frame • Added in the description for <Srv>/ETH-m_A in clause 9.5 • Updated Table 9-27 and 9-28 for editorial correction
0.7	Wd10-12r2	February 2018	• Updated the text for MAC length check in caluse 8.6 • Added a note for ETC3 in clause 11.2 • Updated the text for ETC3 in clause 11.5.3 • Updated the note for ETH-LAG in clause 9.7 • Replaced the Figure 9-69 • Corrected the MI signals in Table 9-30 • Updated the Figures 11-2,4,6,9,11,13,20,22,24,26 for the terminologies ETH_CI and MAC Frames • Added the title of Figures in AppendixX. • Renumbered the Figures 8-119 to 8-129
0.8	Wd10-12r3	February 2018	• Corrected a typo in the title of Fig X.2.

International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

G.8021/Y.1341

(11/2016)

SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

Packet over Transport aspects – Ethernet over Transport aspects

SERIES Y: GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES

Internet protocol aspects – Transport

Characteristics of Ethernet transport network equipment functional blocks

Recommendation ITU-T G.8021/Y.1341

ITU-T G-SERIES RECOMMENDATIONS

TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

INTERNATIONAL TELEPHONE CONNECTIONS AND CIRCUITS	G.100–G.199
GENERAL CHARACTERISTICS COMMON TO ALL ANALOGUE CARRIER-TRANSMISSION SYSTEMS	G.200–G.299
INDIVIDUAL CHARACTERISTICS OF INTERNATIONAL CARRIER TELEPHONE SYSTEMS ON METALLIC LINES	G.300–G.399
GENERAL CHARACTERISTICS OF INTERNATIONAL CARRIER TELEPHONE SYSTEMS ON RADIO-RELAY OR SATELLITE LINKS AND INTERCONNECTION WITH METALLIC LINES	G.400–G.449
COORDINATION OF RADIOTELEPHONY AND LINE TELEPHONY	G.450–G.499
TRANSMISSION MEDIA AND OPTICAL SYSTEMS CHARACTERISTICS	G.600–G.699
DIGITAL TERMINAL EQUIPMENTS	G.700–G.799
DIGITAL NETWORKS	G.800–G.899
DIGITAL SECTIONS AND DIGITAL LINE SYSTEM	G.900–G.999
MULTIMEDIA QUALITY OF SERVICE AND PERFORMANCE – GENERIC AND USER-RELATED ASPECTS	G.1000–G.1999
TRANSMISSION MEDIA CHARACTERISTICS	G.6000–G.6999
DATA OVER TRANSPORT – GENERIC ASPECTS	G.7000–G.7999
PACKET OVER TRANSPORT ASPECTS	G.8000–G.8999
Ethernet over Transport aspects	**G.8000–G.8099**
MPLS over Transport aspects	G.8100–G.8199
Synchronization, quality and availability targets	G.8200–G.8299
Service Management	G.8600–G.8699
ACCESS NETWORKS	G.9000–G.9999

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T G.8021/Y.1341

Characteristics of Ethernet transport network equipment functional blocks

Summary

Recommendation ITU-T G.8021/Y.1341 specifies both the functional components and the methodology that should be used in order to specify the Ethernet transport network functionality of network elements; it does not specify individual Ethernet transport network equipment.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T G.8021/Y.1341	2004-08-22	15	11.1002/1000/7364
1.1	ITU-T G.8021/Y.1341 (2004) Amd. 1	2006-06-06	15	11.1002/1000/8776
2.0	ITU-T G.8021/Y.1341	2007-12-22	15	11.1002/1000/9173
2.1	ITU-T G.8021/Y.1341 (2007) Amd. 1	2009-01-13	15	11.1002/1000/9661
2.2	ITU-T G.8021/Y.1341 (2007) Amd. 2	2010-02-22	15	11.1002/1000/10428
3.0	ITU-T G.8021/Y.1341	2010-10-22	15	11.1002/1000/10900
3.1	ITU-T G.8021/Y.1341 (2010) Amd. 1	2011-07-22	15	11.1002/1000/11137
4.0	ITU-T G.8021/Y.1341	2012-05-07	15	11.1002/1000/11512
4.1	ITU-T G.8021/Y.1341 (2012) Amd. 1	2012-10-29	15	11.1002/1000/11777
4.2	ITU-T G.8021/Y.1341 (2012) Amd. 2	2013-08-29	15	11.1002/1000/12030
5.0	ITU-T G.8021/Y.1341	2015-04-06	15	11.1002/1000/12382
5.1	ITU-T G.8021/Y.1341 (2015) Cor. 1	2015-08-13	15	11.1002/1000/12551
6.0	ITU-T G.8021/Y.1341	2016-11-13	15	11.1002/1000/13095
7.0	ITU-T G.8021/Y.1341	2018-**-**	15	To be allocated

Keywords

Atomic functions, equipment functional blocks, Ethernet transport network.

---

* To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Introduction

This Recommendation forms part of a suite of ITU-T Recommendations covering the full functionality of Ethernet transport network architecture and equipment (e.g., Recommendations ITU-T G.8010/Y.1306 and ITU-T G.8012/Y.1308) and follows the principles defined in Recommendation ITU-T G.805.

This Recommendation specifies a library of basic building blocks and a set of rules by which they may be combined in order to describe equipment used in an Ethernet transport network. The building blocks are based on atomic modelling functions defined in Recommendations ITU-T G.806 and ITU-T G.809. The library comprises the functional building blocks needed to wholly specify the generic functional structure of the Ethernet transport network. In order to be compliant with this Recommendation, the Ethernet functionality of any equipment which processes at least one of the Ethernet transport layers needs to be describable as an interconnection of a subset of these functional blocks contained within this Recommendation. The interconnections of these blocks should obey the combination rules given.

The specification method is based on functional decomposition of the equipment into atomic and compound functions. The equipment is then described by its equipment functional specification (EFS) which lists the constituent atomic and compound functions, their interconnection and any overall performance objectives (e.g., transfer delay, availability, etc.).

Recommendation ITU-T G.8021/Y.1341

Characteristics of Ethernet transport network equipment functional blocks

1 Scope

This Recommendation covers the functional requirements of Ethernet functionality within Ethernet transport equipment.

This Recommendation uses the specification methodology defined in [ITU-T G.806] in general for transport network equipment and is based on the architecture of Ethernet layer networks defined in [ITU-T G.8010], the interfaces for Ethernet transport networks defined in [ITU-T G.8012], and in support of services defined in [ITU-T G.8011]. It also provides processes for Ethernet OAM based on [ITU-T G.8013]. The description is generic and no particular physical partitioning of functions is implied. The input/output information flows associated with the functional blocks serve for defining the functions of the blocks and are considered to be conceptual, not physical.

The functionality defined in this Recommendation can be applied at user-to-network interfaces (UNIs) and network-to-network interfaces (NNIs) of the Ethernet transport network.

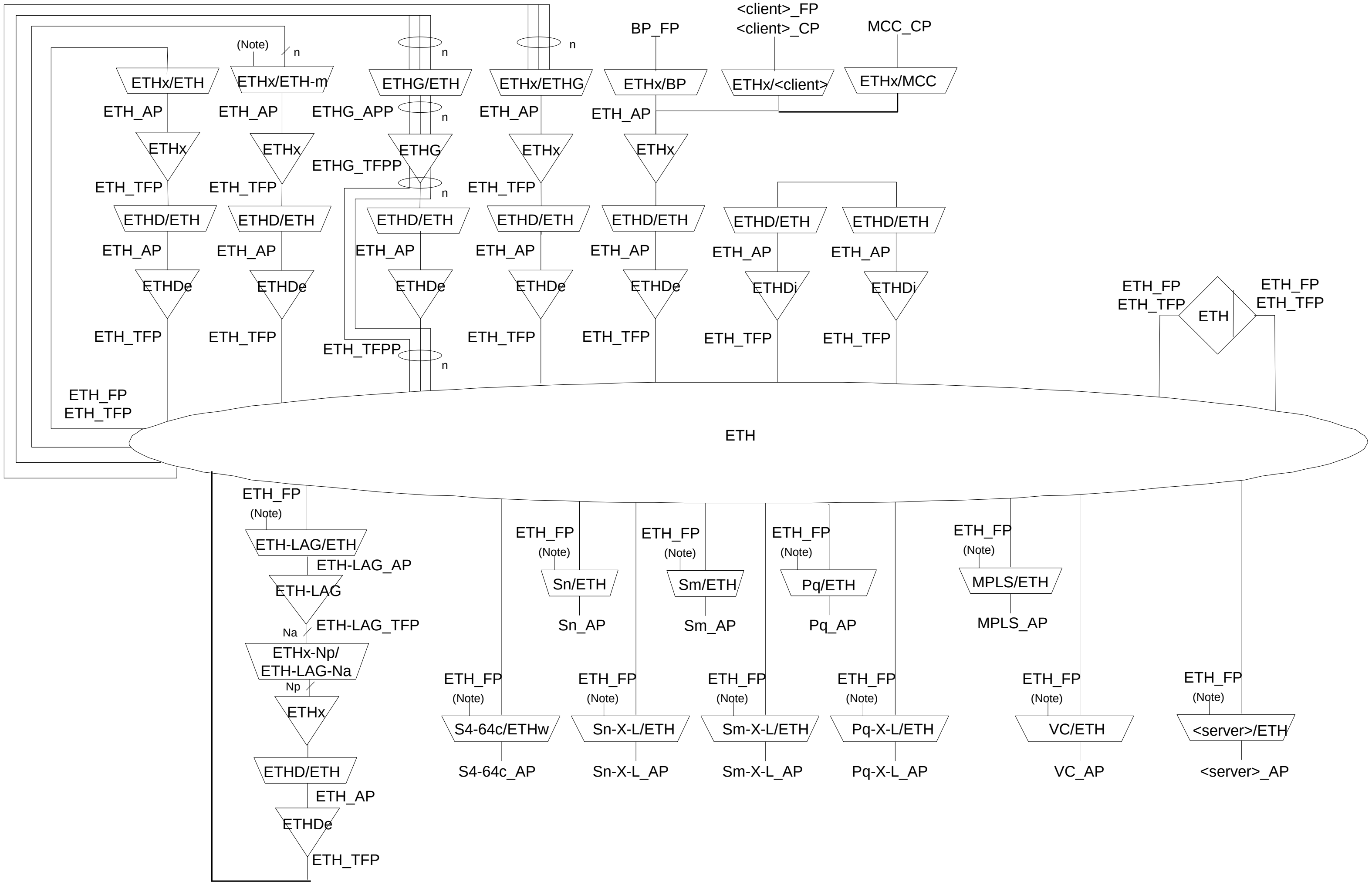
Not every functional block defined in this Recommendation is required for every application. Different subsets of functional blocks from this Recommendation and others (e.g., [ITU-T G.783], [ITU-T G.798], [ITU-T G.806] and [b-ITU-T I.732]) may be assembled in different ways according to the combination rules given in these Recommendations (e.g., [ITU-T G.806]) to provide a variety of different capabilities. Network operators and equipment suppliers may choose which functions must be implemented for each application.

The internal structure of the implementation of this functionality (equipment design) need not be identical to the structure of the functional model, as long as all the details of the externally observable behaviour comply with the equipment functional specification (EFS).

Equipment developed prior to the production of this Recommendation may not comply with all the details in this Recommendation.

The equipment requirements described in this Recommendation are generic and no particular physical partitioning of functions is implied. The input/output information flows associated with the functional blocks define the functions of the blocks and are considered to be conceptual, not physical.

Figure 1-1 presents a summary illustration of the set of atomic functions associated with the Ethernet signal transport. These atomic functions may be combined in various ways to support a variety of Ethernet services, some of which are illustrated in Appendix I. In order to reduce the complexity of the figures, the functions for the processing of management communication channels (e.g., SDH DCC or OTH COMMS) are not shown. For DCC or COMMS functions, refer to the specific layer network descriptions.



NOTE – ETH_TFP interface of adaptation functions towards the ETH_FT functions connects to logical link control. See [ITU-T G.8010] and function definition for details.

Figure 1-1 – Overview of ITU-T G.8021/Y.1341 atomic model functions

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T G.707]	Recommendation ITU-T G.707/Y.1322 (2007), *Network node interface for the synchronous digital hierarchy (SDH).*
[ITU-T G.709]	Recommendation ITU-T G.709/Y.1331 (2016), *Interfaces for the optical transport network (OTN).*
[ITU-T G.783]	Recommendation ITU-T G.783 (2006), *Characteristics of synchronous digital hierarchy (SDH) equipment functional blocks.*
[ITU-T G.798]	Recommendation ITU-T G.798 (2017), *Characteristics of optical transport network hierarchy equipment functional blocks.*
[ITU-T G.805]	Recommendation ITU-T G.805 (2000), *Generic functional architecture of transport networks.*
[ITU-T G.806]	Recommendation ITU-T G.806 (2012), *Characteristics of transport equipment Description methodology and generic functionality.*
[ITU-T G.809]	Recommendation ITU-T G.809 (2003), *Functional architecture of connectionless layer networks.*
[ITU-T G.832]	Recommendation ITU-T G.832 (1998), *Transport of SDH elements on PDH networks – Frame and multiplexing structures.*
[ITU-T G.7041]	Recommendation ITU-T G.7041/Y.1303 (2016), *Generic framing procedure.*
[ITU-T G.7043]	Recommendation ITU-T G.7043/Y.1343 (2004), *Virtual concatenation of plesiochronous digital hierarchy (PDH) signals.*
[ITU-T G.8001]	Recommendation ITU-T G.8001/Y.1354 (2016), *Terms and definitions for Ethernet frames over transport.*
[ITU-T G.8010]	Recommendation ITU-T G.8010/Y.1306 (2004), *Architecture of Ethernet layer networks.*
[ITU-T G.8011]	Recommendation ITU-T G.8011/Y.1307 (2016), *Ethernet service characteristics.*
[ITU-T G.8012]	Recommendation ITU-T G.8012/Y.1308 (2004), *Ethernet UNI and Ethernet NNI.*
[ITU-T G.8013]	Recommendation ITU-T G.8013/Y.1731 (2015), *OAM functions and mechanisms for Ethernet based networks.*
[ITU-T G.8023]	Recommendation ITU-T G.8023 (2018), *Characteristics of equipment functional blocks supporting Ethernet physical layer and Flex Ethernet interfaces.*
[ITU-T G.8031]	Recommendation ITU-T G.8031/Y.1342 (2015), *Ethernet linear protection switching.*
[ITU-T G.8032]	Recommendation ITU-T G.8032/Y.1344 (2015), *Ethernet ring protection switching.*

[ITU-T G.8040] Recommendation ITU-T G.8040/Y.1340 (2005), *GFP frame mapping into Plesiochronous Digital Hierarchy (PDH).*

[ITU-T Z.101] Recommendation ITU-T Z.101 (2016), *Specification and Description Language (SDL) – Basic SDL-2010.*

[IEEE 802.1AB] IEEE 802.1AB (2016), *IEEE Standard for Local and Metropolitan Area Networks: Station and Media Access Control Connectivity Discovery.*

[IEEE 802.1AX] IEEE 802.1AX (2014), *IEEE Standard for Local and Metropolitan Area Networks: Link Aggregation.*

[IEEE 802.1Q] IEEE 802.1Q (2014), *IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks*

[IEEE 802.1X] IEEE 802.1X (2010), *IEEE Standard for Local and metropolitan area networks – Port-Based Network Access Control.*

[IEEE 802.3] IEEE 802.3 (2015), *IEEE Standard for Ethernet*

[MEF 10.3] MEF Technical Specification 10.3 (2013), *Ethernet Service Attributes Phase 3,* Metro Ethernet Forum.

[OIF FLEXE IA] OIF, *Flex Ethernet Implementation Agreement 1.1 (2017)*

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 access point: [ITU-T G.805], [ITU-T G.809]

3.1.2 adaptation: [ITU-T G.809]

3.1.3 adapted information: [ITU-T G.809]

3.1.4 characteristic information: [ITU-T G.809]

3.1.5 client/server relationship: [ITU-T G.809]

3.1.6 connection point: [ITU-T G.805]

3.1.7 connectionless trail: [ITU-T G.809]

3.1.8 consequent actions: [ITU-T G.806]

3.1.9 defect correlations: [ITU-T G.806]

3.1.10 defects: [ITU-T G.806]

3.1.11 dual-ended: [ITU-T G.8001]

3.1.12 Ethernet flow replication point (ETHF_PP): [ITU-T G.8001]

3.1.13 Ethernet replicated information (ETH_PI): [ITU-T G.8001]

3.1.14 Ethernet termination flow replication point (ETHTF_PP): [ITU-T G.8001]

3.1.15 flow: [ITU-T G.809]

3.1.16 flow domain: [ITU-T G.809]

3.1.17 flow domain flow: [ITU-T G.809]

3.1.18 flow point: [ITU-T G.809]

3.1.19 flow point pool: [ITU-T G.809]

3.1.20 flow termination: [ITU-T G.809]

3.1.21 flow termination sink: [ITU-T G.809]

3.1.22 flow termination source: [ITU-T G.809]

3.1.23 generic framing procedure (GFP): [ITU-T G.7041]

3.1.24 jabber: [IEEE 802.3]

3.1.25 layer network: [ITU-T G.809]

3.1.26 link: [ITU-T G.805]

3.1.27 link connection: [ITU-T G.805]

3.1.28 link flow: [ITU-T G.809]

3.1.29 MAC frame: [IEEE 802.3]

3.1.30 media access control (MAC): [IEEE 802.3]

3.1.31 network: [ITU-T G.809]

3.1.32 network connection: [ITU-T G.805]

3.1.33 network flow: [ITU-T G.809]

3.1.34 network operator: [b-ITU-T M.3208.1]

3.1.35 network-to-network interface (NNI): [ITU-T G.8001]

3.1.36 one-way: [ITU-T G.8001]

3.1.37 ordered set: [IEEE 802.3]

3.1.38 performance filters: [ITU-T G.806]

3.1.39 physical layer entity (PHY): [IEEE 802.3]

3.1.40 port: [ITU-T G.809]

3.1.41 reference point: [ITU-T G.805] [ITU-T G.809]

3.1.42 reference points: [ITU-T G.806]

3.1.43 service provider: [b-ITU-T M.3208.1]

3.1.44 single-ended: [ITU-T G.8001]

3.1.45 termination connection point: [ITU-T G.805]

3.1.46 termination flow point: [ITU-T G.809]

3.1.47 termination flow point pool: Refer to clause 6.3.5.5 of [ITU-T G.8010]

3.1.48 timing point: [ITU-T G.806]

3.1.49 traffic conditioning function: [ITU-T G.8001]

3.1.50 traffic shaping function: [ITU-T G.8001]

3.1.51 traffic unit: [ITU-T G.809]

3.1.52 trail: [ITU-T G.805]

3.1.53 trail termination: [ITU-T G.805]

3.1.54 transport: [ITU-T G.809]

3.1.55 transport entity: [ITU-T G.809]

3.1.56 transport processing function: [ITU-T G.809]

3.1.57 two-way: [ITU-T G.8001]

3.1.58 user-to-network interface (UNI): [ITU-T G.8001]

3.2 Terms defined in this Recommendation

None.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

1DM	1-way Delay Measurement
A	Adaptation function
AI	Adapted Information
AIS	Alarm Indication Signal
AP	Access Point
APP	Access Point Pool
APS	Automatic Protection Switching
ATM	Asynchronous Transfer Mode
BER	Bit Error Ratio
BN	Bandwidth Notification
BNM	Bandwidth Notification Message
BPDU	Bridge Protocol Data Unit
BS	Bad Second
C	Connection Function
CBR	Constant Bit Rate
CC	Continuity Check
CCM	Continuity Check Message
CFI	Canonical Format Identifier
CI	Characteristic Information
CK	Clock
COMMS	Communications channel
CoS	Class of Service
CP	Connection Point
CRC	Cyclic Redundancy Check
CSF	Client Signal Fail
D	Data
DA	Destination Address
DCC	Data Communication Channel
DCI	Defect Clear Indication
DCN	Data Communication Network
DE	Drop Eligibility
DEI	Drop Eligible Identifier
DEG	Degraded
DEGM	Degraded M

DEGTHR	Degraded Threshold
DM	Delay Measurement
DMM	Delay Measurement Message
DMR	Delay Measurement Reply
EC	Ethernet Connection
ED	Expected Defect
EDM	Expected Defect Message
EFS	Equipment Functional Specification
EPL	Ethernet Private Line
EPLAN	Ethernet Private Local Area Network
ESMC	Ethernet Synchronization Message Channel
ETC	Ethernet Coding
ETH	Ethernet Media Access Control layer network
ETH_CI	Ethernet Media Access Control Characteristic Information
ETHD	Ethernet MAC layer network Diagnostic function
ETHDe	Ethernet MAC layer network Diagnostic function within MEP
ETHDi	Ethernet MAC layer network Diagnostic function within MIP
ETHG	Ethernet MAC layer network Group
ETH-m	Ethernet MAC layer network – multiplexing
ETH*x*	Ethernet MAC layer network at level *x* (*x* = path, tandem connection, section)
ETY	Ethernet physical layer network
EVC	Ethernet Virtual Connection
EVPL	Ethernet Virtual Private Line
EVPLAN	Ethernet Virtual Private Local Area Network
EXI	Extension Header Identifier
EXM	Extension Header Mismatch
FCS	Frame Check Sequence
FD	Flow Domain
FD	Frame Delay
FDI	Forward Defect Indication
FDF	Flow Domain Flow
FDV	Frame Delay Variation
FF	Flow Forwarding
FOP	Failure Of Protocol
FP	Flow Point
FPP	Flow Point Pool
FS	Frame Start
FT	Flow Termination
GFP	Generic Framing Procedure

GFP-F	Generic Framing Procedure – Frame mapped
GFP-T	Generic Framing Procedure – Transparent mapped
GNM	Generic Notification Message
GS	Good Second
GTCS	Group Traffic Conditioning and Shaping
LAG	Link Aggregation
LAN	Local Area Network
LB	LoopBack
LBM	LoopBack Message
LBR	LoopBack Reply
LCAS	Link Capacity Adjustment Scheme
LCK	Lock
LF	Lost Frames
LFD	Loss of Frame Delineation
LM	Loss Measurement
LMM	Loss Measurement Message
LMR	Loss Measurement Reply
LOC	Loss Of Continuity
LOS	Loss Of Signal
LT	Link Trace
LTM	Link Trace Message
LTR	Link Trace Reply
M_SDU	Media access control Service Data Unit
MAC	Media Access Control
MCC	Maintenance Communication Channel
ME	Maintenance Entity
MEG	Maintenance Entity Group
MEL	Maintenance Entity group Level
MEP	Maintenance entity group End Point
MI	Management Information
MIP	Maintenance entity group Intermediate Point
MMG	Mismerge
MP	Maintenance Point
MPLS	Multi-Protocol Label Switching
NNI	Network-to-Network Interface
OAM	Operations, Administration and Maintenance
ODU	Optical channel Data Unit
ODUk	Optical channel Data Unit – order k
OO	Out of Order

OPC	OpCode
OPU	Optical channel Payload Unit
OSSP	Organization Specific Slow Protocol
OTH	Optical Transport Hierarchy
OTN	Optical Transport Network
OUI	Organizational Unique Identifier
P	Priority
P11s	1544 kbit/s PDH path layer with synchronous 125 µs frame structure according to [b-ITU-T G.704]
P12s	2048 kbit/s PDH path layer with synchronous 125 µs frame structure according to [b-ITU-T G.704]
P31s	34 368 kbit/s PDH path layer with synchronous 125 µs frame structure according to [ITU-T G.832]
P4s	139 264 kbit/s PDH path layer with synchronous 125 µs frame structure according to [ITU-T G.832]
PCP	Priority Code Point
PCS	Physical Convergence Sublayer
PDH	Plesiochronous Digital Hierarchy
PDU	Protocol Data Unit
PFI	Payload FCS Indicator
PHY	Physical layer entity
PI	replication Information
PLM	Payload Mismatch
POH	Path Overhead
PP	replication Point
PP-OS	Preamble, Payload, and Ordered Set information
PRBS	Pseudo-Random Bit Sequence
PSI	Payload Structure Identifier
PT	Payload Type
PTI	Priority Type Identifier
QoS	Quality of Service
R-APS	Ring-Automatic Protection Switching
REC	Received
RES	Reserved
RDI	Remote Defect Indication
RI	Remote Information
RP	Remote Point
RxFCf	Received Frame Count far end
RxFCl	Received Frame Count local

SA	Source Address
SDH	Synchronous Digital Hierarchy
SDU	Service Data Unit
SL	Synthetic Loss
SLM	Synthetic Loss Message
SLR	Synthetic Loss Reply
SNC	Sub-Network Connection
SSD	Server Signal Degrade
SSF	Server Signal Fail
STM-N	Synchronous Transport Module – level N
svd	saved
TA	Target MAC Address
TCI	Tag Control Information
TCM	Tandem Connection Monitoring
TCP	Trail Connection Point
TCS	Traffic Conditioning and Shaping
TF	Transmitted Frames
TFP	Termination Flow Point
TFPP	Termination Flow Point Pool
TI	Timing Information
TID	Transaction Identifier
TLV	Type, Length, Value
TP	Timing Point
TPID	Tag Protocol Identifier
TSD	Trail Signal Degrade
TSF	Trail Signal Fail
TST	Test
TT	Trail Termination
TTL	Time To Live
TxFCf	Transmitted Frame Count far end
TxFCl	Transmitted Frame Count local
UNI	User-to-Network Interface
UNL	Unexpected maintenance entity group Level
UNM	Unexpected Maintenance entity group end point
UNP	Unexpected Period
UNPr	Unexpected Priority
UPI	(Generic Framing Procedure) User Payload Identifier
UPM	User Payload Mismatch
VID	Virtual local area network Identifier

VC	Virtual Channel (asynchronous transfer mode) or Virtual Container (synchronous digital hierarchy)
VCAT	Virtual Concatenation
VC-m	lower order Virtual Channel – order m
VC-n	higher order Virtual Channel – order n
VC-n-Xc	contiguous concatenated Virtual Channel – order n
VC-n-Xv	virtual concatenated Virtual Channel – order n
VLAN	Virtual Local Area Network

5 Methodology and conventions

For the basic methodology to describe transport network functionality of network elements, refer to clause 5 of [ITU-T G.806]. For Ethernet-specific extensions to the methodology, see clause 5 of [ITU-T G.8010].

All process descriptions in clauses 6, 8 and 9 use the SDL methodology defined in [ITU-T Z.101].

Pseudocode in this recommendation uses "switch" statements where each "case" statement is exclusive (i.e., "case" statements do not fall through to each other).

6 Supervision

The generic supervision functions are defined in clause 6 of [ITU-T G.806]. Specific supervision functions for the Ethernet transport network are defined in this clause.

6.1 Defects

6.1.1 Summary of detection and clearance conditions for defects

The defect detection and clearance conditions are based on events. Occurrence or absence of specific events may detect or clear specific defects.

In the following:

Valid means a received value is equal to the value configured via the MI input interface(s).

Invalid means a received value is not equal to the value configured via the MI input interface(s).

The events defined for this Recommendation are summarized in Table 6-1. Events, other than APS or R-APS events are generated by processes in the ETHx_FT_Sk function as defined in clause 9.2.1.2. APS events are generated by the subnetwork connection protection process as defined in clause 9.1.2. R-APS events are generated by the ring protection control process as defined in clause 9.1.3. These processes define the exact conditions for these events, Table 6-1 only provides a quick overview.

Table 6-1 – Overview of events

Event	Meaning
unexpMEL	Reception of a CCM frame with an invalid MEL value
unexpMEG	Reception of a CCM frame with an invalid MEG value, but with a valid MEL value
unexpMEP	Reception of a CCM frame with an invalid MEP value, but with valid MEL and MEG values
unexpPeriod	Reception of a CCM frame with an invalid periodicity value, but with valid MEL, MEG and MEP values

unexpPriority	Reception of a CCM frame with an invalid priority value, but with valid MEL, MEG, MEP and periodicity values
expCCM[i]	Reception of a CCM frame with valid MEL, MEG, MEP and periodicity values, where an MEP is indexed by "i"
RDI[i]=x	Reception by an MEP indexed by "i" of a CCM frame with valid MEL, MEG, MEP and periodicity values and the RDI flag set to x; where x=0 (remote defect clear) and x=1 (remote defect set)
LCK	Reception of a LCK frame
AIS	Reception of an AIS frame
CSF-LOS	Reception of a CSF frame that indicates a client loss of signal
CSF-FDI	Reception of a CSF frame that indicates a client forward defect indication
CSF-RDI	Reception of a CSF frame that indicates a client reverse defect indication
BS	Bad second, a second in which the lost frame ratio exceeds the degraded threshold (MI_LM_DEGTHR)
expAPS	Reception of a valid APS frame
expRAPS	Reception of a valid R-APS frame
APSw	Reception of an APS frame from the working transport entity
APSb	Reception of an APS frame with incompatible "B" bit value
APSr	Reception of an APS frame with incompatible "Requested Signal" value (NOTE)
RAPSpm	Reception by the RPL owner of an R-APS(NR, RB) frame with a node ID that differs from its own

NOTE – One way to detect this event is to detect that the transmitted "Requested Signal" and the received "Requested Signal" values differ, for example in case traffic switching occurs due to a local request.

The occurrence or absence of these events may detect or clear a defect. An overview of the conditions is given in Table 6-2. The notation "#event=x (K*period)" is used to indicate the occurrence of x events within the period as specified between the brackets; 3.25≤K≤3.5.

Table 6-2 gives a quick overview of the detection and clearance conditions for the various defects. In the following clauses 6.1.2, 6.1.3, 6.1.4 and 6.1.5 the precise conditions are specified using SDL diagrams.

Table 6-2 – Overview of defect detection and clearance

Defect	Defect detection	Defect clearance
dLOC[]	#expCCM[] == 0 (K*MI_CC_Period)	expCCM[]
dUNL	unexpMEL	#unexpMEL == 0 (K*CCM_Period)
dUNPr	unexpPriority	#unexpPriority == 0 (K*CCM_Period)
dMMG	unexpMEG	#unexpMEG == 0 (K*CCM_Period)
dUNM	unexpMEP	#unexpMEP == 0 (K*CCM_Period)
dUNP	unexpPeriod	#unexpPeriod == 0 (K*CCM_Period)
dRDI[]	RDI[] == 1	RDI[] == 0
dAIS	AIS	#AIS == 0 (K*AIS_Period)
dLCK	LCK	#LCK == 0 (K*LCK Period)

Table 6-2 – Overview of defect detection and clearance

Defect	Defect detection	Defect clearance
dCSF-LOS	CSF-LOS	#CSF-LOS == 0 (K*CSF_Period or CSF-DCI)
dCSF-FDI	CSF-FDI	#CSF-FDI == 0 (K*CSF_Period or CSF-DCI)
dCSF-RDI	CSF-RDI	#CSF-RDI == 0 (K*CSF_Period or CSF-DCI)
dDEG	#BadSecond == 1 (MI_LM_DEGM*1second)	#BadSecond == 0 (MI_LM_M*1second)
dFOP-CM	APSw	#APSw == 0 (K*normal APS Period)
dFOP-PM	APSb or RAPSpm	expAPS or #RAPSpm == 0 (K*long R-APS frame interval)
dFOP-NR	APSr continues more than 50ms	expAPS
dFOP-TO	#expAPS==0 (K * long APS interval) or #expRAPS==0 (K * long R-APS frame interval)	expAPS or expRAPS

Note that for the case of CCM_Period, AIS_Period, LCK_Period, and CSF_Period the values for the CCM, AIS, LCK, and CSF periods are based on the periodicity as indicated in the CCM, AIS, LCK, or CSF frame that triggered the timer to be started.

For dUNL, dMMG, dUNM, dUNP, dUNPr there may be multiple frames received detecting the same defect but carrying a different periodicity. In that case the longest received period will be used. See the detailed descriptions below.

6.1.2 Continuity supervision

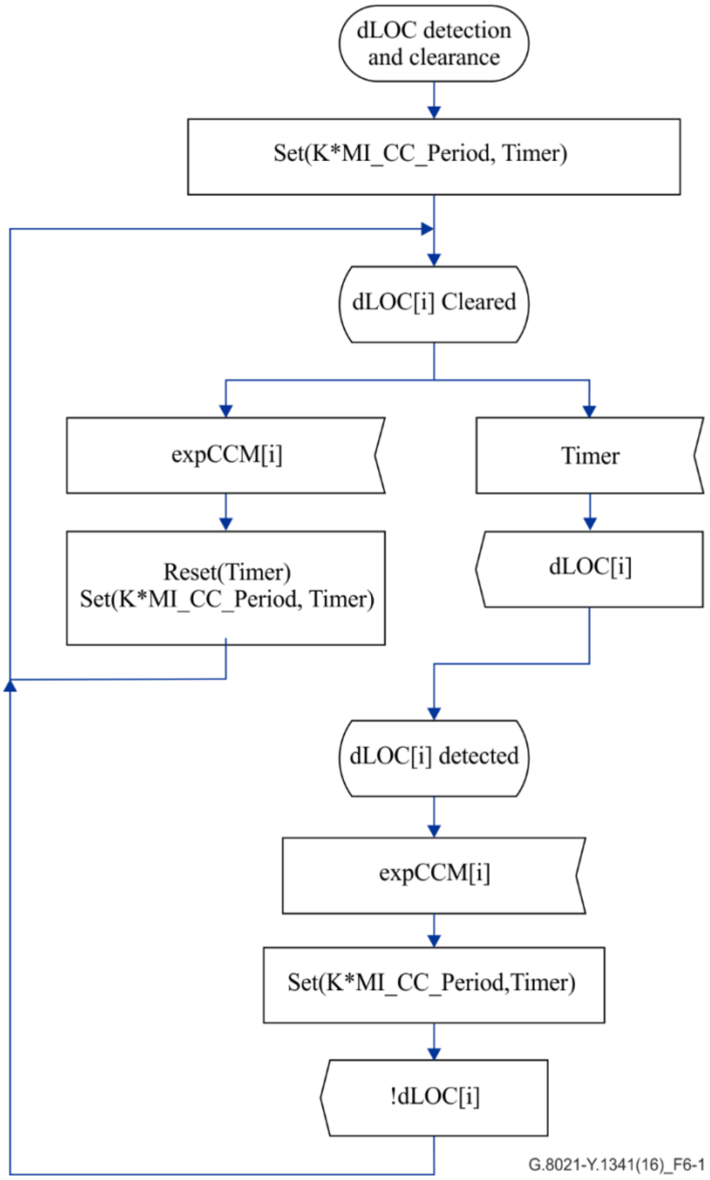


Figure 6-1 – dLOC[] detection and clearance process

6.1.2.1 Loss of continuity defect (dLOC[])

The loss of continuity defect is calculated at the ETH layer. It monitors the presence of continuity in ETH trails.

Its detection and clearance are defined in Figure 6-1. The timer in Figure 6-1 is set to K*MI_CC_Period, where MI_CC_Period corresponds to the configured CCM period and K is such that $3.25 \leq K \leq 3.5$.

NOTE – The dLOC entry/exit criteria defined in this version of the Recommendation are different to those defined in previous versions of this Recommendation (i.e., the 2007 and 2010 versions), because they have been aligned with those defined in clause 21 of [IEEE 802.1Q]. This change impacts only the conditions for defect detection and therefore does not affect interoperability between equipment compliant with this version of the Recommendation (and/or with clause 21 of [IEEE 802.1Q]) and those compliant with older versions of this Recommendation.

6.1.3 Connectivity supervision

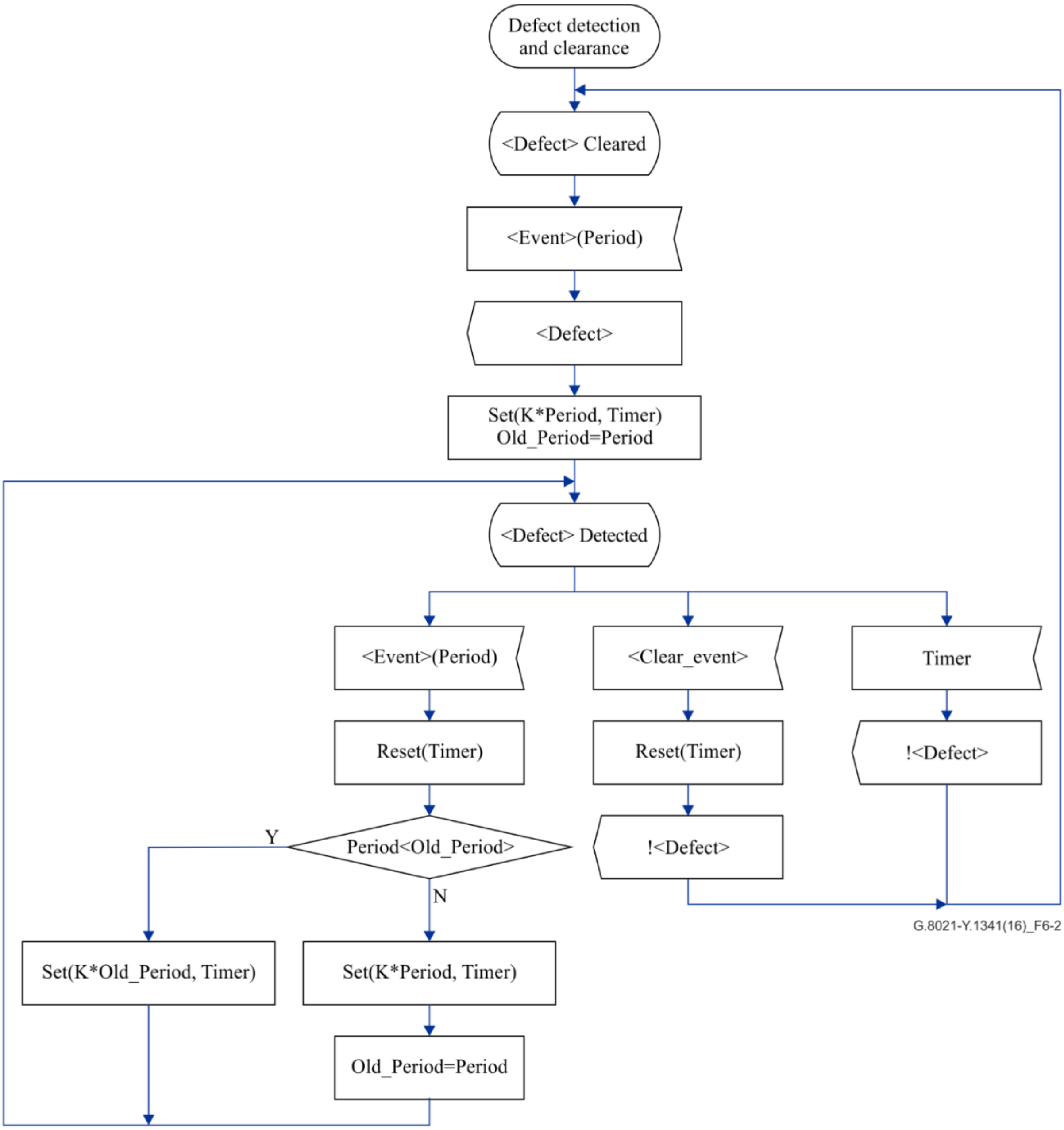


Figure 6-2 – Defect detection and clearance process for dUNL, dMMG, dUNM, dUNP, dUNPr, dAIS, dLCK, and dCSF

Figure 6-2 shows a generic state diagram that is used to detect and clear the dUNL, dMMG, dUNM, dUNP, dUNPr, dAIS, dLCK and dCSF defects. In this diagram <Defect> needs to be replaced with the specific defect and <Event> with the specific event related to this defect. Furthermore, in Figure 6-2, $3.25 \leq K \leq 3.5$.

Figure 6-2 shows that the timer is set based on the last received period value, unless an earlier OAM frame triggering <Event> (and therefore the detection of <Defect>) carried a longer period. As a consequence clearing certain defects may take more time than necessary.

6.1.3.1 Unexpected MEL defect (dUNL)

The unexpected MEL defect is calculated at the ETH layer. It monitors the connectivity in a maintenance entity group.

Its detection and clearance are defined in Figure 6-2. The <Defect> in Figure 6-2 is dUNL. The <Event> in Figure 6-2 is the unexpMEL event (generated by the CCM reception process in clause 8.1.7.3) and the period is the period carried in the CCM frame that triggered this event, unless an earlier CCM frame triggering an unexpMEL event carried a greater period.

6.1.3.2 Mismerge defect (dMMG)

The mismerge defect is calculated at the ETH layer. It monitors the connectivity in a maintenance entity group.

Its detection and clearance are defined in Figure 6-2. The <Defect> in Figure 6-2 is dMMG. The <Event> in Figure 6-2 is the unexpMEG event (as generated by the CCM reception process in clause 8.1.7.3) and the period is the period carried in the CCM frame that triggered the event, unless an earlier CCM frame triggering an unexpMEG event carried a greater period.

6.1.3.3 Unexpected MEP defect (dUNM)

The unexpected MEP defect is calculated at the ETH layer. It monitors the connectivity in a maintenance entity group.

Its detection and clearance are defined in Figure 6-2. The <Defect> in Figure 6-2 is dUNM. The <Event> in Figure 6-2 is the unexpMEP event (as generated by the CCM reception process in clause 8.1.7.3) and the period is the period carried in the CCM frame that triggered the event, unless an earlier CCM frame triggering an unexpMEP event carried a greater period.

6.1.3.4 Degraded signal defect (dDEG)

This defect is only defined for point-to-point ETH connections.

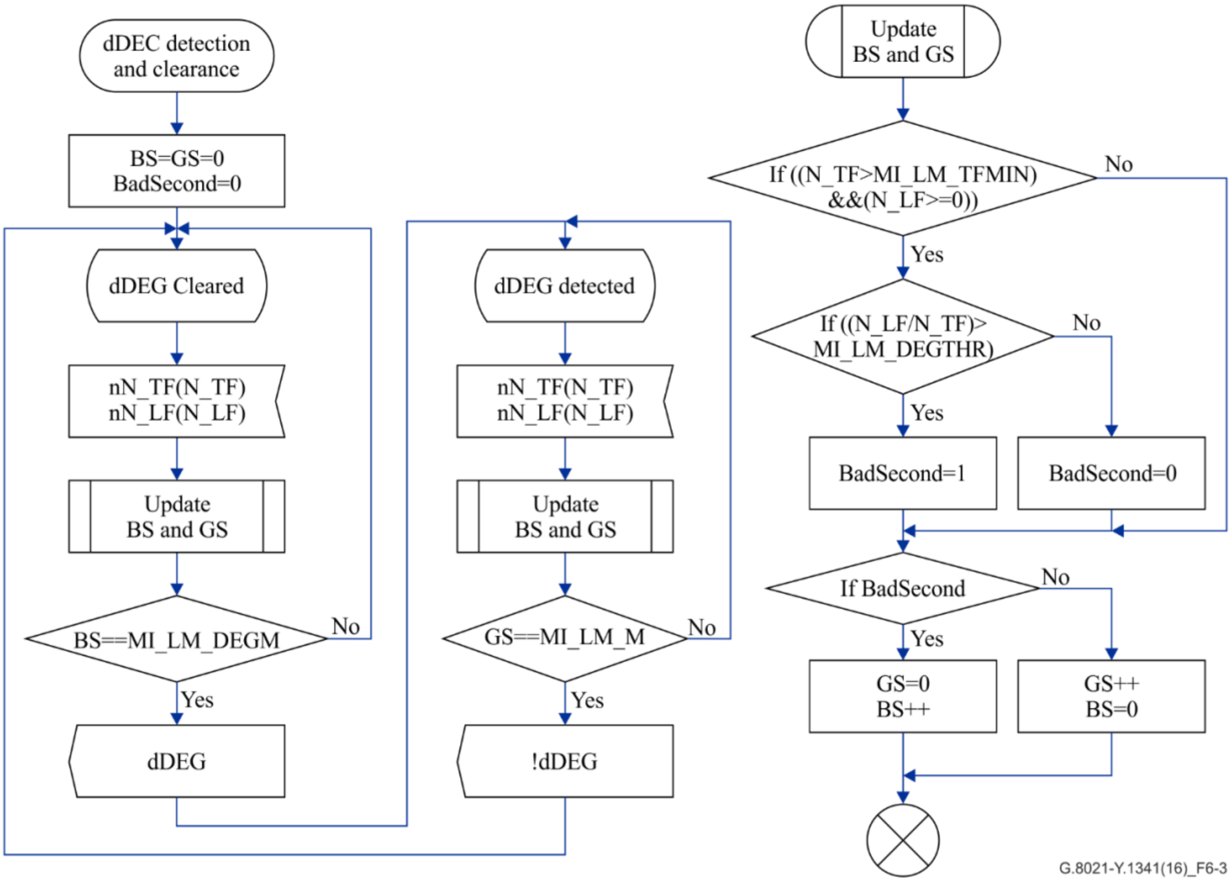


Figure 6-3 – dDEG detection and clearance process

The degraded signal defect is calculated at the ETH layer. It monitors the connectivity of an ETH trail. Its detection and clearance are defined in Figure 6-3.

Every second the state machine receives the one-second counters for near end received and transmitted frames and determines whether the second was a bad second. The defect is detected if there are MI_LM_DEGM consecutive bad seconds and cleared if there are MI_LM_M consecutive good seconds.

In order to declare a bad second the number of transmitted frames must exceed a threshold (MI_LM_TFMIN). Furthermore, if the frame loss ratio (lost frames/transmitted frames) is greater than MI_LM_DEGTHR, a bad second is declared.

6.1.4 Protocol supervision

6.1.4.1 Unexpected periodicity defect (dUNP)

The unexpected periodicity defect is calculated at the ETH layer. It detects the configuration of different periodicities at different MEPs belonging to the same MEG.

Its detection and clearance are defined in Figure 6-2. The <Defect> in Figure 6-2 is dUNP. The <Event> in Figure 6-2 is the unexpPeriod event (as generated by the CCM reception process in clause 8.1.7.3) and the period is the period carried in the CCM frame that triggered the event, unless an earlier CCM frame triggering an unexpPeriod event carried a greater period.

6.1.4.2 Unexpected priority defect (dUNPr)

The unexpected priority defect is calculated at the ETH layer. It detects the configuration of different priorities for CCM at different MEPs belonging to the same MEG.

Its detection and clearance are defined in Figure 6-2. The <Defect> in Figure 6-2 is dUNPr. The <Event> in Figure 6-2 is the unexpPriority event (as generated by the CCM reception process in clause 8.1.7.3) and the period is the period carried in the CCM frame that triggered the event, unless an earlier CCM frame triggering an unexpPriority event carried a greater period.

6.1.4.3 Protection protocol supervision

6.1.4.3.1 Linear or ring protection failure of protocol provisioning mismatch (dFOP-PM)

The failure of protocol provisioning mismatch defect is calculated at the ETH layer. It monitors the provisioning mismatch of:

- linear protection by comparing B bits of the transmitted and the received APS protocol, or
- ring protection by comparing the node ID of the RPL owner and the node ID in a received R-APS(NR, RB) frame.

Its detection and clearance are defined in Table 6-2. dFOP-PM is detected:

- in the case of linear protection, on receipt of an APSb event and cleared on receipt of an expAPS event. These events are generated by the subnetwork connection protection process (clause 9.1.2), or
- in the case of ring protection, on receipt of an RAPSpm event and cleared on receipt of no RAPSpm event during K times the long R-APS frame intervals defined in [ITU-T G.8032], where $3.25 \le K \le 3.5$. These events are generated by the ring protection control process (clause 9.1.3).

6.1.4.3.2 Linear protection failure of protocol no response (dFOP-NR)

The failure of protocol no response defect is calculated at the ETH layer. It monitors incompletion of protection switching by comparing the transmitted "Requested Signal" values and the received "Requested Signal" in the APS protocol.

Its detection and clearance are defined in Table 6-2. dFOP-NR is detected when an APSr event continues for more than 50ms and it is cleared on receipt of the expAPS event. These events are generated by the subnetwork connection protection process (clause 9.1.2). This defect is not applied in the case of a unidirectional protection switching operation.

6.1.4.3.3 Linear protection failure of protocol configuration mismatch (dFOP-CM)

The failure of protocol configuration mismatch defect is calculated at the ETH layer. It monitors working and protection configuration mismatch by detecting the receipt of the APS protocol from the working transport entity.

Its detection and clearance are defined in Table 6-2. dFOP-CM is detected on receipt of an APSw event and cleared on receipt of no APSw event during K times the normal APS transmission period defined in [ITU-T G.8031], where $3.25 \le K \le 3.5$. These events are generated by the subnetwork connection protection process (clause 9.1.2).

6.1.4.3.4 Linear or ring protection failure of protocol time out (dFOP-TO)

The failure of protocol time out defect is calculated at the ETH layer. It monitors the time out defect of:

- linear protection by detecting the prolonged absence of expected APS frames, or
- ring protection by detecting the prolonged absence of expected R-APS frames.

Its detection and clearance are defined in Table 6-2.

In the case of linear protection, dFOP-TO is detected on receipt of no expAPS event during K times the long APS interval defined in [ITU-T G.8031] (where K >= 3.5). dFOP-TO is cleared on receipt of an expAPS event. These events are generated by the subnetwork connection protection process (clause 9.1.2).

In the case of ring protection, dFOP-TO is detected on receipt of no expRAPS event during K times the long R-APS frame intervals defined in [ITU-T G.8032] (where K>=3.5). dFOP-TO is cleared on receipt of an expRAPS event. These events are generated by the ring protection control process (clause 9.1.3).

6.1.5 Maintenance signal supervision

6.1.5.1 Remote defect indicator defect (dRDI[])

The remote defect indicator defect is calculated at the ETH layer. It monitors the presence of an RDI maintenance signal.

dRDI is detected on receipt of the RDI[]=1 event and cleared on receipt of the RDI[]=0 event. These events are generated by the CCM reception process.

6.1.5.2 Alarm indication signal defect (dAIS)

The alarm indication signal defect is calculated at the ETH layer. It monitors the presence of an AIS maintenance signal.

Its detection and clearance conditions are defined in Figure 6-2. The <Defect> in Figure 6-2 is dAIS. The <Event> in Figure 6-2 is the AIS event (as generated by the AIS reception process in clause 9.2.1.2) and the period is the period carried in the AIS frame that triggered the event, unless an earlier AIS frame carried a greater period.

6.1.5.3 Locked defect (dLCK)

The locked defect is calculated at the ETH layer. It monitors the presence of a locked maintenance signal.

Its detection and clearance conditions are defined in Figure 6-2. The <Defect> in Figure 6-2 is dLCK. The <Event> in Figure 6-2 is the LCK event (as generated by the LCK reception process in clause 9.2.1.2) and the period is the period carried in the LCK frame that triggered the event, unless an earlier LCK frame carried a greater period.

6.1.5.4 Client signal fail defect (dCSF)

The CSF (CSF-LOS, CSF-FDI, and CSF-RDI) defect is calculated at the ETH layer. It monitors the presence of a CSF maintenance signal.

Its detection and clearance conditions are defined in Figure 6-2. The <Defect> in Figure 6-2 is dCSF-LOS, dCSF-FDI, or dCSF-RDI. The <Event> in Figure 6-2 is the CSF event (as generated by the CSF extract process in clause 8.1.17) and the period is the period carried in the CSF frame that triggered the event, unless an earlier CSF frame carried a greater period.

The <Clear_event> in Figure 6-2 is the CSF event which indicates defect clear indication (DCI).

6.2 Consequent actions

For consequent actions see [ITU-T G.806] and the specific atomic functions.

6.3 Defect correlations

For defect correlations see the specific atomic functions.

6.4 Performance filters

6.4.1 One-second performance monitoring filters associated with counts

For further study.

6.4.2 Performance monitoring filters associated with gauges

For further study.

7 Information flow across reference points

See clause 7 of [ITU-T G.806] for the generic description of information flow. For Ethernet-specific information flow, see the description of the functions in clause 9.

8 Generic processes for Ethernet equipment

This clause defines processes specific to equipment supporting the Ethernet transport network.

8.1 OAM related processes

8.1.1 OAM MEL filter

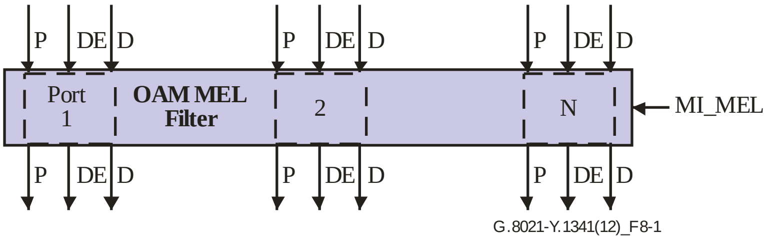


Figure 8-1 – OAM MEL filter process

The OAM MEL filter process filters incoming ETH OAM traffic units based on the MEL they carry. All traffic units with an MEL equal to or lower than the MEL provided by the MI_MEL signal are discarded.

The criteria for filtering depends on the values of the fields in the M_SDU field of the ETH_CI_D signal.

The ETH OAM traffic unit and complementing P and DE signals will be filtered, if

- length/type field = OAM EtherType (89-02 as defined in clause 10 of [ITU-T G.8013]), and
- MEL field <= MI_MEL

Figure 8-1 shows the OAM MEL filter process for multiple ports. Figure 8-2 shows the filtering process that is running per port.

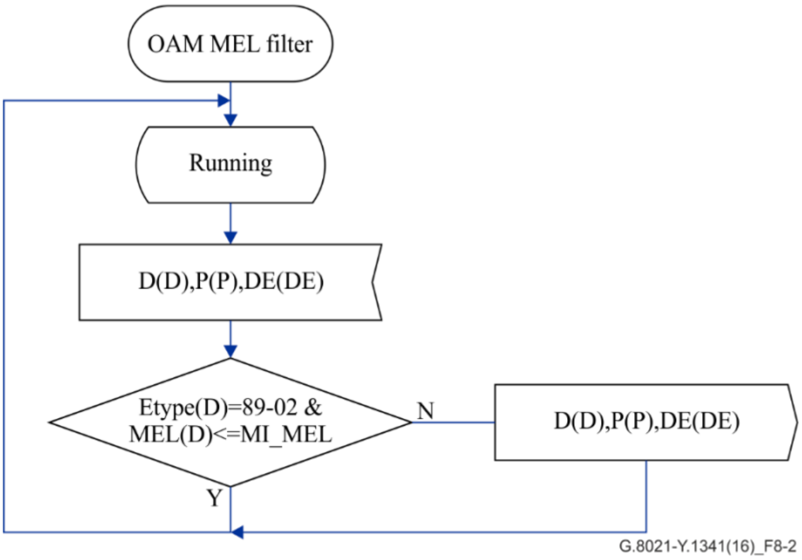


Figure 8-2 – OAM MEL filter behaviour

8.1.2 LCK generation process

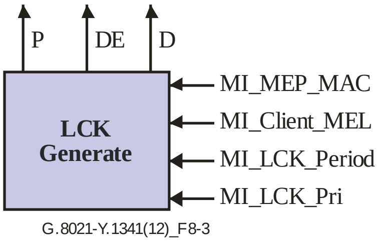


Figure 8-3 – LCK generation process

The LCK generation process generates ETH_CI traffic units where the ETH_CI_D signal contains the LCK signal. Figure 8-4 defines the behaviour of the LCK generation process.

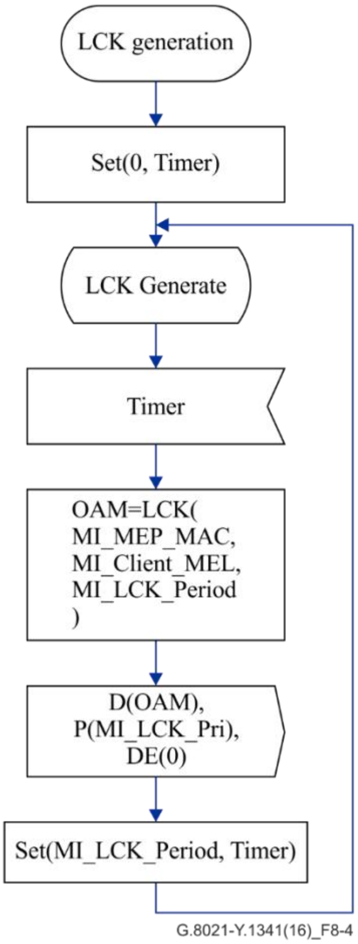


Figure 8-4 – LCK generation behaviour

The LCK generation process continuously generates LCK traffic units; every time the timer expires an LCK traffic unit will be generated. The period between two consecutive traffic units is determined by the MI_LCK_Period input signal. Allowed values are defined in Table 8-1.

Table 8-1 – LCK period values

3-bits	Period value	Comments
000-011	Invalid value	Invalid value for LCK PDUs
100	1s	1 frame per second
101	Invalid value	Invalid value for LCK PDUs
110	1 min	1 frame per minute
111	Invalid value	Invalid value for LCK PDUs

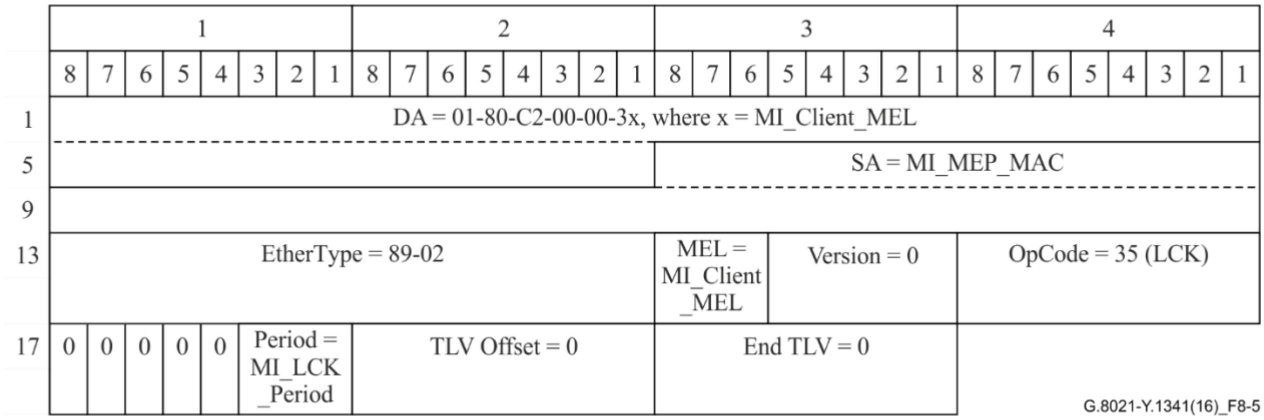
The ETH_CI_D signal contains a source and destination address field and an M_SDU field. The format of the M_SDU field for LCK traffic units is defined in clauses 9.1 and 9.8 of [ITU-T G.8013]. The MEL in the M_SDU field is determined by the MI_Client_MEL input parameter.

The values of the source and destination address fields in the ETH_CI_D signal are determined by the local MAC address (SA) and the multicast class 1 DA as described in [ITU-T G.8013] (DA). The value of the multicast class 1 DA is 01-80-C2-00-00-3x, where x is equal to MI_Client_MEL as defined in clause 10.1 of [ITU-T G.8013]. The value of MI_MEP_MAC should be a valid unicast MAC address.

The periodicity (as defined by MI_LCK_Period) is encoded in the three least significant bits of the flags field in the LCK PDU using the values from Table 8-1.

The LCK (SA, Client_MEL, Period) function generates an LCK traffic unit with the SA, MEL and period fields defined by the values of the parameters. Figure 8-5 below shows the ETH_CI_D signal format resulting from the function call from Figure 8-4:

OAM=LCK(
MI_MEP_MAC,
MI_Client_MEL,
MI_LCK_Period
)



	1	2	3	4
	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1
1	DA = 01-80-C2-00-00-3x, where x = MI_Client_MEL			
5			SA = MI_MEP_MAC	
9				
13	EtherType = 89-02		MEL = MI_Client_MEL Version = 0	OpCode = 35 (LCK)
17	0 0 0 0 0 Period = MI_LCK_Period	TLV Offset = 0	End TLV = 0	

G.8021-Y.1341(16)_F8-5

Figure 8-5 – LCK traffic unit

The value of the ETH_CI_P signal associated with the generated LCK traffic units is defined by the MI_LCK_Pri input parameter; valid values are in the range 0-7.

The value of the ETH_CI_DE signal associated with the generated LCK traffic units is always set to drop ineligible.

8.1.3 Selector process

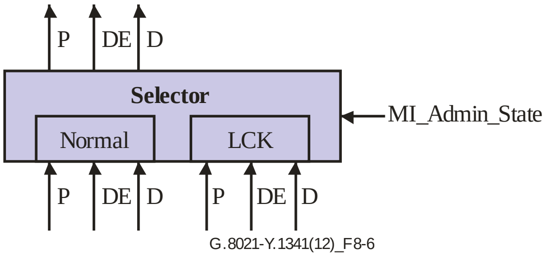


Figure 8-6 – Selector process

The selector process selects the valid signal from the input of the normal ETH_CI signal or the ETH_CI LCK signal (as generated by the LCK generation process). The normal signal is blocked if MI_Admin_State is LOCKED. The behaviour is defined in Figure 8-7.

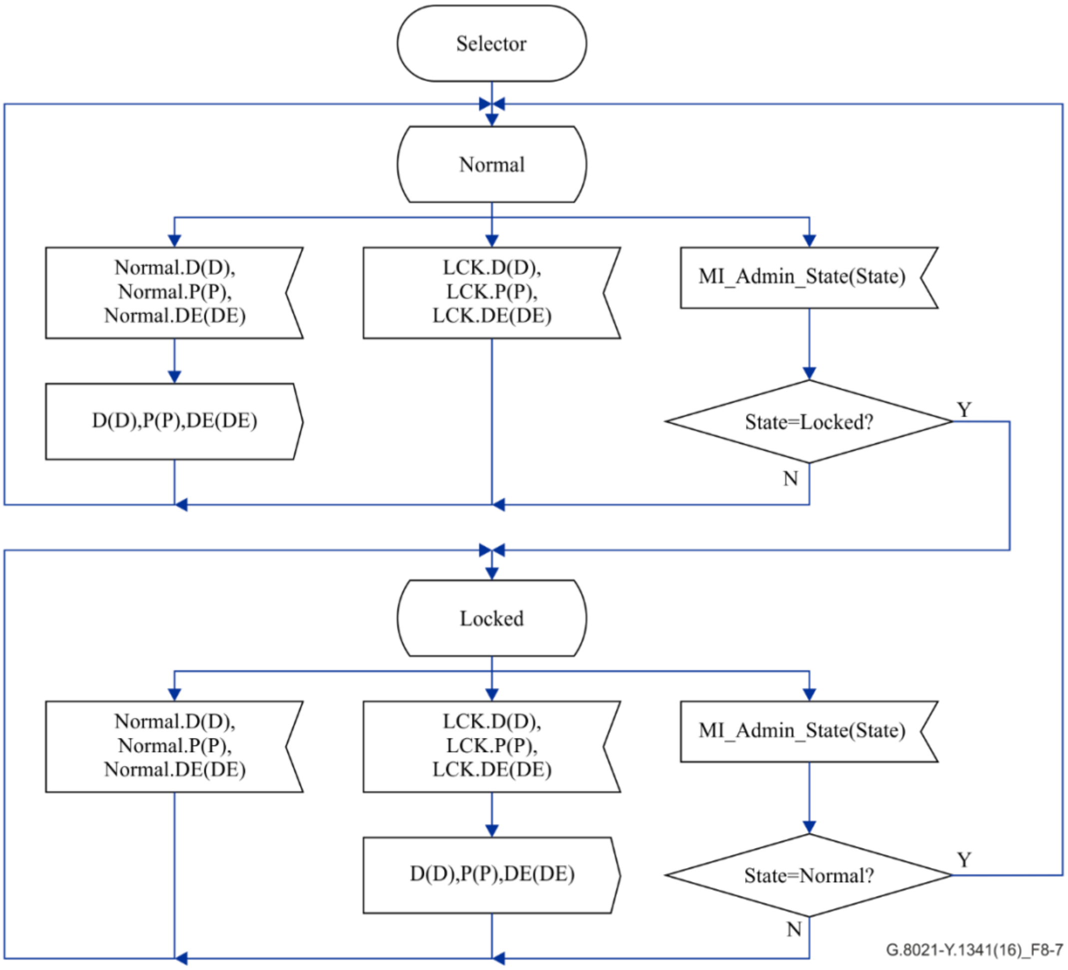


Figure 8-7 – Selector behaviour

8.1.4 AIS insert process

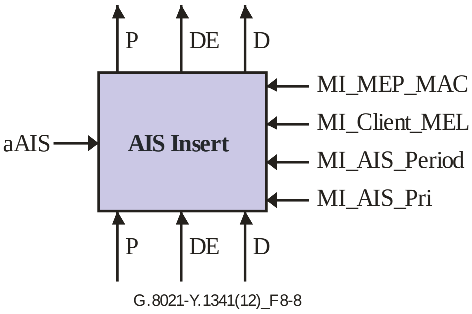


Figure 8-8 – AIS insert process

Figure 8-8 shows the AIS insert process symbol and Figure 8-9 defines the behaviour. If the aAIS signal is true, the AIS insert process continuously generates ETH_CI traffic units where the ETH_CI_D signal contains the AIS signal until the aAIS signal is false. The generated AIS traffic units are inserted in the incoming stream, i.e., the output stream contains the incoming traffic units and the generated AIS traffic units.

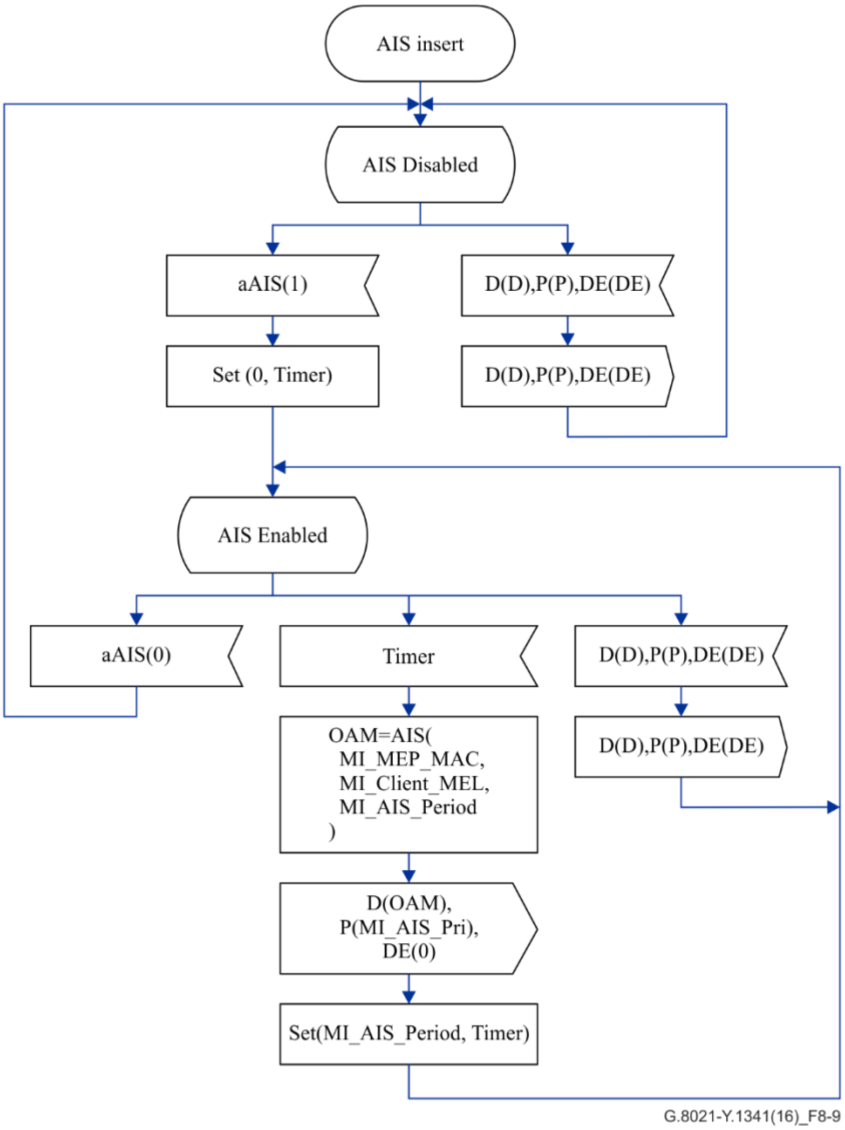


Figure 8-9 – AIS insert behaviour

The period between consecutive AIS traffic units is determined by the MI_AIS_Period parameter. Allowed values are once per second and once per minute; the encoding of these values is defined in Table 8-2. Note that this encoding is the same as for the LCK generation process.

Table 8-2 – AIS period values

3-bits	Period Value	Comments
000-011	Invalid Value	Invalid value for AIS PDUs
100	1s	1 frame per second
101	Invalid Value	Invalid value for AIS PDUs
110	1 min	1 frame per minute
111	Invalid Value	Invalid value for AIS PDUs

The ETH_CI_D signal contains a source and destination address field and an M_SDU field. The format of the M_SDU field for AIS traffic units is defined in clauses 9.1 and 9.7 of [ITU-T G.8013]. The MEL in the M_SDU field is determined by the MI_Client_MEL input parameter.

The values of the source and destination address fields in the ETH_CI_D signal are determined by the local MAC address (SA) and the multicast class 1 DA as described in [ITU-T G.8013] (DA). The value of the multicast class 1 DA is 01-80-C2-00-00-3x, where x is equal to MI_Client_MEL as defined in clause 10.1 of [ITU-T G.8013]. The value of MI_MEP_MAC should be a valid unicast MAC address.

The periodicity (as defined by MI_AIS_Period) is encoded in the three least significant bits of the flags field in the AIS PDU using the values from Table 8-2.

The AIS (SA, Client_MEL, Period) function generates an AIS traffic unit with the SA, MEL and period fields defined by the values of the parameters. Figure 8-10 below shows the ETH_CI_D signal format resulting from the function call from Figure 8-9:

OAM=AIS(
MI_MEP_MAC,
MI_Client_MEL,
MI_AIS_Period
)

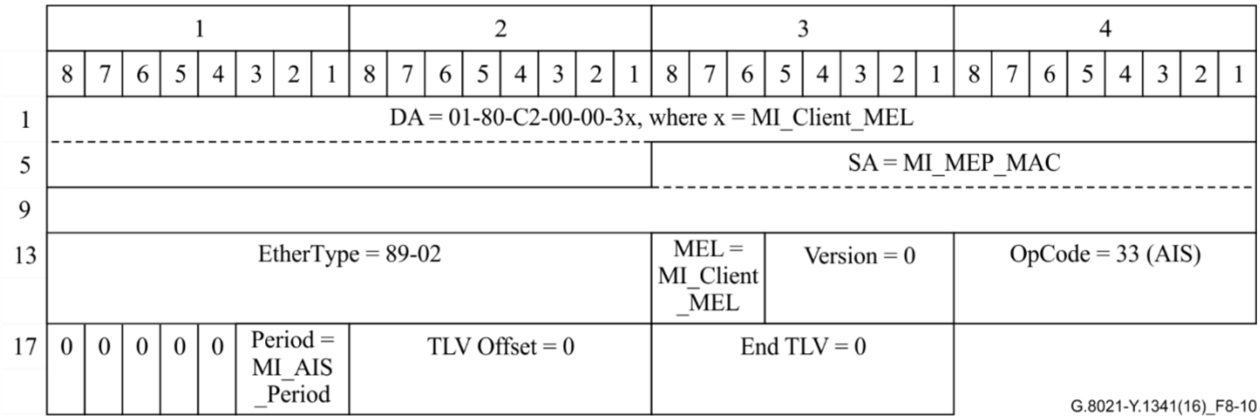


Figure 8-10 – AIS traffic unit

The value of the ETH_CI_P signal associated with the generated AIS traffic units is defined by the MI_AIS_Pri input parameter; valid values are in the range 0-7.

The value of the ETH_CI_DE signal associated with the generated AIS traffic units is always set to drop ineligible.

8.1.5 APS insert process

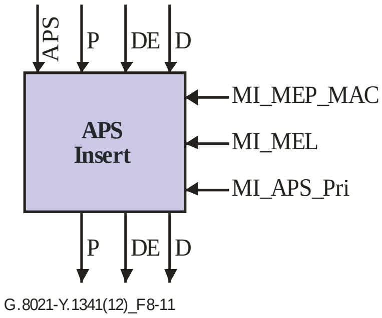


Figure 8-11 – APS insert process

The APS insert process encodes the ETH_CI_APS (APS input signal in Figure 8-11) signal into the ETH_CI_D signal of an ETH_CI traffic unit; the resulting APS traffic unit is inserted into the stream of incoming traffic units, i.e., the outgoing stream consists of the incoming traffic units and the inserted APS traffic units. The ETH_CI_APS signal contains the APS specific information as defined in clause 11.1 of [ITU-T G.8031] (APS format). The behaviour is defined in Figure 8-12.

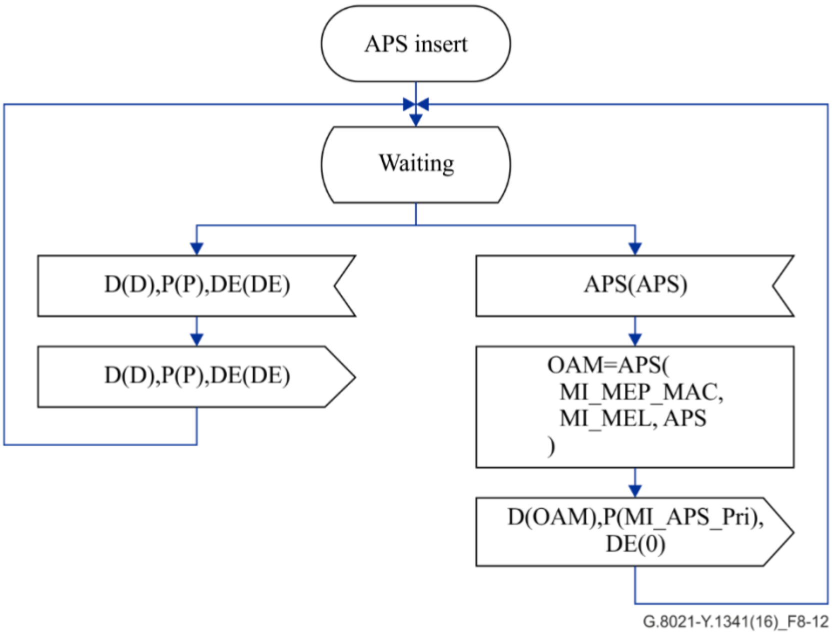


Figure 8-12 – APS insert behaviour

The ETH_CI_D signal contains a source and destination address field and an M_SDU field. The format of the M_SDU field for APS traffic units is defined in clauses 9.1 and 9.10 of [ITU-T G.8013]. The MEL in the M_SDU field is determined by the MI_MEL input parameter.

The values of the source and destination address fields in the ETH_CI_D signal are determined by the local MAC address (SA) and the multicast class 1 DA as described in [ITU-T G.8013] (DA). The value of the multicast class 1 DA is 01-80-C2-00-00-3x, where x is equal to MI_MEL as defined in clause 10.1 of [ITU-T G.8013]. The value of MI_MEP_MAC should be a valid unicast MAC address.

The APS(SA, MEL, APS) function generates an APS traffic unit with the SA, MEL and APS fields defined by the values of the parameters. Figure 8-13 below shows the ETH_CI_D signal format resulting from the function call from Figure 8-12:

OAM=APS(
MI_MEP_MAC,
MI_MEL,
APS
)

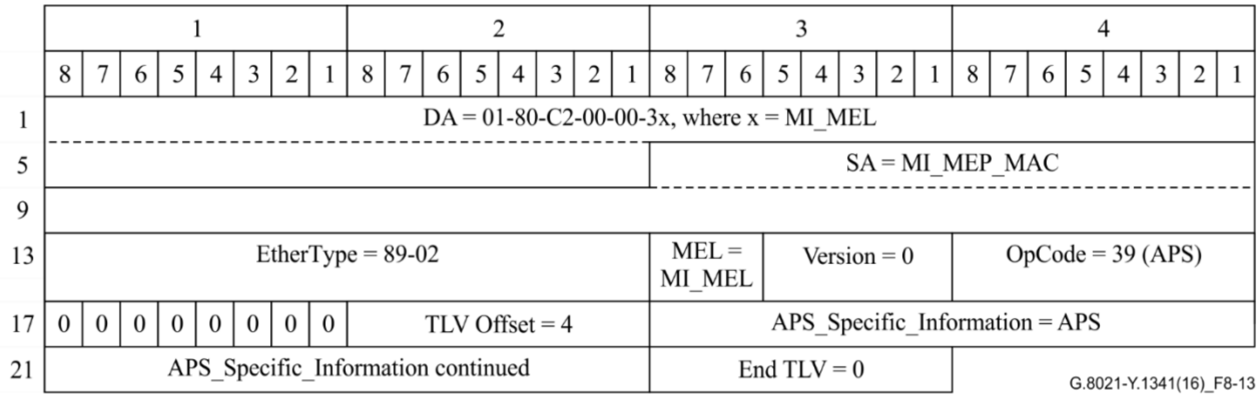


Figure 8-13 – APS traffic unit

The value of the ETH_CI_P signal associated with the generated APS traffic units is determined by the MI_APS_Pri input parameter; valid values are in the range 0-7.

The value of the ETH_CI_DE signal associated with the generated APS traffic units is always set to drop ineligible.

8.1.6 APS extract process

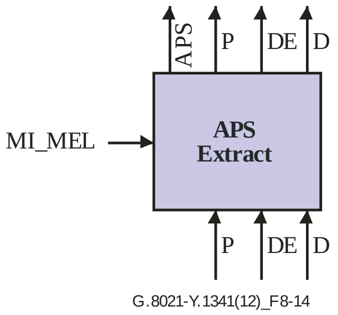


Figure 8-14 – APS extract process

The APS extract process extracts ETH_CI_APS signals from the incoming stream of ETH_CI traffic units. ETH_CI_APS signals are only extracted if they belong to the MEL as defined by the MI_MEL input parameter.

If an incoming traffic unit is an APS traffic unit belonging to the MEL defined by MI_MEL, the ETH_CI_APS signal will be extracted from this traffic unit and the traffic unit will be filtered. The ETH_CI_APS is the APS specific information contained in the received traffic unit. All other traffic units will be transparently forwarded. The encoding of the ETH_CI_D signal for APS frames is defined in clause 9.10 of [ITU-T G.8013].

The criteria for filtering are based on the values of the fields within the M_SDU field of the ETH_CI_D signal:

- length/type field equals the OAM EtherType (89-02)
- MEL field equals MI_MEL
- OAM type equals APS (39), as defined in clause 9.1 of [ITU-T G.8013].

This is defined in Figure 8-15. The function APS(D) extracts the APS specific information from the received traffic unit.

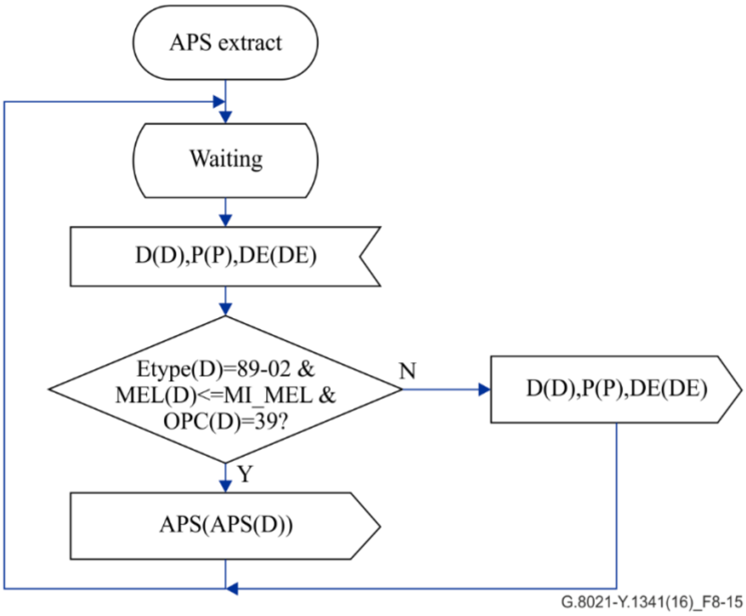


Figure 8-15 – APS extract behaviour

8.1.7 Continuity check (CC) processes

8.1.7.1 Overview

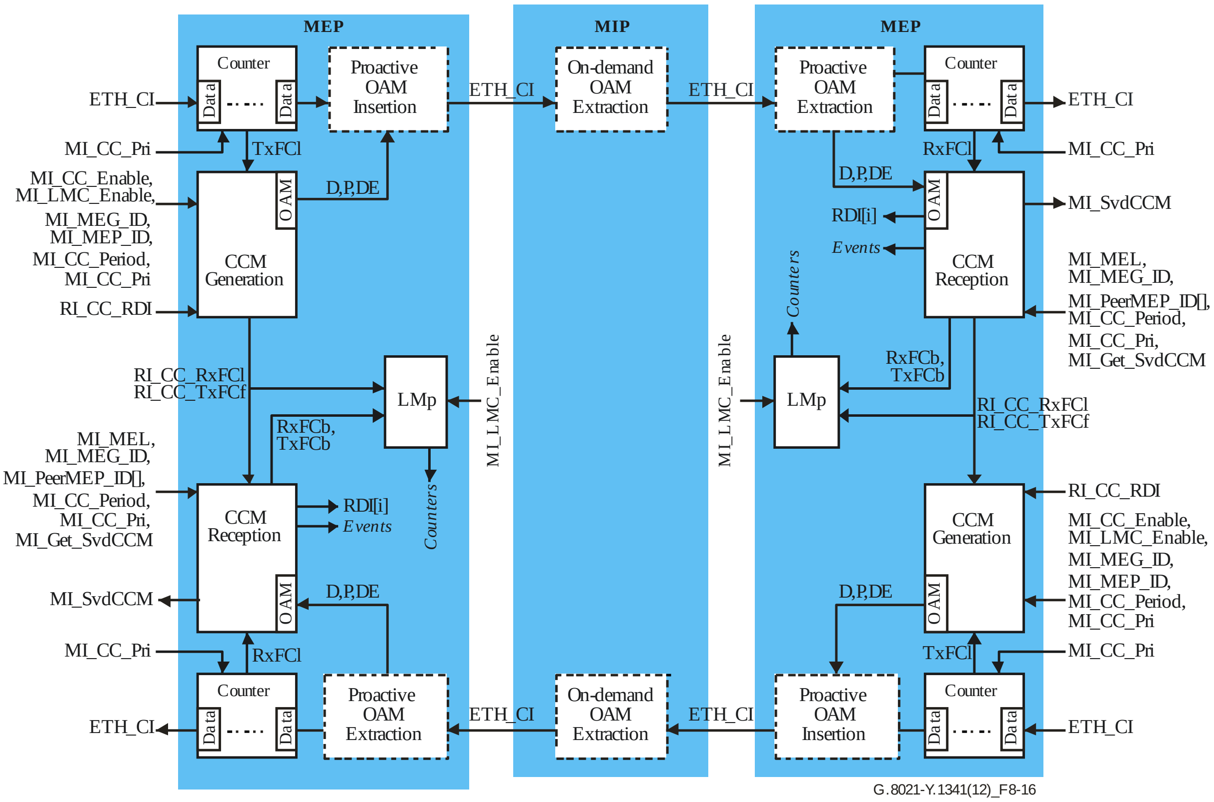


Figure 8-16 – Overview of processes involved with continuity check

Figure 8-16 gives an overview of the processes involved in the CC. The CCM generation process generates the CCM frames if MI_CC_Enable is true. The MI_MEG_ID and MI_MEP_ID are the MEG and MEP IDs of the MEP itself and these IDs are carried in the CCM frame. The CCM frames are generated with a periodicity determined by MI_CC_Period and with a priority determined by MI_CC_Pri. If MI_LMC_Enable is set the CCM frames will also carry loss measurement information. The generated CCM traffic units are inserted in the flow of ETH_CI by the OAM MEP source insertion process.

The CCM frames pass transparently through MIPs.

The OAM MEP sink extraction process extracts the CCM unit from the flow of ETH_CI and the CCM reception process processes the received CCM traffic unit. It compares the received MEG ID with the provisioned MI_MEG_ID, and the received MEP_ID with the provisioned MI_PeerMEP_ID[], that contains the list of all expected peer MEPs in the MEG. Based on the processing of this frame one or more events may be generated that serve as input for the defect detection process (not shown in Figure 8-16).

RDI information is carried in the CCM frame based upon the RI_CC_RDI input. It is extracted in the CCM reception process.

8.1.7.2 CCM generation process

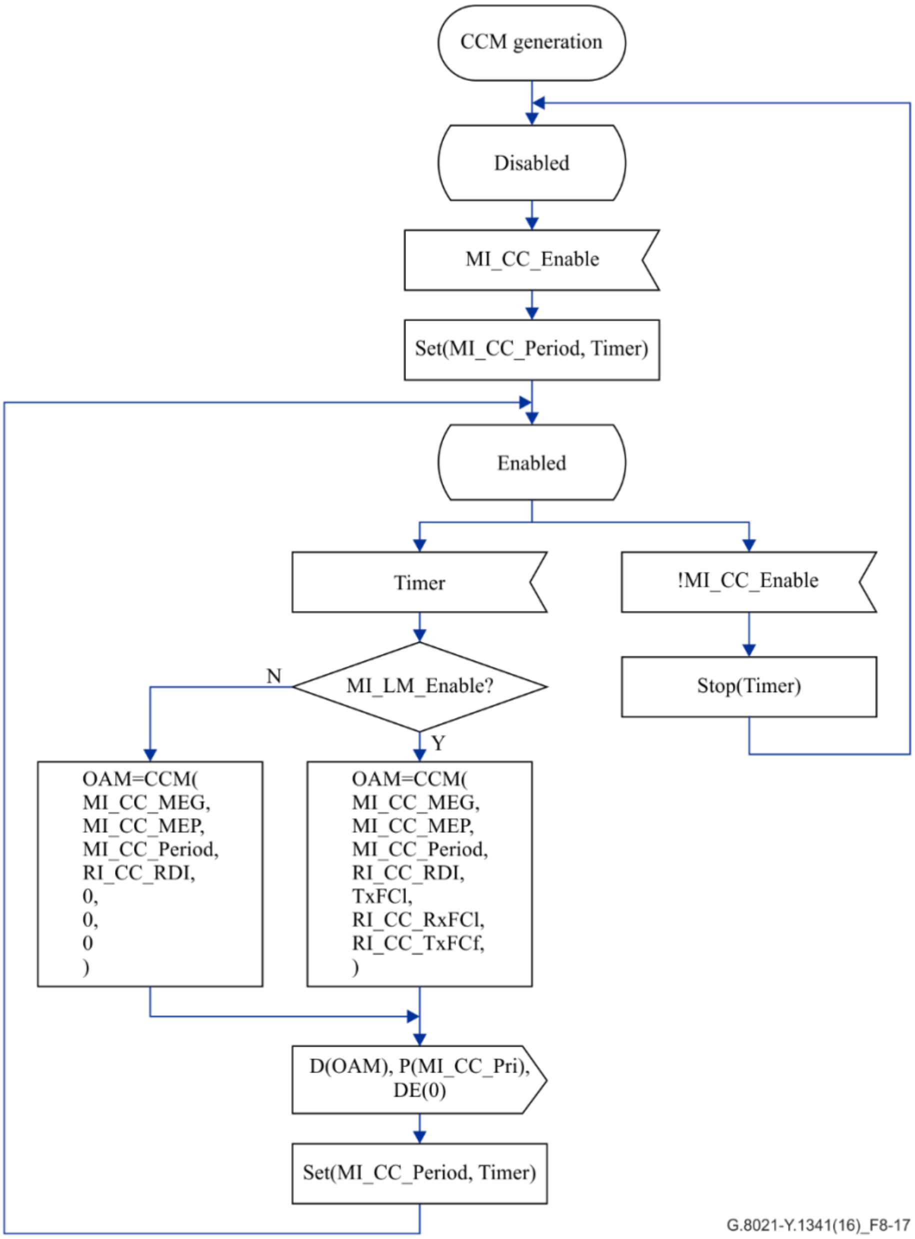


Figure 8-17 – CCM generation behaviour

Figure 8-17 shows the state diagram for the CCM generation process. The CCM generation process can be enabled and disabled using the MI_CC_Enable signal, where the default value is FALSE.

The CCM generation process generates and transmits an OAM frame every MI_CC_Period. The allowed values for MI_CC_Period are defined in Table 8-3.

Table 8-3 – CCM period values

3-bits	Period value	Comments
000	Invalid value	Invalid value for CCM PDUs
001	3.33ms	300 frames per second
010	10ms	100 frames per second
011	100ms	10 frames per second

100	1s	1 frame per second
101	10s	6 frames per minute
110	1 min	1 frame per minute
111	10 min	6 frame per hour

The ETH_CI_D signal contains a source and destination address field and an M_SDU field. The format of the M_SDU field is defined in clauses 9.1 and 9.2 of [ITU-T G.8013].

The value of the destination address field (DA) is the multicast class 1 DA as described in [ITU-T G.8013]. The value of the multicast class 1 DA is 01-80-C2-00-00-3x, where x is equal to MI_MEL as defined in clause 10.1 of [ITU-T G.8013]. This x will be filled in later by the OAM MEP insertion process and will be undefined in this process. The value of the source address will be filled in later by the OAM MEP insertion process and will be undefined in this process.

The M_SDU field contains a CCM PDU. Figure 8-18 below shows the M_SDU field where the CCM specific values are shown. It shows the traffic unit resulting from the function call in Figure 8-17:

```
OAM=CCM(
  MI_CC_MEG,
  MI_CC_MEP,
  MI_CC_Period,
  RI_CC_RDI,
  TxFCl,
  RI_CC_RxFCl,
  RI_CC_TxFCf
)
```

, or if !MI_LMC_Enable:

```
OAM=CCM(
  MI_CC_MEG,
  MI_CC_MEP,
  MI_CC_Period,
  RI_CC_RDI,
  0,
  0,
  0
)
```

The value of the ETH_CI_P signal associated with the generated CCM traffic unit is defined by the MI_CC_Pri input parameter; valid values are in the range 0-7.

The value of the ETH_CI_DE signal associated with the generated CCM traffic units is always set to drop ineligible (0).

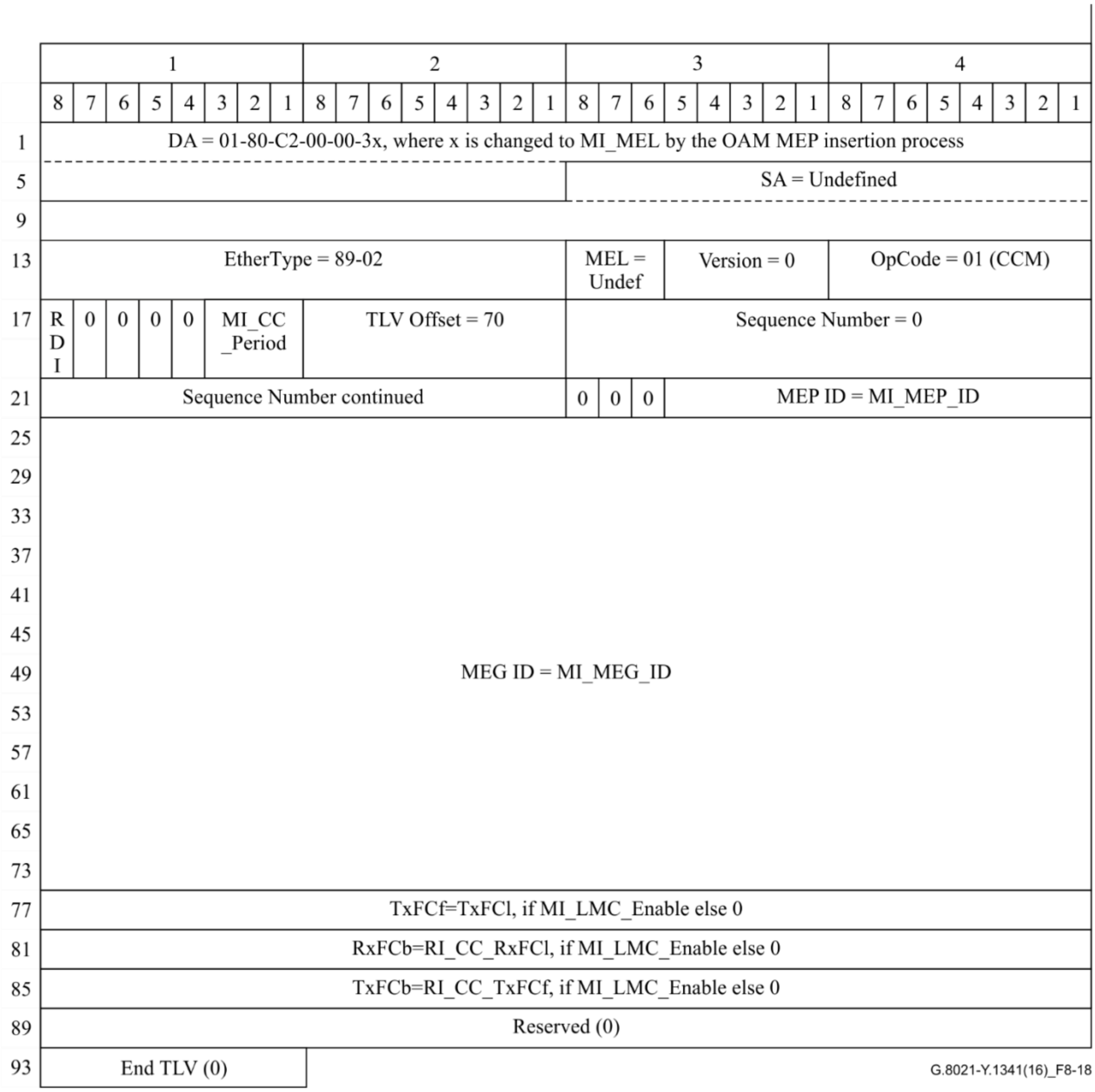


Figure 8-18 – CCM traffic unit

8.1.7.3 CCM reception process

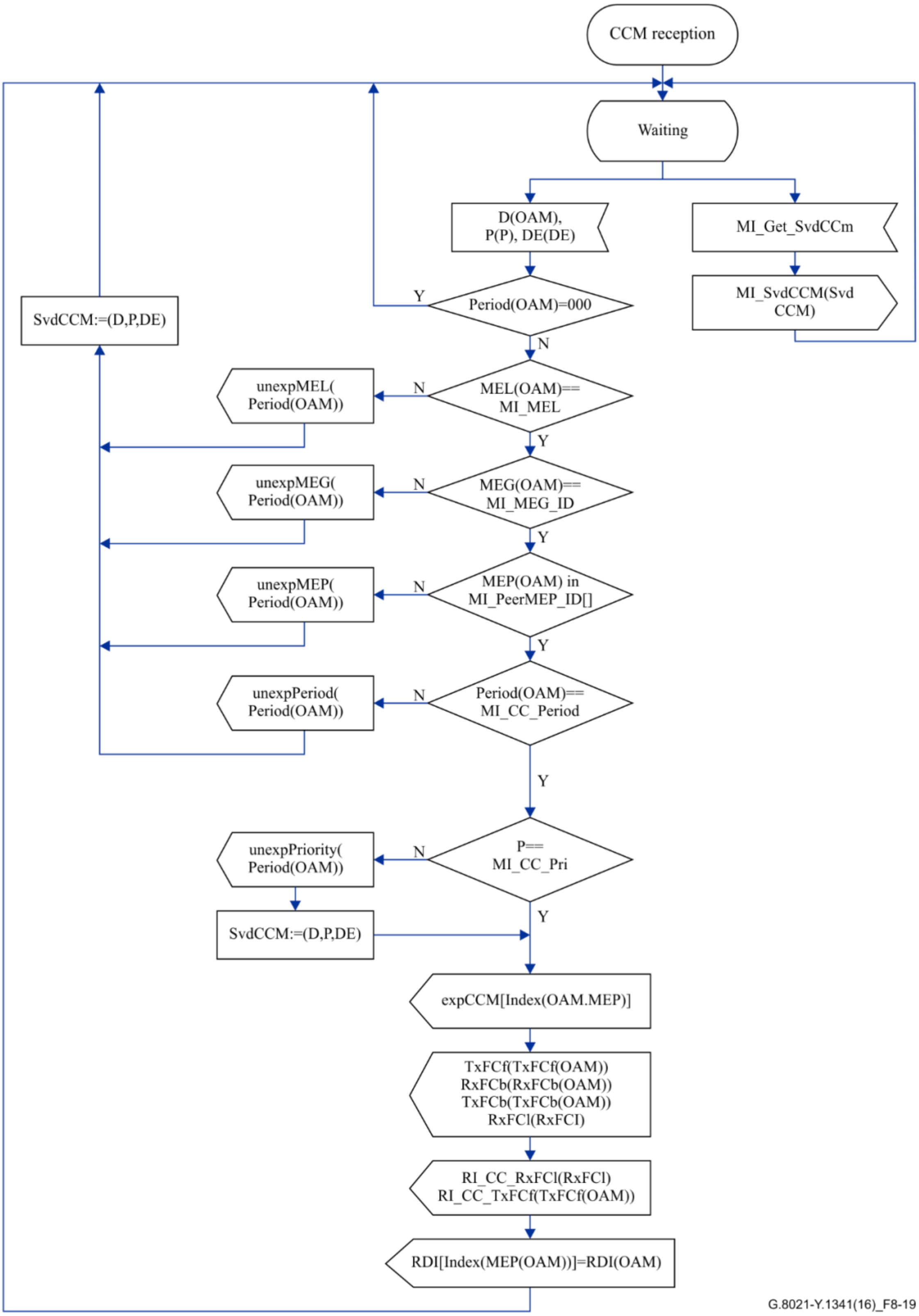


Figure 8-19 – CCM reception behaviour

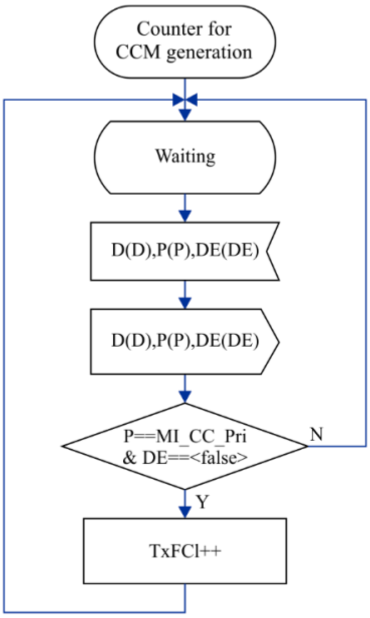
The CCM reception process processes CCM OAM frames. It checks the various fields of the frames and generates the corresponding events (as defined in clause 6). If the version, MEL, MEG, MEP and period are valid, the values of the frame counters are sent to the performance counter process.

Note that unexpPriority event does not prevent the CCM from being processed, since the MEL, MEG, MEP and period are as expected.

8.1.7.4 Counter process

This process counts the number of transmitted and received frames.

The counter process for CCM generation forwards data frames and counts all transmitted ETH_AI frames with priority (P) (i.e., ETH_AI_P) equal to MI_CC_Pri and Drop Eligibility (DE) (i.e., ETH_AI_DE) equal to <false (0)>. The D, P and DE signals are forwarded unchanged as indicated by the dotted lines in Figure 8-16.



G.8021-Y.1341(16)_F8-20

Figure 8-20 – Counter behaviour for CCM generation

The counter process for CCM reception receives ETH_CI and forwards them as ETH_AI traffic units. It counts this number of received ETH_AI traffic units that have priority (P) (i.e., ETH_AI_P) equal to MI_CC_Pri and drop eligibility (DE) (i.e., ETH_AI_DE) equal to <false (0)>.

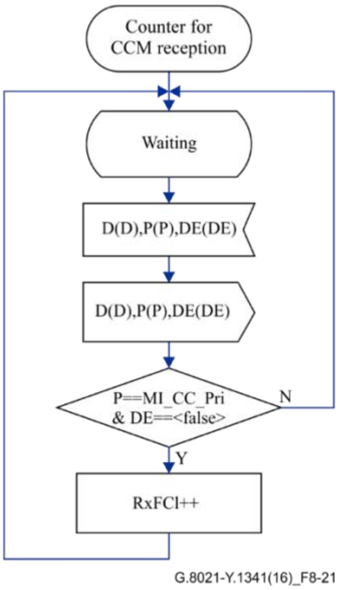


Figure 8-21 – Counter behaviour for CCM reception

8.1.7.5 Proactive loss measurement (LMp) process

This process calculates the number of transmitted and lost frames per second.

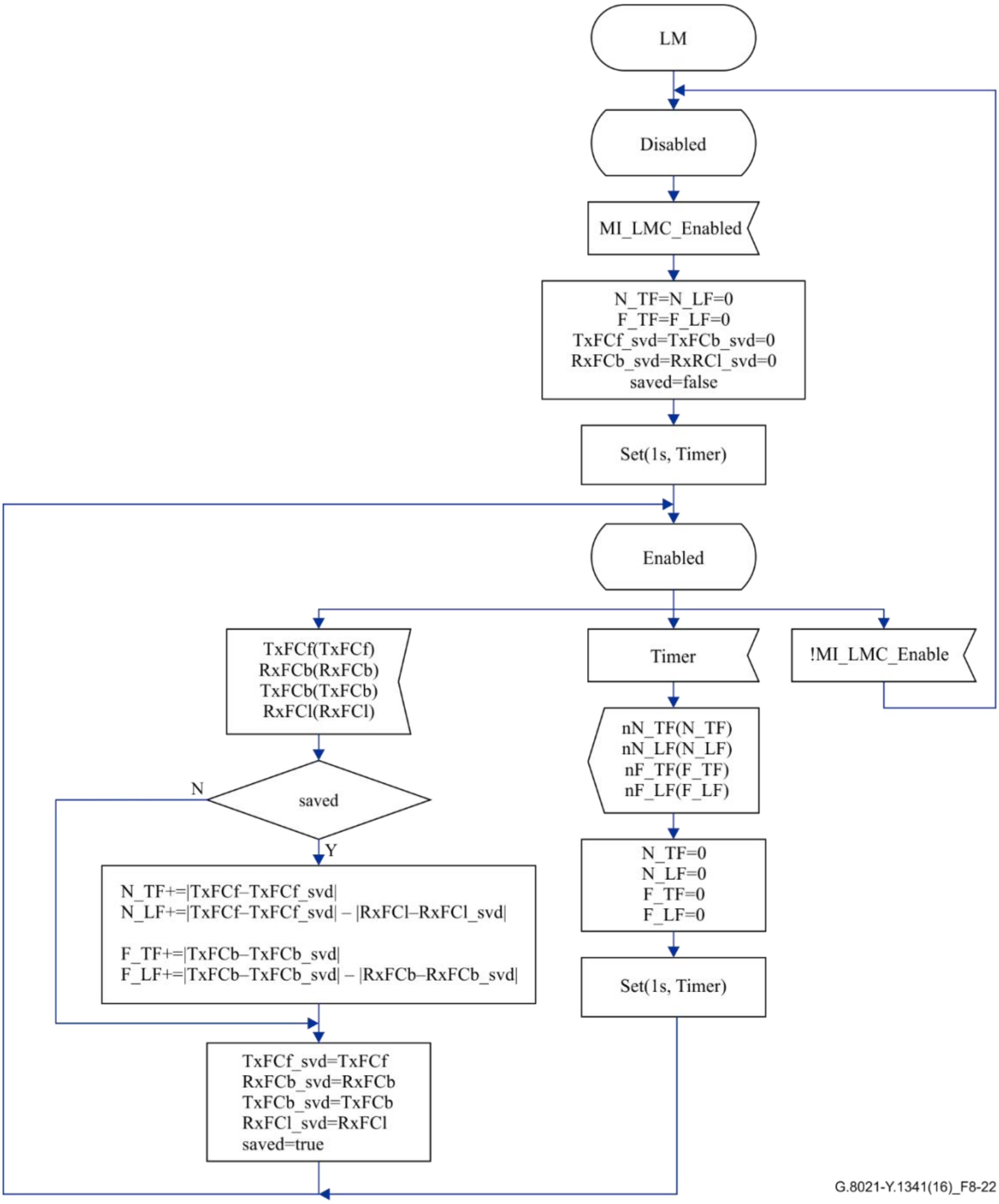


Figure 8-22 – LM process behaviour

It processes the TxFCf, RxFCb, TxFCb, RxFCl values and determines the number of transmitted frames and the number of lost frames. Every second, the number of transmitted and lost frames in that second are sent to the performance monitoring and defect generation processes.

8.1.8 Loopback (LB) processes

8.1.8.1 Overview

Figure 8-23 shows the different processes inside MEPs and MIPs that are involved in the loopback protocol.

The MEP on-demand OAM source insertion process is defined in clause 9.4.1.1, the MEP on-demand OAM sink extraction process in clause 9.4.1.2, the MIP on-demand OAM sink extraction process in clause 9.4.2.2, and the MIP on-demand OAM source insertion process in clause 9.4.2.1. In summary, they insert and extract ETH_CI OAM signals into and from the stream of ETH_CI_D traffic units and the complementing P and D signals going through an MEP and MIP; the extraction is based on MEL and OpCode. Furthermore, the insertion process inserts the correct MEL and SA values in the OAM traffic units. The other processes are defined in this clause.

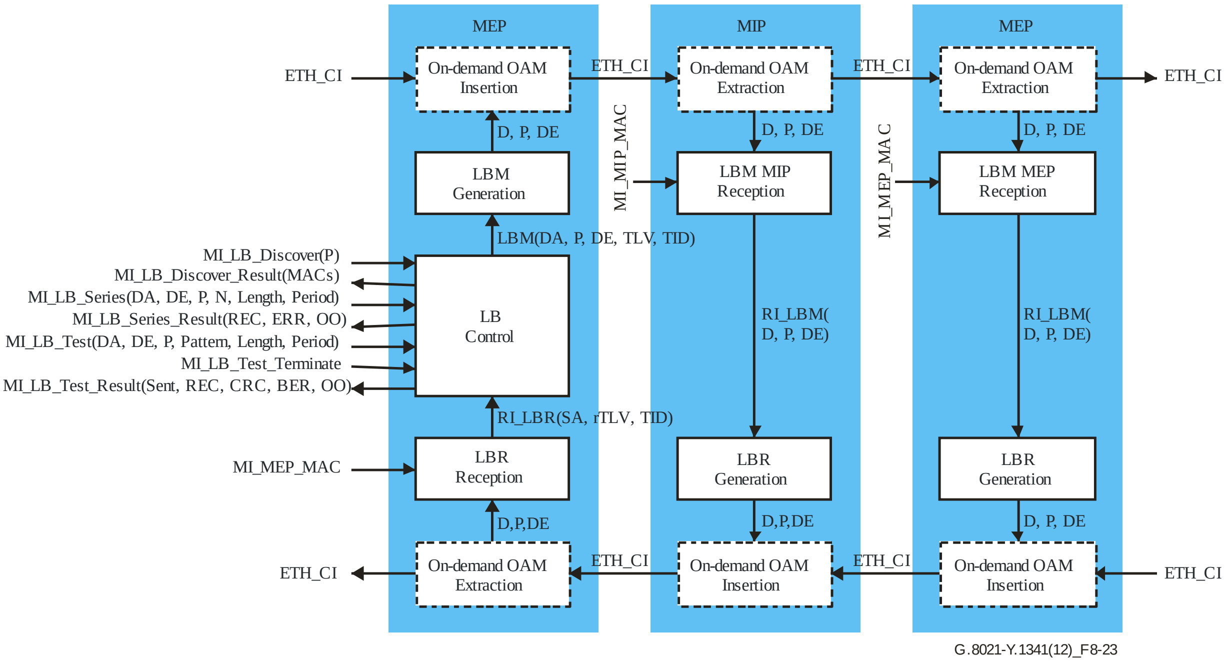


Figure 8-23 – Overview of processes involved with loopback

The LBM protocol is controlled by the LB control process. There are three possible MI signals that can trigger the LB protocol:

- MI_LB_Discover(P): To discover the MAC addresses of the other MEPs in the same MEG.
- MI_LB_Series(DA,DE,P,N,Length,Period): to send a series of N LB messages to a particular MEP/MIP; these LB messages are generated every "Period".
- MI_LB_Test(DA,DE,P,Pattern,Length,Period): to send a series of LB messages carrying a test pattern to a particular MEP; these LB messages are generated every "Period" until the MI_LB_Test_Terminate signal is received.

The details are described later in this clause.

The LBM control protocol triggers the LBM generation process to generate an LBM traffic unit that is received and forwarded by MIPs and received by MEPs in the same MEG. The LBM control process controls the number of LBM generated and the period between consecutive LBM traffic units.

The LBM MIP/MEP reception processes process the received LBM traffic units and as a result the LBR generation process may generate an LBR traffic unit in response. The LBR reception process receives and processes the LBR traffic units. The source address (SA), transaction ID (TID) and TLV values are given to the LBM control process.

The LBM control process processes these received values to determine the result of the requested LB operation. The result is communicated back using the following MI signals:

- MI_LB_Discover_Result(MACs): reports back the MACs that have responded with a valid LBR.
- MI_LB_Series_Result(REC,OO): reports back the total number of received LBR frames (REC), as well as counts of specific errors:
  - OO: number of LBR traffic units that were received out of order (OO).
- MI_LB_Test_Result(Sent, REC, CRC, BER, OO): reports back the total number of LBM frames sent (Sent) as well as the total number of LBR frames received (REC); for the latter counts of specific errors are reported:
  - CRC: number of LBR frames where the CRC in the pattern failed.
  - BER: number of LBR frames where there was a bit error in the pattern.
  - OO: number of LBR frames that were received out of order.

The detailed functionality of the various processes is defined below.

8.1.8.2 LB control process

The LB control process can receive several MI signals to trigger the LB protocol; this is shown in Figure 8-24.

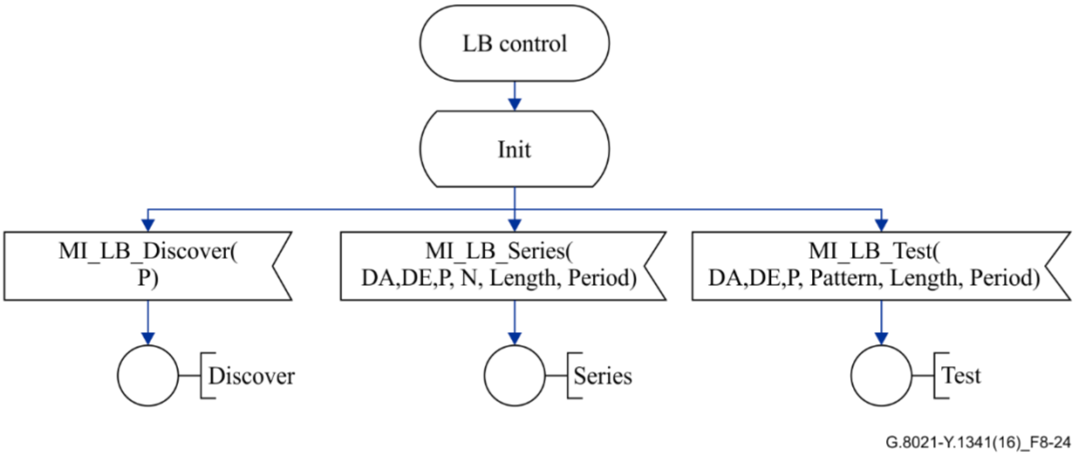


Figure 8-24 –LB control behaviour

Figure 8-25 shows the behaviour if the MI_LB_Discover signal is received.

Figure 8-26 shows the behaviour if the MI_LB_Series signal is received.

Figure 8-27 shows the behaviour if the MI_LB_Test signal is received.

NOTE – The state machine (Figure 8-24 combined with Figures 8-25, 8-26 and 8-27) shows that the LB_Discover, LB_Series and LB_Test actions are mutually exclusive. Furthermore, a new instantiation of any of these actions cannot be initiated until the current action is finished.

MI_LB_Discover behaviour

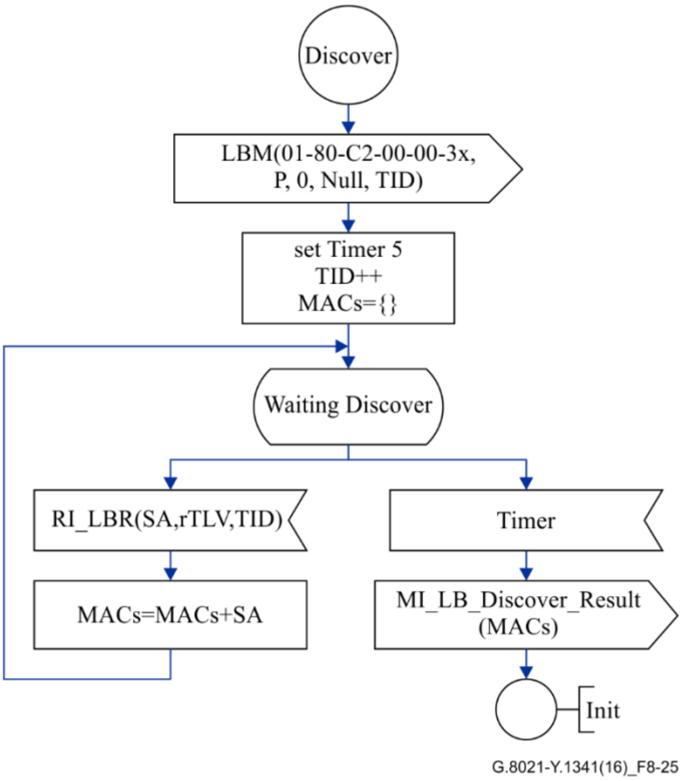


Figure 8-25 – LB control discover behaviour

Figure 8-25 shows the behaviour when an MI_LB_Discover(DE,P) signal is received.

First the LBM generation process is requested to generate an LBM frame by sending the LBM(01-80-c2-00-00-3x, P, 0, Null, TID) signal to the LBM generation process. The DA is set to the class 1 multicast address as defined in [ITU-T G.8013], where the last part (x) will be overwritten with MEL by the OAM MEP insertion process. There are no TLVs included, hence the TLV parameter is set to Null.

After triggering the transmission of the LBM frame, received RI_LBR is processed for 5 seconds (as governed by the timer). Every time the RI_LBR(SA,rTLV,TID) is received the SA is stored in the set of received MACs.

After 5 seconds all the received SAs are reported back using the MI_LB_Discover_Result(MACs) signal and the LBM control process returns to the Init state.

MI_LB_Series behaviour

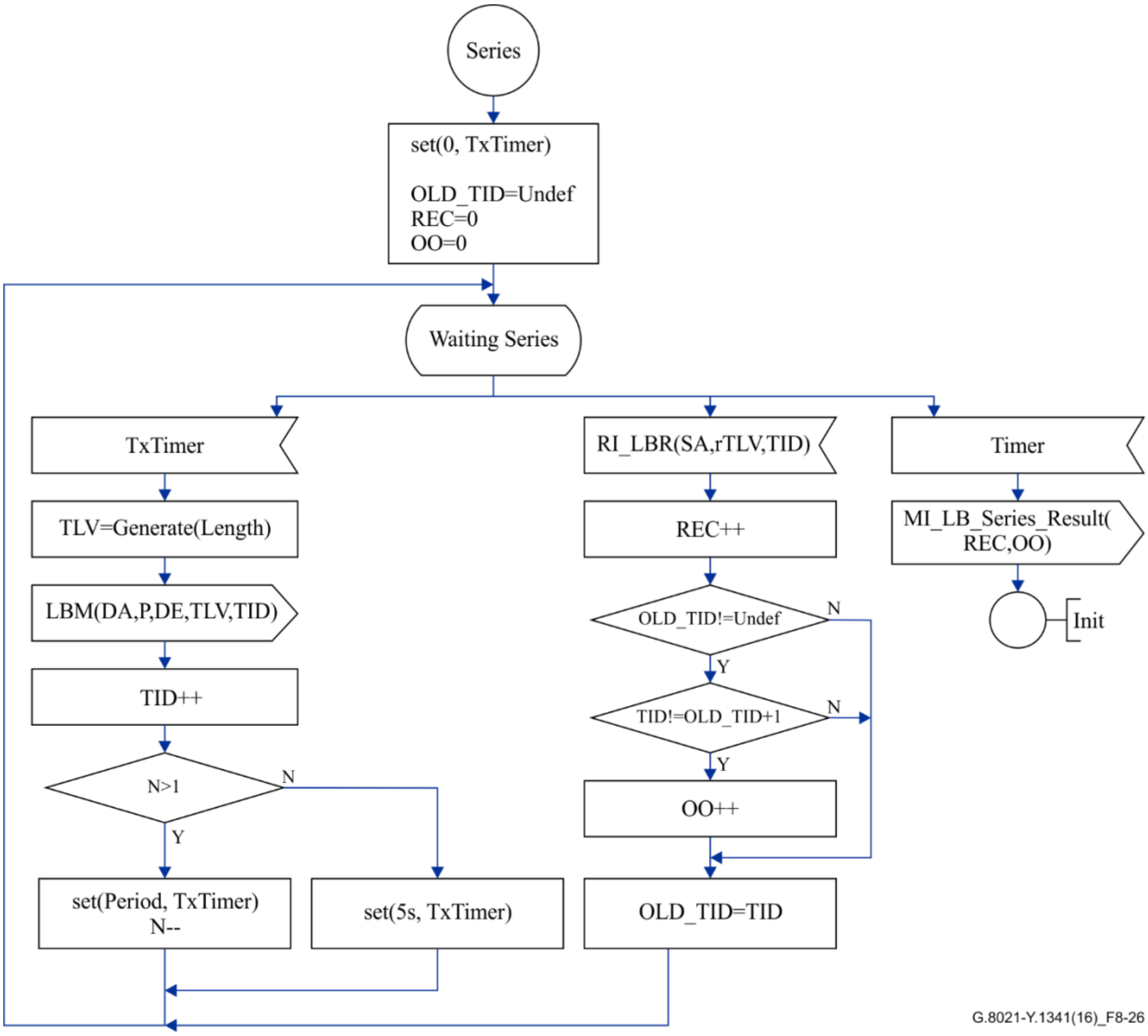


Figure 8-26 – LB control series behaviour

Figure 8-26 defines the behaviour of the LB control process after the reception of the MI_LB_Series(DA,DE,P,N,Length,Period) signal.

The TLV field of the LBM frames is determined by the Generate(Length) function. Generate(Length) generates a Data TLV with length "Length" of an arbitrary bit pattern to be included in the LBM frame.

After the receipt of the MI_LB_Series signal, the LBM generation process is requested N times to generate an LBM frame (where "Period" determines the interval between two LBM frames); this is done by issuing the LBM(DA,P,DE,TLV,TID) signal.

Whenever an RI_LBR(SA, rTLV, TID) signal is received, the number of received LBR frames is increased (REC++). If the TID value from the RI_LBR signal does not consecutively follow the last received TID value, the counter for out of order frames is incremented by one (OO++).

Five seconds after sending the last LBM frame (i.e., after sending the Nth LBM frame) the REC and OO counters are reported back in the MI_LB_Series_Result signal.

MI_LB_Test Behaviour

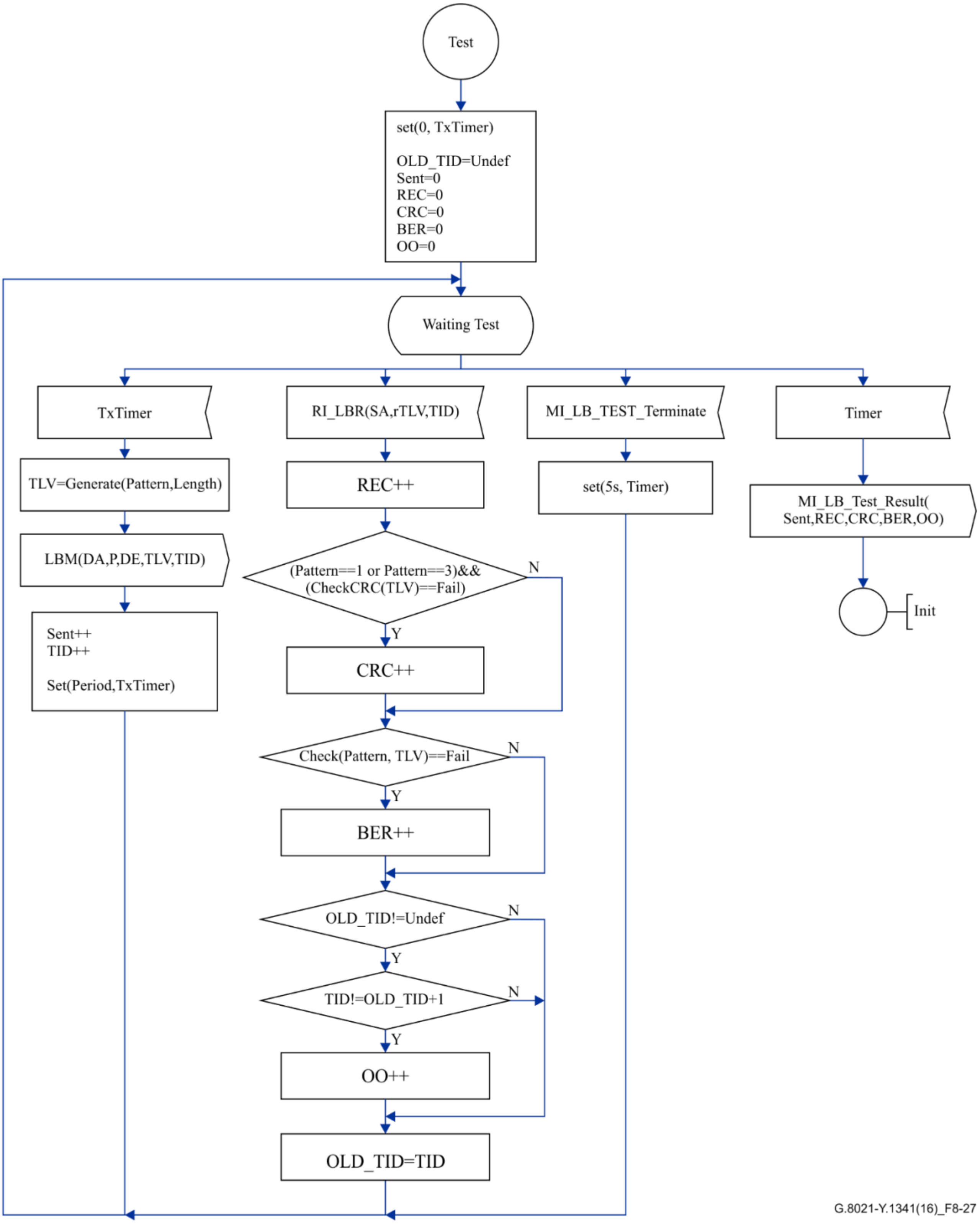


Figure 8-27 – LB control test behaviour

Figure 8-27 defines the behaviour of the LB control process after the reception of the MI_LB_Test(DA,DE,P,Pattern,Length,Period) signal.

Every period an LBM frame is generated until the MI_LB_Test_Terminate signal is received. Five seconds after receiving this MI_LB_Test_Terminate signal the "Sent", REC, CRC, BER and OO counters are reported back using the MI_LB_Test_Result signal.

The TLV field of the LBM frames is determined by the Generate(Pattern, Length) function. For pattern the following types are defined:

0: "Null signal without CRC-32"

1: "Null signal with CRC-32"

2: "PRBS 2^31-1 without CRC-32"

3: "PRBS 2^31-1 with CRC-32"

The length parameter determines the length of the generated TLV.

Generate(Pattern, Length) generates a test TLV with length "Length" to be included in the LBM frame. Therefore, this TLV is passed using the LBM(DA,P,DE,TLV,TID) signal to the LBM generation process.

Upon receipt of the RI_LBR(SA,rTLV,TID) remote information, the received LBR counter is incremented by one (REC++). If the TLV contains a CRC (Pattern 1 or 3) the CRC counter is incremented by one if the CRC check fails. The function Check(Pattern, TLV) compares the received test pattern with the expected test pattern. If there is a mismatch, the BER counter is increased. If the TID value from the RI_LBR signal does not follow the last received TID value, the counter for out of order frames is incremented by one (OO++).

8.1.8.3 LBM generation process

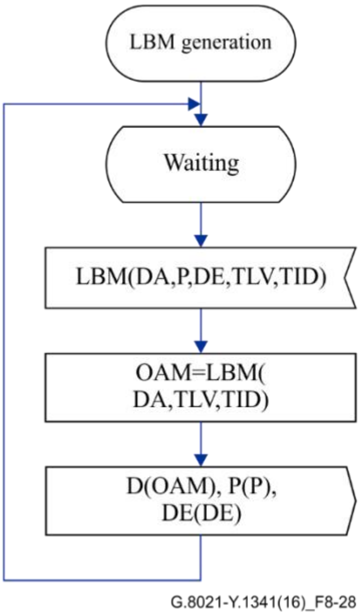


Figure 8-28 – LBM generation behaviour

The LBM generation process generates a single LBM OAM traffic unit (ETH_CI_D) complemented with ETH_CI_P and ETH_CI_DE signals on receipt of the LBM(DA,P,DE,TLV,TID) signal. The process is defined in Figure 8-28.

From the LBM(DA,P,DE,TLV,TID) signal the P field determines the value of the ETH_CI_P signal, the DE field determines the value of the ETH_CI_DE signal. The DA, TLV and TID fields are used in the construction of the ETH_CI_D signal that carries the LBM traffic unit.

The format of the LBM traffic unit and the values are shown in Figure 8-29.

The values of the SA and MEL fields will be determined by the OAM MEP insertion process, as well as the last part (x) of the DA if the DA is set to 01-80-c2-00-00-3x.

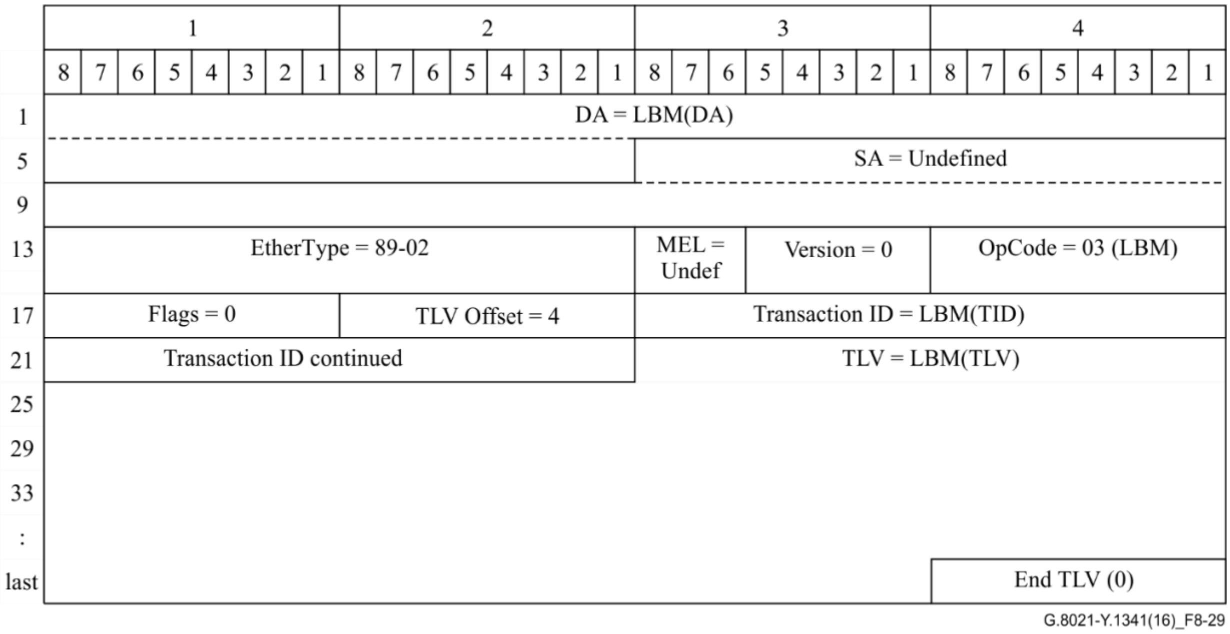


Figure 8-29 – LBM traffic unit

8.1.8.4 MIP LBM reception process

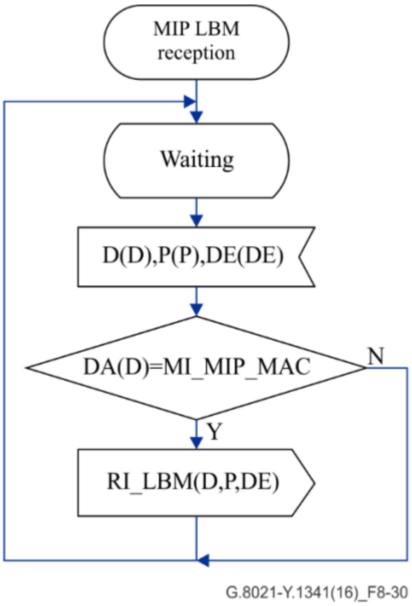


Figure 8-30 – MIP LBM reception behaviour

The MIP LBM reception process receives ETH_CI traffic units containing LBM PDUs complemented by the P and D signals.

The behaviour is defined in Figure 8-30. If the DA field in the traffic unit (D signal) equals the local MAC address (MI_MIP_MAC), the loopback is intended for this MIP and the information is forwarded to the loopback reply generation process using the RI_LBM(D,P,DE) signal; otherwise the information is ignored and no action is taken.

Note that an MIP therefore does not reply to LBM traffic units that have a class 1 multicast address.

8.1.8.5 MEP LBM reception process

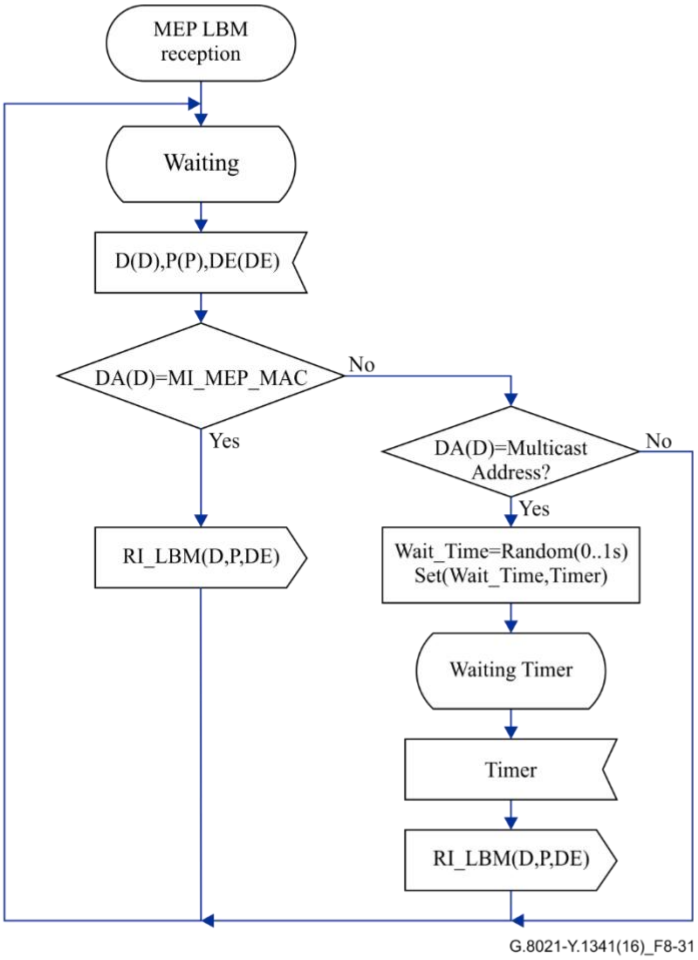


Figure 8-31 – MEP LBM reception behaviour

The MEP LBM reception process receives ETH_CI traffic units containing LBM PDUs complemented by the P and D signals.

The behaviour is defined in Figure 8-31.

If the DA field in the LBM traffic unit (D signal) equals the local MAC address (MI_MEP_MAC), the loopback is intended for this MEP, and the information is forwarded to the loopback reply generation process (RI_LBM(D,P,DE)).

If the DA field in the LBM traffic unit (D signal) is a multicast address, an LBR traffic unit must be generated after a random delay between 0 and 1 second. This is specified by instantiating a separate process, the Send_MC_LBR process. This process chooses a random waiting time between 0 and 1 second and, after waiting for the chosen period of time, the D, P and DE information is forwarded to the loopback reply generation process (RI_LBM(D,P,DE)). Finally, this process instance is terminated.

Since the 0 to 1 second waiting time is performed in a separate process, it does not block the reception and processing of other LBM frames within that waiting period.

8.1.8.6 LBR generation process

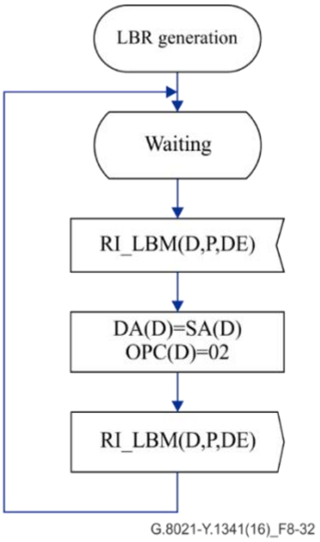


Figure 8-32 – LBR generation behaviour

Note that the LBR generation process is the same for MEPs and MIPs.

Upon receipt of the LBM traffic unit and accompanying signals (RI_LBM(D,P,DE)) from the LBM reception process the LBR generation process generates an LBR traffic unit together with the complementing P and DE signals.

The behaviour is specified in Figure 8-32. The generated traffic unit is the same as the received RI_LBM(D) traffic unit except:

- the DA of the generated LBR traffic unit is the SA of the received LBM traffic unit, and
- the OpCode is set to LBR OpCode.

NOTE – In the generated LBR, in the OAM (MEP) insertion process, the SA will be overwritten with the local MAC address, and the MEL will be overwritten with MI_MEL.

The resulting LBR traffic unit format is shown in Figure 8-33.

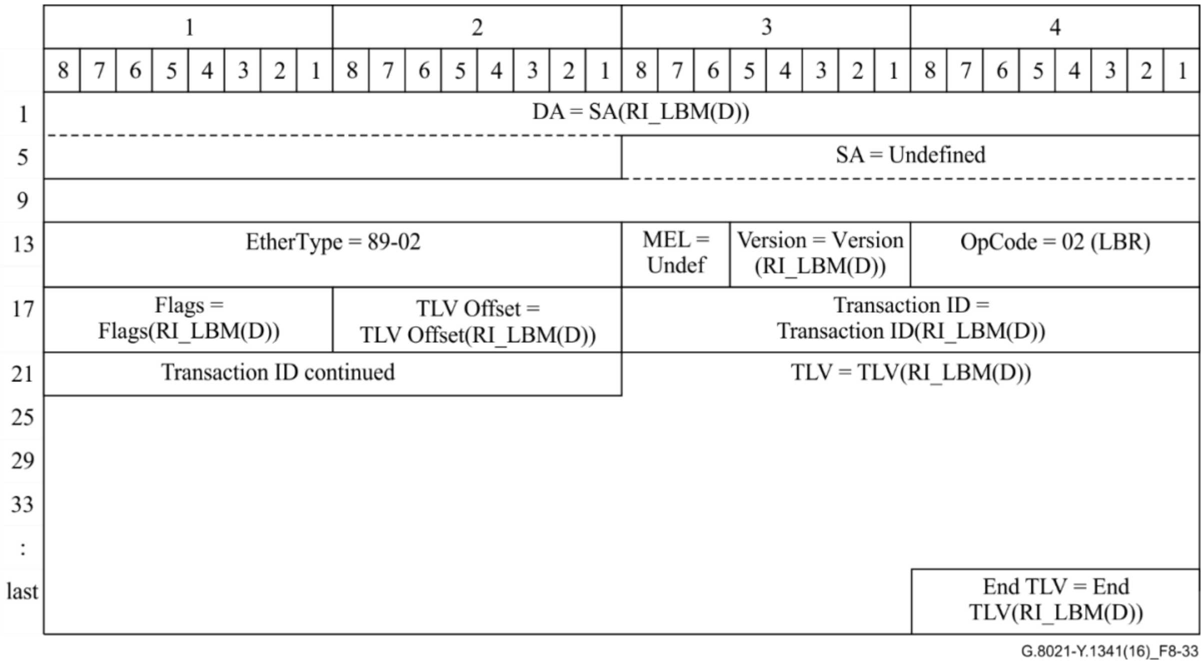


Figure 8-33 – LBR traffic unit

8.1.8.7 LBR reception process

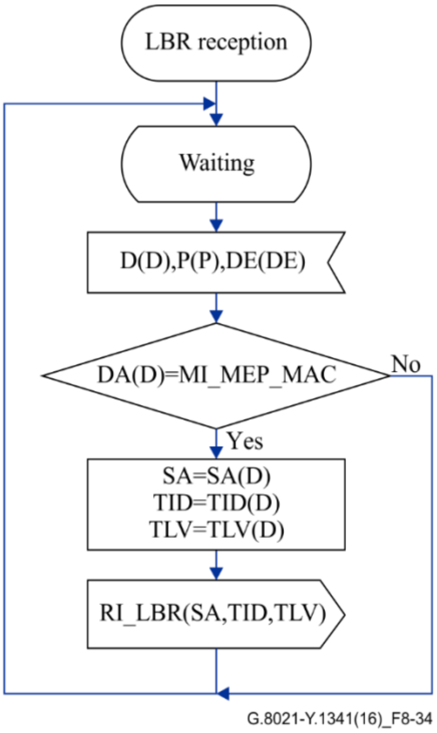


Figure 8-34 – LBR reception behaviour

The LBR reception process receives LBR traffic units (D signal) together with the complementing P and DE signals. The LBR reception process will inspect the DA field in the received traffic unit; if the DA equals the local MAC address (MI_MEP_MAC) the SA, TID and TLV values will be extracted from the LBR PDU and signalled to the LB control process using the RI_LBR(SA,TID,TLV) signal. The behaviour is defined in Figure 8-34.

8.1.9 Loss measurement (LM) processes

8.1.9.1 Overview

Figure 8-35 shows the different processes inside MEPs and MIPs that are involved in the on-demand loss measurement protocol.

The MEP on-demand OAM source insertion process is defined in clause 9.4.1.1, the MEP on-demand OAM sink extraction process in clause 9.4.1.2, the MIP on-demand OAM sink extraction process in clause 9.4.2.2, and the MIP on-demand OAM source insertion process in clause 9.4.2.1. In summary, they insert and extract ETH_CI OAM signals into and from the stream of ETH_CI_D traffic units together with the complementing P and D signals going through an MEP and MIP; the extraction is based on MEL and OpCode. Furthermore, the insertion process inserts the correct MEL and SA values into the OAM traffic units.

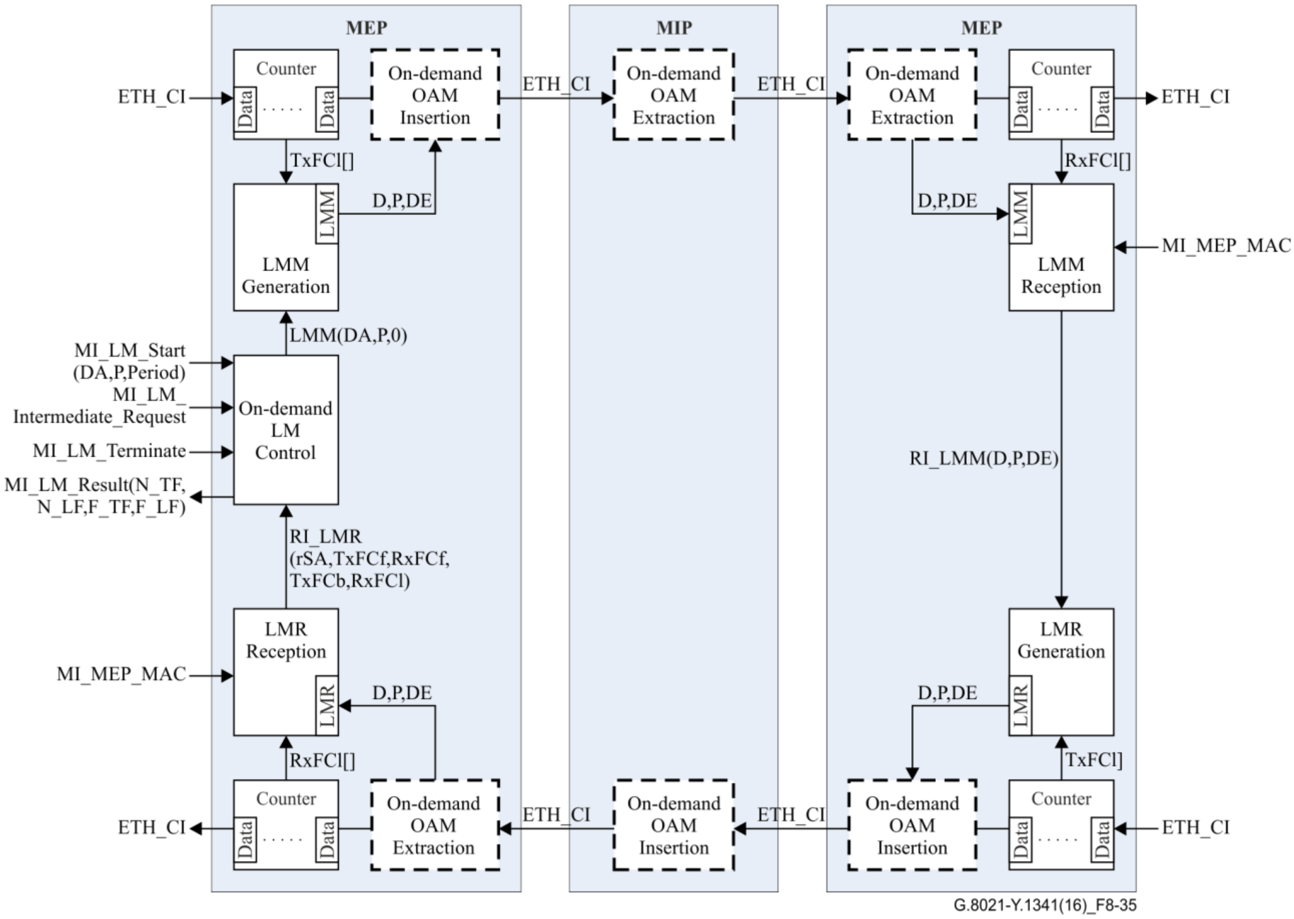


Figure 8-35 – Overview of processes involved with on-demand loss measurement

The on-demand LM control process controls the on-demand LM protocol. The protocol is activated upon receipt of the MI_LM_Start(DA,P,Period) signal and remains activated until the MI_LM_Terminate signal is received.

The result is communicated via the MI_LM_Result(N_TF, N_LF, F_TF, F_LF) signal when the process is terminated by the MI_LM_Terminate signal or when an intermediate result is requested via the MI_LM_Intermediate_Request signal. If the on-demand LM control process activates the multiple monitoring on different CoS levels simultaneously, each result is independently managed per CoS level.

The LMM generation process generates an LMM traffic unit that passes transparently through MIPs, but that will be processed by the LMM reception process in MEPs. The LMR generation process generates an LMR traffic unit in response to the receipt of an LMM traffic unit. The LMR reception process receives and processes the LMR traffic units.

Figure 8-36 shows the different processes inside MEPs and MIPs that are involved in the proactive loss measurement protocol.

The MEP proactive OAM insertion process is defined in clause 9.2.1.1, the MEP OAM proactive extraction process in clause 9.2.1.2, the MIP OAM extraction process in clause 9.4.2.1, and the MIP OAM insertion process in clause 9.4.2.2. In summary, they insert and extract ETH_CI OAM signals into and from the stream of ETH_CI_D traffic units and the complementing P and D signals going through an MEP and MIP; the extraction is based on MEL and OpCode. Furthermore, the insertion process inserts the correct MEL and SA values into the OAM traffic units.

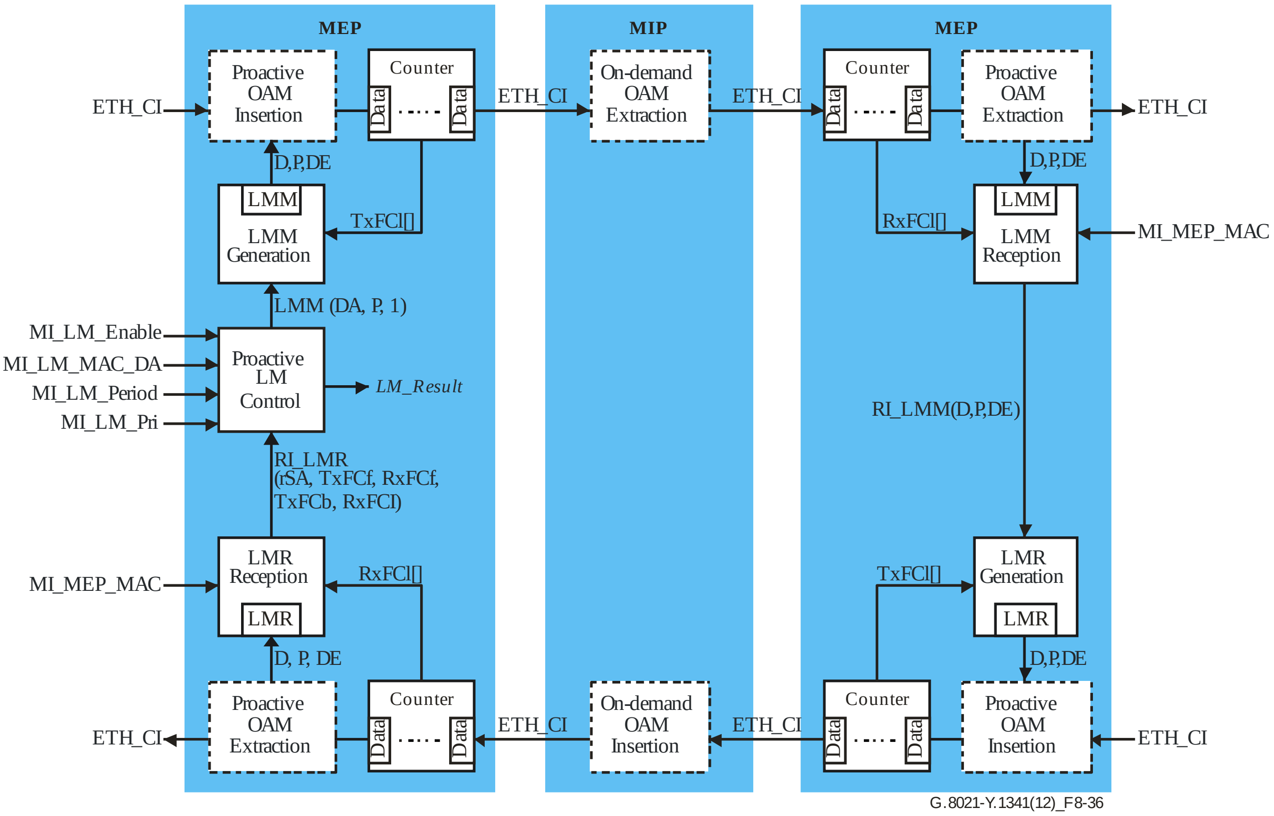


Figure 8-36 – Overview of processes involved with proactive loss measurement

The proactive LM control process controls the proactive LM protocol. If MI_LML_Enable is set the LMM frames are sent periodically. The LMM frames are generated with a periodicity determined by MI_LM_Period and with a priority determined by MI_LM_Pri. The result (N_TF, N_LF, F_TF, F_LF) is reported via an LMR reception. If the proactive LM control process activates the multiple monitoring on different CoS levels simultaneously, each result is independently managed per CoS level.

The behaviour of the processes is defined below.

8.1.9.2 LM control process

The behaviour of the on-demand LM control process is defined in Figure 8-37.

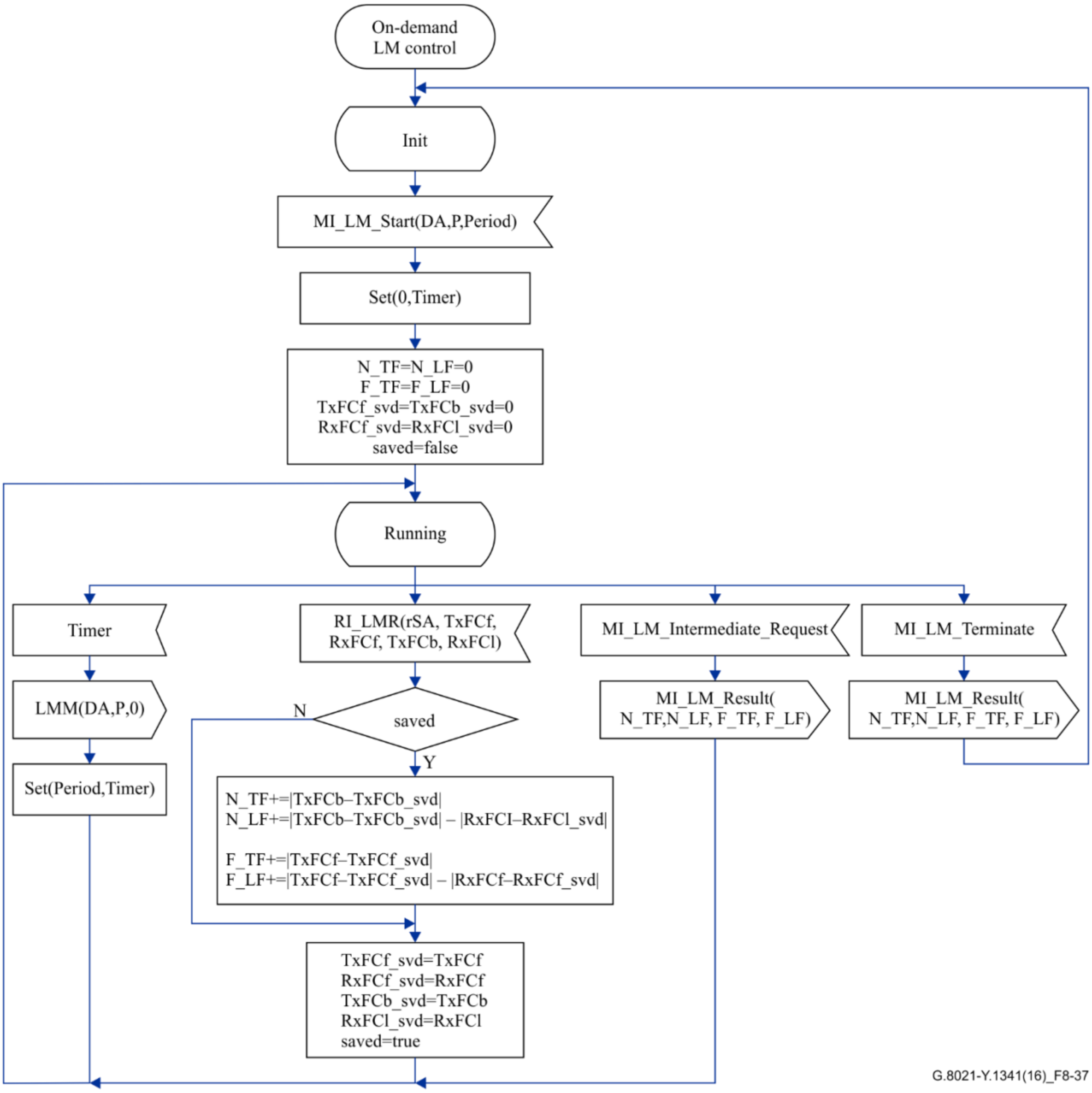


Figure 8-37 – On-demand LM control behaviour

Upon receipt of the MI_LM_Start(DA,P,Period), the LM protocol is started. Every period the generation of an LMM frame is triggered (using the LMM(DA,P,0) signal) until the MI_LM_Terminate signal is received.

The received counters are used to count the near-end and far- end transmitted and lost frames. This result is reported using the MI_LM_Result(N_TF, N_LF, F_TF, F_LF) signal after the receipt of the MI_LM_Terminate signal or of the MI_LM_Intermediate_Request signal.

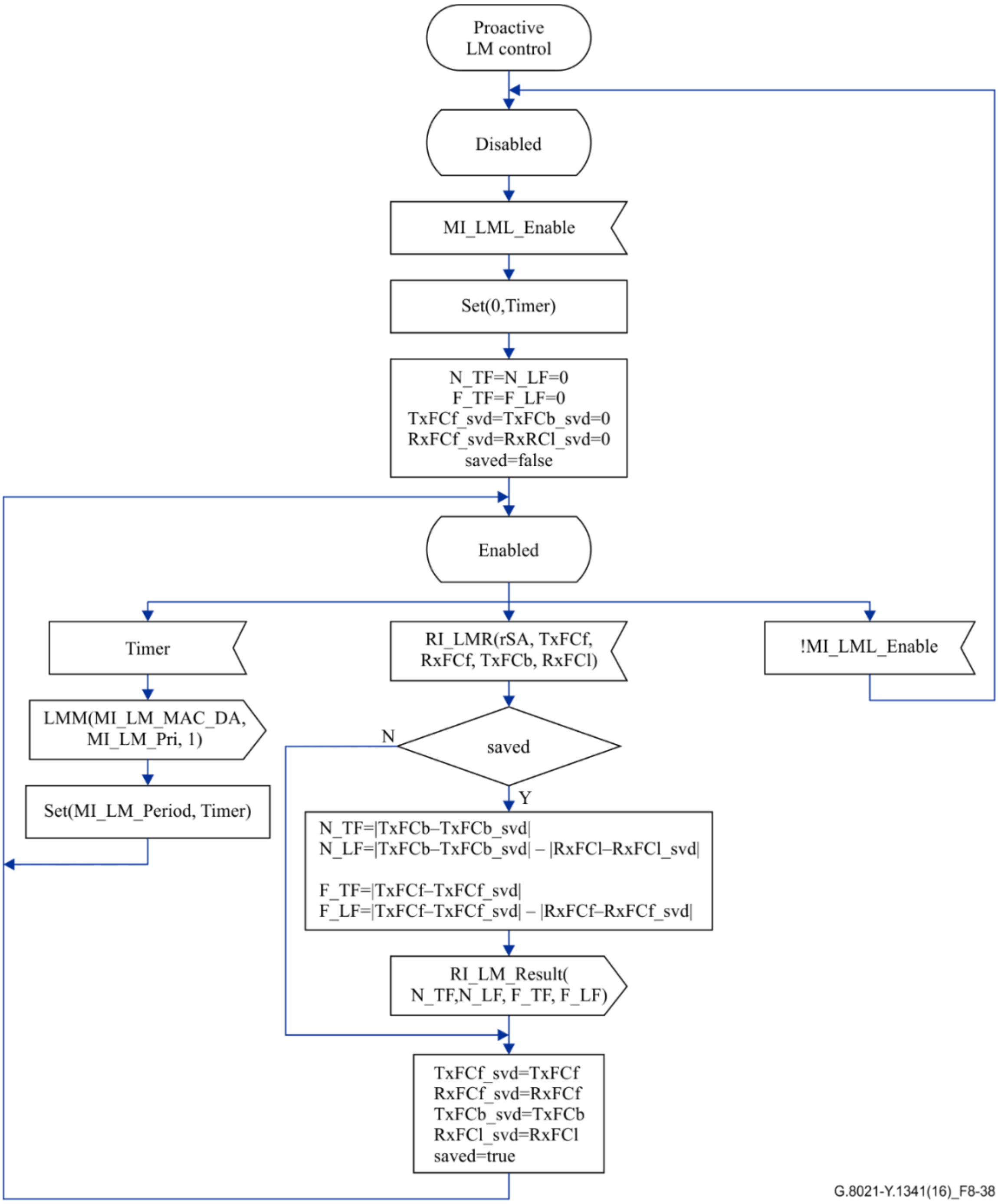


Figure 8-38 – Proactive LM control behaviour

The behaviour of the proactive LM control process is defined in Figure 8-38. If the MI_LML_Enable is asserted, the process starts to generate LMM frames (using the LMM(MI_LM_MAC_DA, MI_LM_Pri, 1) signal). The result (N_TF, N_LF, F_TF, F_LF) is reported via an LMR reception.

8.1.9.3 LMM generation process

This process generates an LMM traffic unit on receipt of the LMM(DA,P,Type) signal.

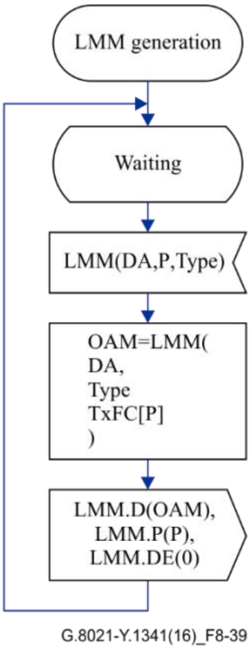


Figure 8-39 – LMM generation behaviour

The LMM traffic unit contains a source and destination address field and an M_SDU field. The format of the M_SDU field for LMM traffic units is defined in clauses 9.1 and 9.12 of [ITU-T G.8013].

The LMM traffic unit is generated by the LMM generate function in Figure 8-39. Figure 8-40 shows the resultant LMM traffic unit. The type signal is set to 1 if it is the proactive OAM, or set to 0 if it is the on-demand OAM operation.

<table>
<tr><td></td><td colspan="8">1</td><td colspan="8">2</td><td colspan="8">3</td><td colspan="8">4</td></tr>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td colspan="32">DA=LMM(DA)</td></tr>
<tr><td>5</td><td colspan="16"></td><td colspan="16">SA=Undefined</td></tr>
<tr><td>9</td><td colspan="32"></td></tr>
<tr><td>13</td><td colspan="16">EtherType=89-02</td><td colspan="3">MEL=
Undef</td><td colspan="5">Version=1</td><td colspan="8">OpCode=43 (LMM)</td></tr>
<tr><td>17</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Type</td><td colspan="8">TLV Offset =12</td><td colspan="16">TxFCf=LMM(Tx)</td></tr>
<tr><td>21</td><td colspan="16">TxFCf Continued</td><td colspan="16">Reserved for RxFCf in LMR=0</td></tr>
<tr><td>25</td><td colspan="16">Reserved Continued</td><td colspan="16">Reserved for TxFCb in LMR=0</td></tr>
<tr><td>29</td><td colspan="16">Reserved Continued</td><td colspan="8">End TLV=0</td><td colspan="8"></td></tr>
</table>

Figure 8-40 – LMM traffic unit

8.1.9.4 LMM reception process

This process processes received LMM traffic units. It checks the destination address, the DA must be either the local MAC address or it should be a multicast class 1 destination address. If this is the case the LMM reception process writes the Rx Counter value to the received traffic unit in the RxFCf field, and forwards the received traffic unit and complementing P and DE signals as remote information to the LMR generation process.

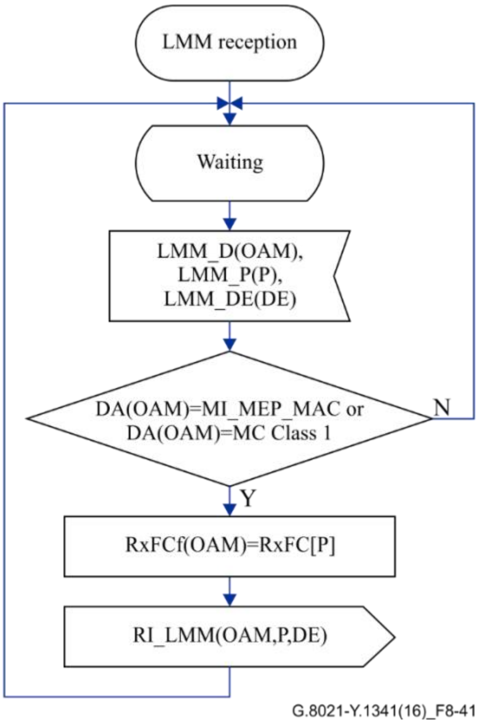


Figure 8-41 – LMM reception behaviour

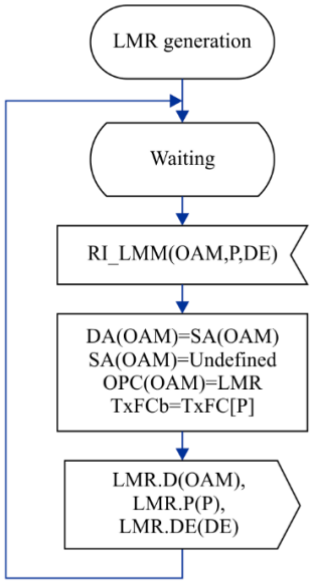
8.1.9.5 LMR generation process

The LMR generation process generates an LMR traffic unit on receipt of RI_LMM signals. The LMR traffic unit is based on the received LMM traffic unit (as conveyed in the RI_LMM_D signal), however:

- the SA of the LMM traffic unit becomes the DA of the LMR traffic unit
- the OpCode is set to LMR
- the TxFCb field is assigned the value of the Tx counter.

NOTE – In the generated LMR, in the OAM (MEP) insertion process, the SA will be overwritten with the local MAC address, and the MEL will be overwritten with MI_MEL.

Note that the RxFCf field is already assigned a value by the LMM reception process.



G.8021-Y.1341(16)_F8-42

Figure 8-42 – LMR generation behaviour

Figure 8-43 shows the resultant LMR traffic unit.

<table>
<tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th></tr>
<tr><td></td><td>8 7 6 5 4 3 2 1</td><td>8 7 6 5 4 3 2 1</td><td>8 7 6 5 4 3 2 1</td><td>8 7 6 5 4 3 2 1</td></tr>
<tr><td>1</td><td colspan="4">DA = SA(RI_LMM(D))</td></tr>
<tr><td>5</td><td colspan="2"></td><td colspan="2">SA = Undefined</td></tr>
<tr><td>9</td><td colspan="4"></td></tr>
<tr><td>13</td><td colspan="2">EtherType = 89-02</td><td>MEL = Undef
Version = version (RI_LMM(D))</td><td>OpCode = 42 (LMR)</td></tr>
<tr><td>17</td><td>Flags = Flags(RI_LMM(D))</td><td>TLV Offset = TLV Offset(RI_LMM(D))</td><td colspan="2">TxFCf = TxFCf(RI_LMM(D))</td></tr>
<tr><td>21</td><td colspan="2">TxFCf continued</td><td colspan="2">RxFCf = RxFCf(RI_LMM(D))</td></tr>
<tr><td>25</td><td colspan="2">RxFCf continued</td><td colspan="2">TxFCb = Tx counter</td></tr>
<tr><td>29</td><td colspan="2">TxFCb continued</td><td>End TLV = End TLV(RI_LMM(D))</td><td></td></tr>
</table>

G.8021-Y.1341(16)_F8-43

Figure 8-43 – LMR traffic unit

8.1.9.6 LMR reception process

This process processes received LMR traffic units. If the DA equals the local MAC address, it extracts the counter values TxFCf, RxFCf, TxFCb from the received traffic unit as well as the SA field. These values together with the value of the Rx counter(RxFCl) are forwarded as RI signals.

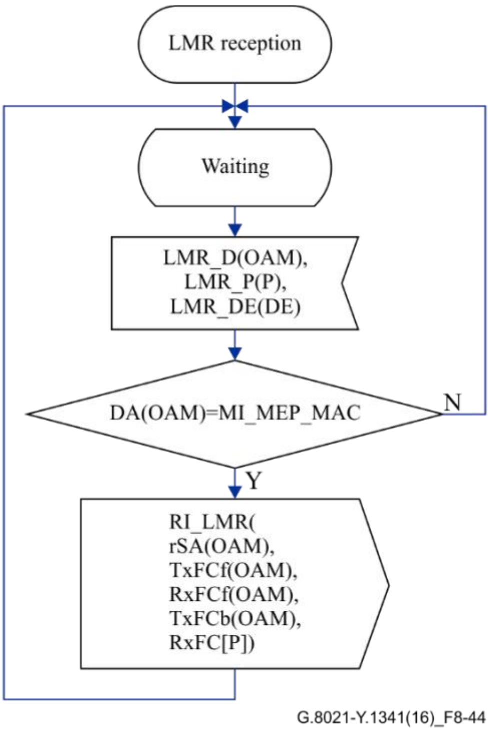


Figure 8-44 – LMR reception behaviour

8.1.9.7 Counter process

This process counts the number of transmitted and received frames.

The counter process for LMM/LMR generation receives ETH_AI and forwards it. It counts the number of ETH_AI traffic units received with ETH_AI_DE to <false (0)>.

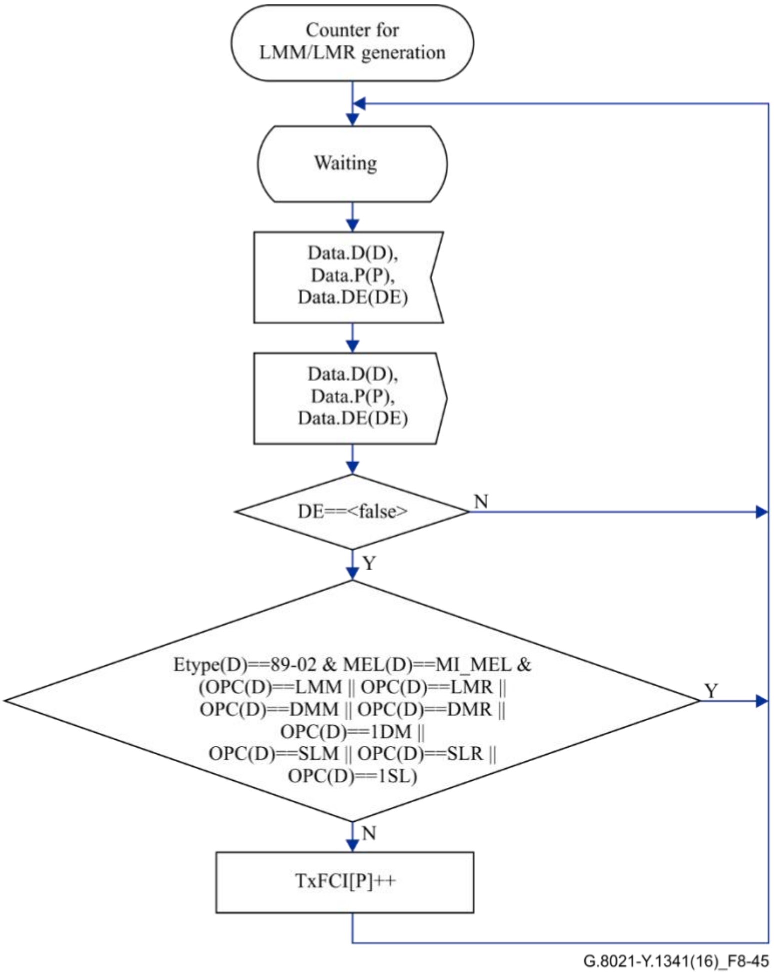


Figure 8-45 – Counter behaviour for LMM/LMR generation

The counter process for LMM/LMR reception receives ETH_CI and forwards them as ETH_AI traffic units. It counts this number of ETH_AI instances with ETH_AI_DE equal to <false (0)>.

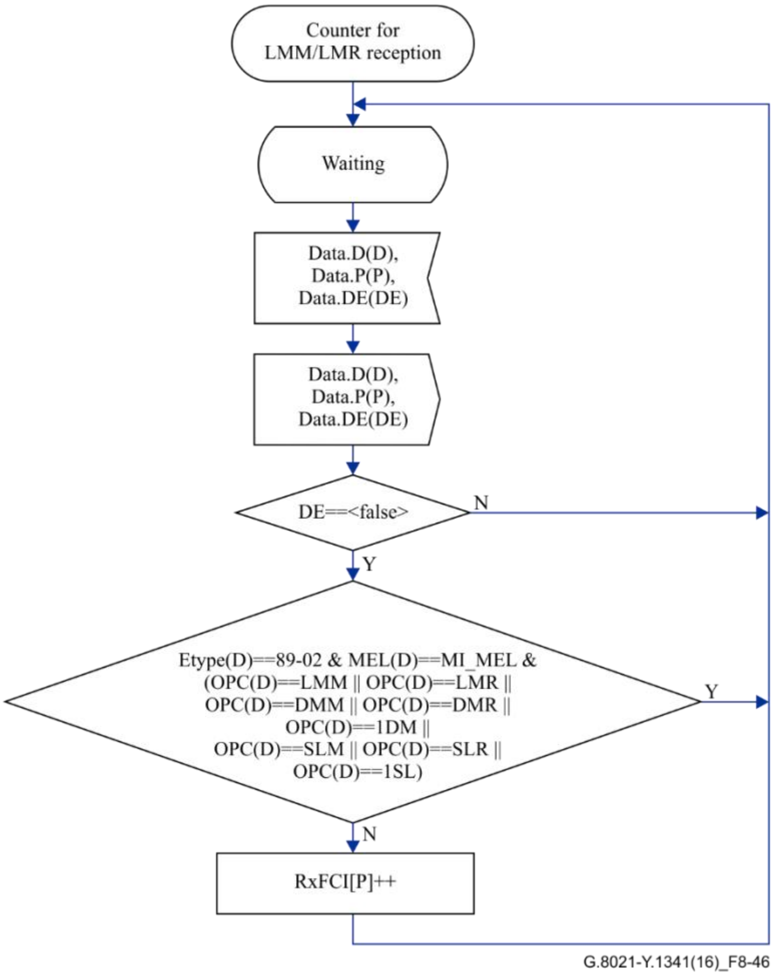


Figure 8-46 – Counter behaviour for LMM/LMR reception

NOTE 1 – To maintain the same behaviour with the earlier versions of this Recommendation, the counter process for LMM/LMR generation and reception excludes the counting of OAM frames which are applicable to both proactive and on-demand performance monitoring (i.e., LMM, LMR, DMM, DMR, 1DM, SLM, SLR and 1SL).

NOTE 2 – The current version of this Recommendation assumes that this process activates the needed TxFCl and RxFCl frame counters before any ETH-LM measurement is initiated. The mechanisms for activating these counters as well as the behaviour when an ETH-LM measurement is initiated before these counters are activated are outside the scope of this version of the Recommendation.

8.1.10 Single-ended delay measurement (DM) processes

8.1.10.1 Overview

Figure 8-47 shows the different processes inside MEPs and MIPs that are involved in the on-demand single-ended delay measurement protocol.

NOTE – In previous versions of this Recommendation, single-ended delay measurement was known as delay measurement. With regard to those definitions, refer to [ITU-T G.8001].

The MEP on-demand OAM source insertion process is defined in clause 9.4.1.1, the MEP on-demand OAM sink extraction process in clause 9.4.1.2, the MIP on-demand OAM sink extraction process in clause 9.4.2.2, and the MIP on-demand OAM source insertion process in clause 9.4.2.1. In summary, they insert and extract ETH_CI OAM signals into and from the stream of ETH_CI_D traffic units and the complementing P and D signals going through an MEP and MIP; the extraction is based on MEL and OpCode. Furthermore, the insertion process inserts the correct MEL and SA values into the OAM traffic units.

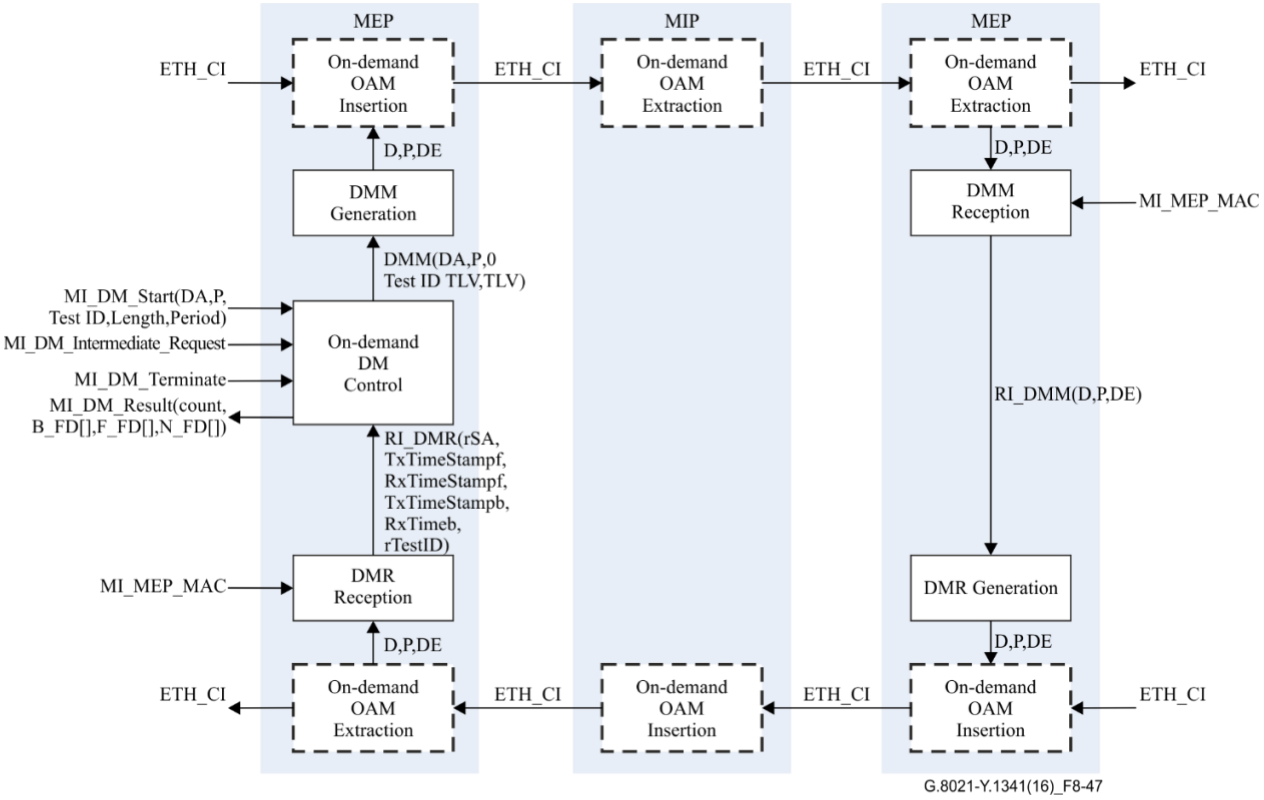


Figure 8-47 – Overview of processes involved with on-demand single-ended delay measurement

The on-demand DM control process controls the on-demand DM protocol. The protocol is activated upon receipt of the MI_DM_Start(DA,P,Test ID,Length,Period) signal and remains activated until the MI_DM_Terminate signal is received. The result is communicated via the MI_DM_Result(count, B_FD[], F_FD[] ,N_FD[]) signal when the process is terminated by the MI_DM_Terminate signal or when an intermediate result is requested via the MI_DM_Intermediate_Request signal. If the on-demand DM control process activates the multiple monitoring on different CoS levels simultaneously, each result is independently managed per CoS level. Optional test ID TLVs can be utilized to distinguish each measurement if multiple measurements are simultaneously activated in an ME. If the protocol is used in multipoint-to-multipoint environments, the multicast class 1 address can be used for a DA and the test result is independently managed per peer node.

The DMM generation process generates DMM traffic units that pass through MIPs transparently, but are received and processed by DMM reception processes in MEPs. The DMR generation process may generate a DMR traffic unit in response. This DMR traffic unit also passes transparently through MIPs, but is received and processed by DMR reception processes in MEPs.

At the source MEP side, the DMM generation process stamps the value of the local time to the TxTimeStampf field in the DMM message when the first bit of the frame is transmitted. Note well that at the sink MEP side, the DMM reception process stamps the value of the local time to the RxTimeStampf field in the DMM message when the last bit of the frame is received.

The DMR generation and reception process stamps with the same way as the DMM generation and reception process.

Figure 8-48 shows the different processes inside MEPs and MIPs that are involved in the proactive single-ended delay measurement protocol.

The MEP proactive OAM insertion process is defined in clause 9.2.1.1, the MEP OAM proactive extraction process in clause 9.2.1.2, the MIP OAM extraction process in clause 9.4.2.1, and the MIP OAM insertion process in clause 9.4.2.2. In summary, they insert and extract ETH_CI OAM signals into and from the stream of ETH_CI_D traffic units and the complementing P and D signals going through an MEP and MIP; the extraction is based on MEL and OpCode. Furthermore, the insertion process inserts the correct MEL and SA values into the OAM traffic units.

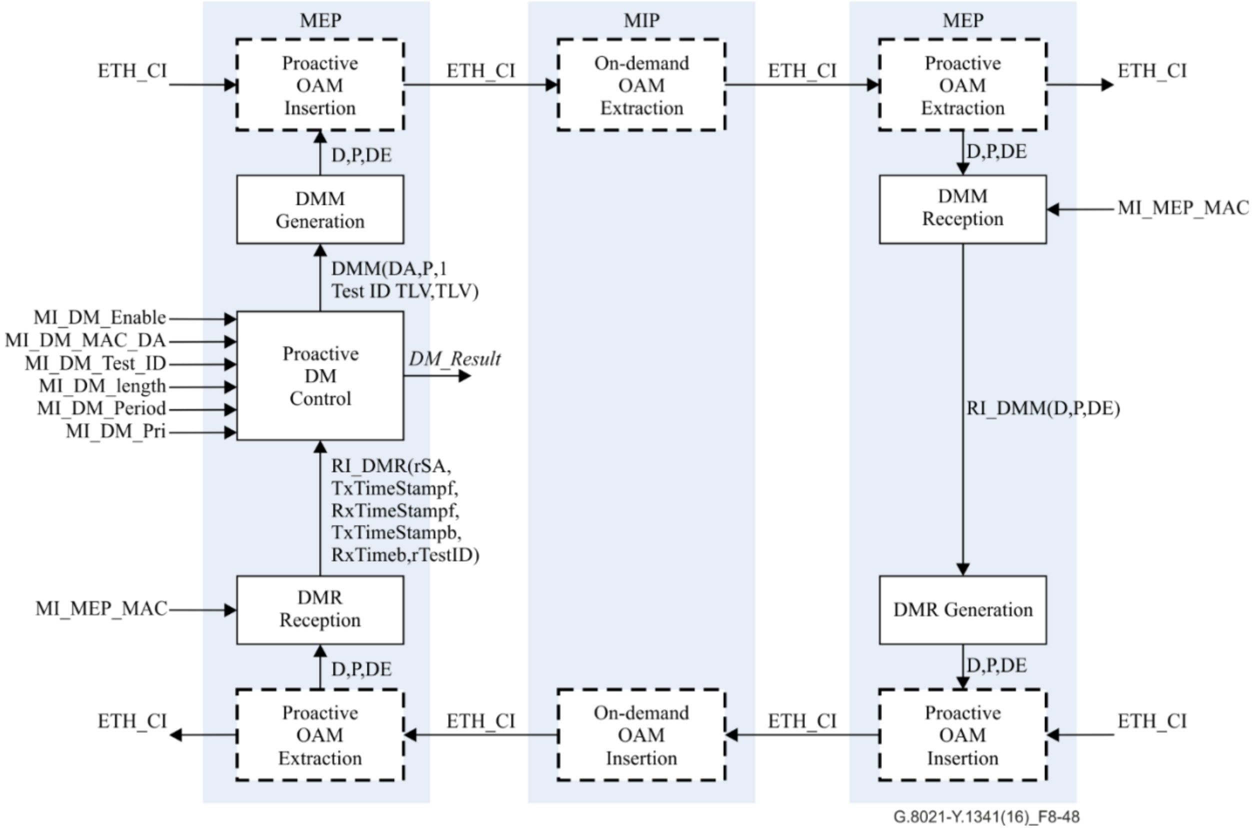


Figure 8-48 – Overview of processes involved with proactive single-ended delay measurement

The proactive DM control process controls the proactive DM protocol. If MI_DM_Enable is set the DMM frames are sent periodically. The DMM frames are generated with a periodicity determined by MI_DM_Period and with a priority determined by MI_DM_Pri. The result (B_FD, F_FD, N_FD) is reported via a DMR reception. If the proactive DM control process activates the multiple monitoring on different CoS levels simultaneously, each result is independently managed per CoS level. Optional test ID TLVs can be utilized to distinguish each measurement if multiple measurements are simultaneously activated in an ME. If the protocol is used in multipoint-to-multipoint environments, the multicast class 1 address can be used for a DA and the test result is independently managed per peer node.

8.1.10.2 DM control process

The behaviour of the on-demand DM control process is defined in Figure 8-49.

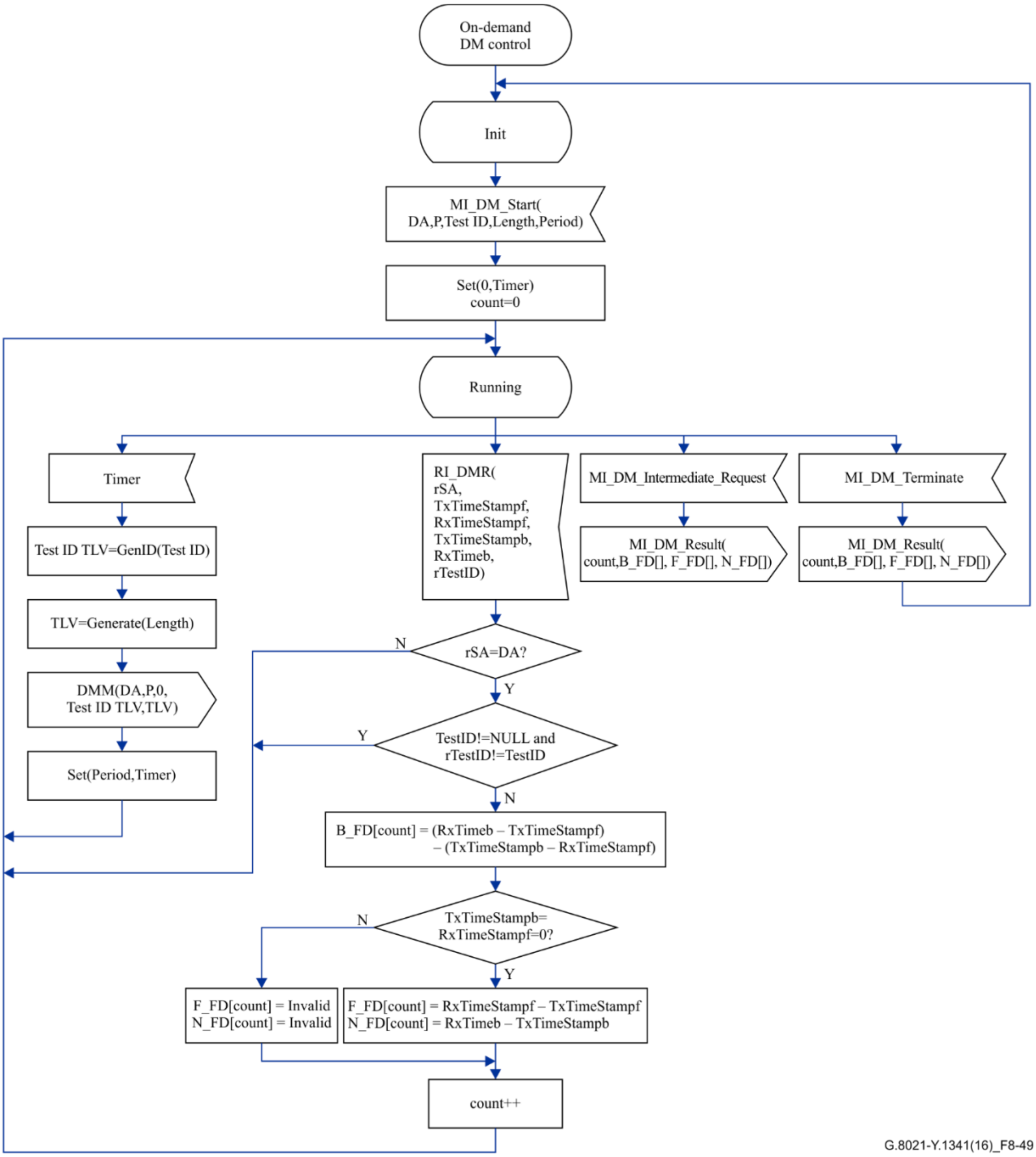


Figure 8-49 – On-demand DM control behaviour

Upon receipt of the MI_DM_Start(DA,P,Test ID,Length,Period), the DM protocol is started. Every period the generation of a DMM frame is triggered (using the DMM(DA,P,0,Test ID TLV,TLV) signal) until the MI_DM_Terminate signal is received. The TLV field of the DMM frames can have two types of TLVs. The first one is the test ID TLV, which is optionally used for a discriminator of each test and the value Test ID is included in the TLV. The second one is the data TLV, which is determined by the Generate(Length) function. Generate(Length) generates a data TLV with length "Length" of an arbitrary bit pattern to be included in the DMM frame.

Upon receipt of a DMR traffic unit the delay value recorded by this particular DMR traffic unit is calculated. This result is reported using the MI_DM_Result(count, B_FD[], F_FD[] ,N_FD[]) signal after the receipt of the MI_DM_Terminate signal or of the MI_DM_Intermediate_Request signal. Note that the measurements of F_FD and N_FD are not supported by peer MEP if both TxTimeStampb and TxTimeStampf are zero.

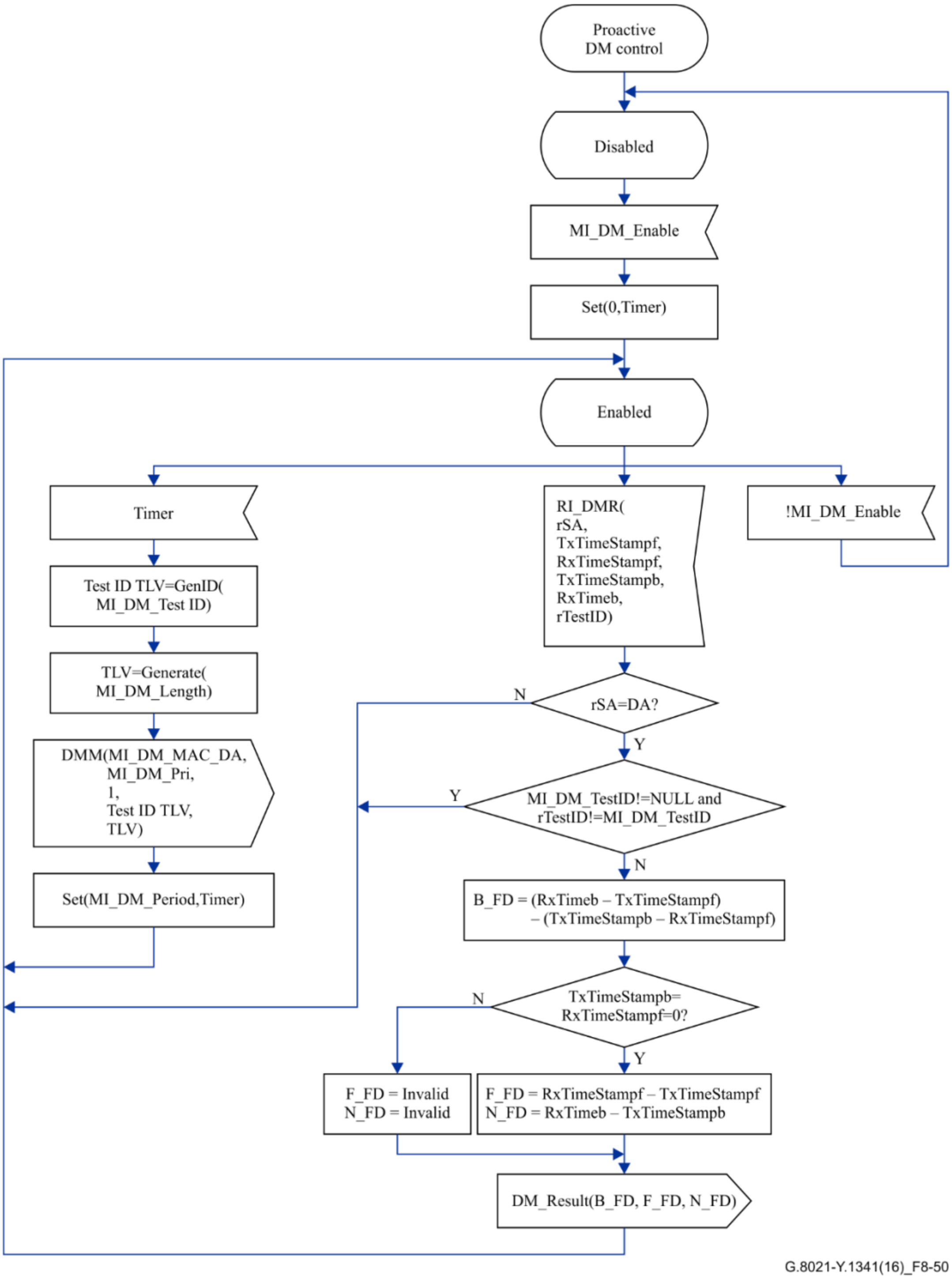


Figure 8-50 – Proactive DM control behaviour

The behaviour of the proactive DM control process is defined in Figure 8-50. If the MI_DM_Enable is asserted, the process starts to generate DMM frames (using the DMM(MI_DM_MAC_DA,MI_DM_Pri,1, Test ID TLV,TLV) signal). The result (B_FD, F_FD, N_FD) is reported via a DMR reception.

8.1.10.3 DMM generation process

The behaviour of the DMM generation process is defined in Figure 8-51.

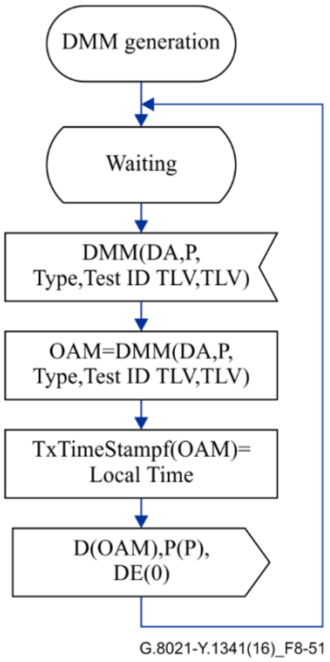


Figure 8-51 – DMM generation behaviour

Upon receiving the DMM(DA,P,Type,Test ID TLV,TLV), a single DMM traffic unit is generated together with the complementing P and DE signals. The DA of the generated traffic unit is determined by the DMM(DA) signal. The TxTimeStampf field is assigned the value of the local time.

The P signal value is defined by DMM(P). The DE signal is set to 0. The type signal is set to 1 if it is the proactive OAM, or set to 0 if it is the on-demand OAM operation. The test ID signal is determined by the DMM(Test ID TLV) signal. The TLV signal is determined by the DMM(TLV) signal.If both the test ID TLV and data TLV are included in the DMM PDU, it is recommended that the test ID TLV be located at the beginning of the optional TLV field. It makes for easier classification of the test ID in the received PDUs.

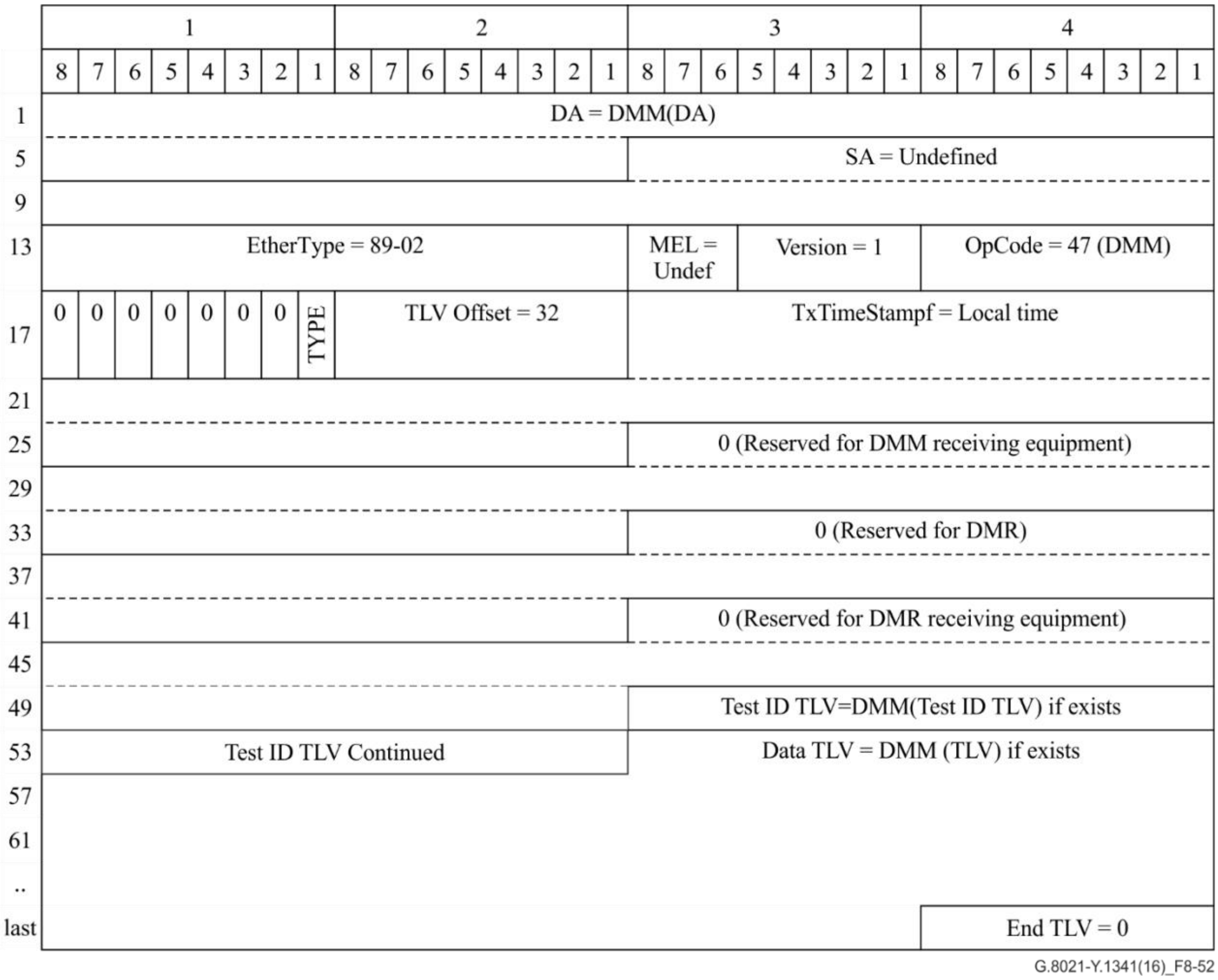


Figure 8-52 – DMM traffic unit

8.1.10.4 DMM reception process

The DMM reception process processes the received DMM traffic units and the complementing P and DE signals. The behaviour is defined in Figure 8-53.

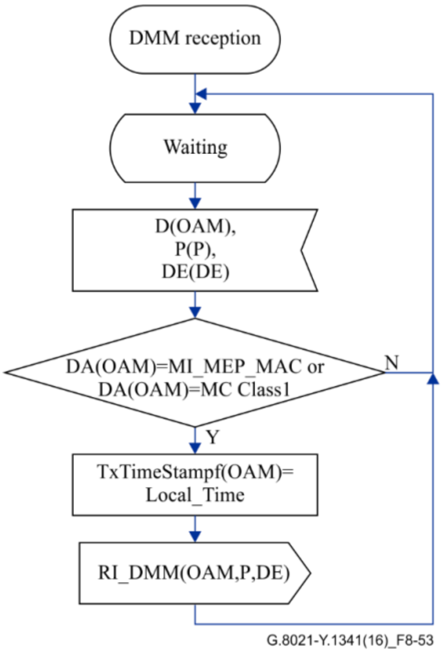


Figure 8-53 – DMM reception behaviour

First the DA is checked, it should be the local MAC address or a multicast class 1 address, otherwise the frame is ignored.

If the DA is the local MAC or a multicast class 1 address the RxTimeStampf field is assigned the value of the local time and traffic unit and the complementing P and DE signals are forwarded as remote information to the DMR generation process.

8.1.10.5 DMR generation process

The DMR generation process generates a DMR traffic unit and its complementing P and DE signals. The behaviour is defined in Figure 8-54.

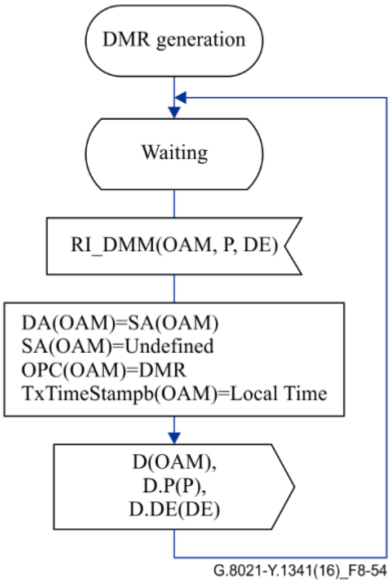


Figure 8-54 – DMR generation behaviour

Upon receipt of the remote information containing a DMM traffic unit, the DMR generation process generates a DMR traffic unit and forwards it to the OAM insertion process.

As part of the DMR generation the:

- DA of the DMR traffic unit is the SA of the original DMM traffic unit.
- The OpCode is changed into DMR OpCode.
- The TxTimeStampb field is assigned the value of the local time.
- All the other fields (including TLVs and padding after the End TLV) are copied from the remote information containing the original DMM traffic unit.

The resulting DMR traffic unit is shown in Figure 8-55.

NOTE – In the generated DMR, in the OAM (MEP) insertion process, the SA will be overwritten with the local MAC address, and the MEL will be overwritten with MI_MEL.

The TLVs are copied from the remote information containing the original DMM traffic unit. If multiple TLVs exist, the order of the TLVs is unchanged.

	1	2	3	4
	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1
1	DA=SA(RI_DMM(D))			
5			SA=Undefined	
9				
13	EtherType=89-02		MEL= Undef Version= Version (RI_DMM(D))	OpCode=46 (DMR)
17	Flags= Flags(RI_DMM(D))	TLV Offset= TLV Offset(RI_DMM(D))	TxTimeStampf=TxTimeStampf(RI_DMM(D))	
21				
25			RxTimeStampf=RxTimeStampf(RI_DMM(D))	
29				
33			TxTimeStampb=Local Time	
37				
41			0 (Reserved for DMR reception process)	
45				
49			Test ID TLV=Test ID(RI_DMM(D)) if exists	
53	Test ID TLV Continued		Data TLV= TLV (RI_DMM(D)) if exists	
57				
61				
:				
last				End TLV= End TLV(RI_DMM(D))

Figure 8-55 – DMR traffic unit

8.1.10.6 DMR reception process

The DMR reception process processes the received DMR traffic units and the complementing P and DE signals. The behaviour is defined in Figure 8-56.

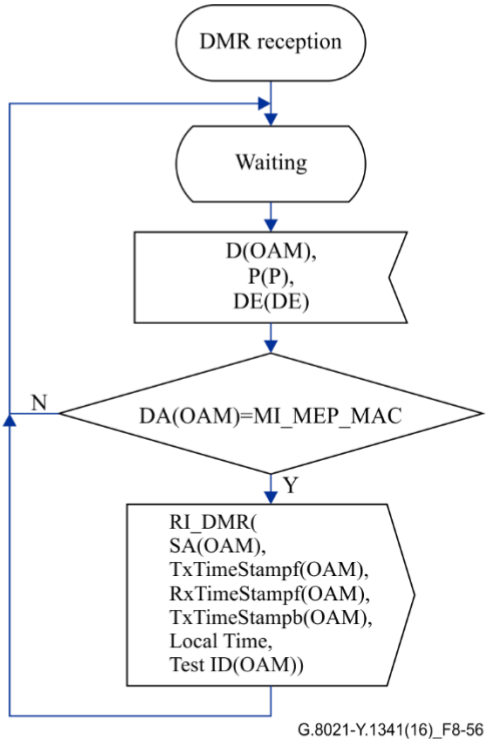


Figure 8-56 – DMR reception behaviour

Upon receipt of a DMR traffic unit the DA field of the traffic unit is checked. If the DA field equals the local MAC address, the DMR traffic unit is processed further, otherwise it is ignored.

If the DMR traffic unit is processed, the TxTimeStampf, RxTimeStampf, TxTimeStampb and test ID are extracted from the traffic unit and signalled together with the local time.

8.1.11 Dual-ended delay measurement (1DM) processes

8.1.11.1 Overview

Figure 8-57 shows the different processes inside MEPs and MIPs that are involved in the on-demand dual-ended delay measurement protocol.

NOTE – In previous versions of this Recommendation, dual-ended delay measurement was known as one-way delay measurement. With regard to those definitions, refer to [ITU-T G.8001].

The MEP on-demand OAM source insertion process is defined in clause 9.4.1.1, the MEP on-demand OAM sink extraction process in clause 9.4.1.2, and the MIP on-demand OAM sink extraction process in clause 9.4.2.2. In summary, they insert and extract ETH_CI OAM signals into and from the stream of ETH_CI_D traffic units and the complementing P and DE signals going through an MEP and MIP; the extraction is based on MEL and OpCode. Furthermore, the insertion process inserts the correct MEL and SA values into the OAM traffic units.

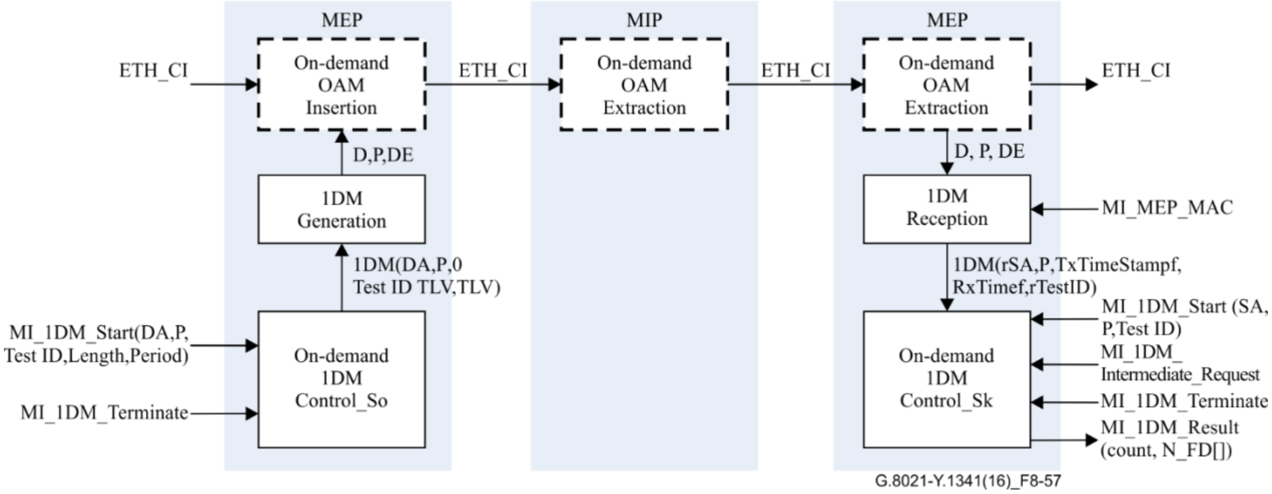


Figure 8-57 – Overview of processes involved with on-demand dual-ended delay measurement

The on-demand 1DM protocol is controlled by the on-demand 1DM Control_So and 1DM Control_Sk processes. The on-demand 1DM Control_So process triggers the generation of 1DM traffic units upon receipt of an MI_1DM_Start(DA,P,Test ID,Length,Period) signal. The on-demand 1DM Control_Sk process processes the information from received 1DM traffic units after receiving the MI_1DM_Start(SA,P,Test ID) signal. The result is communicated by the sink MEP when the on-demand 1DM Control_Sk process is terminated by the MI_1DM_Terminate signal or when an intermediate result is requested via the MI_1DM_Intermediate_Request signal.

The 1DM generation process generates 1DM messages that pass transparently through MIPs and are received and processed by the 1DM reception process in MEPs.

At the source MEP side, the 1DM generation process stamps the value of the local time to the TxTimeStampf field in the 1DM message when the first bit of the frame is transmitted. Note well that at the sink MEP side, the 1DM reception process records the value of the local time when the last bit of the frame is received.

Figure 8-58 shows the different processes inside MEPs and MIPs that are involved in the proactive dual-ended delay measurement protocol.

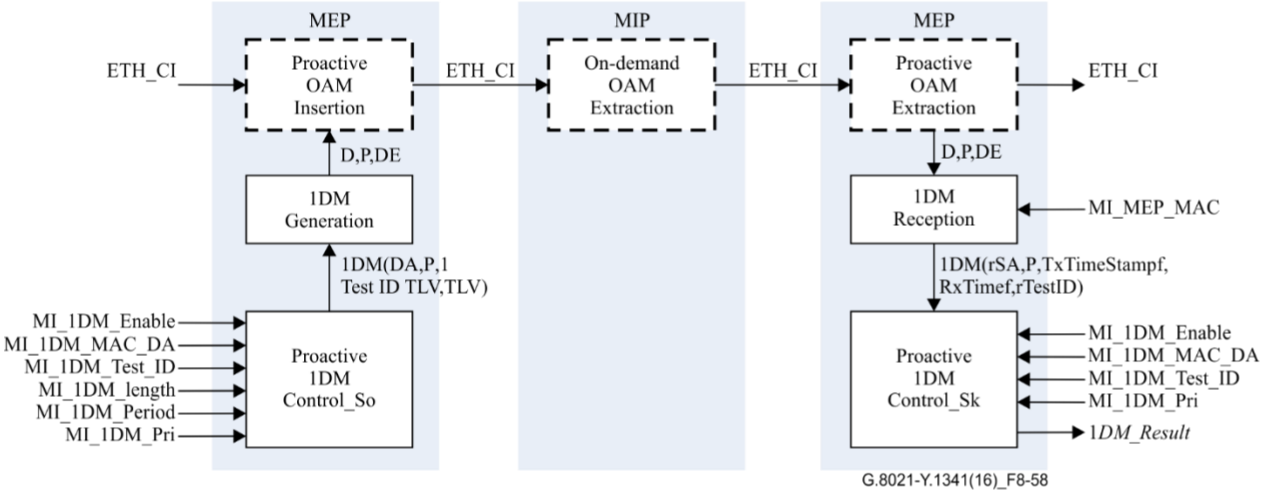


Figure 8-58 – Overview of processes involved with proactive dual-ended delay measurement

The MEP proactive-OAM source insertion process is defined in clause 9.2.1.1, the MEP proactive OAM sink extraction process in clause 9.2.1.2, and the MIP on-demand OAM sink extraction process in clause 9.4.2.2.

The proactive 1DM Control_So process triggers the generation of 1DM traffic units if MI_1DM_Enable signal is set. The 1DM frames are generated with a periodicity determined by MI_1DM_Period and with a priority determined by MI_1DM_Pri. The result (N_FD) is reported via a 1DM reception by the 1DM Control_Sk process.

8.1.11.2 1DM Control_So Process

Figure 8-59 shows the behaviour of the on-demand 1DM Control_So Process. Upon receipt of the MI_1DM_Start(DA,P,Test ID,Length,Period) signal the 1DM protocol is started. The protocol will run until the receipt of the MI_1DM_Terminate signal.

If the DM protocol is running every period (as specified in the MI_1DM_Start signal) the generation of a 1DM message is triggered by generating the 1DM(DA,P,0,Test ID TLV,TLV) signal towards the 1DM generation process. The TLV field of the 1DM frames can have two types of TLVs. The first one is the test ID TLV, which is optionally used for a discriminator of each test and the value Test ID is included in the TLV. The second one is the data TLV, which is determined by the Generate(Length) function. Generate(Length) generates a data TLV with length "Length" of an arbitrary bit pattern to be included in the 1DM frame.

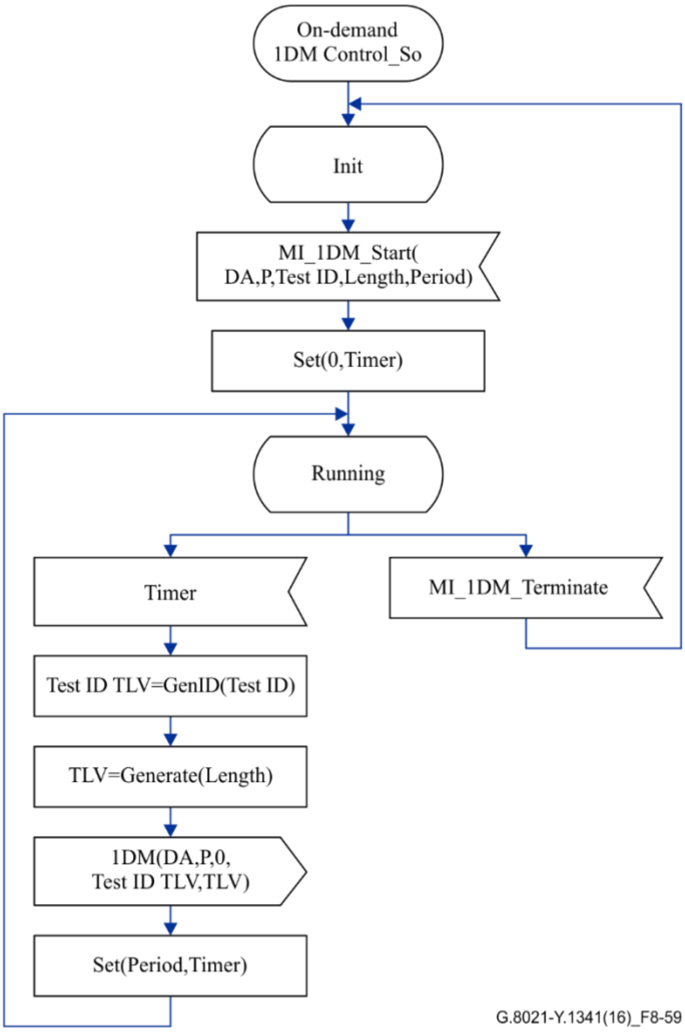


Figure 8-59 – On-demand 1DM Control_So behaviour

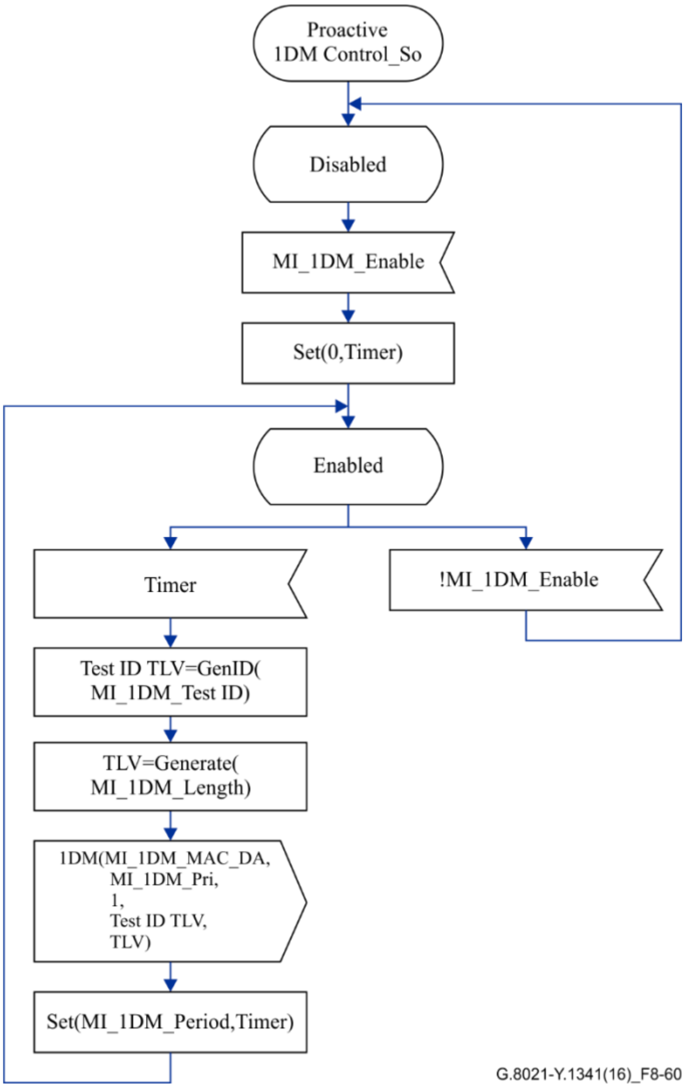


Figure 8-60 – Proactive 1DM Control_So behaviour

The behaviour of the proactive 1DM control process is defined in Figure 8-60.

If the MI_1DM_Enable is asserted, the process starts to generate 1DM frames (using the 1DM(MI_1DM_MAC_DA,MI_1DM_Pri,1, Test ID TLV,TLV) signal.

8.1.11.3 1DM generation process

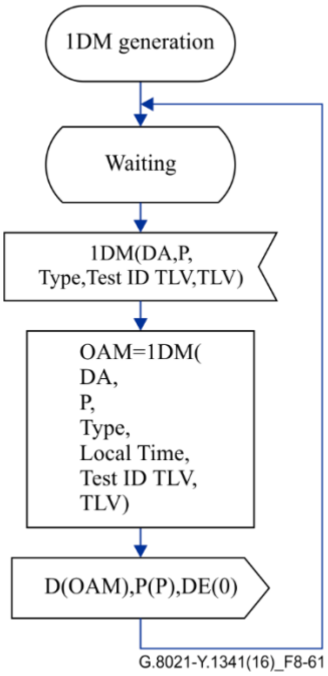


Figure 8-61 – 1DM generation behaviour

Figure 8-61 shows the 1DM generation process. Upon receiving the 1DM(DA,P,Type,Test ID TLV,TLV) signal a single 1DM traffic unit is generated by the OAM=1DM (DA,P,Type, LocalTime, Test ID TLV, TLV) call.

Together with this 1DM traffic unit the complementing P and DE signals are generated. The DA of the generated 1DM traffic unit is determined by the 1DM(DA) signal. The TxTimeStampf field is assigned the value of the local time. The value of the P signal is determined by the 1DM(P) signal. The DE signal is set to 0. The type signal is set to 1 if it is the proactive OAM, or set to 0 if it is the on-demand OAM operation. The test ID signal is determined by the 1DM(Test ID TLV) signal. The TLV signal is determined by the 1DM(TLV) signal.

The resulting traffic unit is shown in Figure 8-62.

NOTE – In the generated 1DM traffic unit, in the OAM (MEP) insertion process, the SA will be assigned the local MAC address, and the MEL will be assigned by MI_MEL.

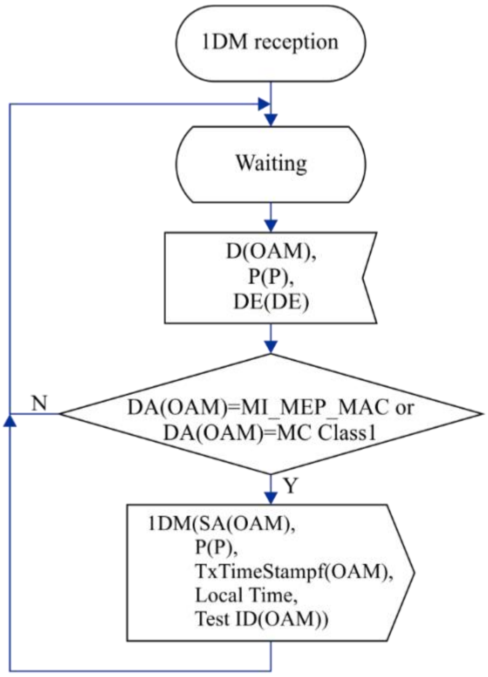
If both the test ID TLV and data TLV are included in the 1DM PDU, it is recommended that the test ID TLV be located at the beginning of the optional TLV field. It makes for easier classification of the test ID in the received PDUs.

	1	2	3	4
	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1
1	DA=1DM(DA)			
5			SA=Undefined	
9				
13	EtherType=89-02		MEL= Undef Version=1	OpCode=45 (1DM)
17	0 0 0 0 0 0 Type	TLV Offset =16	TxTimeStampf=Local Time	
21				
25			0 (Reserved for 1DM receiving equipment)	
29				
33			Test ID TLV=1DM(Test ID TLV) if exists	
37	Test ID TLV Continued		Data TLV=1DM(TLV) if exists	
41				
45				
:				
last				End TLV (0)

Figure 8-62 – 1DM traffic unit

8.1.11.4 1DM reception process

The 1DM reception process processes the received 1DM traffic units and the complementing P and DE signals. The behaviour is defined in Figure 8-63.



G.8021-Y.1341(16)_F8-63

Figure 8-63 – 1DM reception behaviour

Upon receipt of a 1DM traffic unit the DA field is checked. The 1DM traffic unit is processed if the DA is equal to the local MAC address or multicast class 1 MAC address. Otherwise, the traffic unit is ignored.

If the 1DM traffic unit is processed the SA and TxTimeStampf fields are extracted and forwarded to the 1DM Control_Sk process together with the local time using the 1DM(rSA,rP,TxTimeStampf,RxTimef,rTestID) signal.

8.1.11.5 1DM Control_Sk Process

Figure 8-64 shows the behaviour of the on-demand 1DM Control_Sk process. The MI_1DM_Start(SA,P,TestID) signal starts the processing of 1DM messages coming from an MEP with SA as the MAC address. The protocol runs until the receipt of the MI_1DM_Terminate signal.

While running the process processes the received 1DM(rSA,rP,TxTimeStampf,RxTimef,rTestID) information. First the rSA is compared with the SA from the MI_1DM_Start (SA) signal. If the rSA is not equal to this SA, the information is ignored. Next the rP is compared with the priority from the MI_1DM_Start (P) signal. If the rP is not equal to this P, the information is ignored. Finally the rTestID is compared with the TestID from the MI_1DM_Start (Test ID) signal. If the MI_1DM_Start (Test ID) signal is configured and rTestID is available but both values are different, the information is ignored. Otherwise the delay from the single received 1DM traffic unit is calculated. This result is reported using the MI_1DM_Result(count, N_FD[]) signal after the receipt of the MI_1DM_Terminate signal or of the MI_1DM_Intermediate_Request signal.

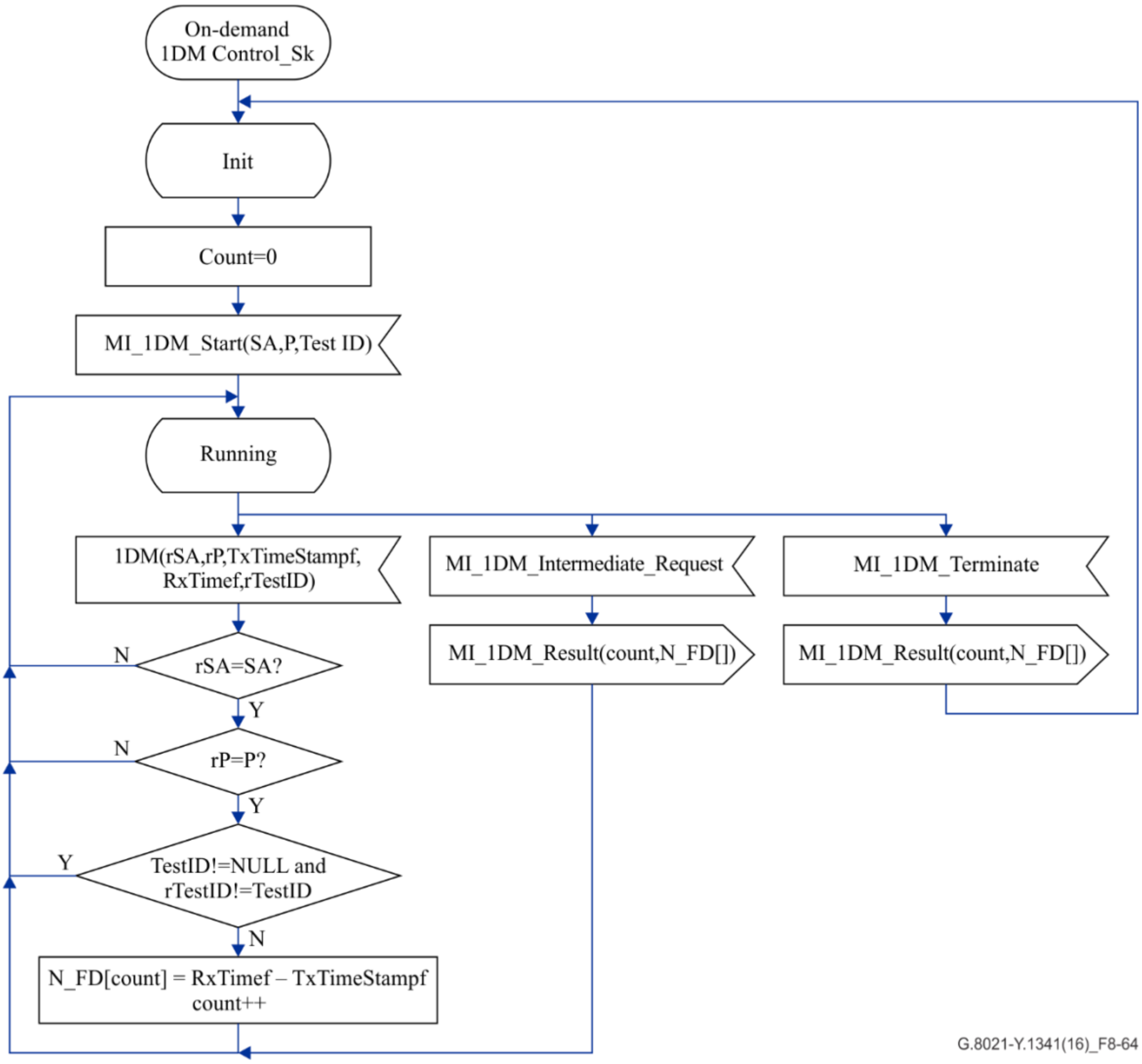


Figure 8-64 – On-demand 1DM Control_Sk process

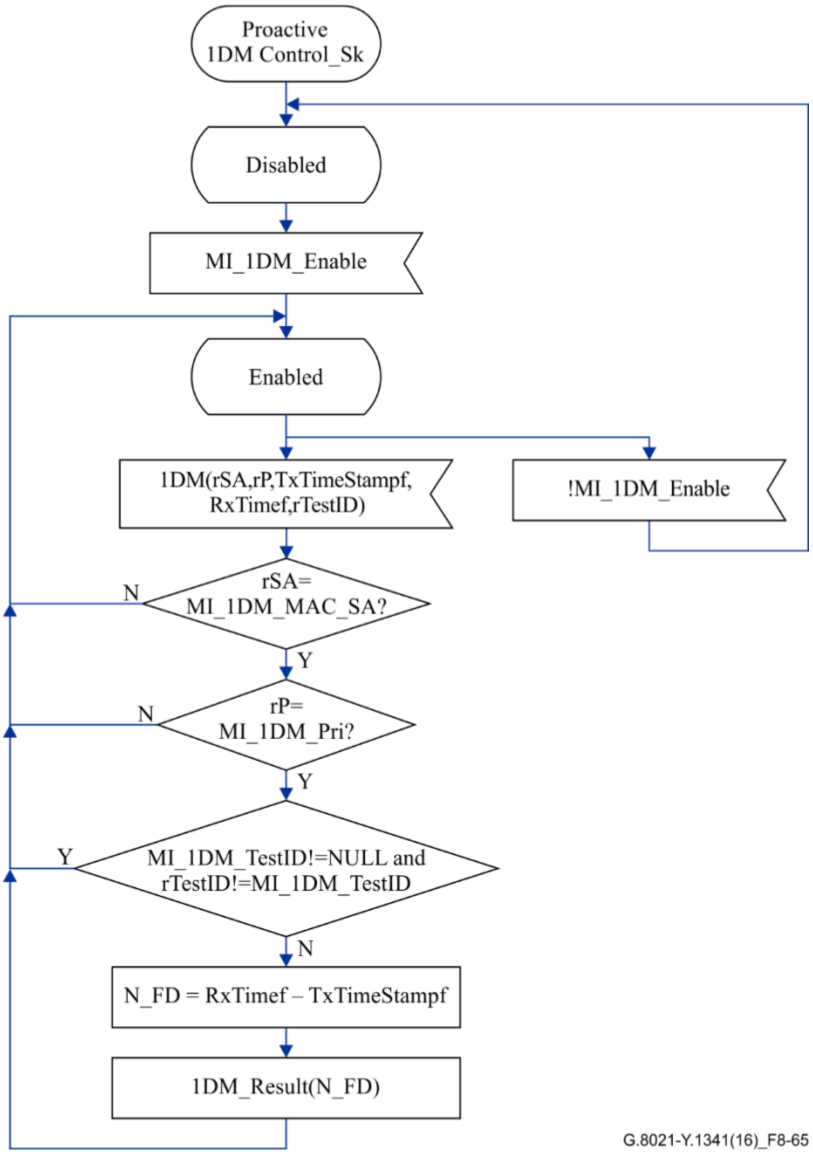


Figure 8-65 – Proactive 1DM Control_Sk process

The behaviour of the proactive 1DM Control_Sk Process is defined in Figure 8-65. If the MI_1DM_Enable is asserted, the result (N_FD) is reported via a 1DM reception.

8.1.12 Test (TST) processes

8.1.12.1 Overview

Figure 8-66 shows the different processes inside MEPs and MIPs that are involved in the test protocol.

The MEP on-demand OAM source insertion process is defined in clause 9.4.1.1, the MEP on-demand OAM sink extraction process in clause 9.4.1.2, the MIP on-demand OAM sink extraction process in clause 9.4.2.2. In summary, they insert and extract ETH_CI OAM signals into and from the stream of ETH_CI_D traffic units together with the complementing P and DE signals going through an MEP and MIP; the extraction is based on MEL and OpCode. Furthermore, the insertion process inserts the correct MEL and SA values into the OAM traffic units.

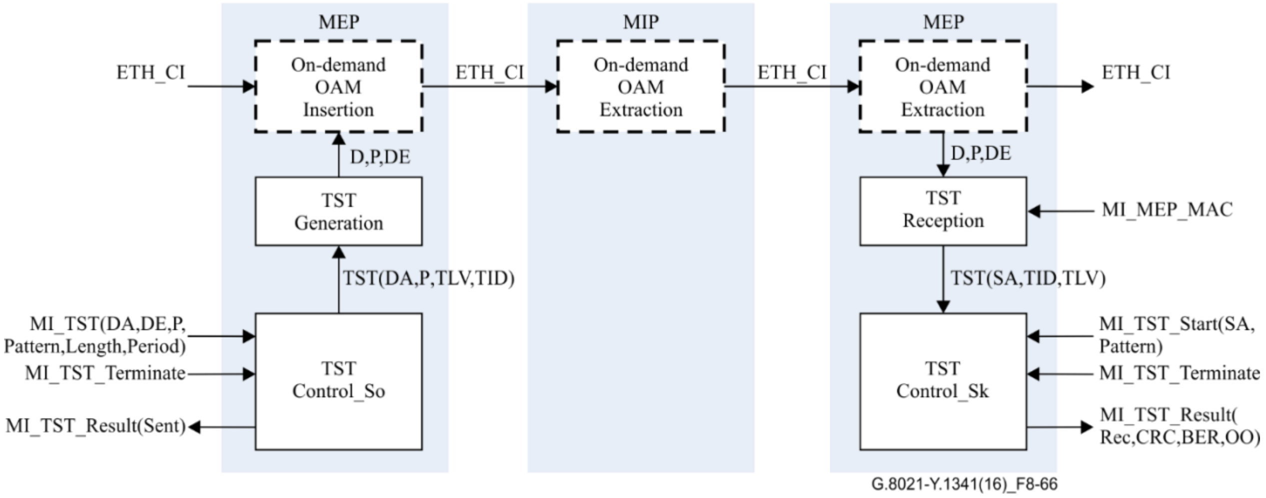


Figure 8-66 – Overview of processes involved with the test protocol

The TST protocol is controlled by the TST Control_So and TST Control_Sk processes. The TST Control_So process triggers the generation of TST traffic units after the receipt of an MI_TST_Start(DA,DE,P,Pattern,Length,Period) signal. The TST Control_Sk process processes the information from received TST traffic units after receiving the MI_TST_Start(SA,Pattern) signal.

The TST generation process generates TST messages that pass transparently through MIPs and are received and processed by the TST reception process in MEPs.

The processes are defined below.

8.1.12.2 TST Control_So process

Figure 8-67 defines the behaviour of the TST Control_So process. This process starts the transmission of TST traffic units after receiving the MI_Test(DA,DE,P,Pattern,Length,Period) signal. Each transmission of TST traffic units is triggered by the generation of the TST(DA,P,DE,TLV,TID) signal. This is continued until the receipt of the MI_Test_Terminate signal. After receiving this signal the number of triggered TST traffic units is reported back using the MI_Test_Result(Sent) signal.

The TLV field of the TST frames is determined by the Generate(Pattern, Length) function. For "Pattern" the following types are defined:

0: "Null signal without CRC-32"

1: "Null signal with CRC-32"

2: "PRBS 2^31-1 without CRC-32"

3: "PRBS 2^31-1 with CRC-32"

The length parameter determines the length of the generated TLV.

Generate(Pattern, Length) generates a test TLV with length "Length" to be included in the TST frame. Therefore, this TLV is passed using the TST(DA,P,DE,TLV,TID) signal to the TST generation process.

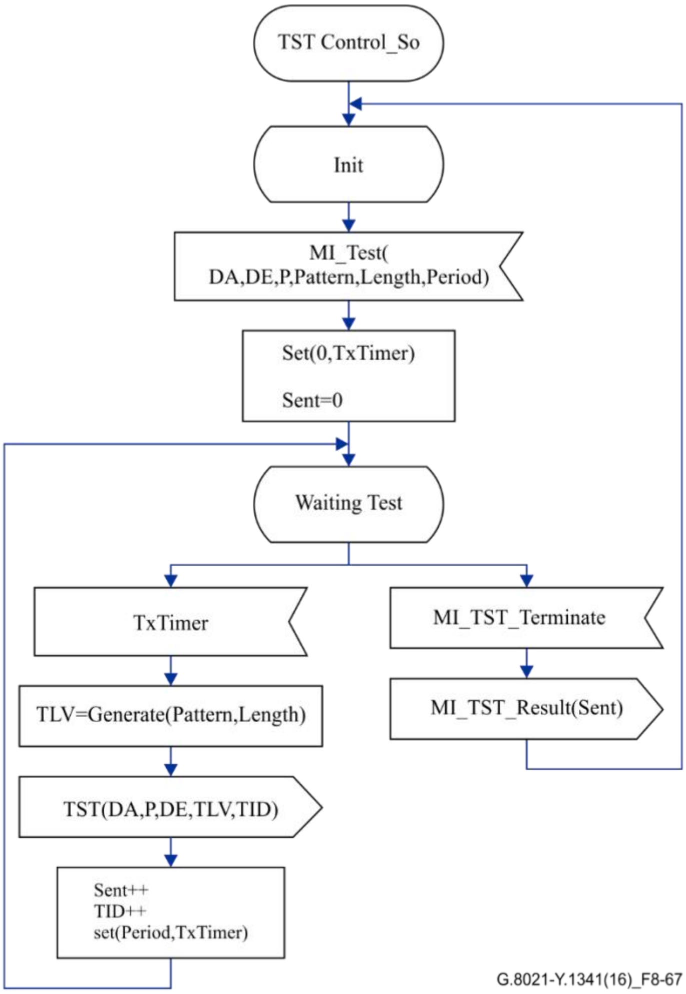


Figure 8-67 – TST Control_So behaviour

8.1.12.3 TST generation process

Figure 8-68 defines the behaviour of the TST generation process.

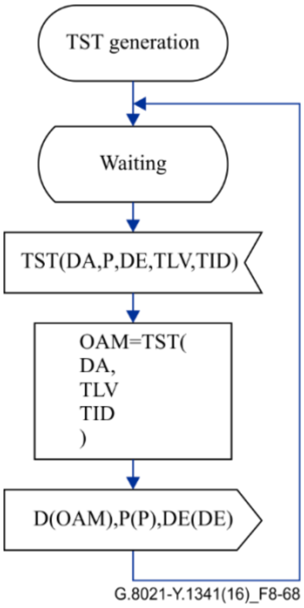


Figure 8-68 – TST generation behaviour

Upon receiving the TST(DA,P,DE,TLV,TID), a single TST traffic unit is generated together with the complementing P and DE signals. The TST traffic unit is generated by:

OAM=TST(DA,TLV,TID).

The DA of the generated TST traffic unit is determined by the TST(DA) signal. The transaction identifier field gets the value of TST(TID); the TLV field is populated with TST(TLV). The resulting TST traffic unit is shown in Figure 8-69.

NOTE – In the generated TST traffic unit, in the OAM (MEP) insertion process, the SA will be assigned the local MAC address, and the MEL will be assigned by MI_MEL.

The P signal is determined by the TST(P) signal.

The DE signal is determined by the TST(DE) signal.

<table>
<tr><td></td><td colspan="8">1</td><td colspan="8">2</td><td colspan="8">3</td><td colspan="8">4</td></tr>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td colspan="32">DA=TST(DA)</td></tr>
<tr><td>2</td><td colspan="16"></td><td colspan="16">SA=Undefined</td></tr>
<tr><td>3</td><td colspan="32"></td></tr>
<tr><td>4</td><td colspan="16">EtherType=89-02</td><td colspan="3">MEL=
Undef</td><td colspan="5">Version=0</td><td colspan="8">OpCode=37(TST)</td></tr>
<tr><td>5</td><td colspan="8">Flags=0</td><td colspan="8">TLV Offset =4</td><td colspan="16">Transaction Identifier=TST(TID)</td></tr>
<tr><td>6</td><td colspan="16">Transaction Identifier Continued</td><td colspan="16">TST(TLV)</td></tr>
<tr><td>7</td><td colspan="32"></td></tr>
<tr><td>8</td><td colspan="32"></td></tr>
<tr><td>9</td><td colspan="16"></td><td colspan="8">End TLV=0</td><td colspan="8"></td></tr>
</table>

Figure 8-69 – TST traffic unit

8.1.12.4 TST reception process

Figure 8-70 defines the behaviour of the TST reception process.

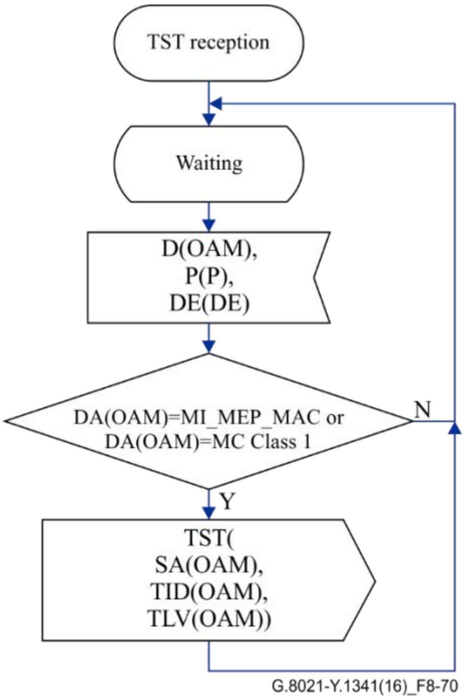


Figure 8-70 – TST reception behaviour

First the DA is checked, it should be the local MAC address (as configured via MI_MEP_MAC) or a multicast class 1 address, otherwise the frame is ignored.

If the DA is the local MAC or a multicast class 1 address the SA, TID and TLV fields from the TST traffic unit are forwarded using the TST signal.

8.1.12.5 TST Control_Sk process

Figure 8-71 shows the behaviour of the TST Control_Sk process. The MI_TST_Start (SA) signal starts the processing of TST messages coming from an MEP with SA as the MAC address. The protocol is running until the receipt of the MI_TST_Terminate signal.

While running, the process processes the received TST(rSA,rTLV,TID) information. First the rSA is compared with the SA from the MI_TST_Start (SA) signal. If the rSA is not equal to this SA, the information is ignored. Otherwise the received information is processed.

First, the received TST counter is incremented by one (REC++). Furthermore, if the TLV contains a CRC (Pattern 1 or 3), the CRC counter is incremented by one (CRC++) if the CRC check fails. The function Check(Pattern, TLV) compares the received test pattern with the expected test pattern. If there is a mismatch the BERR counter is incremented by one. If the TID value from the RI_LBR signal does not follow the last received TID value the counter for out of order frames is incremented by one (OO++).

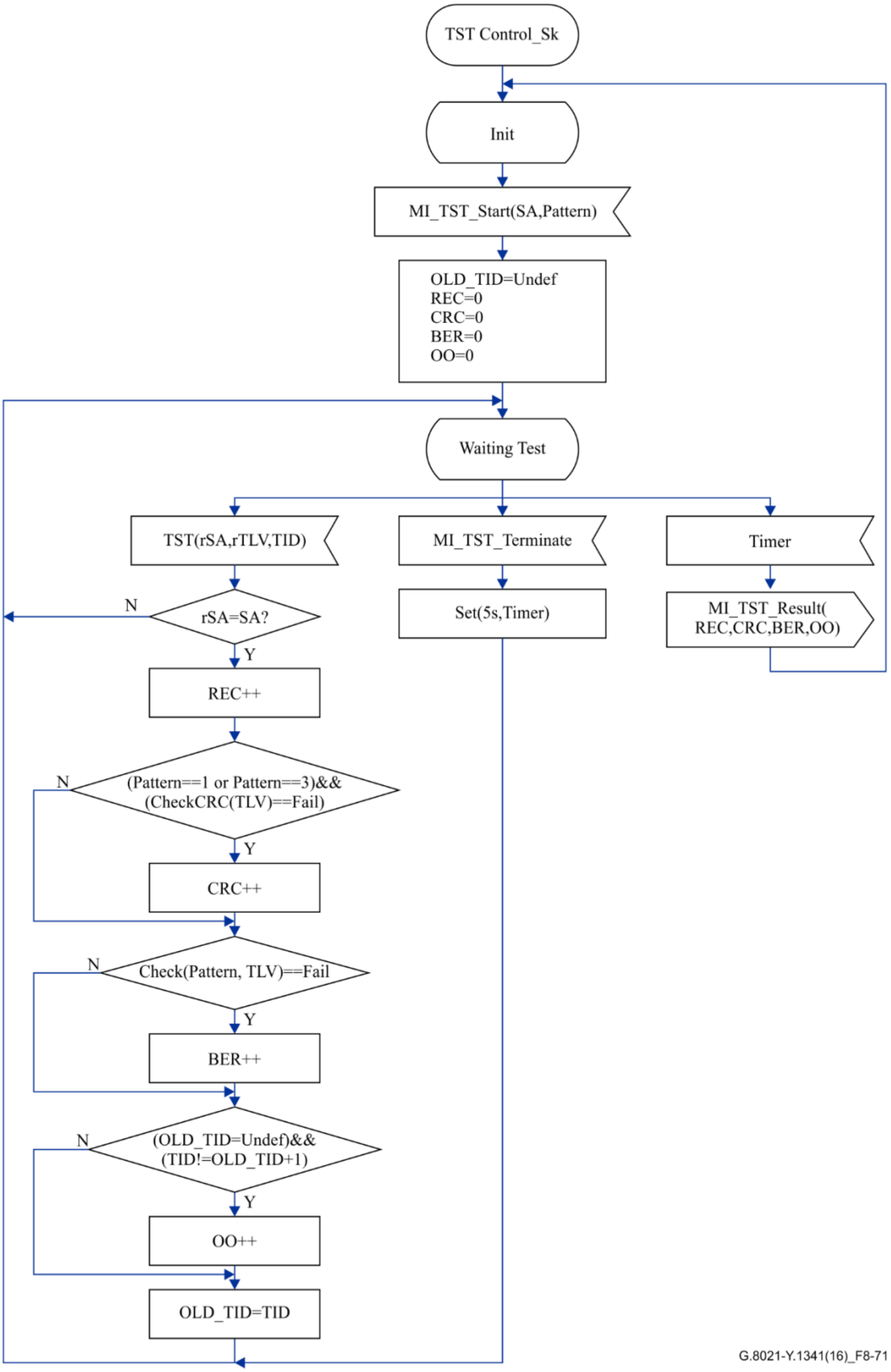


Figure 8-71 – TST Control_Sk behaviour

8.1.13 Link trace (LT) processes

8.1.13.1 Overview

Figure 8-72 shows the different processes involved in the link trace protocol.

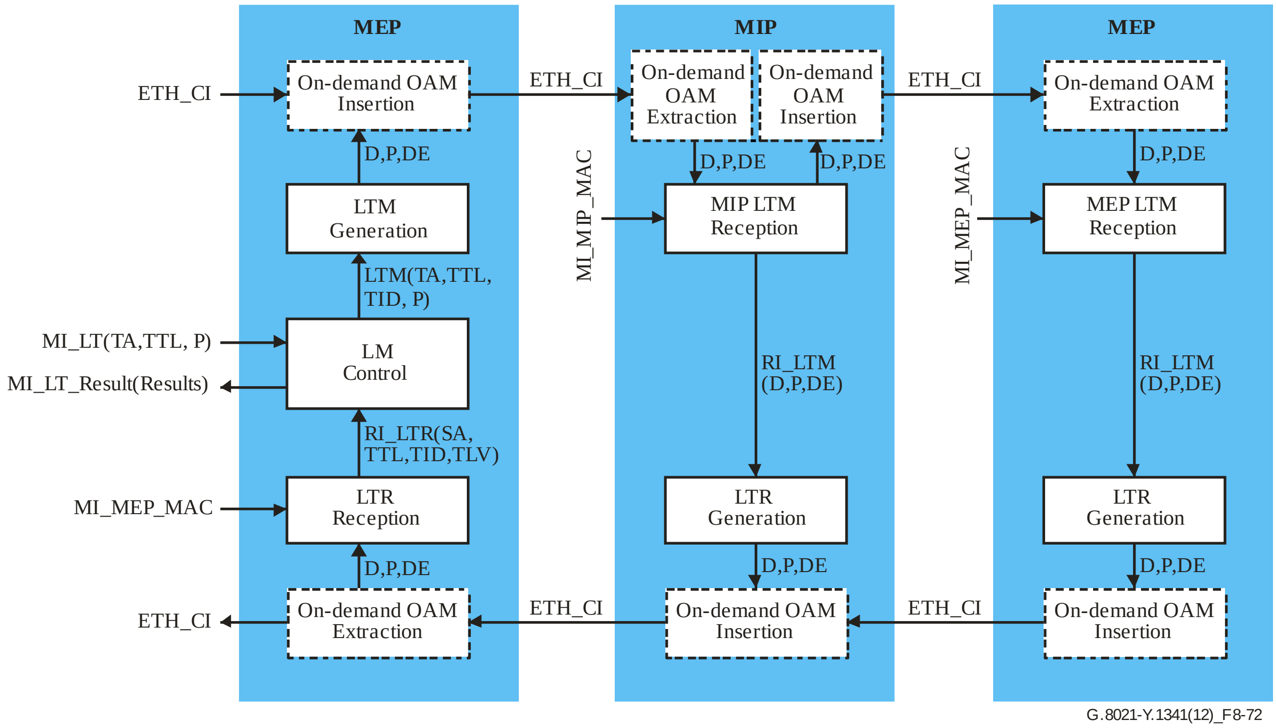


Figure 8-72 – LT protocol overview

The link trace protocol is started upon receipt of an MI_LT(TA, TTL, P) signal. The result of the process will be communicated back via the MI_LT_Result(Results) signal.

The LM control will trigger the transmission of an LTM traffic unit and then wait for the LTR traffic units that are sent in reply to this LTM traffic unit.

The LTM traffic unit is processed by MIP LTM reception processes and by MEP LTM reception processes. Depending on the DA given in the MI_LT(TA, TTL, P) signal these processes may decide to trigger the transmission of an LTR traffic unit back to the source of the LTM traffic unit.

NOTE – In the 2008 version of Recommendation ITU-T G.8013/Y.1731 the LTM traffic unit is received by an ETH-LT responder process which solely resides in a network element and acts as an alternative process for LTM MIP reception. Similarly, the trigger of sending an LTR traffic unit is decided by the ETH-LT responder.

8.1.13.2 LT control process

Figure 8-73 shows the behaviour of the LT control process.

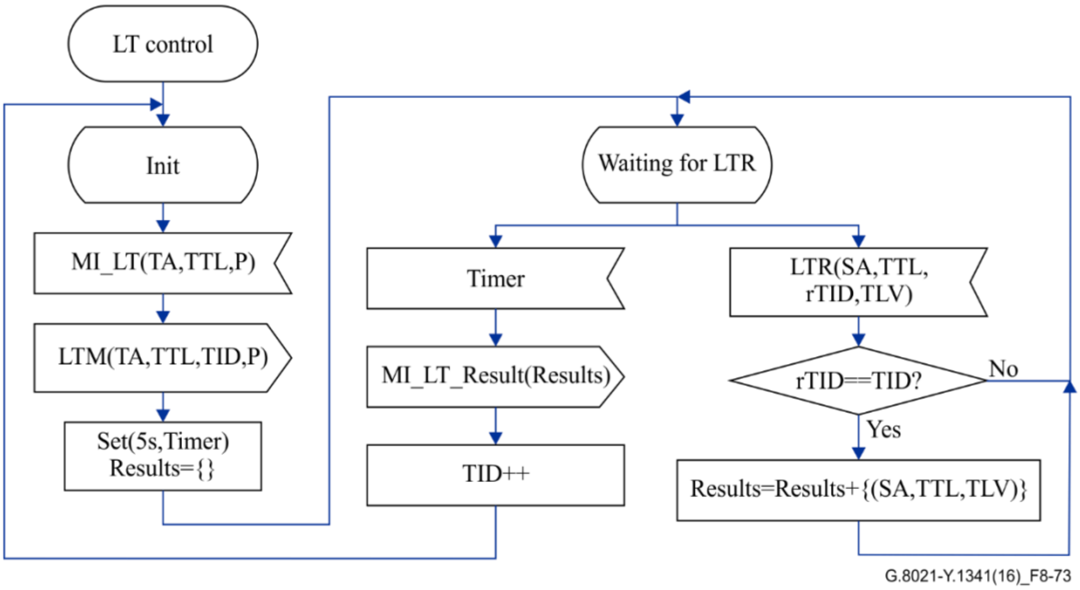


Figure 8-73 – LT control behaviour

After receiving the MI_LT(TA, TTL, P) input signal, the transmission of an LTM traffic unit is triggered. In the "Waiting for LTR" state, the LTM control process waits for the LTR traffic units that will be sent in response. The waiting period is five seconds. For each received LTR traffic unit the TID value in the received LTM traffic unit is compared with the one that was sent in the LTM traffic unit. If they are equal, the SA, TTL and TLV values are stored in the results. These results are communicated back using the MI_LT_Results signal after the five second waiting period is over.

8.1.13.3 LTM generation process

Figure 8-74 shows the behaviour of the LTM generation process.

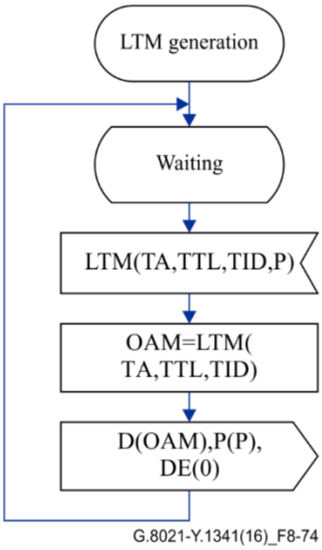
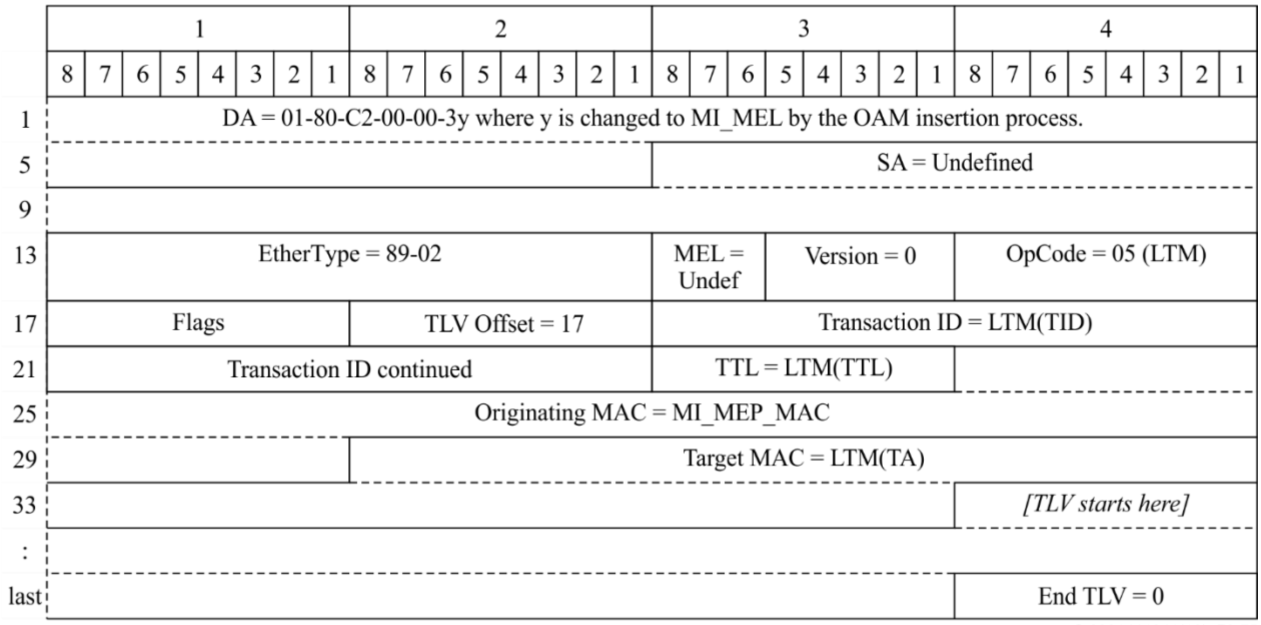


Figure 8-74 – LTM generation behaviour

The LTM generation process generates an LTM traffic unit with the function:

OAM=LTM(TA, TTL, TID) and the result is shown in Figure 8-75.

NOTE – In the generated LTM traffic unit, in the OAM (MEP) insertion process, the SA will be assigned the local MAC address, and the MEL will be assigned by MI_MEL. The value of the multicast class 2 DA is 01-80-C2-00-00-3y, where y is equal to {MI_MEL + 8} as defined in clause 10.1 of [ITU-T G.8013]. The usage of flags is specified in clause 9.5.2 of [ITU-T G.8013].

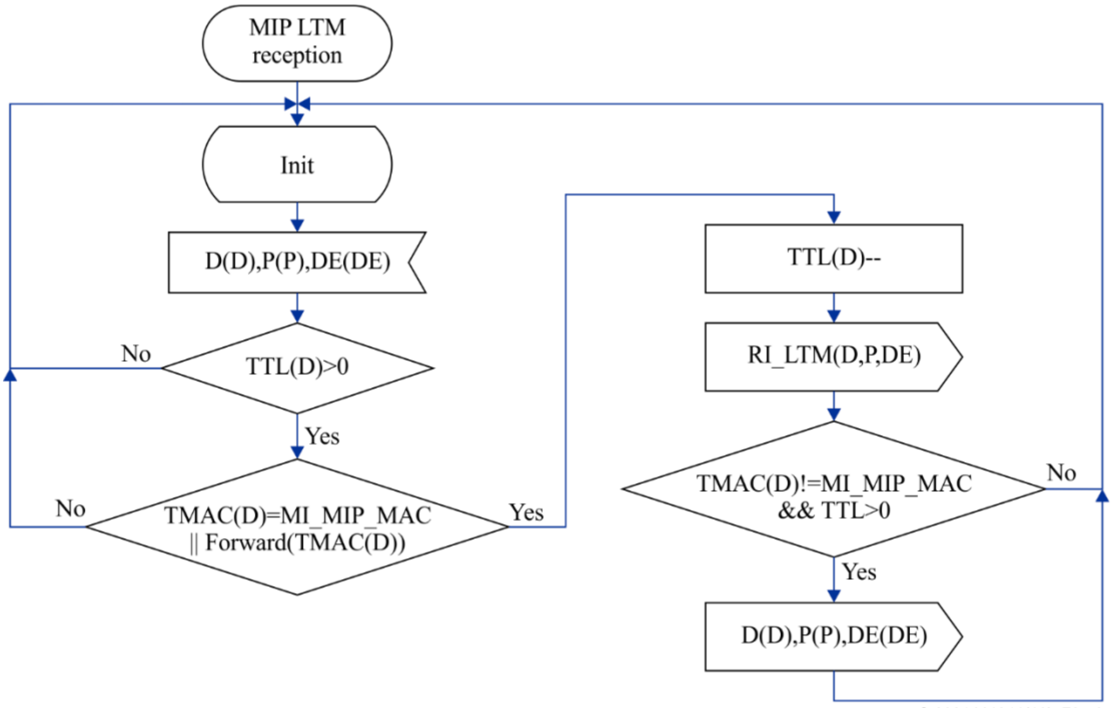


G.8021-Y.1341(16)_F8-75

Figure 8-75 – LTM traffic unit

8.1.13.4 MIP LTM reception process

Figure 8-76 shows the behaviour of the MIP LTM reception process.



G.8021-Y.1341(16)_F8-76

Figure 8-76 – MIP LTM reception behaviour

Upon receipt of an LTM traffic unit, first the TTL is checked, only LTM traffic units with a TTL>0 are processed. Thereafter, the target MAC (TMAC) of the LTM traffic unit is checked.

There are two reasons to send back an LTR traffic unit. The first is if the TMAC in the LTM traffic unit is the MAC address of the MIP itself.

The second reason is summarized in Figure 8-65 as Forward(TMAC(D)). This function returns true if:

- the network element that the MIP LTM reception process resides in would forward a normal data traffic unit with its DA equal to the TMAC to a single port (forwarding port), and
- the MIP LTM reception process resides in the egress port which equals to the "forwarding port" (LTM in egress port), or the MIP LTM reception process resides in the ingress port which does not equal to the "forwarding port" (LTM in ingress port).

Furthermore, after triggering the transmission of an LTR traffic unit, the LTM traffic unit is forwarded if the TMAC was not the MAC of the MIP and if the TTL>0.

8.1.13.5 MEP LTM reception process

Figure 8-77 shows the behaviour of the MEP LTM reception process.

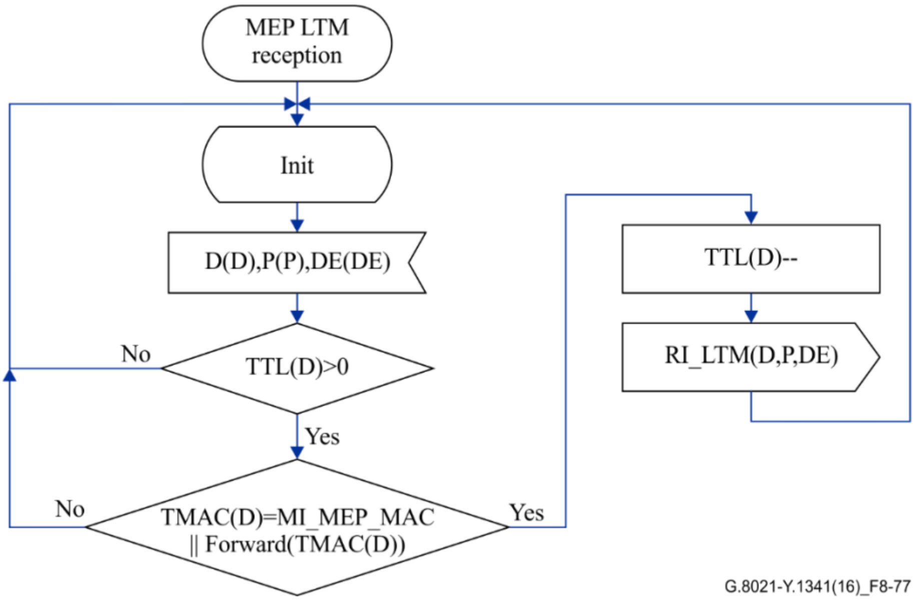


Figure 8-77 – MEP LTM reception behaviour

Upon receipt of an LTM traffic unit first the TTL is checked, only LTM traffic units with a TTL>0 are processed. Thereafter the Target MAC (TMAC) of the LTM traffic unit is checked. Conditions to send back an LTR traffic unit are similar with ones for MIP LTM reception process. The first is if the TMAC in the LTM traffic unit is the MAC address of the MEP itself. The second is summarized in Figure 8-77 as Forward(TMAC(D)). This function returns true if:

- the network element the MEP LTM reception process resides in would forward a normal data traffic unit with its DA equal to the TMAC to a single port (forwarding port), and
- the MEP LTM reception process resides in the egress port which equals to the "forwarding port" (LTM in egress port), or the MEP LTM reception process resides in the ingress port which does not equal to the "forwarding port" (LTM in ingress port).

Note that the LTM traffic unit is not forwarded anymore regardless of the value of TMAC.

8.1.13.6 LTR generation process

Figure 8-78 shows the behaviour of the LTR generation process.

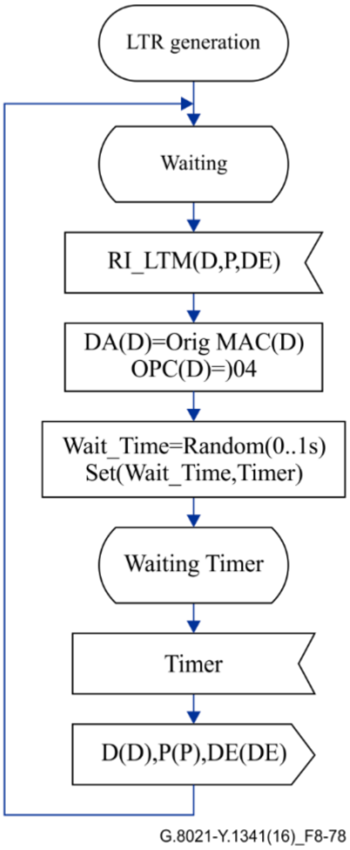


Figure 8-78 – LTR generation behaviour

The LTR generation process generates the LTR traffic unit to be sent back, based on the LTM traffic unit. The DA of the LTR traffic unit is the originating MAC (Orig MAC) as contained in the LTM traffic unit. The OpCode is the LTR OpCode. The resulting LTR traffic unit is shown in Figure 8-79. The SA and MEL will be overwritten by the OAM insertion process. The LTR traffic unit is sent back after a random delay between 0 and 1 second. The usage of flags is specified in clause 9.6.2 of [ITU-T G.8013].

The resulting frame is shown in Figure 8-79.

NOTE – In the generated LTR, in the OAM (MEP) insertion process, the SA will be overwritten with the local MAC address, and the MEL will be overwritten with MI_MEL.

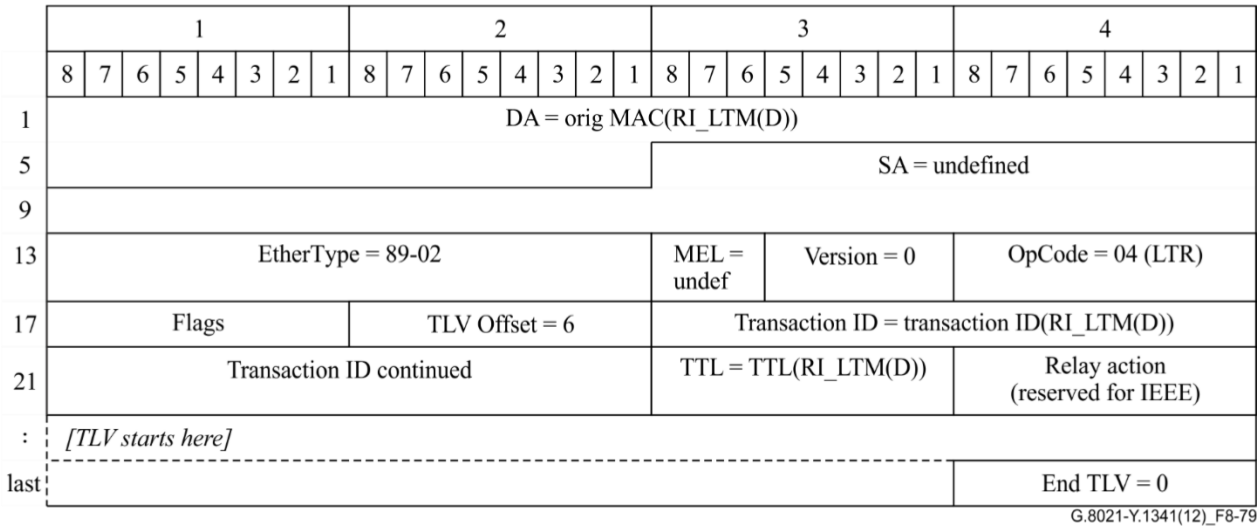


Figure 8-79 – LTR traffic unit

8.1.13.7 LTR reception process

Figure 8-80 shows the behaviour of the LTR reception process.

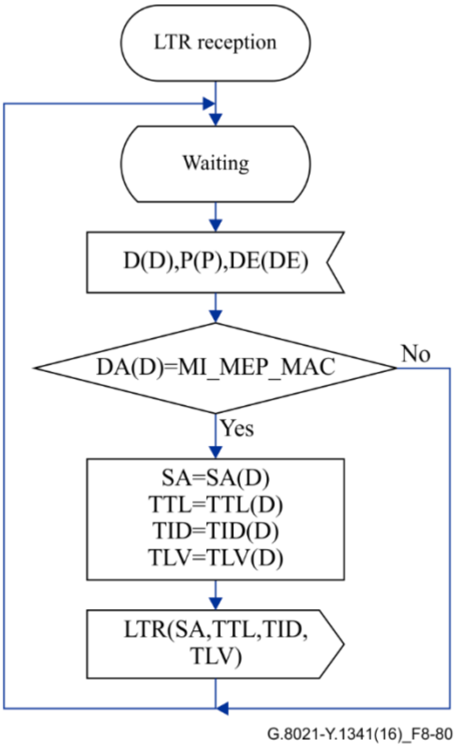


Figure 8-80 – LTR reception behaviour

The LTR reception process checks the DA of the received LTR traffic unit and passes the SA, TTL, TID and TLV fields from the LTR traffic unit to the LT control process.

8.1.14 Single-ended synthetic loss measurement (SL) processes

8.1.14.1 Overview

Figure 8-81 shows the different processes inside MEPs and MIPs that are involved in the on-demand single-ended synthetic loss measurement protocol.

NOTE – In previous versions of this Recommendation, single-ended synthetic loss measurement was known as synthetic loss measurement. With regard to those definitions, refer to [ITU-T G.8001].

The MEP on-demand OAM insertion process is defined in clause 9.4.1.1, the MEP OAM on-demand extraction process in clause 9.4.1.2, the MIP OAM extraction process in clause 9.4.2.1, and the MIP OAM insertion process in clause 9.4.2.2. In summary, they insert and extract ETH_CI OAM signals into and from the stream of ETH_CI_D traffic units and the complementing P and D signals going through an MEP and MIP; the extraction is based on MEL and OpCode. Furthermore, the insertion process inserts the correct MEL and SA values into the OAM traffic units.

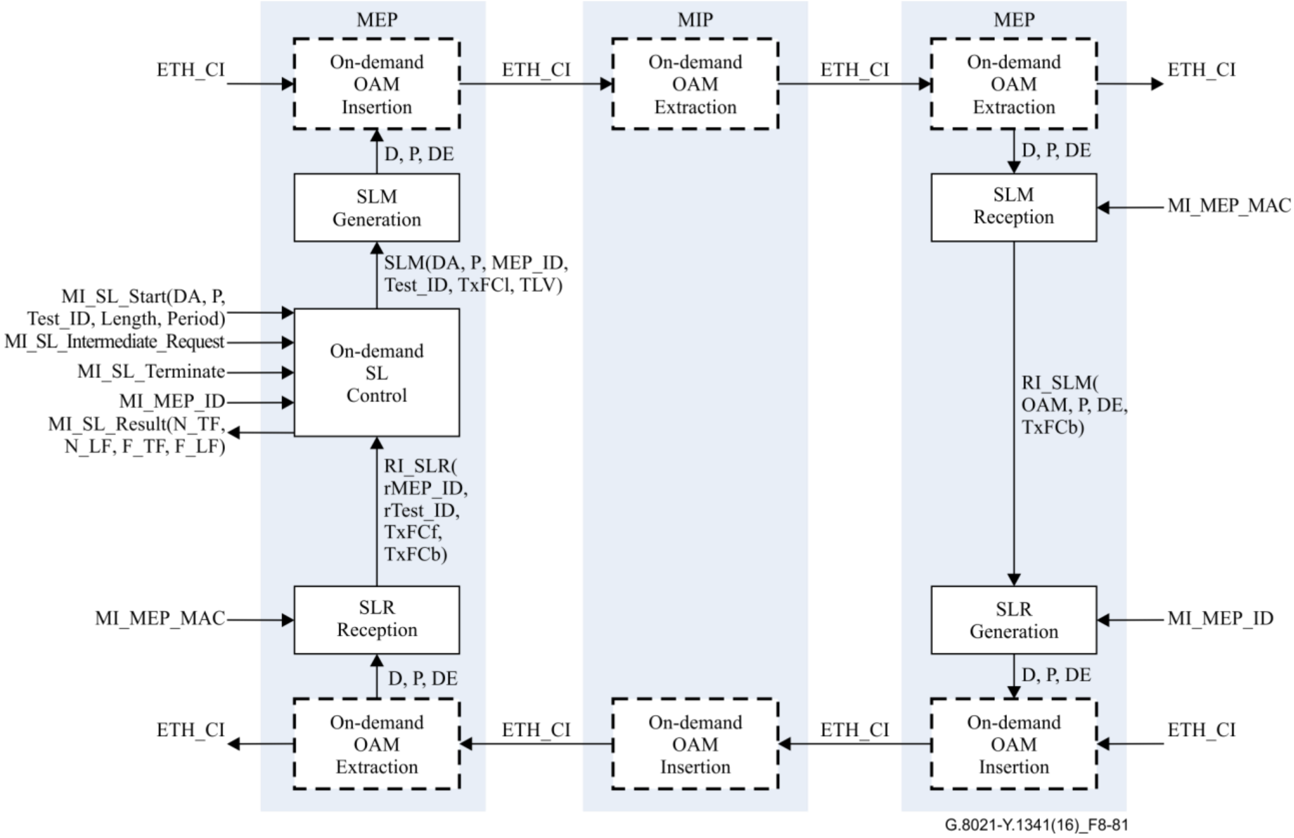


Figure 8-81 – Overview of processes involved with an on-demand single-ended synthetic loss measurement protocol

The SL protocol is controlled by the on-demand SL control process.

The on-demand SL control process is activated upon receipt of the MI_SL_Start(DA,P,Test_ID,Length,Period) signal and remains activated until the MI_SL_Terminate signal is received. The measured synthetic loss values are output via the MI_SL_Result(N_TF,N_LF,F_TF,F_LF) signal when the process is terminated by the MI_SL_Terminate signal or when an intermediate result is requested via the MI_SL_Intermediate_Request signal.

The SLM generation process generates SLM traffic units that pass through MIPs transparently, but are received and processed by SLM reception processes in MEPs. The SLR generation process may generate an SLR traffic unit in response. This SLR traffic unit also passes transparently through MIPs, but is received and processed by SLR reception processes in MEPs.

Figure 8-82 shows the different processes inside MEPs and MIPs that are involved in the proactive single-ended synthetic loss measurement protocol.

The MEP proactive OAM insertion process is defined in clause 9.2.1.1, the MEP OAM proactive extraction process in clause 9.2.1.2, the MIP OAM extraction process in clause 9.4.2.1, and the MIP OAM insertion process in clause 9.4.2.2. In summary, they insert and extract ETH_CI OAM signals into and from the stream of ETH_CI_D traffic units and the complementing P and D signals going through an MEP and MIP; the extraction is based on MEL and OpCode. Furthermore, the insertion process inserts the correct MEL and SA values into the OAM traffic units.

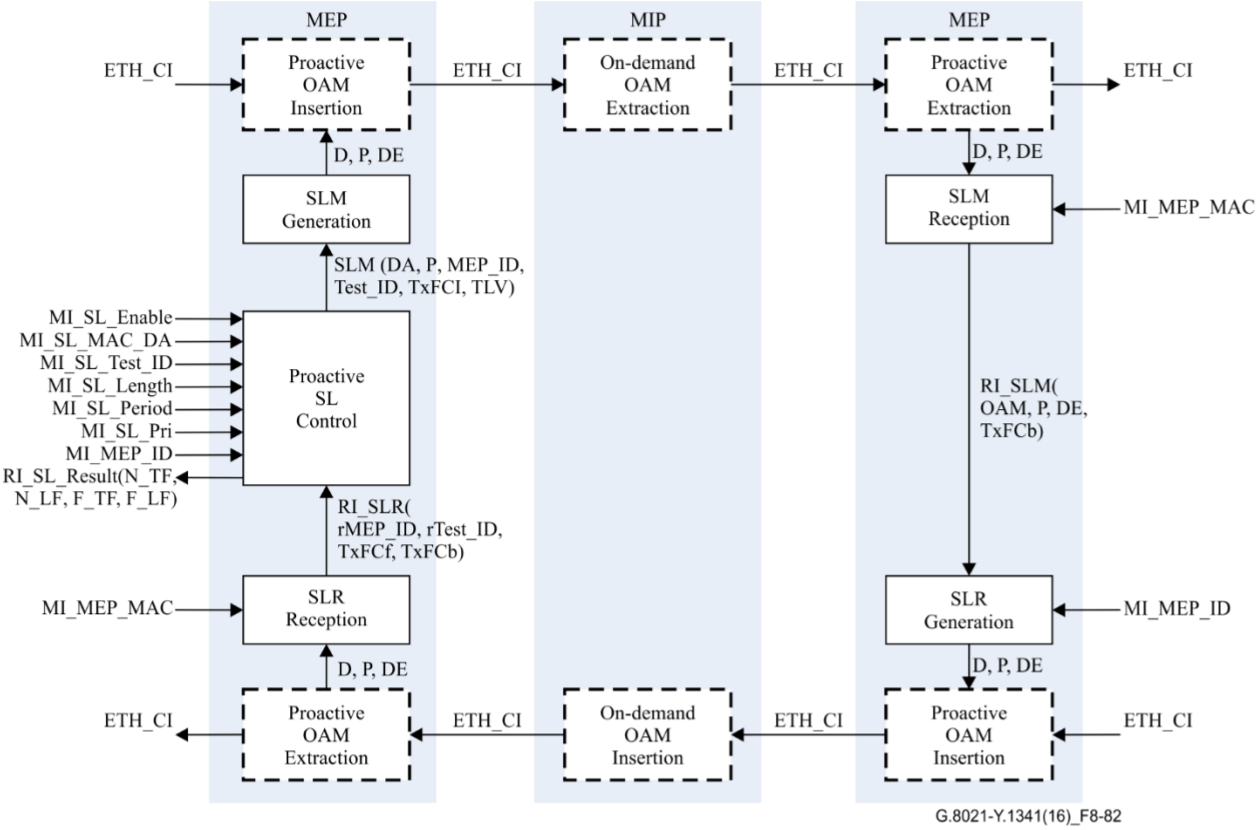


Figure 8-82 – Overview of processes involved with a proactive single-ended synthetic loss measurement protocol

The SL protocol is controlled by the proactive SL control processes.

The proactive SL control process is activated upon receipt of the MI_SL_Enable signal and remains activated until the signal is deactivated. The measured results are output every 1s using the RI_SL_Result (N_TF, N_LF, F_TF, F_LF) signal.

8.1.14.2 SL control process

The behaviour of the on-demand SL control process is defined in Figure 8-83. There are multiple instances of the on-demand SL control process, each handling an independent stream of SLM frames.

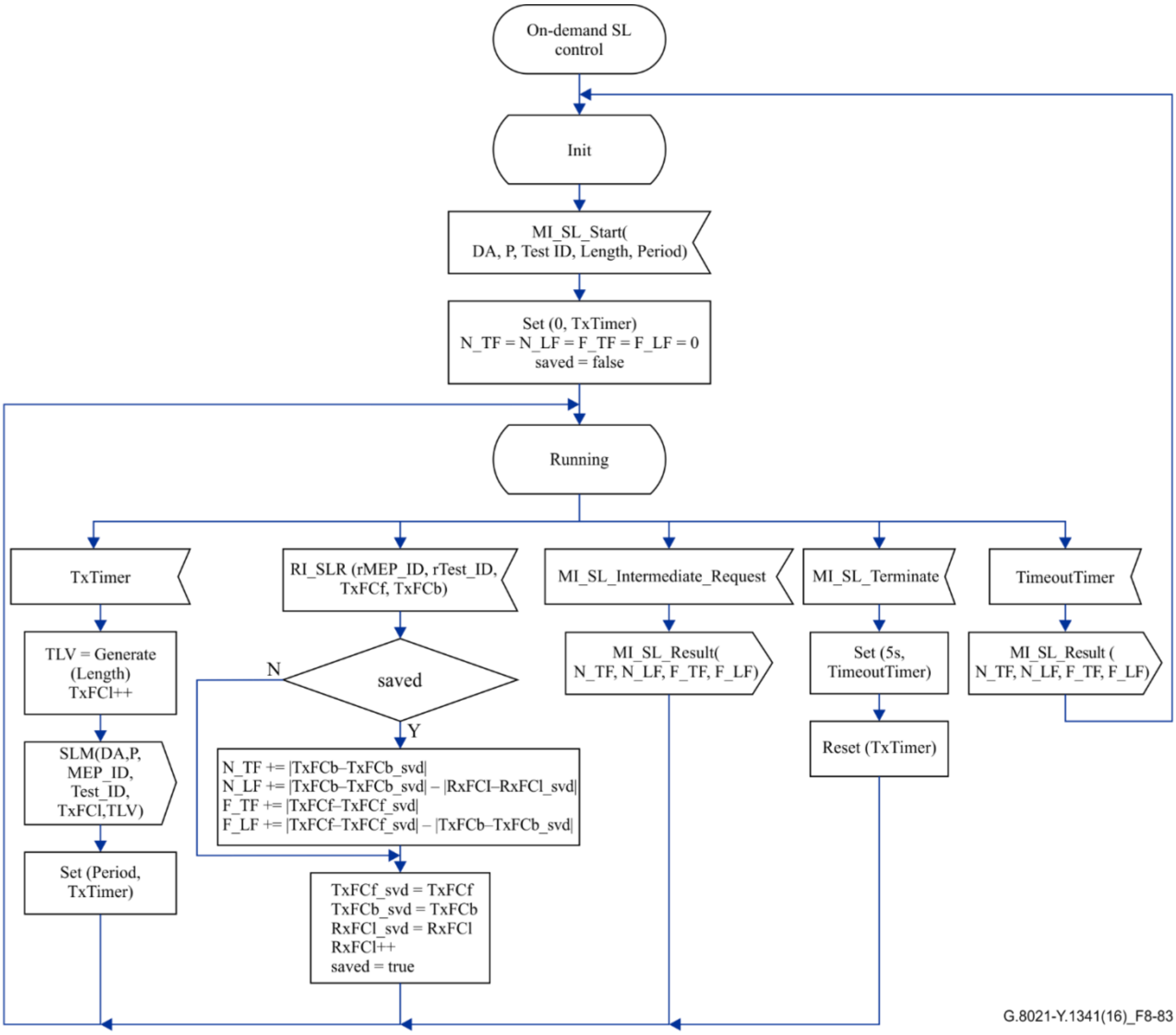


Figure 8-83 – On-demand SL control behaviour

Upon receipt of the MI_SL_Start(DA,P,Test_ID,Length,Period), the SL protocol is started. Every designated period the generation of an SLM frame is triggered (using the SLM(DA,P,MEP_ID,Test_ID,TxFCl,TLV) signal), until the MI_SL_Terminate signal is received. The MEP_ID is the MI_MEP_ID of the MEP itself. The TLV field of the SLM frames is determined by the Generate(Length) function. Generate(Length) generates a data TLV with length "Length" of an arbitrary bit pattern, as described in clause 8.1.8.2. If the length is 0, the TLV is set to NULL.

Upon receipt of an SLR traffic unit, the received counter values are used to count the near-end and far-end transmitted and lost synthetic frames. This result is reported using the MI_SL_Result(N_TF,N_LF,F_TF,F_LF) signal after the receipt of the MI_SL_Terminate signal or of the MI_SL_Intermediate_Request signal.

The behaviour of the proactive SL Control process is defined in Figure 8-84. There are multiple instances of the proactive SL Control process, each handling an independent stream of SLM frames.

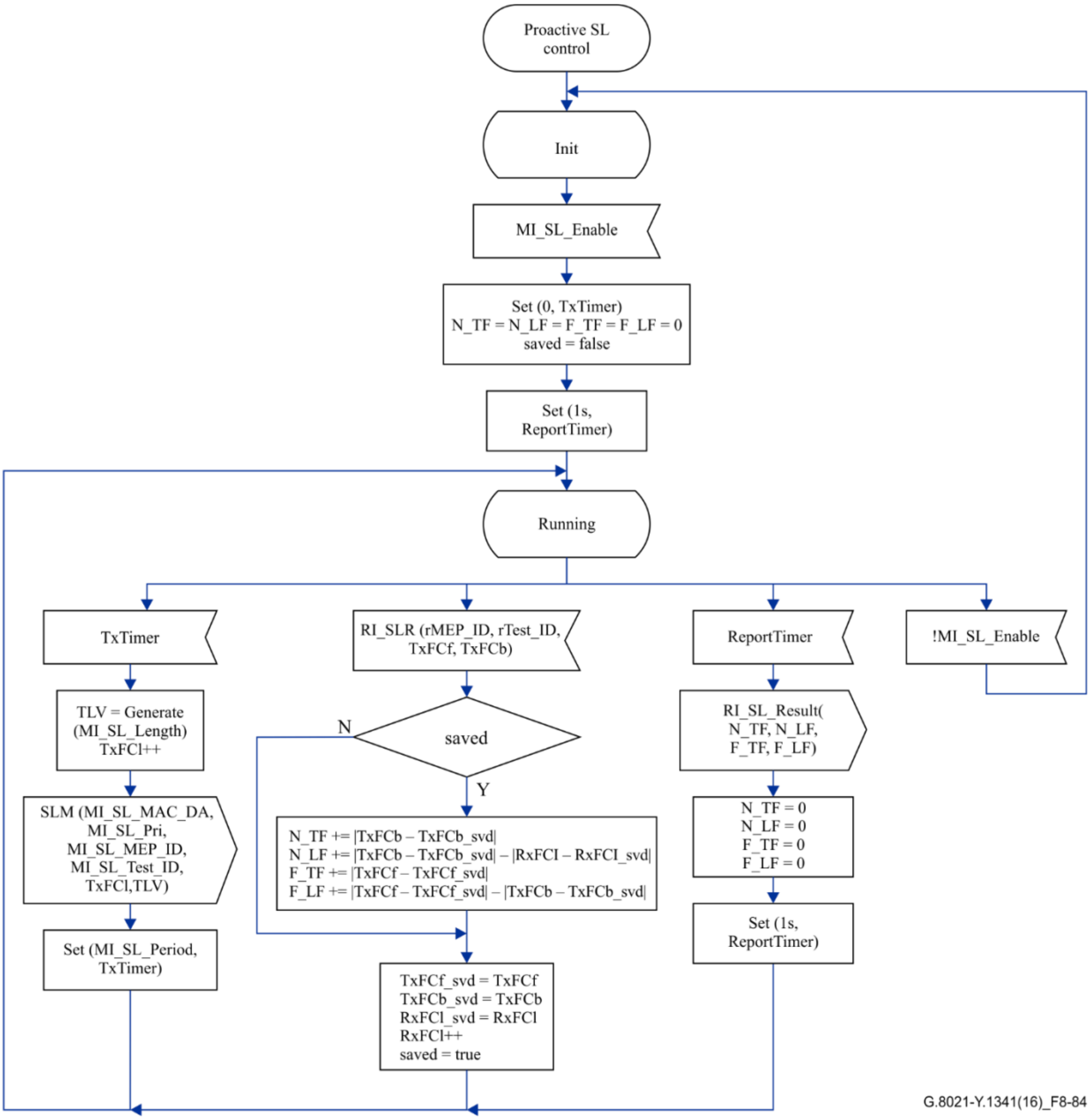


Figure 8-84 – Proactive SL control behaviour

Upon receipt of the MI_SL_Enable, the SL protocol is started. Every designated MI_SL_Period the generation of an SLM frame is triggered (using the SLM(MI_SL_MAC_DA,MI_SL_Pri,MI_MEP_ID,MI_SL_Test_ID,TxFCl,TLV) signal). The TLV field of the SLM frames is determined by the Generate(MI_SL_Length) function. Generate(MI_SL_Length) generates a data TLV with MI_SL_ Length of an arbitrary bit pattern, as described in clause 8.1.8.2. If the MI_SL_Length is 0, the TLV is set to NULL.

Upon receipt of an SLR traffic unit, the received counter values are used to count the near-end and far-end transmitted and lost synthetic frames. The calculation is performed every 1s and the RI_SL_Result(N_TF, N_LF, F_TF, N_LF) signal is generated.

8.1.14.3 SLM generation process

The behaviour of the SLM generation process is defined in Figure 8-85.

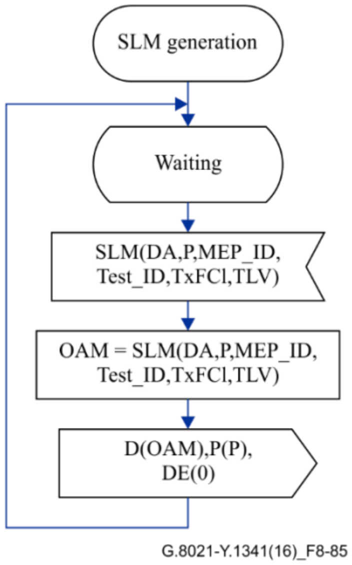


Figure 8-85 – SLM generation behaviour

Upon receiving the SLM(DA,P,MEP_ID,Test_ID,TxFCl,TLV), a single SLM traffic unit is generated together with the complementing P and DE signals. The DA, Source_MEP_ID, Test_ID and TxFCf of the generated traffic unit are determined by the DA, MEP_ID, Test_ID and TxFCl respectively in the SLM(DA,P,MEP_ID,Test_ID,TxFCl,TLV) signal. If not NULL, the specified TLV is appended to the traffic unit as shown in Figure 8-86.

The P signal value is defined by SLM(P). The DE signal is set to 0.

<table>
<tr><td></td><td colspan="8">1</td><td colspan="8">2</td><td colspan="8">3</td><td colspan="8">4</td></tr>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td colspan="32">DA=SLM(DA)</td></tr>
<tr><td>5</td><td colspan="16"></td><td colspan="16">SA=Undefined</td></tr>
<tr><td>9</td><td colspan="32"></td></tr>
<tr><td>13</td><td colspan="16">EtherType=89-02</td><td colspan="3">MEL= Undef</td><td colspan="5">Version=0</td><td colspan="8">OpCode=55 (SLM)</td></tr>
<tr><td>17</td><td colspan="8">Flags=0</td><td colspan="8">TLV Offset = 16</td><td colspan="16">Source_MEP_ID = SLM(MI_MEP_ID)</td></tr>
<tr><td>21</td><td colspan="16">0 (Reserved for Responder_MEP_ID)</td><td colspan="16">Test_ID = SLM(Test_ID)</td></tr>
<tr><td>25</td><td colspan="16">Test_ID Continued</td><td colspan="16">TxFCf = SLM(TxFCl)</td></tr>
<tr><td>29</td><td colspan="16">TxFCf Continued</td><td colspan="16">Reserved for TxFCb</td></tr>
<tr><td>33</td><td colspan="16">Reserved Continued</td><td colspan="16">TLV = SLM(TLV) if exists</td></tr>
<tr><td>37</td><td colspan="32"></td></tr>
<tr><td>41</td><td colspan="32"></td></tr>
<tr><td>45</td><td colspan="32"></td></tr>
<tr><td>:</td><td colspan="32"></td></tr>
<tr><td>last</td><td colspan="24"></td><td colspan="8">End TLV (0)</td></tr>
</table>

Figure 8-86 – SLM traffic unit

8.1.14.4 SLM reception process

The SLM reception process processes the received SLM traffic units and the complementing P and DE signals. The behaviour is defined in Figure 8-87.

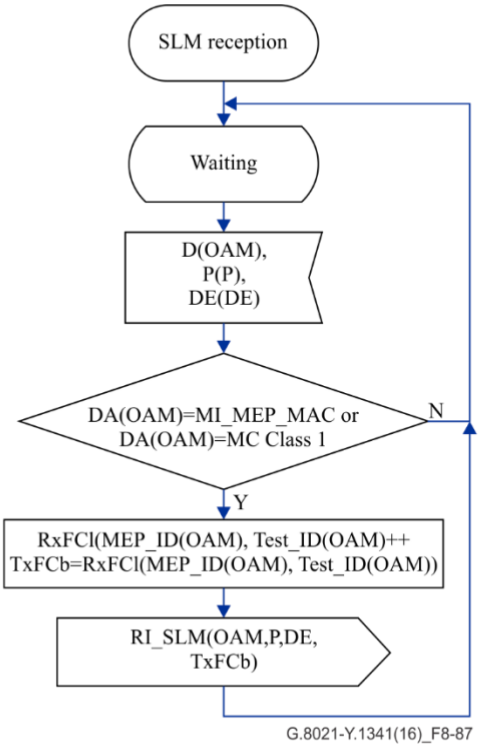


Figure 8-87 – SLM reception behaviour

First the DA is checked, it should be the local MAC address or a multicast class 1 address, otherwise the frame is ignored.

If the DA is the local MAC or a multicast class 1 address, the MEP_ID and the Test_ID fields are extracted from the traffic unit. The local received counter RxFCl maintained per MEP_ID and Test_ID values, is incremented. The received OAM information, P and DE signals, as well as the local TxFCb value are forwarded as remote information to the SLR generation process using the RI_SLM(OAM,P,DE, TxFCb) signal.

NOTE – The SLM reception process allocates and maintains local resources for the counter RxFCl per MEP_ID and Test_ID. To facilitate the automatic release of local resources, a timer for monitoring no receipt of SLM can be utilized. The SLM reception process must ensure that there is no discontinuity in RxFCl for a given MEP ID and Test ID for a given interval (e.g., 5 minutes) after the last received SLM for that MEP ID and Test ID. A detailed mechanism for the release is out of the scope of this Recommendation.

8.1.14.5 SLR generation process

The SLR generation process generates an SLR traffic unit and its complementing P and DE signals. The behaviour is defined in Figure 8-88.

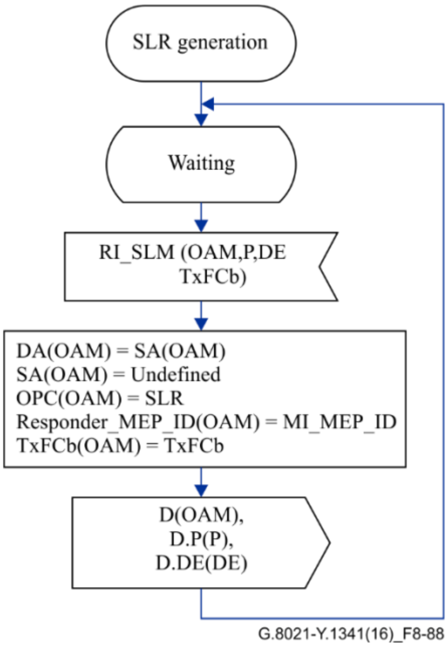


Figure 8-88 – SLR generation behaviour

Upon receipt of the RI_SLM (OAM,P,DE,TxFCb) signal containing an SLM traffic unit, the SLR generation process generates an SLR traffic unit and forwards it to the MEP OAM insertion process.

As part of the SLR generation:

– the DA of the SLR traffic unit is the SA of the original SLM traffic unit

– the OpCode is changed into SLR OpCode

– the responder MEP_ID is set to MI_MEP_ID

– TxFCb field is assigned the TxFCb value passed in the SLR(TxFCb)

– the other fields and optional TLVs are copied from the SLM.

The resulting SLR traffic unit is shown in Figure 8-89.

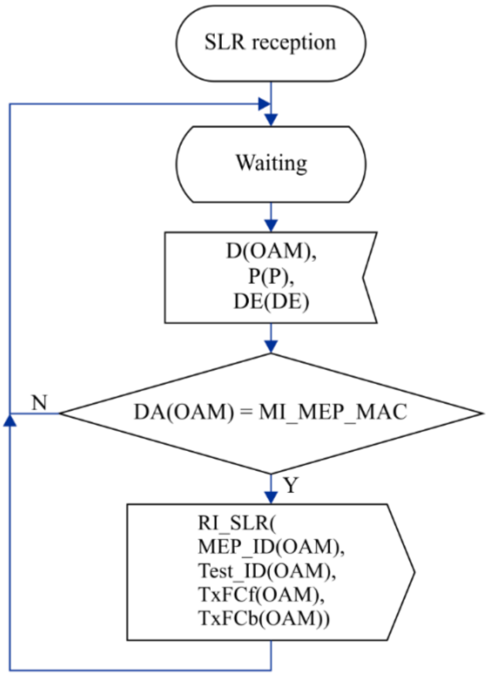
NOTE – In the generated SLR, in the OAM (MEP) insertion process, the SA will be overwritten with the local MAC address, and the MEL will be overwritten with MI_MEL.

	1	2	3	4
	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1
1	DA=SA(RI_SLM (OAM))			
5			SA=Undefined	
9				
13	EtherType=89-02		MEL= Undef Version=0	OpCode=54(SLR)
17	Flags=Flags (RI_SLM(OAM))	TLV Offset = TLV Offset((RI_SLM(OAM))	Source_MEP_ID = Source_MEP_ID((RI_SLM(OAM))	
21	Responder_MEP_ID = MI_MEP_ID		Test_ID = Test_ID((RI_SLM(OAM))	
25	Test_ID Continued		TxFCf = TxFCf((RI_SLM(OAM))	
29	TxFCf Continued		TxFCb = (RI_SLM(TxFCb))	
33	TxFCb Continued		TLV = TLV((RI_SLM(OAM)) if exists	
37				
41				
45				
:				
last				End TLV = End TLV(RI_SLM(OAM))

Figure 8-89 – SLR traffic unit

8.1.14.6 SLR reception process

The SLR reception process processes the received SLR traffic units and the complementing P and DE signals. The behaviour is defined in Figure 8-90.



G.8021-Y.1341(16)_F8-90

Figure 8-90 – SLR reception behaviour

Upon receipt of an SLR traffic unit, the DA field of the traffic unit is checked. If the DA field equals the local MAC address, the SLR traffic unit is processed further, otherwise it is ignored.

If the SLR traffic unit is processed, Test_ID, TxFCf, TxFCb, responder MEP_ID, are extracted from the traffic unit and signalled, using the RI_SLR(MEP_ID, Test_ID,TxFCf,TxFCb) signal.

8.1.15 Dual-ended synthetic loss measurement (1SL) processes

8.1.15.1 Overview

Figure 8-91 shows the different processes inside MEPs and MIPs that are involved in the on-demand dual-ended synthetic loss measurement protocol.

NOTE ‑ In previous versions of this Recommendation, dual-ended synthetic loss measurement was known as one-way synthetic loss measurement. With regard to those definitions, refer to [ITU-T G.8001].

The MEP on-demand OAM source insertion process is defined in clause 9.4.1.1, the MEP on-demand OAM sink extraction process in clause 9.4.1.2, the MIP on-demand OAM sink extraction process in clause 9.4.2.2. In summary, they insert and extract ETH_CI OAM signals into and from the stream of ETH_CI_D traffic units and the complementing P and DE signals going through an MEP and MIP; the extraction is based on MEL and OpCode. Furthermore, the insertion process inserts the correct MEL and SA values into the OAM traffic units.

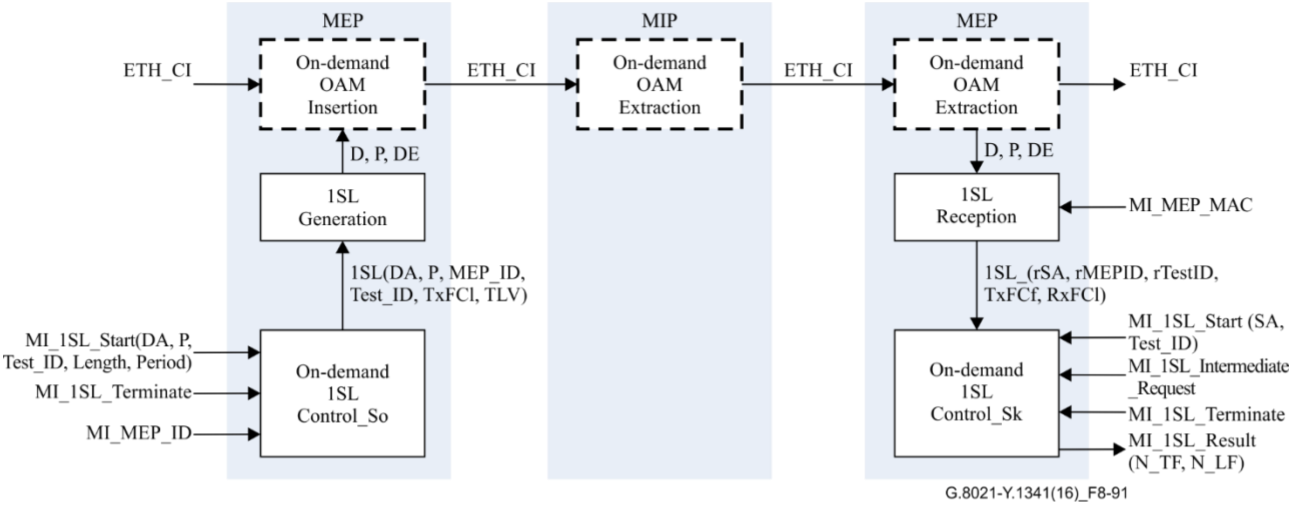


Figure 8-91 – Overview of processes involved with on-demand dual-ended synthetic loss measurement

The on-demand 1SL protocol is controlled by the on-demand 1SL Control_So and 1SL Control_Sk processes. The on-demand 1SL Control_So process triggers the generation of 1SL traffic units upon receipt of an MI_1SL_Start(DA,P, Test_ID,Length,Period) signal. The on-demand 1SL Control_Sk process processes the information from received 1SL traffic units after receiving the MI_1SL_Start(SA,Test_ID) signal. The result is communicated by the sink MEP when the process is terminated by the MI_1SL_Terminate signal or when an intermediate result is requested via the MI_1SL_Intermediate_Request signal.

The 1SL generation process generates 1SL messages that pass transparently through MIPs and are received and processed by the 1SL reception process in MEPs.

Figure 8-92 shows the different processes inside MEPs and MIPs that are involved in the proactive dual-ended synthetic loss measurement protocol.

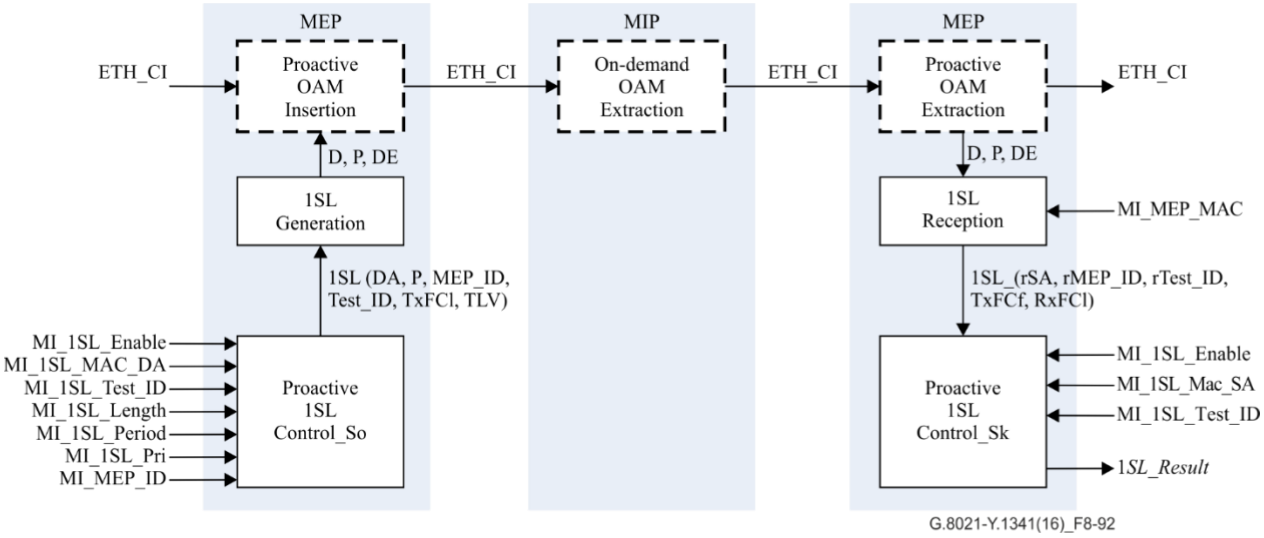


Figure 8-92 – Overview of processes involved with proactive dual-ended synthetic loss measurement

The MEP proactive-OAM source insertion process is defined in clause 9.2.1.1, the MEP proactive OAM sink extraction process in clause 9.2.1.2, and the MIP on-demand OAM sink extraction process in clause 9.2.2.2.

The proactive 1SL protocol is controlled by the proactive 1SL Control_So and 1SL Control_Sk processes. The proactive 1SL Control_So process triggers the generation of 1SL traffic units if MI_1SL_Enable signal is set. The 1SL frames are generated with a periodicity determined by MI_1SL_Period and with a priority determined by MI_1SL_Pri. The result is reported every one second by the 1SL Control_Sk process.

8.1.15.2 1SL Control_So process

Figure 8-93 shows the behaviour of the on-demand 1SL Control_So process. Upon receipt of the MI_1SL_Start(DA,P,Test_ID, Length, Period) signal the 1SL protocol is started. The protocol will run until the receipt of the MI_1SL_Terminate signal.

If the 1SL protocol is running, every period (as specified in the MI_1SL_Start signal) the generation of a 1SL message is triggered by generating the 1SL(DA,P,MEP_ID,Test_ID,TxFCl,TLV) signal towards the 1SL generation process. The MEP_ID is the MI_MEP_ID of the MEP itself. The TLV is determined by the Generate(Length) function. Generate(Length) generates a data TLV with length "Length" of an arbitrary bit pattern, as described in clause 8.1.8.2. If the length is 0, the TLV is set to NULL.

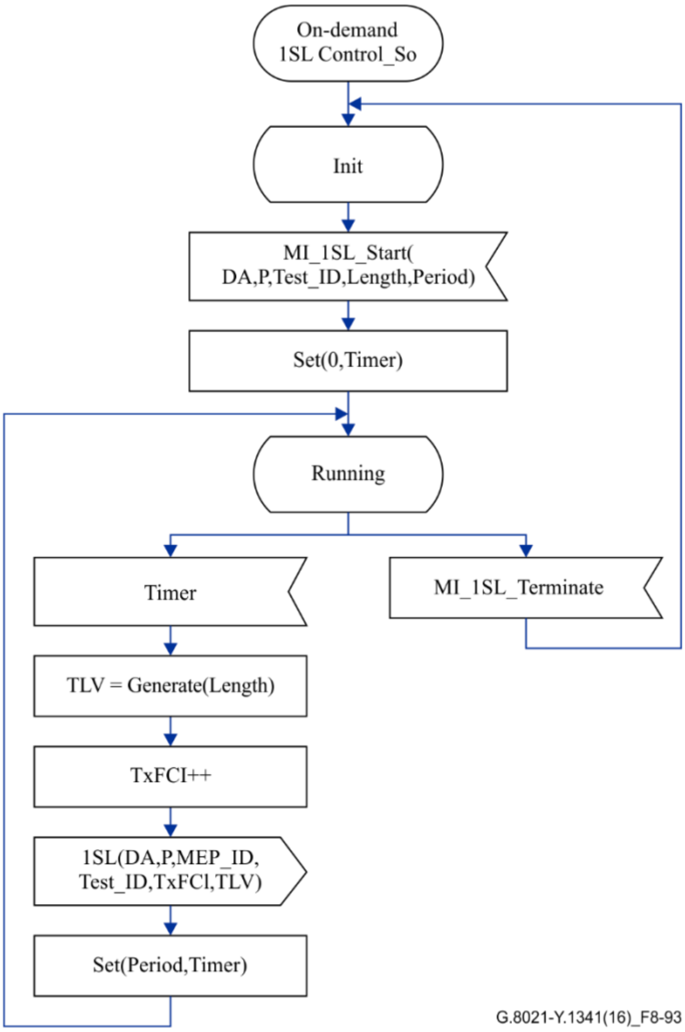


Figure 8-93 – On-demand 1SL Control_So behaviour

The behaviour of the proactive 1SL control process is defined in Figure 8-94.

If the MI_1SL_Enable is asserted, the process starts to generate 1SL frames (using the 1SL (MI_1SL_MAC_DA, MI_1SL_Pri, MI_MEP_ID, MI_1SL_Test_ID, TxFCl, TLV) signal.

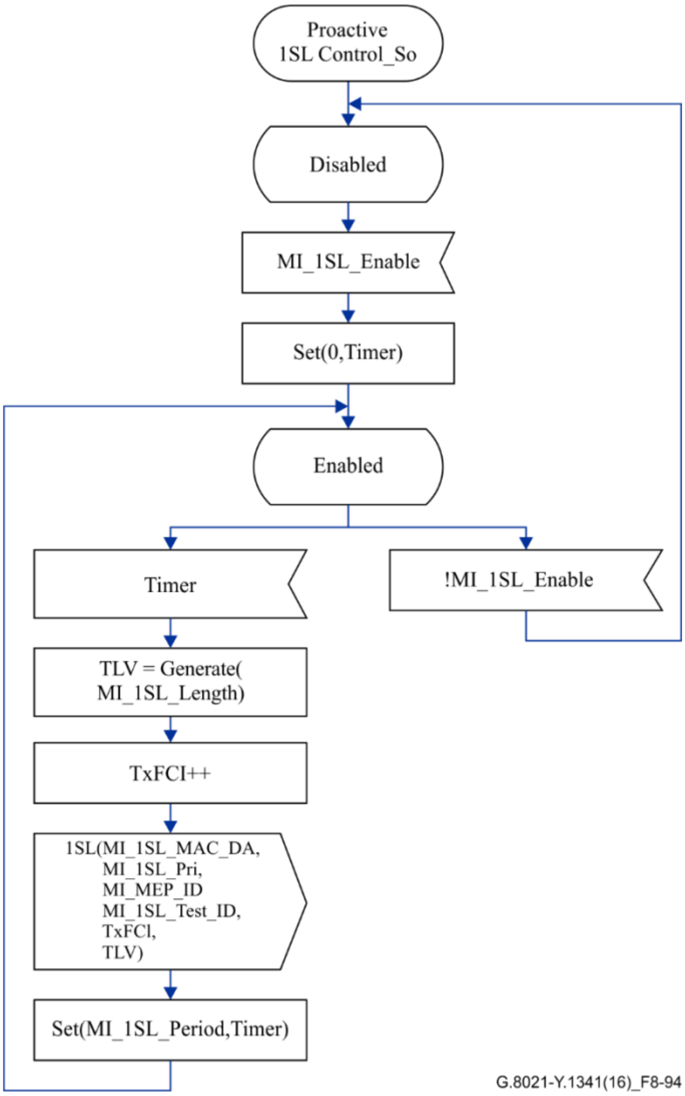


Figure 8-94 – Proactive 1SL Control_So behaviour

8.1.15.3 1SL generation process

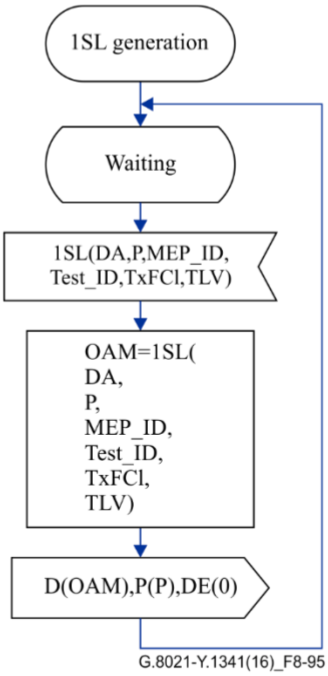


Figure 8-95 – 1SL generation behaviour

Figure 8-95 shows the 1SL generation process. Upon receiving the 1SL(DA, P, MEP_ID, Test_ID, TxFCl, TLV) signal, a single 1SL traffic unit is generated, along with the complementing P and DE signals.

The DA, source_MEP_ID, Test_ID and TxFCl of the generated traffic unit are determined by the DA, MEP_ID, Test_ID and TxFCl respectively in the 1SL(DA,P,MEP_ID,Test_ID,TxFCl,TLV) signal. If not NULL, the specified TLV is appended to the traffic unit as shown.

The value of the P signal is determined by the 1SL(P) signal. The DE signal is set to 0.

The resulting traffic unit is shown in Figure 8-96.

NOTE – In the generated 1SL traffic unit, in the OAM (MEP) insertion process, the SA will be assigned the local MAC address, and the MEL will be assigned by MI_MEL.

	1	2	3	4
	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1
1	DA=1SL(DA)			
5			SA=Undefined	
9				
13	EtherType=89-02		MEL= Undef Version=0	OpCode=53 (1SL)
17	Flags=0	TLV Offset = 16	Source_MEP_ID = 1SL(MI_MEP_ID)	
21	0 (not used)		Test_ID = 1SL(Test_ID)	
25	Test_ID Continued		TxFCf = 1SL(TxFCl)	
29	TxFCf Continued		0 (Reserved)	
33	0 (Reserved)		TLV = 1SL(TLV) if exists	
37				
41				
45				
:				
last				End TLV (0)

Figure 8-96 – 1SL traffic unit

8.1.15.4 1SL reception process

The 1SL reception process processes the received 1SL traffic units and the complementing P and DE signals. The behaviour is defined in Figure 8-97.

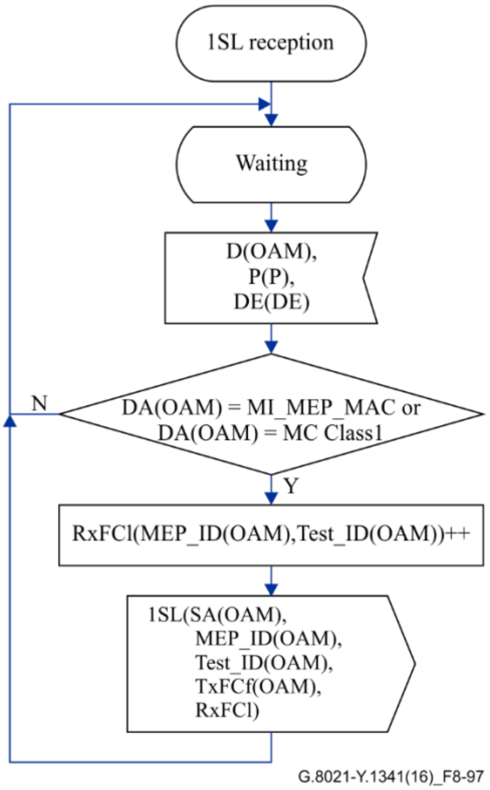


Figure 8-97 – 1SL Reception behaviour

Upon receipt of a 1SL traffic unit, the DA field is checked. The 1SL traffic unit is processed if the DA is equal to the local MAC address or a multicast class 1 address and ignored otherwise.

If the 1SL traffic unit is processed, the SA, source_MEP_ID, Test_ID and TxFCf fields are extracted and the appropriate RxFCl counter is incremented. The values are forwarded to the 1SL Control_Sk Process using the 1SL(rSA, rMEP_ID, rTest_ID, TxFCf, RxFCl) signal.

8.1.15.5 1SL Control_Sk process

Figure 8-98 shows the behaviour of the on-demand 1SL Control_Sk process. The MI_1SL_Start(SA,Test_ID) signal starts the processing of 1SL messages coming from an MEP with SA as the MAC address. The protocol runs until the receipt of the MI_1SL_Terminate signal.

While running, the process processes the received 1SL(rSA, rMEP_ID, rTest_ID, TxFCf, RxFCl) information. First the rSA is compared with the SA from the MI_1SL_Start (SA,Test_ID) signal. If the rSA is not equal to this SA, the information is ignored. Next the rTest_ID is compared with the Test_ID from the MI_1SL_Start (SA,Test_ID) signal. If the Test_ID signal is configured and rTest_ID is available but both values are different, the information is ignored. Otherwise the loss from the single received 1SL traffic unit is calculated. This result is reported using the MI_1SL_Result(N_TF, N_LF) signal after receiving the MI_1SL_Terminate signal or of the MI_1SL_Intermediate_Request signal.

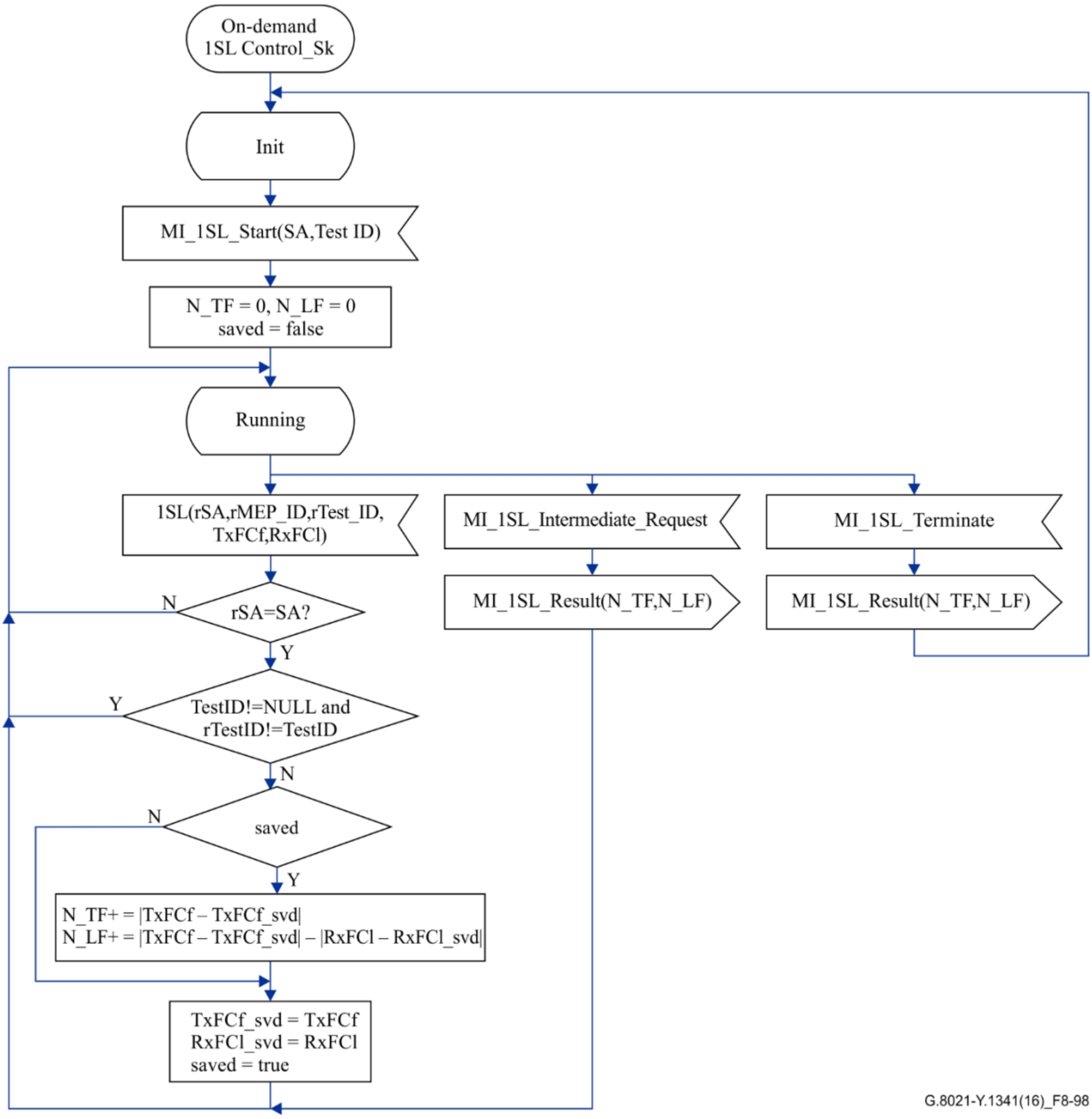


Figure 8-98 – On-demand 1SL Control_Sk process

The behaviour of the proactive 1SL Control_Sk process is defined in Figure 8-99. If the MI_1SL_Enable is asserted, the result (N_TF, N_LF) is reported every one second.

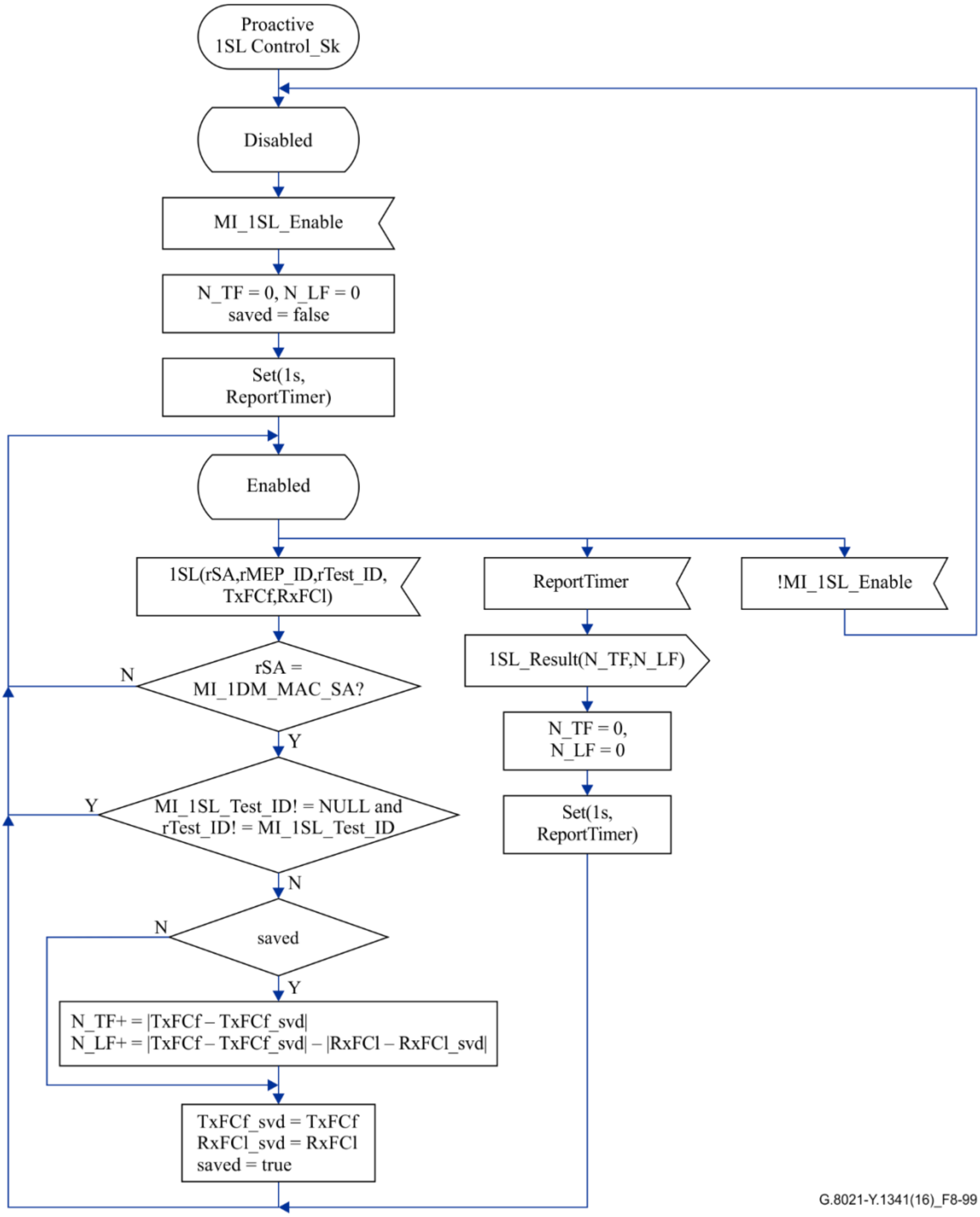


Figure 8-99 – Proactive 1SL Control_Sk process

8.1.16 CSF insert process

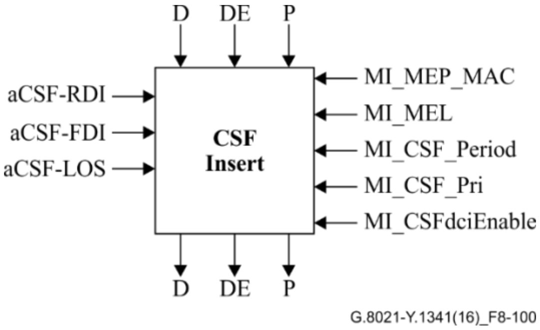


Figure 8-100 – CSF insert process

Figure 8-100 shows the CSF insert process symbol and Figure 8-101 defines the behaviour. If any of the aCSF-RDI, aCSF-FDI or aCSF-LOS signals are true, the CSF insert process continuously generates ETH_CI traffic units where the ETH_CI_D signal contains the CSFtraffic unit until the condition no longer holds, ie all of aCSF-RDI, aCSF-FDI and aCSF-LOS are false. At this point, CSF traffic unit(s) with DCI (Defect Clear Information) are generated indicating that the defect has been cleared, if MI_CSFdciEnable = True.

NOTE 1 – Figure 8-101 shows a case where a single CSF traffic unit with DCI is generated. However, the detail transmission condition (e.g., transmission period, the number of traffic unit) is out of scope of this Recommendation.

The generated CSF traffic units are inserted in the incoming stream, i.e., the output stream contains the incoming traffic units and the generated CSF traffic units.

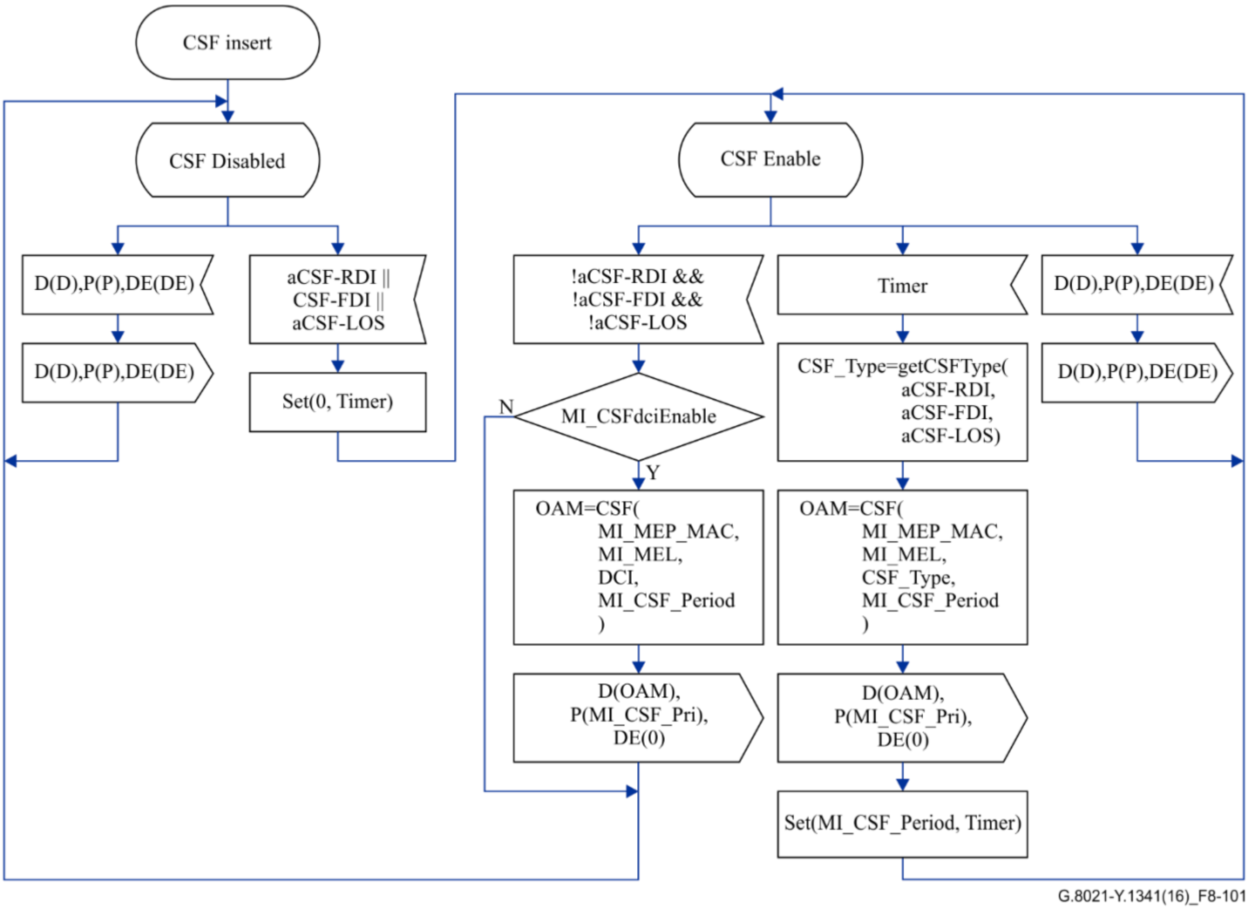


Figure 8-101 – CSF insert behaviour

If exactly one of aCSF-RDI, aCSF-FDI and aCSF-LOS is set, the getCSFType() function returns RDI, FDI or LOS as appropriate. The behaviour of getCSFType() when more than one of the conditions are set is for further study.

NOTE 2 – As described in [ITU-T Y.1731], triggering CSF is client and application specific. Ideally all clients and applications should ensure that at most one of the conditions is set at any given time.

The period between consecutive CSF traffic units is determined by the MI_CSF_Period parameter. Allowed values are once per second and once per minute; the encoding of these values is defined in Table 8-4. Note that these encoding are the same as for the LCK/AIS generation process.

Table 8-4 – CSF period values

3-bits	Period value	Comments
000	Invalid value	Invalid value for CSF PDUs
001	FFS	FFS
010	FFS	FFS
011	FFS	FFS
100	1s	1 frame per second
101	FFS	FFS
110	1 min	1 frame per minute
111	FFS	FFS

The ETH_CI_D signal contains a source and destination address field and an M_SDU field. The format of the M_SDU field for CSF traffic units is defined in clauses 9.1 and 9.21 of [ITU-T G.8013]. The MEL in the M_SDU field is determined by the MI_ MEL input parameter.

The values of the source and destination address fields in the ETH_CI_D signal are determined by the local MAC address (SA) and the multicast class 1 DA as described in [ITU-T G.8013] (DA). The value of the multicast class 1 DA is 01-80-C2-00-00-3x, where x is equal to MI_MEL as defined in clause 10.1 of [ITU-T G.8013]. The value of MI_MEP_MAC should be a valid unicast MAC address.

The CSF_Type is encoded in the three bits of the flags field in the CSF PDU using the values from Table 8-5.

Table 8-5 – CSF type values

Value	Type	Comments
000	LOS	Client loss of signal
001	FDI/AIS	Client forward defect indication
010	RDI	Client reverse defect indication
011	DCI	Client defect clear indication

The periodicity (as defined by MI_CSF_Period) is encoded in the three least significant bits of the flags field in the CSF PDU using the values from Table 8-4.

The CSF (SA, MEL, type, period) function generates a CSF traffic unit with the SA, MEL, type and period fields defined by the values of the parameters. Figure 8-102 below shows the ETH_CI_D signal format resulting from the function call from Figure 8-101:

OAM=CSF(
MI_MEP_MAC,
MI_MEL,
CSF_Type,
MI_CSF_Period
)

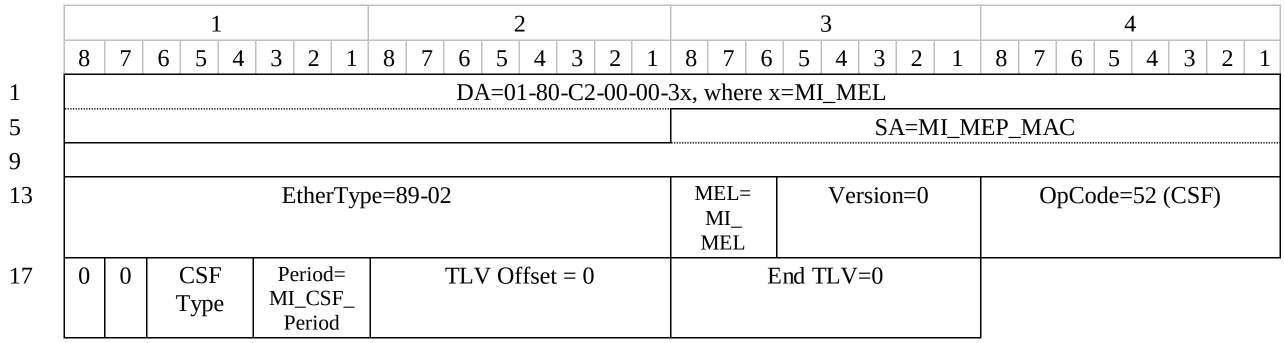


Figure 8-102 – CSF traffic unit

8.1.17 CSF extract process

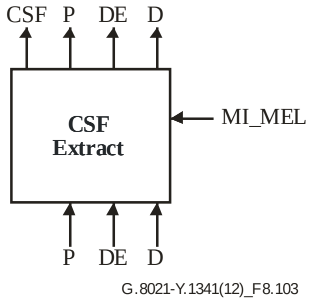


Figure 8-103 – CSF extract process

The CSF extract process extracts ETH_CI_CSF signals from the incoming stream of ETH_CI traffic units. ETH_CI_CSF signals are only extracted if they belong to the MEL as defined by the MI_MEL input parameter.

If an incoming traffic unit is a CSF traffic unit belonging to the MEL defined by MI_MEL, the ETH_CI_CSF signal will be extracted from this traffic unit and the traffic unit will be filtered. The ETH_CI_CSF is the CSF specific information contained in the received traffic unit. All other traffic units will be transparently forwarded. The encoding of the ETH_CI_D signal for CSF frames is defined in clause 9.12 of [ITU-T G.8013].

The criteria for filtering are based on the values of the fields within the M_SDU field of the ETH_CI_D signal:

- length/type field equals the OAM EtherType (89-02)
- MEL field equals MI_MEL
- OAM type equals CSF (52), as defined in clause 9.12 of [ITU-T G.8013].

This is defined in Figure 8-103. The function CSF(D) extracts the CSF specific information from the received traffic unit.

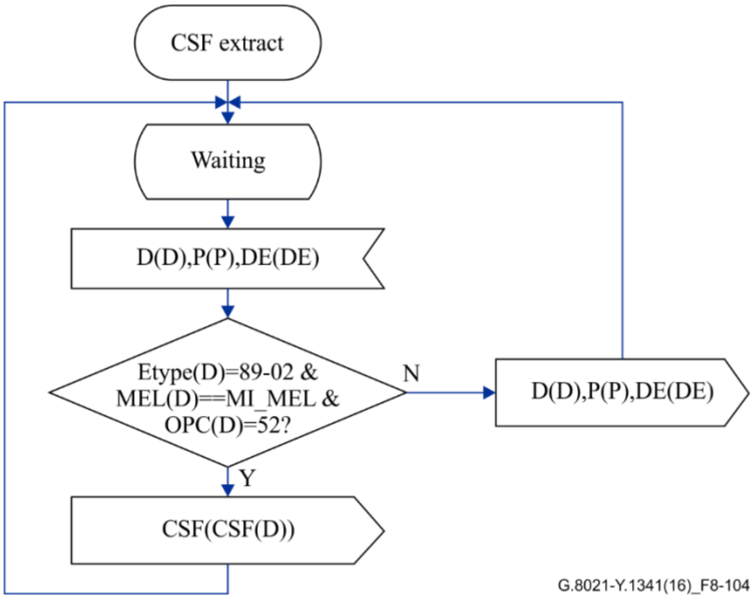


Figure 8-104 – CSF extract behaviour

8.1.18 BNM insert process

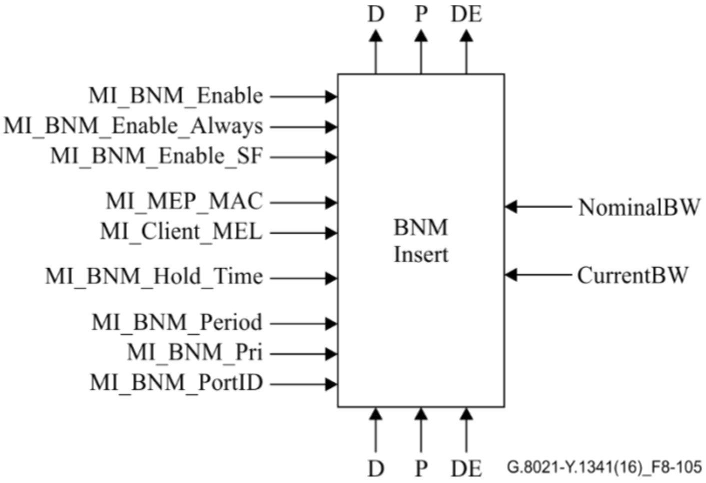


Figure 8-105 – BNM insert process

Figure 8-105 shows the BNM symbol and Figure 8-106 defines the behaviour. The NominalBW and CurrentBW are continuously signalled from the server layer, and contain respectively the nominal full transmission bandwidth of the link at the server layer, and the current available transmission bandwidth.

NOTE 1 – The NominalBW and CurrentBW are generated by adaptation functions of some specific server layer technology such as microwave links.

When MI_BNM_Enable is set, the BNM insert process monitors the current and nominal transmission bandwidths, and when the current transmission bandwidth falls below the nominal bandwidth for a given hold time, it generates ETH_CI traffic units where the ETH_CI_D signal contains a BNM traffic unit. If MI_BNM_Enable_Always is set, ETH_CI traffic units where the ETH_CI_D signal contains a BNM traffic unit are also transmitted periodically when there is no degradation. If MI_BNM_Enable_SF is set, ETH_CI traffic units where the ETH_CI_D signal contains a BNM traffic unit are also transmitted periodically when the link fails in the transmit direction (i.e., when the current transmission bandwidth is 0).

When the current transmission bandwidth changes, MI_BNM_Hold_Time specifies the hold time before the first notification is sent. Allowed values are between 0 and 10s (in increments of 10ms). At the end of the hold time, a number of BNM notifications containing the new value are sent quickly (the exact number and period is implementation-specific) in order to increase the reliability of the notification.

NOTE 2 – BNM notifications are expected to be used where the server layer is a microwave link that uses adaptive bandwidth modulation. A hold time is used to prevent notifications if the degradation is very short, such as might be caused by an object passing through the line of sight of the microwave link. The applicability of BNM notifications to other technologies is for further study.

The traffic units are generated with the Source MAC specified by MI_MEP_MAC, the MEG level specified by MI_Client_MEL, and the priority specified by MI_BNM_Pri. During degradation or link failure, they are generated periodically at the period specified by MI_BNM_Period; allowed values are 1s, 10s and 1min. BNM_Fast_Period and BNM_Fast_Count in the Figure 8-106 are implementation specific parameters that allow sending a number of the first BNM frames more quickly. The value of BNM_Fast_Period must be less than or equal to MI_BNM_Period. If MI_BNM_PortID is set, the Port ID field is set to the value specified in MI_BNM_PortID. Otherwise the Port ID field is set to 0 to indicate that no Port ID was configured.

The generated BNM traffic units are inserted in the incoming stream, i.e., the output stream contains the incoming traffic units and the generated BNM traffic units.

The BNM insert process continues generating BNM traffic units until the current bandwidth is restored to the nominal bandwidth. At that point a number of the final BNM traffic units are generated with the current bandwidth set equal to the nominal bandwidth. If MI_BNM_Enabled_Always is not set, generation of BNM traffic units then ceases. Otherwise, BNM traffic units continue to be generated periodically at the period specified by MI_BNM_Period.

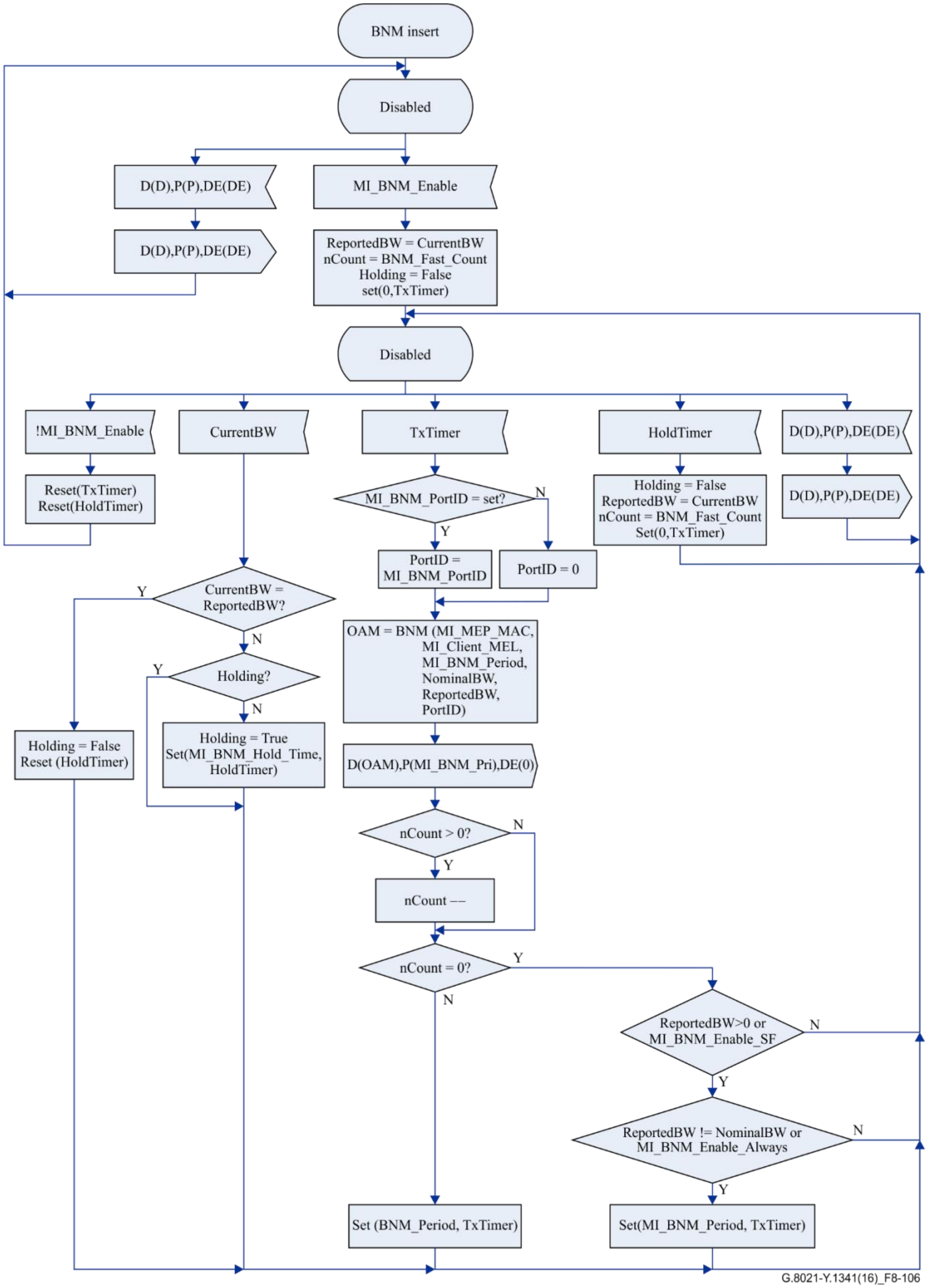


Figure 8-106 – BNM insert behaviour

To prevent very frequent changes in the notified bandwidth, server layer should avoid reporting consecutive changes of the CurrentBW within an implementation specific time: the filtering mechanism is implementation and server layer specific.

The BNM(SA, MEL, Period, NominalBW, CurrentBW, PortID) function generates an ETH_CI traffic unit containing a source and destination address field and an M_SDU field. The source address is set to the given SA, and the destination address is set to the multicast class 1 DA as described in [ITU-T G.8013]. The format of the M_SDU field for BNM traffic units is defined in [ITU-T G.8013]. The MEL, Period, Current Bandwidth, Nominal Bandwidth and Port ID fields are set to the given values. Figure 8-107 below shows the ETH_CI_D signal format resulting from the function call from Figure 8-106:

OAM=BNM(
MI_MEP_MAC,
MI_Client_MEL,
MI_BNM_Period,
NominalBW
ReportedBW,
PortID
)

<table>
<tr><th></th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th><th colspan="8">4</th></tr>
<tr><th></th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th></tr>
<tr><td>1</td><td colspan="32">DA=01-80-C2-00-00-3x, where x=MI_MEL</td></tr>
<tr><td>5</td><td colspan="16"></td><td colspan="16">SA=MI_MEP_MAC</td></tr>
<tr><td>9</td><td colspan="32"></td></tr>
<tr><td>13</td><td colspan="16">EtherType=89-02</td><td colspan="3">MEL=
MI_
MEL</td><td colspan="5">Version=0</td><td colspan="8">OpCode=32 (GNM)</td></tr>
<tr><td>17</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="3">Period=
MI_BNM
_Period</td><td colspan="8">TLV Offset = 13</td><td colspan="8">SubOpCode=1 (BNM)</td><td colspan="8">Nominal Bandwidth</td></tr>
<tr><td>21</td><td colspan="24">Nominal Bandwidth Continued</td><td colspan="8">Current Bandwidth</td></tr>
<tr><td>25</td><td colspan="24">Current Bandwidth Continued</td><td colspan="8">Port ID</td></tr>
<tr><td>29</td><td colspan="24">Port ID Continued</td><td colspan="8">End TLV (0)</td></tr>
</table>

Figure 8-107 – BNM traffic unit

NOTE 3 – The Period field in the generated BNM Traffic Unit is always set to MI_BNM_Period, even for the initial traffic units generated after the expiry of the hold time, which are transmitted at an implementation-specific faster period. This ensures the correct operation of the BNM extract process in the receiving MEP.

8.1.19 BNM extract process

The BNM extract process processes the received BNM traffic units and the complementing P and DE signals.

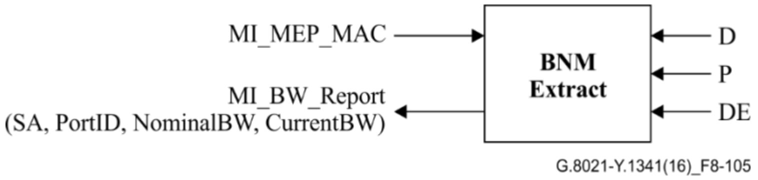


Figure 8-108 – BNM extract process

Figure 8-108 shows the BNM extract process symbol and Figure 8-109 defines the behaviour. When BNM traffic units are received, if the DA is equal to the MEP's MAC, or it is a multicast class 1 address, then the SA, Port ID, Current Bandwidth and Nominal Bandwidth are extracted from the traffic unit and, if different to the previous values, are passed to the Management System via the MI_BW_Report(SA, PortID, NominalBW, CurrentBW) signal.

NOTE 1 – Use of BNM for protection switching is for further study.

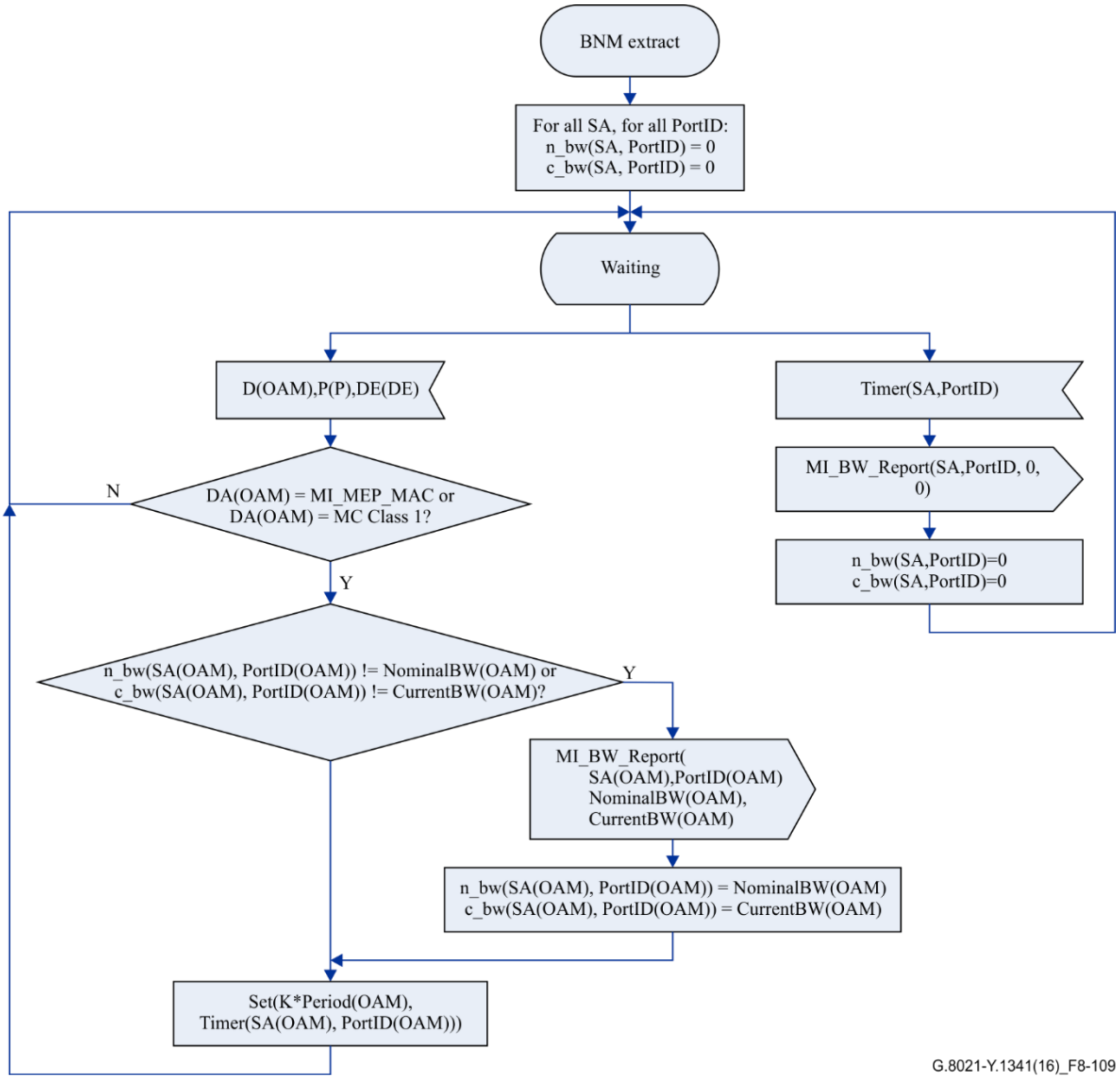


Figure 8-109 – BNM extract behaviour

Based on the received BNM frames and/or timer expiration the MEP is able to report the following information:

- Full bandwidth conditions when c_bw = n_bw ≠ 0
- Degraded conditions when c_bw < n_bw and c_bw ≠ 0
- Link faults conditions when c_bw = 0 and n_bw ≠ 0
- Unknown link conditions when c_bw = n_bw = 0

NOTE 2 – The c_bw/n_bw is the value in MI_BW_Report, not the value in the BNM.

When the MEP reports unknown link conditions, the management system, if needed, can correlate this information with other network information (e.g., the network topology, the alarms, and SF status of this or other links) to determine which is the actual condition of the link.

A timer is used in the BNM Extract process to detect when BNM traffic units are no longer being received. This is set to K times the period extracted from the Period field in the last received traffic unit. The BNM Extract process therefore does not require any local Management Information (MI) to set the period.

8.1.20 Expected Defect (ED) processes

8.1.20.1 Overview

Figure 8-110 shows the different processes inside MEPs that are involved in Expected Defect Message signals carried in MCC protocol data units.

In the source side of ETHx to MCC adaptation function, EDM signals are generated in EDM generation process when MI_EDM_Enable is set. MCC generation process encapsulates the signals into MCC PDUs and generates ETH_AI_D traffic units together with the complementing P and DE signals. In the sink side, the MCC reception process receives ETH_AI traffic units and extracts EDM signals from MCC PDUs. Finally EDM reception process terminates the signals and generates MI_EDM_Received (MEP_ID, Duration) signals to EMF function.

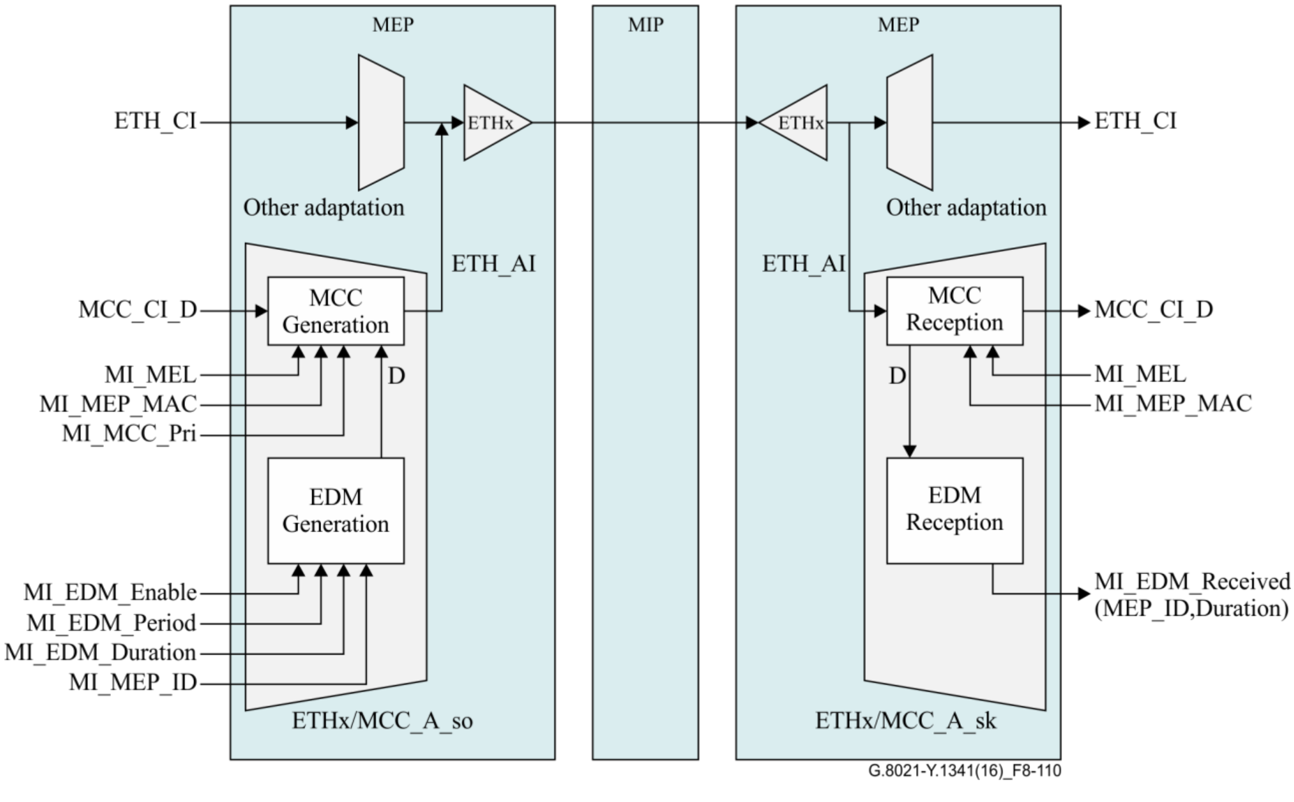


Figure 8-110 – Overview of Expected Defect processes

8.1.20.2 EDM Generation process

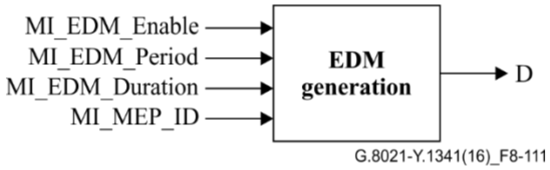


Figure 8-111 – EDM Generation process

Figure 8-111 shows the EDM Generation process symbol and Figure 8-112 defines the behaviour. When MI_EDM_Enable is set, the process generates EDM signal. Based on the EDM signals, MCC PDUs are generated at the MCC Generation process. As a result, MCC PDUs are signalled to peer MEPs that CCM transmission will be interrupted or has not yet commenced, and hence that Loss of Continuity defects and consequent actions should be suppressed.

EDM signals are generated periodically at the specified period and containing the specified Duration until MI_EDM_Enable is unset.

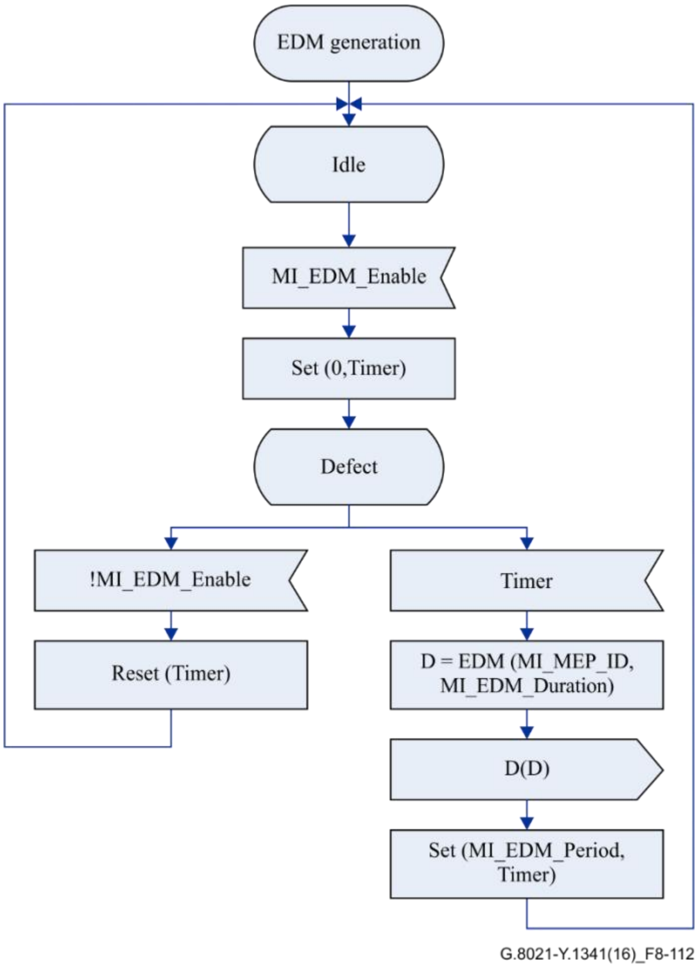


Figure 8-112 – EDM Generation behaviour

In the MCC Generation process, ETH_AI traffic units containing a source and destination address field and an M_SDU field are generated. The format of the M_SDU field for MCC and EDM information is defined in [ITU-T G.8013]. The EDM signal contains the MEP ID set to MI_MEP_ID and the Expected Defect Duration set to MI_EDM_Duration. In addition, MCC PDUs are generated with the priority set to MI_MCC_Pri, the SA set to the local MAC address by MI_MEP_MAC and the MEL set to MI_MEL. The value of the multicast class 1 DA is 01-80-C2-00-00-3x, where x is equal to MI_MEL, as defined in clause10.1 of [ITU-T G.8013]. Figure 8-113 shows the ETH_CI_D signal format resulting from the EDM Generation process and MCC Generation process.

<table>
<tr><th></th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th><th colspan="8">4</th></tr>
<tr><th></th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th></tr>
<tr><td>1</td><td colspan="32">DA=01-80-C2-00-00-3x, where x=MI_MEL</td></tr>
<tr><td>5</td><td colspan="16"></td><td colspan="16">SA=MI_MEP_MAC</td></tr>
<tr><td>9</td><td colspan="32"></td></tr>
<tr><td>13</td><td colspan="16">EtherType=89-02</td><td colspan="3">MEL=
MI_
MEL</td><td colspan="5">Version=0</td><td colspan="8">OpCode=41 (MCC)</td></tr>
<tr><td>17</td><td colspan="8">Flags=0</td><td colspan="8">TLV Offset = 10</td><td colspan="16">OUI=00-19-A7</td></tr>
<tr><td>21</td><td colspan="8">OUI Continued</td><td colspan="8">SubOpCode=1 (EDM)</td><td colspan="16">MEP ID=MI_MEP_ID</td></tr>
<tr><td>25</td><td colspan="32">Expected Duration=MI_EDM_Duration</td></tr>
<tr><td>25</td><td colspan="8">End TLV (0)</td><td colspan="24"></td></tr>
</table>

Figure 8-113 – EDM traffic unit

8.1.20.3 EDM Reception process

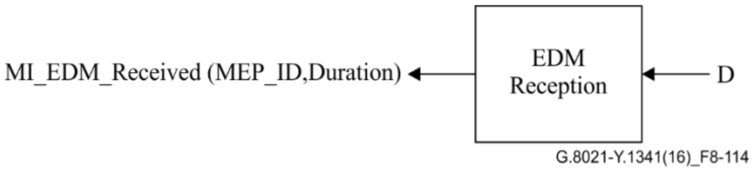


Figure 8-114 – EDM Reception process

Figure 8-114 shows the EDM Reception process symbol and Figure 8-115 defines the behaviour. When EDM signals are received, then the MEP ID and Expected Defect Duration are extracted from the EDM signals and passed to the EMF via the MI_EDM_Received (MEP_ID, Duration) signal.

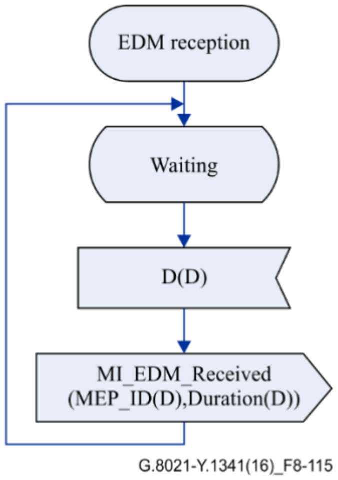


Figure 8-115 – EDM Reception behaviour

NOTE – It is expected that the EMF handles the MI_EDM_Received (MEP_ID, Duration) signal by unsetting MI_CC_Enable in the corresponding ETHx_FT_Sk or ETHG_FT_Sk function as appropriate, for the specified duration, if it has been configured to enable this functionality by the user. Further examples can be found in Appendix IX.

8.2 Queueing process

The queueing process buffers the received ETH_CI traffic units for output (see Figure 8-116). The queueing process is also responsible for discarding the ETH_CI traffic units if their rate at the ETH_FP is higher than the rate that the <server>_AP can accommodate, as well as for maintaining PM counters for discarded frames. Additional performance monitor counters (MI_PM_count) per [IEEE 802.1Q] are for further study.

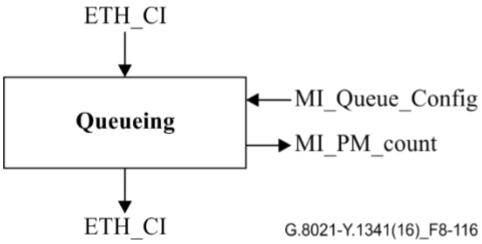


Figure 8-116 – Queueing process

The queueing process is configured using the MI_Queue_Config input parameter. This parameter specifies the mapping of ETH_CI traffic units into the available queues based on the value of the ETH_CI_P signal.

Furthermore, it specifies whether the value of the ETH_CI_DE signal should be taken into account when discarding frames. If this needs to be taken into account, ETH_CI traffic units with the ETH_CI_DE set to drop eligible should have a higher probability of being discarded than ETH_CI traffic units with ETH_CI_DE set to drop ineligible.

8.3 Filter process

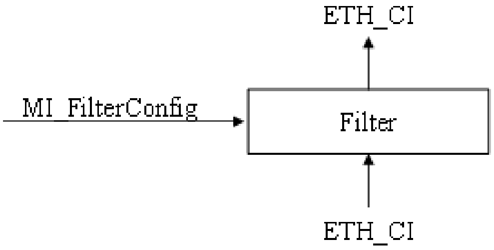


Figure 8-117 – Filter process

The filter process maintains the filter action for each of the 33 group MAC addresses indicating control frames as defined in clause 6.3 of [ITU-T G.8012]. Valid filter actions are "pass" and "block". The filter action for these 33 MAC addresses can be configured separately. If the destination address of the incoming ETH_CI traffic unit matches one of the above addresses, the filter process shall perform the corresponding configured filter action:

- Block: The frame is discarded by the filter process.
- Pass: The frame is passed unchanged through the filter process.

If none of the above addresses match, the ETH_CI traffic unit is passed.

Valid filter actions for specific services are indicated in [ITU-T G.8011] as the service attributes. The default filter action value shall be "pass" for all frames with the exception of MAC control frames for which the default value shall be "block".

8.4 Replicate process

See Figure 8-118.

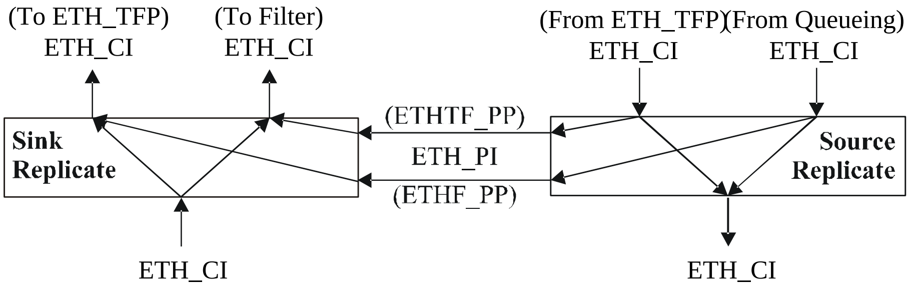


Figure 8-118 – Replicate processes

The <Srv>/ETH_A_So replicate process shall:

- replicate ETH_CI traffic units received on the input from the queueing process and deliver them as ETH_PI to the ETHF_PP interface and the 802.3 protocols process;
- replicate ETH_CI traffic units received on the input from the ETH_TFP and deliver them as ETH_PI to the ETHTF_PP interface and the 802.3 protocols process.

The <Srv>/ETH_A_Sk replicate process shall:

- replicate ETH_CI traffic units received on the input from the 802.3 protocols process and deliver them to the ETH_TFP and to the filter process;
- deliver ETH_PI traffic units received on the input from the ETHF_PP interface to the ETH_TFP;
- deliver ETH_PI traffic units received on the input from the ETHTF_PP to the filter process.

8.5 802.3 protocol processes

[ITU-T G.8023] defines the atomic functions for the 802.3 protocol processes, as defined in [IEEE 802.3].

8.6 MAC length check process

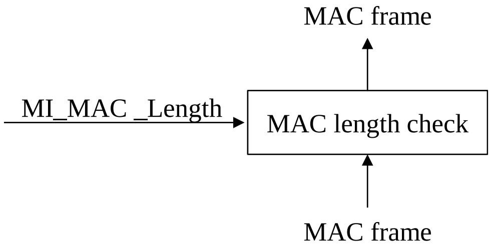


Figure 8-119 – MAC length check process

This process checks whether the length of the MAC frame is allowed. MAC frames longer than MI_MAC_Length are discarded.

NOTE – MAC frames shorter than 64 bytes are only foreseen on non-Ethernet interfaces, e.g., in connection with removal of VLAN tags. Such frames are padded to a minimum length of 64 bytes when forwarded on Ethernet interfaces, according to clause 4 of [IEEE 802.3].

Table 8-6 shows the values corresponding to the IEEE defined frame lengths.

Table 8-6 – IEEE 802.3 MI_MAC_Length values

Frame type	MI_MAC_Length
Basic	1518
Q-tagged	1522
Envelope	2000

8.7 MAC frame counter process

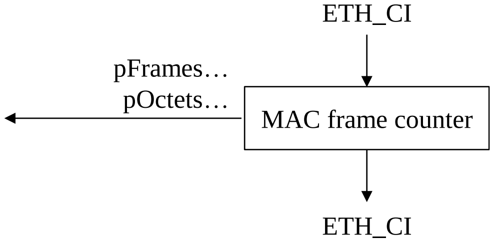


Figure 8-120 – MAC frame counter process

This process passes ETH_CI traffic units and counts the number of traffic units and octets that are passed.

For source side:

pOctetsTransmittedOK as per clause 30 of [IEEE 802.3].

pFramesTransmittedOK as per clause 30 of [IEEE 802.3].

For sink side:

pOctetsReceivedOK as per clause 30 of [IEEE 802.3].

pFramesReceivedOK as per clause 30 of [IEEE 802.3].

8.8 Server-specific common processes

For some server signals MAC FCS generation is not supported. This will be defined in the server-specific adaptation functions.

8.8.1 MAC FCS generation process

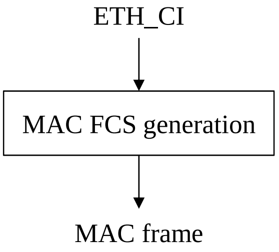


Figure 8-121 – MAC FCS generation process

The MAC FCS is calculated over the received ETH_CI traffic units and is inserted into the MAC FCS fields of the transmitted MAC frames as defined in clause 4.2.3 of [IEEE 802.3].

8.8.2 MAC FCS check process

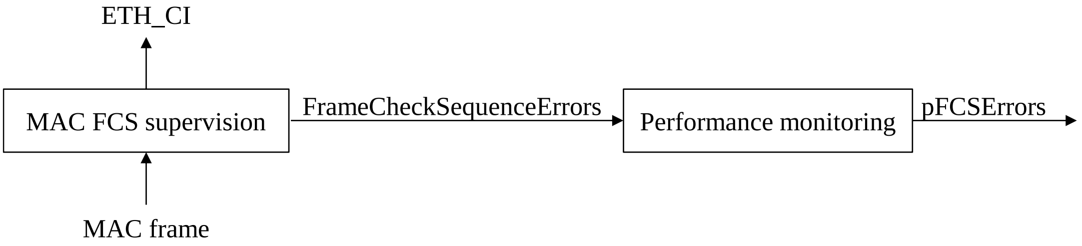


Figure 8-122 – MAC FCS check process

The MAC FCS is calculated over the received MAC frames and checked as specified in clause 4.2.4.1.2 of [IEEE 802.3]. If errors are detected, the frame is discarded. Errored frames are indicated by FrameCheckSequenceErrors.

8.8.3 802.1AB/X protocols processes

802.1AB/X protocols processes include source and sink handling of 802.1X and 802.1AB protocols, as shown in Figures 8-123 and 8-124.

The following clauses specify processes for each of the illustrated process blocks.

8.8.3.1 802.1X protocol process

The 802.1X protocol block implements the port-based network access control as per [IEEE 802.1X].

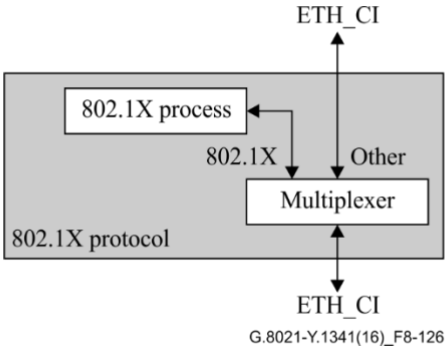


Figure 8-123 – 802.1X protocol process

In the sink direction, the multiplexer separates the ETH_CI traffic units carrying 802.1X PDUs from the rest of the ETH_CI traffic units based on MAC address 01-80-C2-00-00-03. The former are delivered to the 802.1X process, the latter are passed on in the sink direction. In the source direction, the ETH_CI traffic units carrying 802.1X PDUs are multiplexed with the rest of the ETH_CI traffic units.

In the function descriptions in which it appears, the 802.1X process is optional.

8.8.3.2 802.1AB protocol process

The 802.1AB protocol block implements the link layer discovery protocol as per [IEEE 802.1AB].

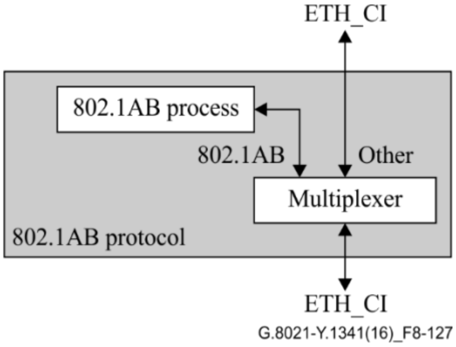


Figure 8-124 – 802.1AB protocol process

In the sink direction, the multiplexer separates the ETH_CI traffic units carrying 802.1AB PDUs from the rest of the ETH_CI traffic units. The former are delivered to the 802.1AB process, the latter are passed on in the sink direction. In the source direction, the ETH_CI traffic units carrying 802.1AB PDUs are multiplexed with the rest of the ETH_CI traffic units. Frames are defined by: MAC address 01-80-C2-00-00-0E, EtherType 88-CC.

In the function description in which it appears, the 802.1AB process is optional.

8.8.4 Link quality supervision

Counts of transmitted and received octets and frames are maintained in <Srv>/ETH_A functions per the requirements of clause 30 of [IEEE 802.3]. Discarded jabber frames are counted in the xTSi[G]/ETH_A_So function defined in [ITU-T G.8023].

Additional link quality performance monitors as per clause 30 of [IEEE 802.3] are for further study.

8.8.5 FDI/BDI generation and detection

For further study.

8.8.6 ETH-specific GFP-F process

8.8.6.1 ETH-specific GFP-F source process

See clause 8.5.4.1.1 of [ITU-T G.806]. GFP pFCS generation is disabled (FCSenable=false). The UPI value for frame-mapped Ethernet shall be inserted (as defined in Table 6-3 of [ITU-T G.7041]). The Ethernet frames are inserted into the client payload information field of the GFP-F frames according to clause 7.1 of [ITU-T G.7041]. Client management frame insertion is governed by the consequent actions.

Consequent actions

aCSF-RDI ← CI_SSFrdi and CSFrdifdiEnable and CSFEnable

aCSF-FDI ← CI_SSFfdi and CSFrdifdiEnable and CSFEnable

aCSF-LOS ← CI_SSF and CSFEnable

8.8.6.2 ETH-specific GFP-F sink process

See clause 8.5.4.1.2 of [ITU-T G.806]. GFP pFCS checking, GFP p_FCSError, p_FDis are not supported (FCSdiscard=false). The UPI value for frame-mapped Ethernet shall be expected (as defined in Table 6-3 of [ITU-T G.7041]). The Ethernet frames are extracted from the client payload information field of the GFP-F frames according to clause 7.1 of [ITU-T G.7041]. The generic defects and consequent actions are extended as follows.

Defects

dCSF-RDI: GFP client signal fail-remote defect indication (dCSF-RDI) is raised when a GFP client management frame with the RDI UPI (as defined in Table 6-4 of [ITU-T G.7041]) is received. dCSF-RDI is cleared when no such GFP client management frame is received in N x 1000 ms (a value of 3 is suggested for N), a valid GFP client data frame is received, or a GFP client management frame with the DCI UPI is received.

dCSF-FDI: GFP client signal fail-forward defect indication (dCSF-FDI) is raised when a GFP client management frame with the FDI UPI (as defined in Table 6-4 of [ITU-T G.7041]) is received. dCSF-FDI is cleared when no such GFP client management frame is received in N x 1000 ms (a value of 3 is suggested for N), a valid GFP client data frame is received, or a GFP client management frame with the DCI UPI is received.

dCSF-LOS: GFP client signal fail-loss of signal (dCSF-LOS) is raised when a GFP client management frame with the LOS UPI (as defined in Table 6-4 of [ITU-T G.7041]) is received. dCSF-LOS is cleared when no such GFP client management frame is received in N x 1000 ms (a value of 3 is suggested for N), a valid GFP client data frame is received, or a GFP client management frame with the DCI UPI is received.

Consequent actions

aSSFrdi ← dCSF-RDI and CSFrdifdiEnable

aSSFfdi ← dCSF-FDI and CSFrdifdiEnable

aSSF ← GFP_SF or dUPM or dCSF-LOS

Defect correlations

cCSF ← (dCSF-RDI or dCSF-FDI or dCSF-LOS) and (not dUPM) and (not GFP_SF) and CSF_Reported.

The GFP_SF term refers collectively to the set of defects detected in the Common GFP-F sink process (see clause 8.5.3.2 of [ITU-T G.806]), the server-specific GFP-F sink process (see clause 8.5.2.2 of [ITU-T G.806]), or the server-specific process (see clause 11) with the consequent action of aGFP_SF. This includes dEXM, dLFD, any server-specific defects related to the GFP-F mapping, and server layer TSF.

8.9 QoS related processes

8.9.1 Queue

The queue process stores received ETH_CI traffic units and associated signals, and forwards a traffic unit if requested to do so by the connected process.

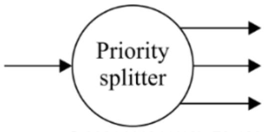
Figure 8-125 – Queue process

There are several parameters on the queue:

- Queue depth: The maximum size of the queue in bytes. An incoming ETH_CI traffic unit is dropped if there is insufficient space to hold the whole unit.
- Dropping threshold: If the queue is filled beyond this threshold, incoming ETH_CI traffic units accompanied by the ETH_CI_DE signal set are dropped.

8.9.2 Priority splitter

The priority splitter process forwards received ETH_CI onto different output ports depending on the value of the ETH_CI_P signal.



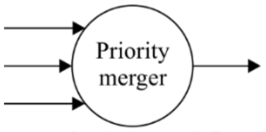
G.8021-Y.1341(16)_F8-129

Figure 8-126 – Priority splitter process

The mapping of ETH_CI_P values to output ports of the priority splitter process needs to be configured.

8.9.3 Priority merger

The priority merger process forwards received ETH_CI on one of its input ports to a single output port.



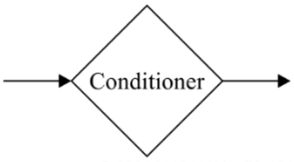
G.8021-Y.1341(16)_F8-130

Figure 8-127 – Priority merger process

Nothing has to be configured on this process.

8.9.4 Conditioner

The conditioner determines the conformance of the incoming ETH_CI traffic units. The level of conformance is expressed as one of three colours; green, yellow or red.



G.8021-Y.1341(16)_F8-131

Figure 8-128 – Conditioner process

Red conformance means that the ETH_CI traffic unit is discarded; yellow conformance means that for the ETH_CI traffic units the associated ETH_CI_DE signal is set to true; green conformance means that the ETH_CI traffic unit is forwarded unchanged and the ETH_CI_DE signal is set to false.

Compliance for a bandwidth profile is described by four parameters. The parameters are:

1) Committed information rate (CIR) expressed as bits per second. CIR must be ≥ 0.
2) Committed burst size (CBS) expressed as bytes. When CIR > 0, CBS must be ≥ maximum transmission unit size allowed to enter the function.
3) Excess information rate (EIR) expressed as bits per second. EIR must be ≥ 0.
4) Excess burst size (EBS) expressed as bytes. When EIR > 0, EBS must be ≥ maximum Ethernet frame allowed to enter the network.

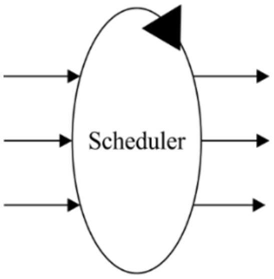
Two additional parameters are used to determine the behaviour of the bandwidth profile algorithm. The algorithm is said to be in colour-aware mode when each incoming Ethernet frame already has a level of conformance colour associated with it and that colour is taken into account in determining the level of conformance to the bandwidth profile parameters. The bandwidth profile algorithm is said to be in colour-blind mode when the level of conformance colour (if any) already associated with each incoming Ethernet frame, is ignored in determining the level of conformance. Colour-blind mode support is required at the UNI. Colour-aware mode is optional at the UNI.

1) Coupling flag (CF) must have only one of two possible values, 0 or 1.
2) Colour mode (CM) must have only one of two possible values, "colour-blind" and "colour-aware".

All these parameters have to be configured at the conditioner function. The conformance algorithm is defined in [MEF 10.3].

8.9.5 Scheduler

The scheduler process forwards ETH_CI from its input ports to the corresponding output ports of the scheduler function according to a specified scheduling algorithm.



G.8021-Y.1341(16)_F8-132

Figure 8-129 – Scheduler process

The scheduling algorithm and its parameters must be configured.

The scheduling algorithms are for further study.

9 Ethernet MAC layer (ETH) functions

Figure 1-1 illustrates all the ETH layer network, server and client adaptation functions. The information crossing the ETH flow point (ETH_FP) is referred to as the ETH characteristic information (ETH_CI). The information crossing the ETH access point (ETH_AP) is referred to as ETH adapted information (ETH_AI).

ETH sublayers can be created by expanding an ETH_FP as illustrated in Figure 9-1.

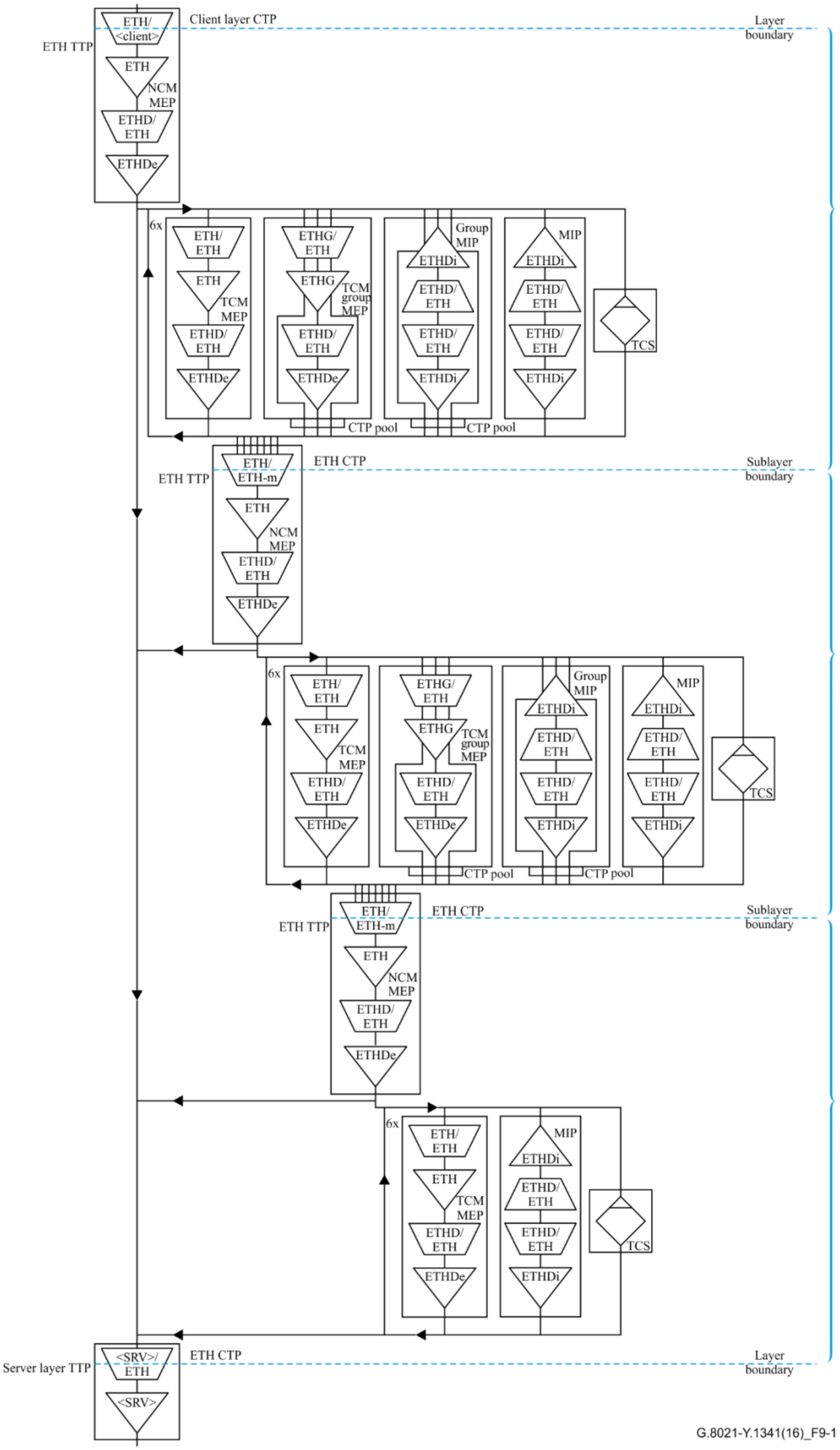

ETH/
<client>
Client layer CTP
Layer
boundary
ETH TTP
ETH
NCM
MEP
ETHD/
ETH
ETHDe
ETHx
Sublayer
6x
ETH/
ETH
ETH
TCM
MEP
ETHD/
ETH
ETHDe
ETHG/
ETH
ETHG
TCM
group
MEP
ETHD/
ETH
ETHDe
Group
MIP
ETHDi
ETHD/
ETH
ETHD/
ETH
ETHDi
MIP
ETHDi
ETHD/
ETH
ETHD/
ETH
ETHDi
TCS
CTP pool
CTP pool
ETH/
ETH-m
ETH CTP
ETH TTP
Sublayer
boundary
ETH
NCM
MEP
ETHD/
ETH
ETHDe
6x
ETH/
ETH
ETH
TCM
MEP
ETHD/
ETH
ETHDe
ETHG/
ETH
ETHG
TCM
group
MEP
ETHD/
ETH
ETHDe
Group
MIP
ETHDi
ETHD/
ETH
ETHD/
ETH
ETHDi
MIP
ETHDi
ETHD/
ETH
ETHD/
ETH
ETHDi
TCS
CTP pool
CTP pool
ETHx
Sublayer
ETH/
ETH-m
ETH CTP
ETH TTP
Sublayer
boundary
ETH
NCM
MEP
ETHD/
ETH
ETHDe
6x
ETH/
ETH
ETH
TCM
MEP
ETHD/
ETH
ETHDe
MIP
ETHDi
ETHD/
ETH
ETHD/
ETH
ETHDi
TCS
ETHx
Sublayer
<SRV>/
ETH
ETH CTP
Server layer TTP
Layer
boundary
<SRV>
G.8021-Y.1341(16)_F9-1

Figure 9-1 – ETH sublayering

Figure 9-1 illustrates the basic flow termination and adaptation functions involved and the possible ordering of these functions. The ETHx/ETH-m functions multiplex ETH_CI streams. The ETHx and ETHG flow termination functions insert and extract the proactive [ITU-T G.8013/Y.1731] OAM information (e.g., CCM). The ETHDy flow termination functions insert and extract the on-demand [ITU-T G.8013/Y.1731] OAM information (e.g., LBM, LTM). The ETHx/ETH and ETHG/ETH adaptation functions insert and extract the administrative and control [ITU-T G.8013/Y.1731] OAM information (e.g., LCK, APS).

Any combination that can be constructed by following the directions in the figure is allowed. Some recursion is allowed as indicated by the arrows upwards; the number next to the arrow defines the number of recursions allowed.

Note that the ETHx sublayers in Figure 9-1 correspond to the ETH0 (top), ETH1 (middle) and ETH2 (bottom) in Figure 7-5 of [ITU-T G.8010].

NOTE 1 – ETHx/ETHG adaptation function is not included in Figure 9-1 because this atomic function is not used in ETH MEP and MIP functions described in clause 9.8.

ETH characteristic information

The ETH_CI is a stream of ETH_CI traffic units complemented with ETH_CI_P, ETH_CI_DE, ETH_CI_SSF and ETH_CI_SSD signals. An ETH_CI traffic unit defines the ETH_CI_D signal as illustrated in Figure 9-2. Each ETH_CI traffic unit contains a source address (SA) field, a destination address (DA) field and an M_SDU field, this can be further decomposed into a length/type field and a payload field; the payload field may be padded.

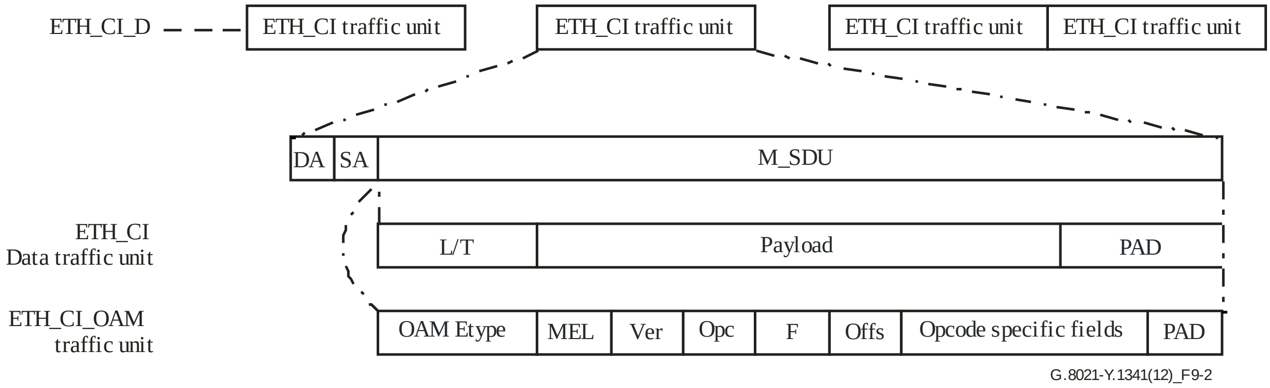


Figure 9-2 – ETH characteristic information

The SA and DA field contain 48 byte MAC addresses as defined in [IEEE 802.3].

There are two types of ETH_CI traffic units: data traffic units and OAM traffic units. If the L/T field equals the OAM Etype value (89-02 as defined in clause 10 of [ITU-T G.8013]) the ETH_CI traffic unit is an ETH_CI OAM traffic unit, otherwise it is an ETH_CI data traffic unit.

The payload field of an ETH_CI OAM traffic unit can be decomposed into the maintenance entity group level field (MEL), the version field (Ver), the OpCode field (Opc), the flags field (F), the TLV Offset field (Offs) and OpCode specific fields. This structure of ETH_CI OAM traffic units is defined in clause 9 of [ITU-T G.8013].

Functions for traffic units

The following functions are used in this Recommendation to indicate the various fields of a traffic unit:

- SA(Traffic_Unit): returns the value of the SA field in the traffic unit.
- DA(Traffic_Unit): returns the value of the DA field in the traffic unit.
- Etype(Traffic_Unit): returns the value of the EtherType field in the traffic unit.
- OPC(OAM Traffic_Unit): returns the value of the OpCode field in the OAM traffic unit; returns an undefined value if the traffic unit is not an OAM traffic unit.
- MEL(OAM Traffic_Unit): returns the value of the maintenance entity group level field in the OAM traffic unit; returns an undefined value if the traffic unit is not an OAM traffic unit.

Flags(OAM Traffic_Unit): returns the value of the flags field in the OAM traffic unit; returns an undefined value if the traffic unit is not an OAM traffic unit.

NOTE 2 – The ETH_CI contains no VID field as the ETH_CI is defined per VLAN.

ETH adapted information

The ETH_AI is a stream of ETH_AI traffic units complemented with the following signals: ETH_AI_P, ETH_AI_DE, ETH_AI_TSF and ETH_AI_TSD. The ETH_AI traffic units define the ETH_AI_D signal. The ETH_AI traffic unit structure is shown in Figure 9-3.

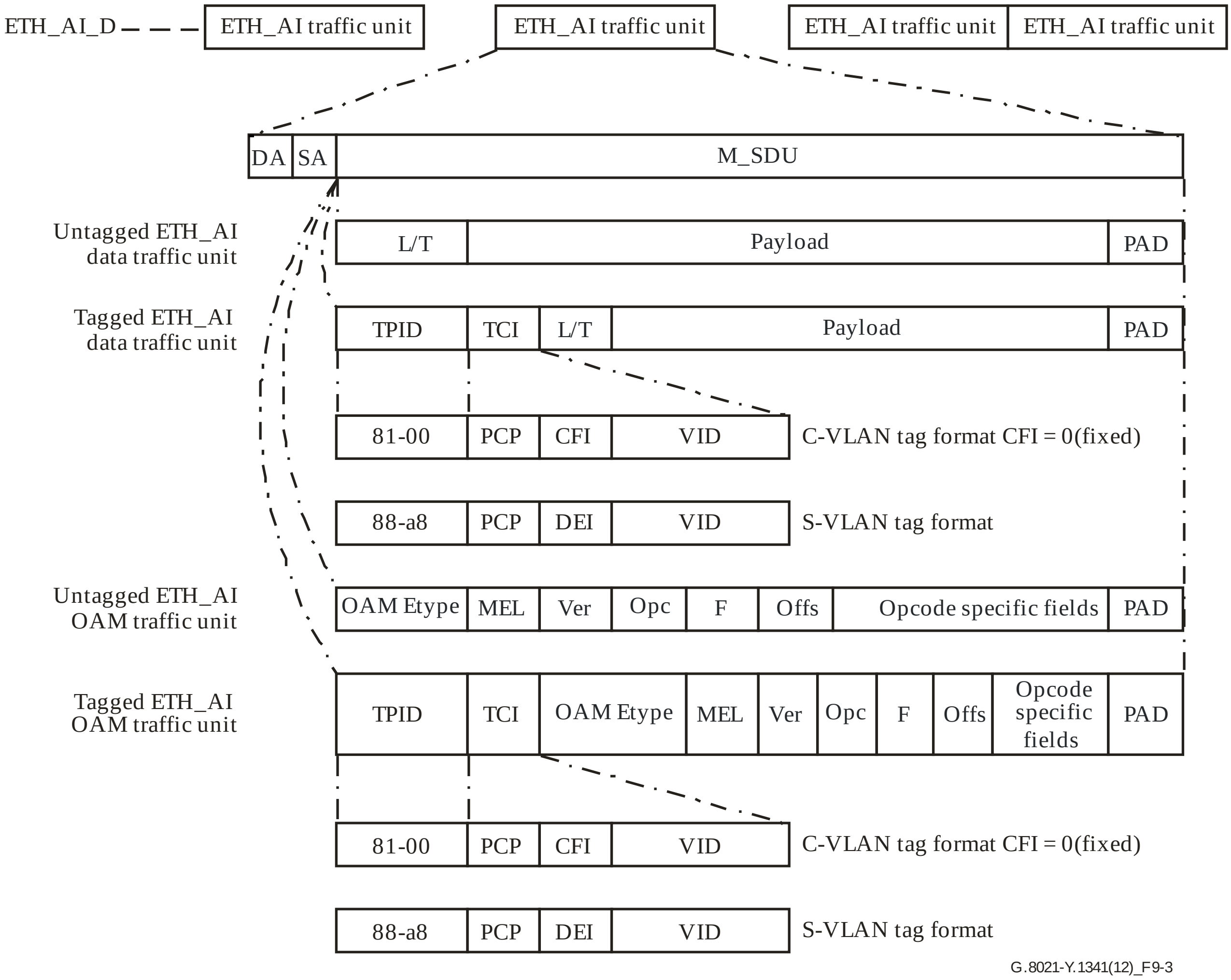


Figure 9-3 – ETH adapted information

The ETH_AI traffic unit contains the M_SDU and the DA and SA fields. The M_SDU field can be further decomposed into L/T, payload and PAD fields. These fields are the same as in ETH_CI traffic units.

There are four types of ETH_AI traffic units: untagged data, tagged data, untagged OAM and tagged OAM traffic units. The untagged and tagged types are defined in [IEEE 802.1Q]. The OAM traffic units are defined in [ITU-T G.8013].

The L/T field determines the type of the ETH_AI traffic unit:

- If the L/T field contains the OAM EtherType value, the traffic unit is an untagged OAM traffic unit; otherwise,
- if the L/T field contains one of the tag protocol identifier (TPID) values indicated in Figure 9-3, and the succeeding field to the tag control information (TCI) value corresponds to the OAM EtherType value, the traffic unit is a tagged OAM traffic unit; otherwise,
- if the L/T field contains neither the OAM EtherType value nor the TPID values, the traffic unit is an untagged data traffic unit; otherwise,
- the traffic unit is a tagged data traffic unit.

The payload field of an ETH_AI OAM traffic unit can be decomposed into the maintenance entity group level field (MEL), the version field (Ver), the OpCode field (Opc), the flags field (F), TLV Offset field (Offs) and OpCode specific fields. This structure of ETH_AI OAM traffic units is the same as ETH_CI OAM traffic units defined in clause 9 of [ITU-T G.8013].

There are two types of tagged traffic units: C-VLAN tagged and S-VLAN tagged. Each of these types has its own TPID value, 81-00 for C-VLAN tagged and 88-a8 for S-VLAN tagged as defined in clause 9.5 of [IEEE 802.1Q].

In a tagged frame (C-VLAN and S-VLAN tagged) a tag control information (TCI) field follows the TPID field. This field consists of a priority code point (PCP), VLAN ID (VID) and canonical format identifier (CFI) for C-VLAN tagged traffic units, or drop eligible indicator (DEI) field for S-VLAN tagged traffic units.

The PCP field may be used to carry the ETH_CI_P and ETH_CI_DE signal values from an ETH_FP. The DEI field may be used to carry the ETH_CI_DE signal from an ETH_FP.

All ETH_AI traffic units may come from one ETH_FP or different ETH_FPs (in the case of multiplexing in ETHx/ETH-m_A function). In the latter case the VID field value is used to identify the ETH_FP where the traffic unit is associated.

Note that because of the stacking of ETH sublayers, ETH_CI of a client ETH sublayer is encapsulated in ETH_AI to be transferred via a server ETH sublayer. Figure 9-4 shows an ETH_CI OAM traffic unit encapsulated in an ETH_AI data traffic unit. The grey fields constitute the original ETH_CI OAM traffic unit. The encapsulating traffic unit is no longer an OAM traffic unit, but a tagged traffic unit. Adding a VLAN tag hides the OAM information, and transforms an ETH_CI OAM traffic unit into a tagged ETH_AI Data traffic unit.

DA	SA	TPID	TCI	OAM Etype	MEL	Ver	Opc	F	Offs	Opcode specific OAM information	PAD

Figure 9-4 – Tagged ETH_AI carrying ETH_CI OAM

This ETH_AI tagged traffic unit will be transformed into an ETH_CI data traffic unit by the ETHx_FT source function, resulting in an ETH_CI data traffic unit carrying a client layer ETH_CI OAM traffic unit.

9.1 ETH connection functions (ETH_C)

The information flow and processing of the ETH_C function is defined with reference to Figures 9-5 and 9-6. The ETH_C function connects ETH characteristic information from its input ports to its output ports. As the process does not affect the nature of characteristic information, the reference points on either side of the ETH_C function are the same as illustrated in Figure 9-5.

The connection process is unidirectional and as such no differentiation in sink and source is required.

In addition, the ETH_C function supports the following protection schemes:

- – 1+1 unidirectional SNC/S protection without APS protocol.
- – 1+1 unidirectional SNC/S protection with an APS protocol.
- – 1+1 bidirectional SNC/S protection with an APS protocol.
- – 1:1 bidirectional SNC/S protection with an APS protocol.
- – Ring protection with an APS protocol.

The protection functionality is described in clauses 9.1.2 and 9.1.3.

NOTE 1 – The SNC/S protection processes have a dedicated sink and source behaviour.

Symbol

The ETH connection function, as shown in Figure 9-5, forwards ETH_CI signals at its input ports to its output ports.

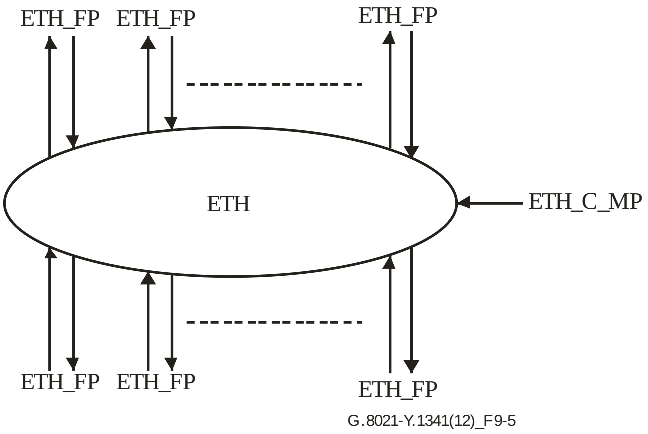


Figure 9-5 – ETH_C symbol

The actual forwarding is performed using flow forwarding processes ETH_FF interconnecting the input and output ports.

Interfaces

Table 9-1 – ETH_C interfaces

Inputs	Outputs
Per ETH_FP: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_APS	**Per ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_APS

Table 9-1 – ETH_C interfaces

Inputs	Outputs
ETH_CI_SSF ETH_CI_SSD **ETH_C_MP**: ETH_C_MI_Create_FF ETH_C_MI_Modify_FF ETH_C_MI_Delete_FF **ETH_C_MP per flow forwarding process**: ETH_C_MI_FF_Set_PortIds ETH_C_MI_FF_ConnectionType ETH_C_MI_FF_Flush_Learned ETH_C_MI_FF_Flush_Config ETH_C_MI_FF_Group_Default ETH_C_MI_FF_ETH_FF ETH_C_MI_FF_Ageing ETH_C_MI_FF_Learning ETH_C_MI_FF_STP_Learning_State[i] **ETH_C_MP per SNC/S protection process**: ETH_C_MI_PS_WorkingPortId ETH_C_MI_PS_ProtectionPortId ETH_C_MI_PS_ProtType ETH_C_MI_PS_OperType ETH_C_MI_PS_HoTime ETH_C_MI_PS_WTR ETH_C_MI_PS_ExtCMD ETH_C_MI_PS_BridgeType ETH_C_MI_PS_SD_Protection **ETH_C_MP per Ring protection process**: ETH_C_MI_RAPS_PortIds[0..1] ETH_C_MI_RAPS_RPL_Owner_Node ETH_C_MI_RAPS_RPL_Neighbour_Node ETH_C_MI_RAPS_Propagate_TC[1…M] ETH_C_MI_RAPS_Compatible_Version ETH_C_MI_RAPS_Revertive ETH_C_MI_RAPS_Sub_Ring_Without_Virtual_Channel ETH_C_MI_RAPS_HoTime ETH_C_MI_RAPS_WTR ETH_C_MI_RAPS_GuardTime ETH_C_MI_RAPS_ExtCMD ETH_C_MI_RAPS_RingID	**ETH_C_MP per SNC/S protection process:** ETH_C_MI_cFOP-PM ETH_C_MI_cFOP-CM ETH_C_MI_cFOP-NR ETH_C_MI_cFOP-TO ETH_C_MI_PS_RequestState ETH_C_MI_PS_ReqestedSignal ETH_C_MI_PS_BridgedSignal **ETH_C_MP per Ring protection process:** ETH_C_MI_cFOP-PM ETH_C_MI_cFOP-TO[0..1] ETH_C_MI_RAPS_NodeState ETH_C_MI_RAPS_PortState[0..1]

Processes

The processes associated with the ETH_C function are depicted in Figure 9-6.

ETH_CI traffic units are forwarded between input and output ETH flow points by means of an ETH flow forwarding process. ETH flow points may be allocated within a protection group.

NOTE 2 – Neither the number of input/output signals to the connection function, nor the connectivity, is specified in this Recommendation. That is a property of individual network elements.

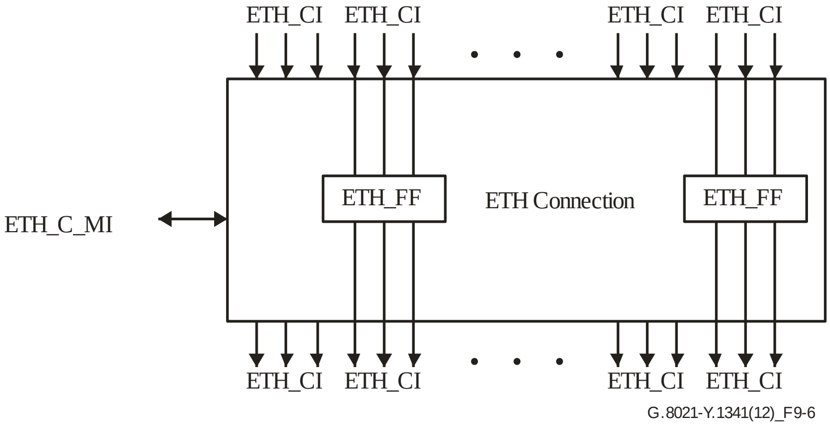


Figure 9-6 – ETH connection function with ETH_FF processes

The flow forwarding process ETH_FF is described in clause 9.1.1.

Defects	None.
Consequent actions	None.
Defect correlations	None.
Performance monitoring	None.

9.1.1 ETH flow forwarding process (ETH_FF)

The ETH flow forwarding process, as shown in Figure 9-6, forwards ETH_CI signals at its input ports to its output ports. The forwarding may take into account the value of the DA field of the ETH_CI traffic unit.

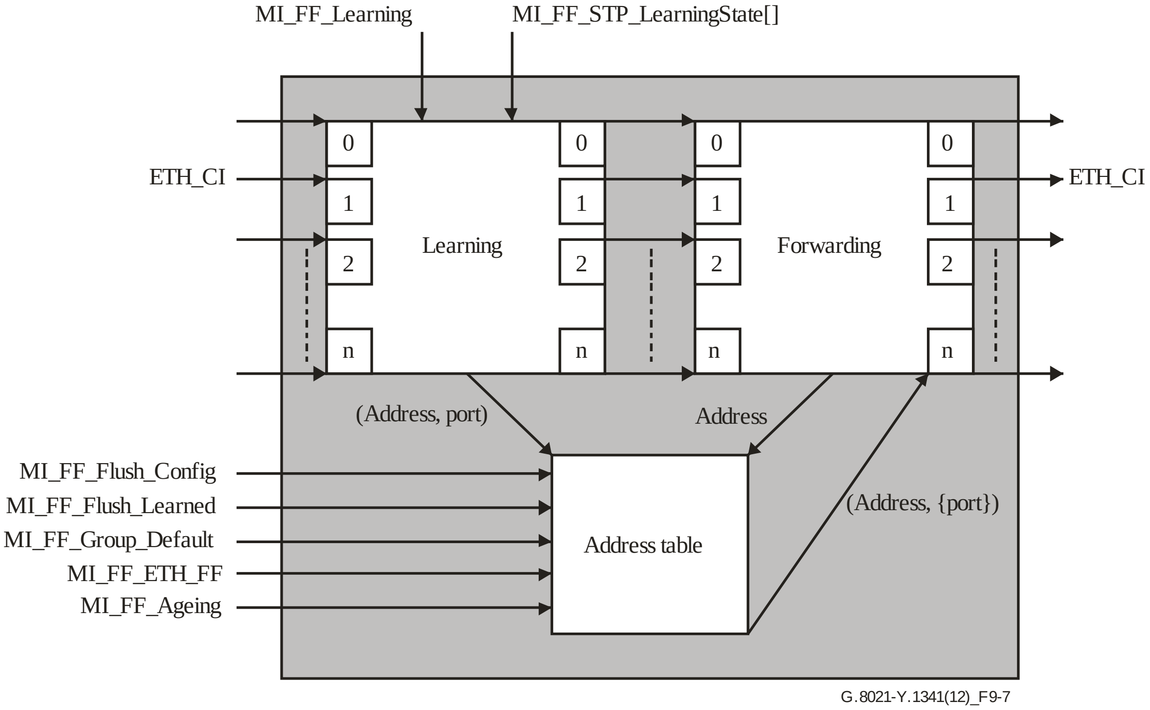


Figure 9-7 – ETH flow forwarding process

Figure 9-7 shows the ETH_FF in the case of the individual VLAN learning (IVL) mode. In this mode each ETH_FF has its own address table. Figure 9-8 shows the process for the case of the shared VLAN learning (SVL) mode. In this mode two or more ETH_FF share the address table process.

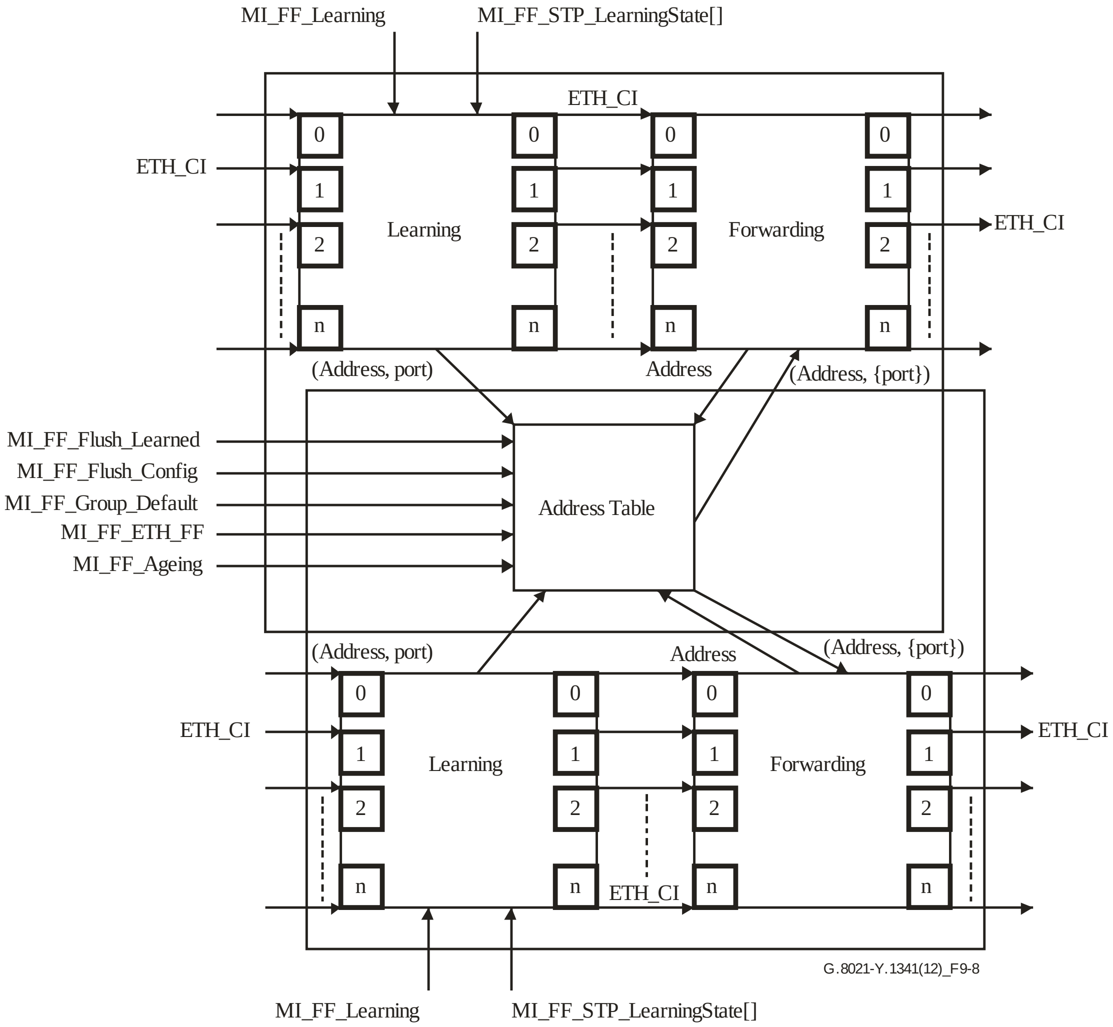


Figure 9-8 – ETH flow forwarding process in SVL mode

Address table process

The Address table process maintains a list of tuples (Address, {ports}). This list may be configured using the MI_FF_ETH_FF input signal and by the learning process.

A tuple received from the learning process is only stored in the address table process if there is no entry present for that MAC address that has been configured by the MI_FF_ETH_FF input signal.

The MI_FF_Ageing is used to provision the ageing time period for entries configured from the learning process. Entries received from the learning process are removed from the address table ageing time period after it was received. If before the ageing time period has expired a new entry for the same MAC address is received, the ageing time period starts again.

There is one specific value of MI_FF_Ageing: "never". This means that the entries received from the learning process are never removed.

All the tuples received from the learning process can be cleared using the MI_FF_Flush_Learned command.

All the tuples that are entered via the MI_FF_ETH_FF can be cleared using the MI_FF_Flush_Config command. Individual entries are removed via the MI_FF_ETH_FF signal.

The address table process processes address requests from the forwarding process, and responds with the tuple (Address, {port}) for the specified address. For unicast MAC addresses, if the tuple does not exist the port set ({port}) is empty. For multicast MAC addresses, if the tuple does not exist the port set ({port}) contains the ports as configured using the MI_FF_Group_Default input signal.

Learning process

If the value of MI_FF_Learning is enabled, the learning process reads the SA field of the incoming ETH_CI traffic unit, and forwards a tuple (Address, {port}) to the address table process. The address contains the value of the SA field of the ETH_CI traffic unit, and the port is the port on which the traffic unit was received.

If the value of MI_FF_Learning is disabled, the learning process does not submit information to the addresstable process.

In both cases the ETH_CI itself is forwarded unchanged to the output of the learning process.

Forwarding process

The parameters of MI_Create_FF, MI_Modify_FF, and MI_Delete_FF are used to provision the flow forwarding process.

The MI_FF_Set_PortIds parameter is used to provision TBD.

The MI_FF_ConnectionType parameter is used to provision TBD.

The MI_FF_STP_LearningState[i] input signal is provisioned per port [i]; it can be used to configure a specific port to be in the learning state. If a port is in the learning state this means that all frames received on that port will be discarded by the learning process, and therefore not forwarded to the forwarding process; however the (Address, {port}) tuple may be submitted to the address table process before the frame is dropped (depending on the value of MI_FF_Learning).

The forwarding process reads the DA field of the incoming ETH_CI traffic unit and sends this to the address table process, the addresstable will send a tuple (Address, {port}) back in response. It will forward the ETH_CI on all ports listed in the port set field of the tuple. If the port set is empty, the ETH_CI will be forwarded on all ports (flooding). In all cases the ETH_CI is never forwarded on the same port as it was received on.

9.1.2 Subnetwork connection protection process

SNC protection with sublayer monitoring based on TCM is supported.

Figure 9-9 shows the involved atomic functions in SNC/S. The ETHx_FT_Sk provides the TSF/TSD protection switching criterion via the ETHx/ETH_A_Sk function (SSF/SSD) to the ETH_C function.

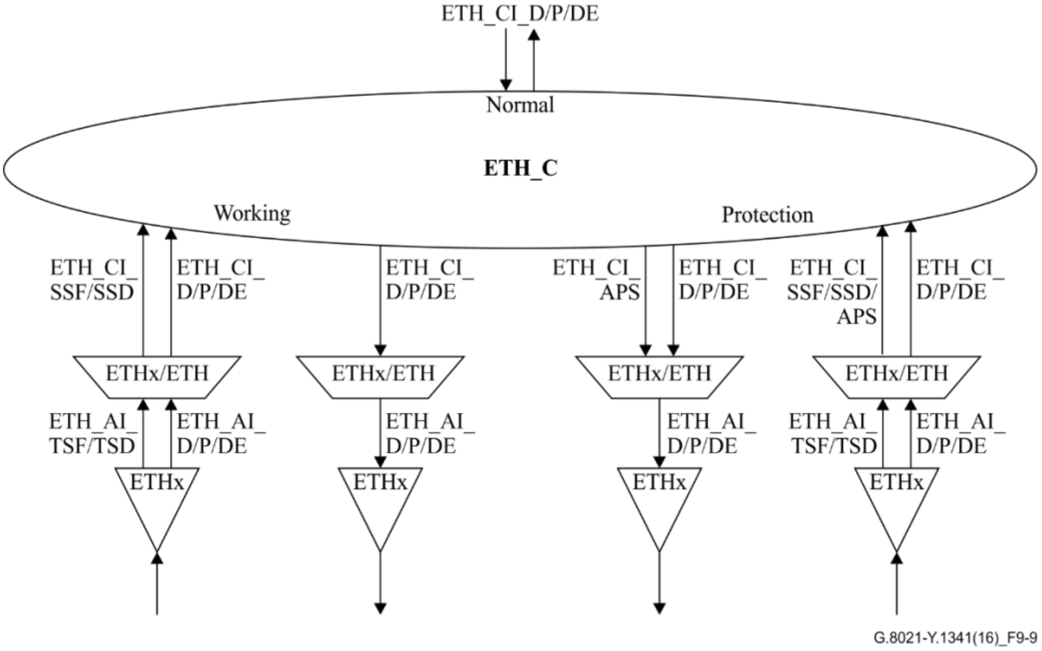


Figure 9-9 – SNC/S atomic functions

NOTE 1 – Since SNC/S is ETH subnetwork protection with sublayer monitoring, ETHx flow termination and ETHx/ETH adaptation functions in Figure 9-9 correspond to ETHT (tandem connection) sublayer where this abbreviation is described in Amendment 1 to [ITU-T G.8010].

The protection functions at both ends operate the same way, by monitoring the working and protection subnetwork connections for defects, evaluating the system status taking into consideration the priorities of defect conditions and of external switch requests, and switching the appropriate subnetwork flow point (i.e., working or protection) to the protected (sub)network flow point.

The signal flows associated with the ETH_C SNC protection process are described with reference to Figure 9-10. The protection process receives control parameters and external switch requests at the MP reference point. The report of status information at the MP reference point is for further study.

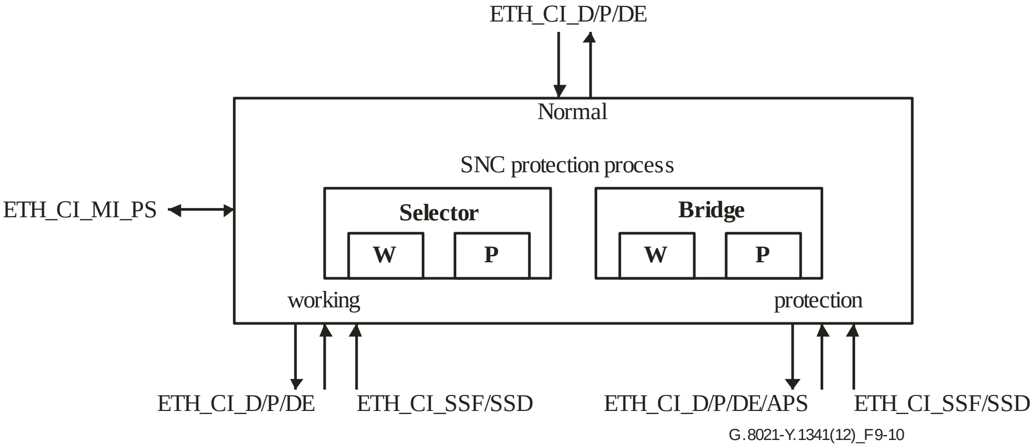


Figure 9-10 – SNC/S protection process

Source direction

For a 1+1 architecture, the CI coming from the normal (protected) ETH_FP is bridged permanently to both the working and protection ETH_FP.

For a 1:1 architecture, the CI coming from the normal (protected) ETH_FP is switched to either the working or the protection ETH_FP. A switch-over from working to protection ETH_FP or vice versa is initiated by the switch initiation criteria defined below.

Sink direction

For a 1+1 or 1:1 architecture, the CI coming from either the working or protection ETH_FP is switched to the normal (protected) ETH_FP. A switch-over from working to protection ETH_FP or vice versa is initiated by the switch initiation criteria defined below.

Switch initiation criteria

Automatic protection switching is based on the defect conditions of the working and protection (sub)network connections, for SNC/S protection server signal fail (SSF) and server signal degrade (SSD).

In order to allow interworking between nested protection schemes, a hold-off timer is provided. The hold-off timer delays switch initiation, in case of signal fail, in order to allow a nested protection to react and clear the fault condition. The hold-off timer is started by the activation of signal fail and runs for the hold-off time. Protection switching is only initiated if signal fail is still present at the end of the hold-off time. The hold-off time shall be provisionable between 0 and 10 s in steps of 100 ms; this is defined in clause 11.12 of [ITU-T G.8031].

Protection switching can also be initiated by external switch commands received via the MP or a request from the far end via the received ETH_CI_APS. Depending on the mode of operation, internal states (e.g., wait-to-restore) may also affect a switch-over.

See the switching algorithm described in [ITU-T G.8031].

Switching time

Refer to [ITU-T G.8031].

Switch restoration

In the revertive mode of operation, the protected signal shall be switched back from the protection (sub)network connection to the working (sub)network connection when the working (sub)network connection has recovered from the fault.

To prevent frequent operation of the protection switch due to an intermittent fault, a failed working (sub)network connection must become fault-free for a certain period of time before it is used again. This period, called the wait-to-restore (WTR) period, should be of the order of 5-12 minutes and should be capable of being set. The WTR is defined in clause 11.13 of [ITU-T G.8031].

In the non-revertive mode of operation no switch back to the working (sub)network connection is performed when it has recovered from the fault.

Configuration

The following configuration parameters are defined in [ITU-T G.8031]:

- ETH_C_MI_PS_WorkingPortId configures the working port.
- ETH_C_MI_PS_ProtectionPortId configures the protection port.
- ETH_C_MI_PS_ProtType configures the protection type.
- ETH_C_MI_PS_OperType configures to be in revertive mode.
- ETH_C_MI_PS_HoTime configures the hold-off timer.
- ETH_C_MI_PS_WTR configures the wait-to-restore timer.
- ETH_C_MI_PS_ExtCMD configures the protection group command.

– ETH_C_MI_PS_BridgeType configures the type of bridge used for 1:1 SNC protection switching.
– ETH_C_MI_PS_SD_Protection configures the ability of an SNC protection switching process to trigger protection switching upon SD.

Reporting

The following reporting parameters are defined in [ITU-T G.8031]:

ETH_C_MI_PS_RequestState

ETH_C_MI_PS_RequestedSignal

ETH_C_MI_PS_BridgedSignal

Defects

The function detects dFOP-PM, dFOP-CM, dFOP-NR and dFOP-TO defects in case the APS protocol is used.

Consequent actions

None.

Defect correlations

cFOP-PM ← dFOP-PM and (not CI_SSF)

cFOP-CM ← dFOP-CM

cFOP-NR ← dFOP-NR and (not CI_SSF)

cFOP-TO ← dFOP-TO and (not dFOP-CM) and (not CI_SSF)

NOTE 2 – In case of cFOP-PM/NR/TO, CI_SSF of the protection transport entity is used.

9.1.3 Ring protection control process

Ring protection with inherent, sub-layer, or test trail monitoring is supported.

Figure 9-11 shows a subset of the atomic functions involved, and the signal flows associated with the ring protection control process. This is only an overview of the Ethernet ring protection control process as specified in [ITU-T G.8032]. The ETH_FT_Sk provides the TSF protection switching criterion via the ETHDi/ETH_A_Sk function (SSF). [ITU-T G.8032] specifies the requirements, options and the ring protection protocol supported by the ring protection control process.

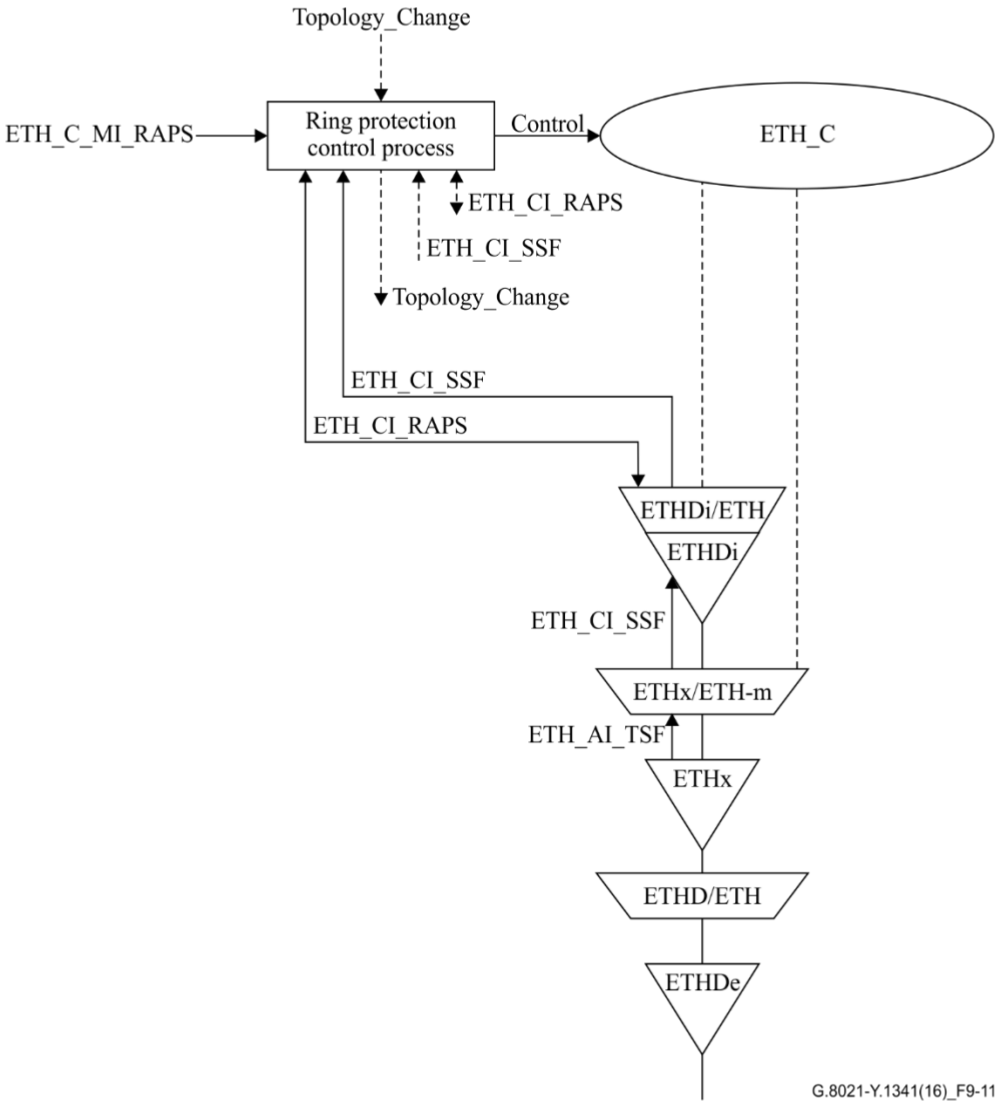


Figure 9-11 – Ring protection atomic functions and control process

Configuration

The following configuration parameters are defined in [ITU-T G.8032]:

– ETH_C_MI_RAPS_PortIds[0..1] associates the given ring port (0 / 1) to an ETH flow point in the ETH_FF function controlled by the ERP control process.

NOTE 1– ETH_C_MI_RAPS_PortIds is set by the EMF based on ERP configuration and is not exposed to the operator as a configuration parameter of the equipment management interface.

– ETH_C_MI_RAPS_RPL_Owner_Node configures the node type.

– ETH_C_MI_RAPS_RPL_Neighbour_Node configures the adjacency of a node to the RPL owner.

– ETH_C_MI_RAPS_Propagate_TC[1…M] configures the flush logic of an interconnection node.

– ETH_C_MI_RAPS_Compatible_Version configures the backward compatibility logic.

– ETH_C_MI_RAPS_Revertive configures the revertive mode.

– ETH_C_MI_RAPS_Sub_Ring_Without_Virtual_Channel configures the sub-ring type.

– ETH_C_MI_RAPS_HoTime configures the hold-off timer.

- ETH_C_MI_RAPS_WTR configures the wait-to-restore timer.
- ETH_C_MI_RAPS_GuardTime configures the guard timer.
- ETH_C_MI_RAPS_ExtCMD configures the protection command.
- ETH_C_MI_RAPS_RingID configures the Ring ID.

Reporting

The following reporting parameters are defined in [ITU-T G.8032]:

- ETH_C_MI_RAPS_NodeState reports the current ring node state.
- ETH_C_MI_RAPS_PortState[0..1] reports the given ring port's forwarding state.

Defects

The function detects dFOP-PM and dFOP-TO[0..1] in case the R-APS protocol is used.

Consequent actions

None.

Defect correlations

cFOP-PM ← dFOP-PM

cFOP-TO[i] ← dFOP-TO[i] and (not CI_SSF) and (not RAPS_Block)

NOTE 2 – As indicated in [ITU-T G.8032], cFOP-TO is not reported if a ring port has a link level failure (operationally disabled), or is administratively locked or blocked from R-APS message reception. The ETHDi/ETH_A signals the CI_SSF, when a ring port has a link level failure (operationally disabled), or it is administratively locked. The Ring Protection Control Process signals the RAPS_Block, when a ring port is blocked from R-APS message reception. Clause 10.4 of [ITU-T G.8032] describes examples of the RAPS_Block condition.

9.2 ETH termination functions

9.2.1 ETHx flow termination functions (ETHx_FT)

The bidirectional ETH flow termination (ETHx_FT) function is performed by a co-located pair of ETH flow termination source (ETHx_FT_So) and sink (ETHx_FT_Sk) functions.

9.2.1.1 ETHx flow termination source function (ETHx_FT_So)

Symbol

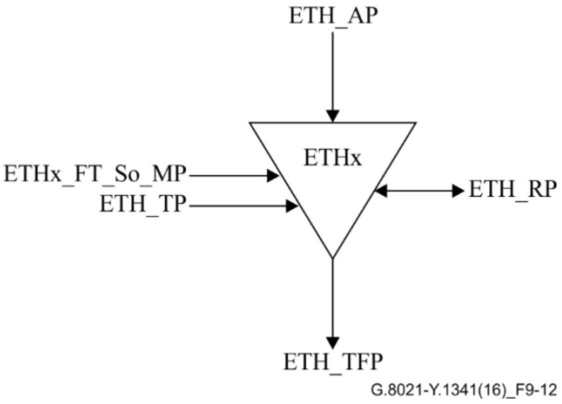


Figure 9-12 – ETHx_FT_So symbol

Interfaces

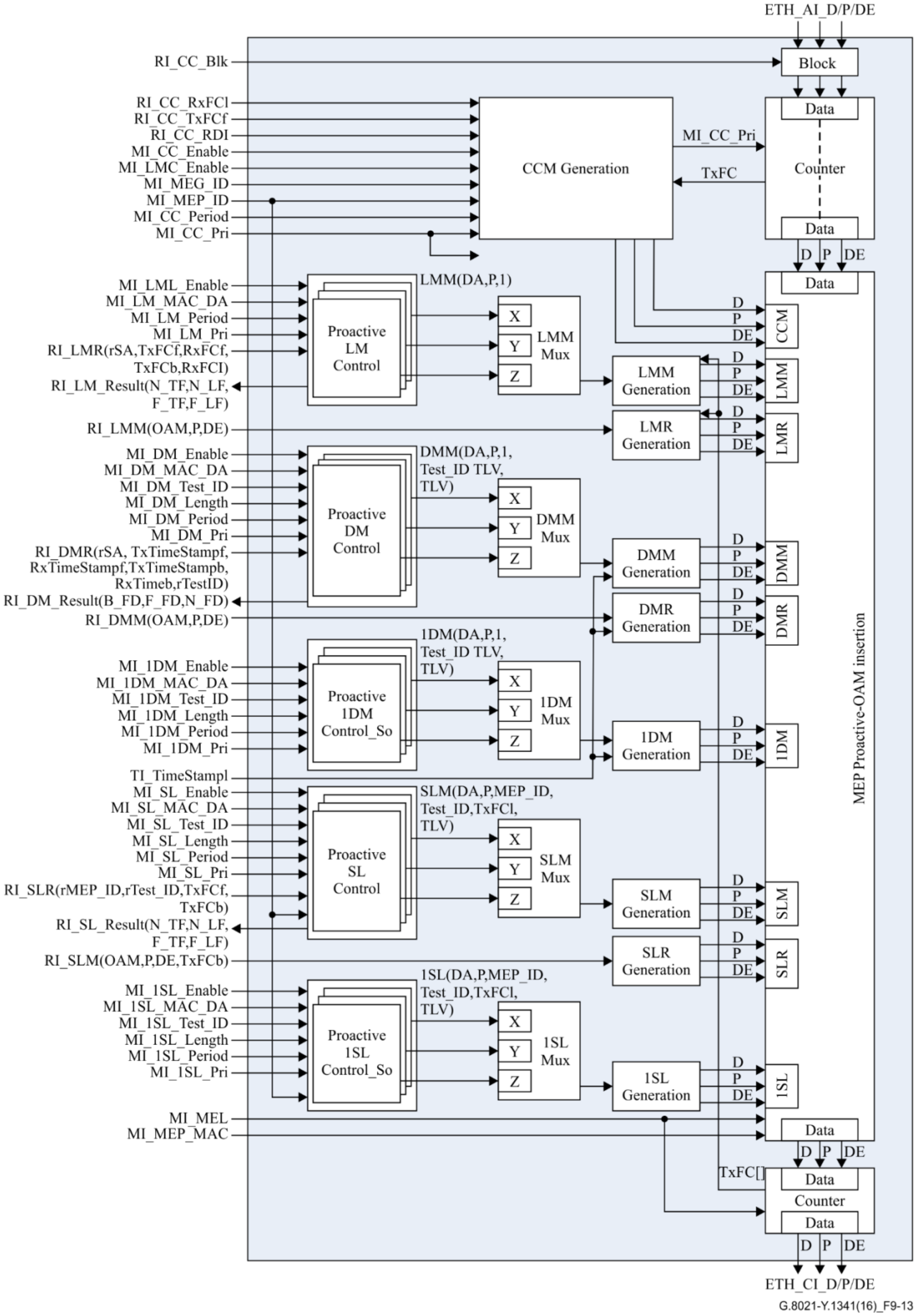
Table 9-2 – ETHx_FT_So interfaces

Inputs	Outputs
ETH_AP: ETH_AI_D ETH_AI_P ETH_AI_DE	**ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE
ETH_RP: ETH_RI_CC_RxFCl ETH_RI_CC_TxFCf ETH_RI_CC_RDI ETH_RI_CC_Blk ETH_RI_LMM(OAM,P,DE) ETH_RI_LMR(rSA,TxFCf,RxFCf,TxFCb,RxFCl) [1...M_{LM}] ETH_RI_DMM(OAM,P,DE) ETH_RI_DMR(rSA,TxTimeStampf, RxTimeStampf,TxTimeStampb,RxTimeb, rTestID)[1...M_{DM}] ETH_RI_SLM(OAM,P,DE,TxFCb) ETH_RI_SLR(rMEP_ID,rTest_ID, TxFCf, TxFCb) [1...M_{SL}]	**ETH_RP**: ETH_RI_LM_Result(N_TF,N_LF,F_TF,F_LF) [1...M_{LM}] ETH_RI_DM_Result(B_FD,F_FD,N_FD) [1...M_{DM}] ETH_RI_SL_Result(N_TF,N_LF,F_TF,F_LF) [1...M_{SL}]
ETH_TP: ETHx_FT_So_TI_TimeStampl	
ETHx_FT_So_MP: ETHx_FT_So_MI_MEL ETHx_FT_So_MI_MEP_MAC ETHx_FT_So_MI_CC_Enable ETHx_FT_So_MI_LMC_Enable ETHx_FT_So_MI_MEG_ID ETHx_FT_So_MI_MEP_ID ETHx_FT_So_MI_CC_Period ETHx_FT_So_MI_CC_Pri ETHx_FT_So_MI_LML_Enable[1...M_{LM}] ETHx_FT_So_MI_LM_MAC_DA[1...M_{LM}] ETHx_FT_So_MI_LM_Period[1...M_{LM}] ETHx_FT_So_MI_LM_Pri[1...M_{LM}] ETHx_FT_So_MI_DM_Enable[1...M_{DM}] ETHx_FT_So_MI_DM_MAC_DA[1...M_{DM}] ETHx_FT_So_MI_DM_Test_ID[1...M_{DM}] ETHx_FT_So_MI_DM_Length[1...M_{DM}] ETHx_FT_So_MI_DM_Period[1...M_{DM}] ETHx_FT_So_MI_DM_Pri[1...M_{DM}] ETHx_FT_So_MI_1DM_Enable[1...M_{1DM}] ETHx_FT_So_MI_1DM_MAC_DA[1...M_{1DM}] ETHx_FT_So_MI_1DM_Test_ID[1...M_{1DM}] ETHx_FT_So_MI_1DM_Length[1...M_{1DM}]	

Table 9-2 – ETHx_FT_So interfaces

Inputs	Outputs
ETHx_FT_So_MI_1DM_Period[1...M_{1DM}]	
ETHx_FT_So_MI_1DM_Pri[1...M_{1DM}]	
ETHx_FT_So_MI_SL_Enable[1...M_{SL}]	
ETHx_FT_So_MI_SL_MAC_DA[1...M_{SL}]	
ETHx_FT_So_MI_SL_Test_ID[1...M_{SL}]	
ETHx_FT_So_MI_SL_Length[1...M_{SL}]	
ETHx_FT_So_MI_SL_Period[1...M_{SL}]	
ETHx_FT_So_MI_SL_Pri[1...M_{SL}]	
ETHx_FT_So_MI_1SL_Enable[1...M_{1SL}]	
ETHx_FT_So_MI_1SL_MAC_DA[1...M_{1SL}]	
ETHx_FT_So_MI_1SL_Test_ID[1...M_{1SL}]	
ETHx_FT_So_MI_1SL_Length[1...M_{1SL}]	
ETHx_FT_So_MI_1SL_Period[1...M_{1SL}]	
ETHx_FT_So_MI_1SL_Pri[1...M_{1SL}]	

Processes



G.8021-Y.1341(16)_F9-13

Figure 9-13 – ETHx_FT_So process

MEP proactive OAM insertion process

This process inserts the OAM traffic units in the stream of ETH_CI, sets the MEL field to MI_MEL and sets the SA field to MI_MEP_MAC.

If the DA of the OAM traffic unit is a class 1 multicast DA, the OAM insertion process updates the DA to reflect the correct MEL.

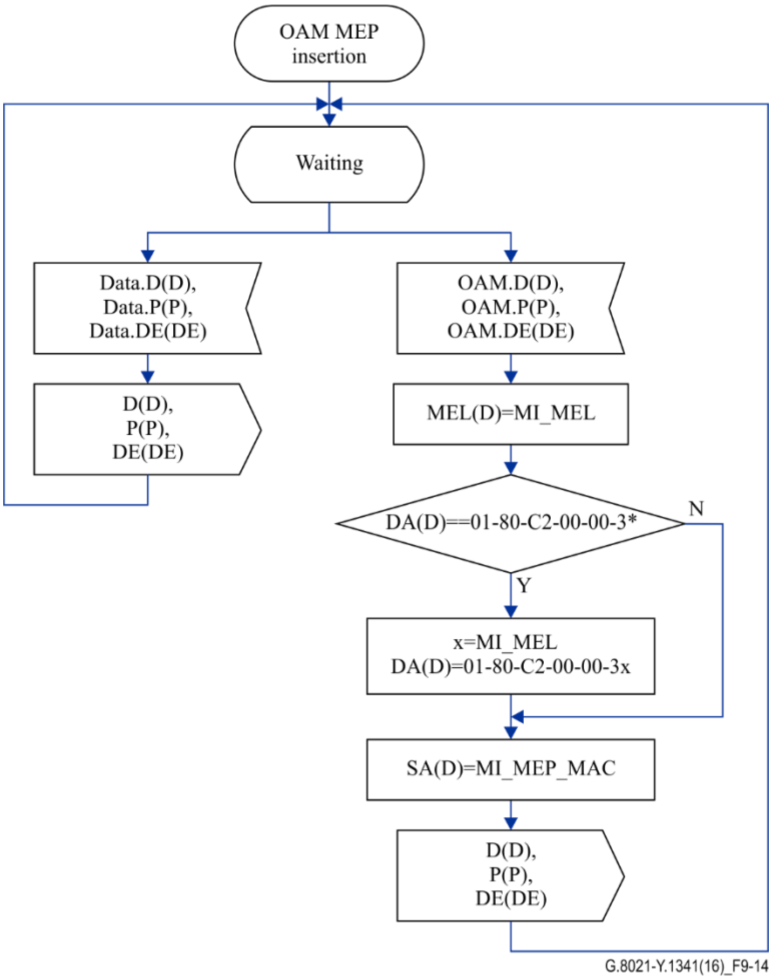


Figure 9-14 – OAM MEP insertion behaviour

CCM generation process

This process is defined in clause 8.1.7 where the CC protocol is defined. Clause 8.1.7.2 defines the CCM generation process.

Block process

When RI_CC_Blk is raised, the block process will discard all ETH_CI information it receives. If RI_CC_Blk is cleared, the received ETH_CI information will be passed to the output port.

Counter process

This process is defined in clauses 8.1.7.4 and 8.1.9.7. It is used to count frames for proactive loss measurements with CCM and proactive LM protocols, respectively.

Proactive LM control

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.2 defines the proactive LM control process.

LMM generation

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.3 defines the LMM generation process.

LMR generation

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.5 defines the LMR generation process.

LMM Mux

The LMM Mux process interleaves the signal sets LMM(DA,P,1) from the input ports (X, Y, Z).

Proactive DM control

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.2 defines the DM control process.

DMM generation

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.3 defines the DMM generation process.

DMR generation

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.5 defines the DMR generation process.

DMM Mux

The DMM Mux process interleaves the signal sets DMM(DA,P,1,Test ID TLV, TLV) from the input ports (X, Y, Z).

Proactive 1DM Control_So

This process is defined in clause 8.1.11 where the 1DM protocol is defined. Clause 8.1.11.2 defines the 1DM Control_So process.

1DM generation

This process is defined in clause 8.1.11 where the 1DM protocol is defined. Clause 8.1.11.3 defines the 1DM generation process.

1DM Mux

The 1DM Mux process interleaves the signal sets 1DM(DA,P,1,Test ID TLV, TLV) from the input ports (X, Y, Z).

Proactive SL control

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.2 defines the SL control process.

SLM generation

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.3 defines the SLM generation process.

SLR Generation

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.5 defines the SLR generation process.

SLM Mux

The SLM Mux process interleaves the signal sets SLM(DA,P,MEP_ID,Test_ID,TxFCl,TLV) from the input ports (X, Y, Z).

Proactive 1SL Control_So

This process is defined in clause 8.1.15 where the 1SL protocol is defined. Clause 8.1.15.2 defines the 1SL Control_So process.

1SL generation

This process is defined in clause 8.1.15 where the 1SL protocol is defined. Clause 8.1.15.3 defines the 1SL generation process.

1SL Mux

The 1SL Mux process interleaves the signal sets 1SL(DA,P, MEP_ID,Test_ID, TxFCl, TLV) from the input ports (X, Y, Z).

Defects	None.
Consequent actions	None.
Defect correlations	None.
Performance monitoring	None.

9.2.1.2 ETHx flow termination sink function (ETHx_FT_Sk)

The ETHx_FT_Sk process diagram is shown in Figure 9-15.

Symbol

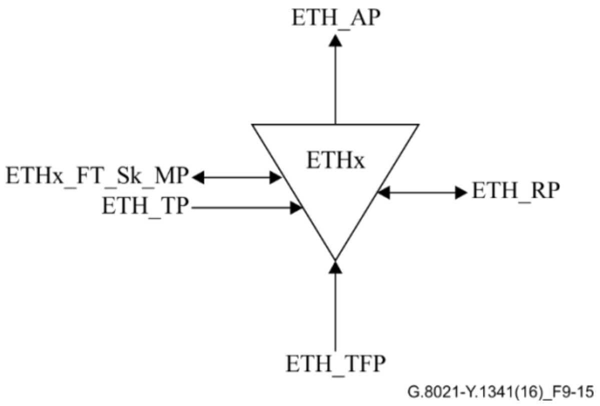


Figure 9-15 – ETHx_FT_Sk symbol

Interfaces

Table 9-3 – ETHx_FT_Sk interfaces

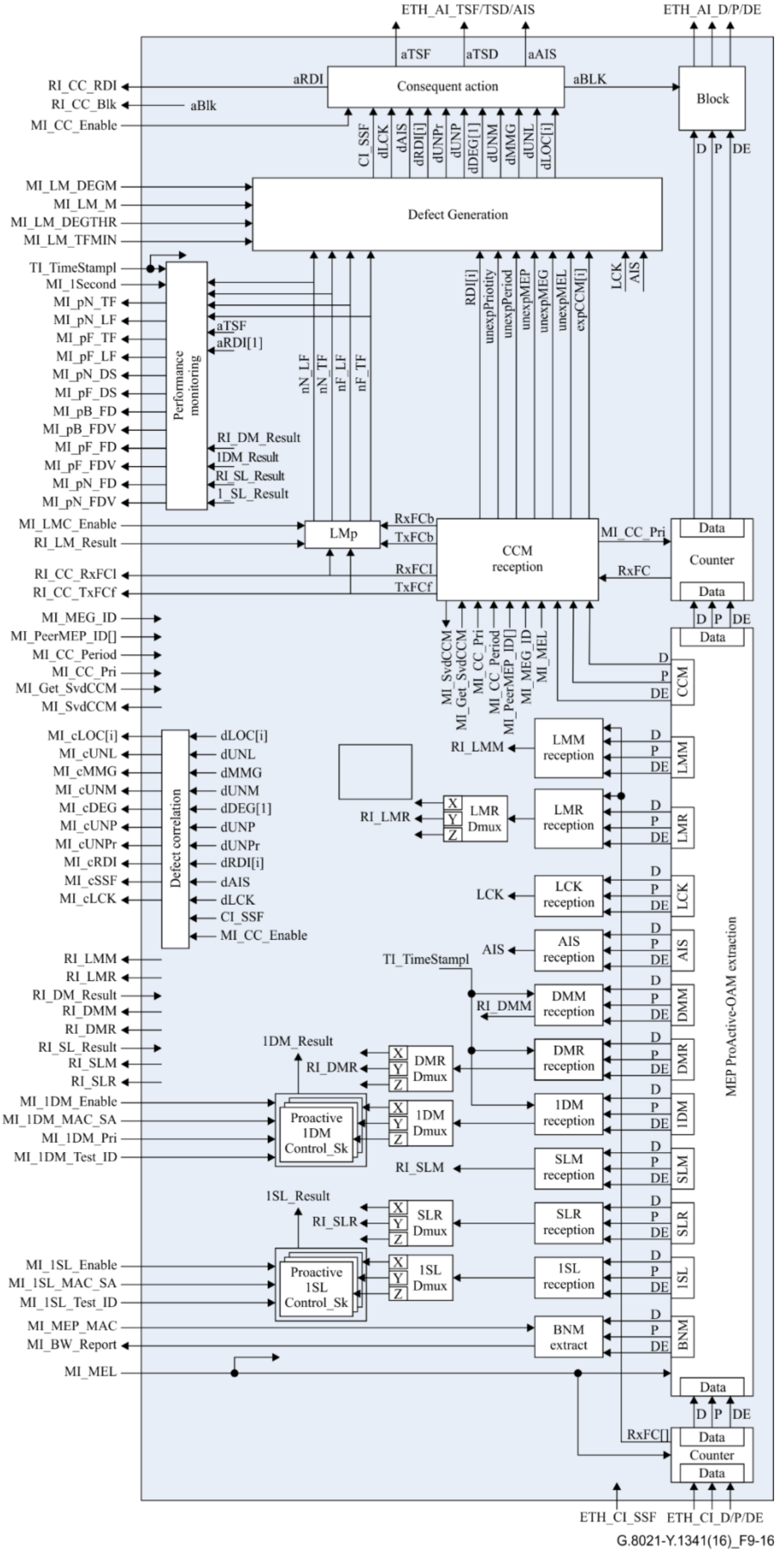
Inputs	Outputs
ETH_TFP: ETH_CI_D ETH_CI_P	**ETH_AP**: ETH_AI_D ETH_AI_P

Table 9-3 – ETHx_FT_Sk interfaces

Inputs	Outputs
ETH_CI_DE ETH_CI_SSF **ETH_RP**: ETH_RI_LM_Result(N_TF,N_LF,F_TF,F_LF) [1...M_{LM}] ETH_RI_DM_Result(B_FD,F_FD,N_FD) [1...M_{DM}] ETH_RI_SL_Result(N_TF,N_LF,F_TF,F_LF) [1...M_{SL}] **ETH_TP:** ETHx_FT_Sk_TI_TimeStampl **ETHx_FT_Sk_MP**: ETHx_FT_Sk_MI_CC_Enable ETHx_FT_Sk_MI_LMC_Enable ETHx_FT_Sk_MI_1Second ETHx_FT_Sk_MI_LM_DEGM ETHx_FT_Sk_MI_LM_M ETHx_FT_Sk_MI_LM_DEGTHR ETHx_FT_Sk_MI_LM_TFMIN ETHx_FT_Sk_MI_MEL ETHx_FT_Sk_MI_MEG_ID ETHx_FT_Sk_MI_PeerMEP_ID[i] ETHx_FT_Sk_MI_CC_Period ETHx_FT_Sk_MI_CC_Pri ETHx_FT_Sk_MI_GetSvdCCM ETHx_FT_Sk_MI_1DM_Enable[1...M_{1DM}] ETHx_FT_Sk_MI_1DM_MAC_SA[1...M_{1DM}] ETHx_FT_Sk_MI_1DM_Pri[1...M_{1DM}] ETHx_FT_Sk_MI_1DM_Test_ID[1...M_{1DM}] ETHx_FT_Sk_MI_1SL_Enable[1...M_{1SL}] ETHx_FT_Sk_MI_1SL_MAC_SA[1...M_{1SL}] ETHx_FT_Sk_MI_1SL_Test_ID[1...M_{1SL}] ETHx_FT_Sk_MI_MEP_MAC	ETH_AI_DE ETH_AI_TSF ETH_AI_TSD ETH_AI_AIS **ETH_RP**: ETH_RI_CC_RxFCl ETH_RI_CC_TxFCf ETH_RI_CC_RDI ETH_RI_CC_Blk ETH_RI_LMM(OAM,P,DE) ETH_RI_LMR(rSA,TxFCf,RxFCf,TxFCb,RxFCl) [1...M_{LM}] ETH_RI_DMM(OAM,P,DE) ETH_RI_DMR(rSA,TxTimeStampf, RxTimeStampf,TxTimeStampb,RxTimeb, rTestID) [1...M_{DM}] ETH_RI_SLM(OAM,P,DE,TxFCb) ETH_RI_SLR(rMEP_ID,rTest_ID,TxFCf, TxFCb) [1...M_{SL}] **ETHx_FT_Sk_MP**: ETHx_FT_Sk_MI_cLOC[i] ETHx_FT_Sk_MI_cUNL ETHx_FT_Sk_MI_cMMG ETHx_FT_Sk_MI_cUNM ETHx_FT_Sk_MI_cDEG ETHx_FT_Sk_MI_cUNP ETHx_FT_Sk_MI_cUNPr ETHx_FT_Sk_MI_cRDI ETHx_FT_Sk_MI_cSSF ETHx_FT_Sk_MI_cLCK ETHx_FT_Sk_MI_pN_TF ETHx_FT_Sk_MI_pN_LF ETHx_FT_Sk_MI_pF_TF ETHx_FT_Sk_MI_pF_LF ETHx_FT_Sk_MI_pF_DS ETHx_FT_Sk_MI_pN_DS ETHx_FT_Sk_MI_pB_FD ETHx_FT_Sk_MI_pB_FDV ETHx_FT_Sk_MI_pF_FD ETHx_FT_Sk_MI_pF_FDV ETHx_FT_Sk_MI_pN_FD ETHx_FT_Sk_MI_pN_FDV ETHx_FT_Sk_MI_SvdCCM ETHx_FT_Sk_MI_BW_Report(SA, PortID, NominalBW, CurrentBW)

NOTE 1 – If the delay measurement message rate is smaller than one second, there will be more than one set of primitive values (i.e., pB_FD, pB_FDV, pF_FD, pF_FDV, pN_FD, pN_FDV) for some 1-second period. If the delay measurement message rate is larger than one second, there will be no set of primitive values for some 1-second period.

Processes



G.8021-Y.1341(16)_F9-16

Figure 9-16 – ETHx_FT_Sk process

MEP proactive OAM extraction process

The MEP proactive OAM extraction process extracts OAM traffic units that are processed in the ETHx_FT_Sk process from the stream of traffic units according to the following pseudo code:

```
if (TYPE=<ETHOAM>) and (MEL=MI_MEL) then
  switch(OPC) {
  case <CCM>: extract ETH-CCM OAM traffic unit and forward to CCM Port
  case <AIS>: extract ETH-AIS OAM traffic unit and forward to AIS Port
  case <LCK>: extract ETH-LCK OAM traffic unit and forward to LCK Port
  case <LMM>: extract ETH-LMM OAM traffic unit and forward to LMM Port
  case <LMR>: extract ETH-LMR OAM traffic unit and forward to LMR Port
  case <DMM>: extract ETH-DMM OAM traffic unit and forward to DMM Port
  case <DMR>: extract ETH-DMR OAM traffic unit and forward to DMR Port
  case <1DM>: extract ETH-1DM OAM traffic unit and forward to 1DM Port
  case <SLM>: extract ETH-SLM OAM traffic unit and forward to SLM port
  case <SLR>: extract ETH-SLR OAM traffic unit and forward to SLR port
  case <1SL>: extract ETH-1SL OAM traffic unit and forward to 1SL Port
  case <GNM>: switch(SubOPC) {
    case <BNM>: extract ETH-BN OAM traffic unit and forward to BNM Port
    default: forward ETH_CI traffic unit to Data port
  }
  default: forward ETH_CI traffic unit to Data port
  }
elseif (TYPE=<ETH0AM>) and (MEL<MI_MEL) and (OPC=CCM) then
  extract ETH-CCM OAM traffic unit and forward to CCM Port
else
  forward ETH CI traffic unit to Data Port
endif
```

NOTE 2 – Further filtering of OAM traffic units is performed by the OAM MEL filter process which forms part of the ETH adaptation functions specified in clause 9.3.

ETH_AIS reception process

This process generates the AIS event upon receipt of the AIS traffic unit from the OAM MEP extraction process.

ETH_LCK reception process

This process generates the LCK event upon receipt of the LCK traffic unit from the OAM MEP extraction process.

LMM reception

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.4 defines the LMM reception process.

LMR reception

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.6 defines the LMR reception process.

LMR Demux

The LMR Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P signal can be used for the selection of the port.

DMM reception

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.4 defines the DMM reception process.

DMR reception

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.6 defines the DMR reception process.

DMR Demux

The DMR Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P and/or Test_ID signal can be used for the selection of the port.

1DM reception

This process is defined in clause 8.1.11 where the 1DM protocol is defined. Clause 8.1.11.4 defines the 1DM reception process.

1DM Demux

The 1DM Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P and/or Test_ID signal can be used for the selection of the port.

Proactive 1DM Control_Sk

This process is defined in clause 8.1.11 where the 1DM protocol is defined. Clause 8.1.11.5 defines the 1DM Control_Sk process.

SLM reception

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.4 defines the SLM reception process.

SLR reception

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.6 defines the SLR reception process.

SLR Demux

The SLR Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P and/or Test_ID signal can be used for the selection of the port.

1SL reception

This process is defined in clause 8.1.15 where the 1SL protocol is defined. Clause 8.1.15.4 defines the 1SL reception process.

1SL Demux

The 1SL Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P and/or Test_ID signal can be used for the selection of the port.

Proactive 1SL Control_Sk

This process is defined in clause 8.1.15 where the 1SL protocol is defined. Clause 8.1.15.5 defines the 1SL Control_Sk process.

Block process

When aBlk is raised, the block process will discard all ETH_CI information it receives. If aBLK is cleared, the received ETH_CI information will be passed to the output port.

LMp process

This process is defined in clause 8.1.7.5.

Defect generation process

This process detects and clears the defects (dLOC[i], dUNL, dMMG, dUNM, dDEG, dUNP, dUNPr, dRDI[i], dAIS, dLCK) as defined in clause 6, where [i] = maintenance entity.

CCM reception process

This process is defined in clause 8.1.7.3.

Counter process

This process is defined in clauses 8.1.7.4 and 8.1.9.7. It is used to count frames for proactive loss measurements with CCM and proactive LM protocols, respectively.

BNM Extract process

This process is defined in clause 8.1.19.

Defects

This function detects dLOC[i], dUNL, dMMG, dUNM, dDEG, dUNP, dUNPr, dRDI[i], dAIS, dLCK.

Consequent actions

aBLK ← (dUNL or dMMG or dUNM)

Note that dUNP and dUNPr does not contribute to aBLK because a mismatch of periodicity is not considered to be a security issue.

aTSF ← (dLOC[1..n] and MI_CC_Enable) or (dAIS and not(MI_CC_Enable)) or (dLCK and not(MI_CC_Enable)) or dUNL or dMMG or dUNM or CI_SSF

aTSD ← dDEG[1] and (not aTSF)

aAIS ← aTSF

aRDI ← aTSF

Defect correlations

cLOC[i] ← dLOC[i] and (not dAIS) and (not dLCK) and (not CI_SSF) and (MI_CC_Enable)

cUNL ← dUNL

cMMG ← dMMG

cUNM ← dUNM

cDEG[1] ← dDEG[1] and (not dAIS) and (not dLCK) and (not CI_SSF) and (not (dLOC[1..n] or dUNL or dMMG or dUNM)) and (MI_CC_Enable))

cUNP ← dUNP

cUNPr ← dUNPr

cRDI ← (dRDI[1..n]) and (MI_CC_Enable)

cSSF ← CI_SSF or dAIS

cLCK ← dLCK and (not dAIS)

Performance monitoring

pN_TF ← N_TF
pN_LF ← N_LF
pF_TF ← F_TF
pF_LF ← F_LF
pN_DS ← aTSF
pF_DS ← aRDI[1]
pB_FD ← B_FD
pB_FDV ← B_FDV
pF_FD ← F_FD
pF_FDV ← F_FDV
pN_FD ← N_FD
pN_FDV ← N_FDV

NOTE 3 – A detail calculation formula for FDV is for further study.

9.2.2 ETH group flow termination functions (ETHG_FT)

The bidirectional ETH group flow termination (ETHG_FT) function is performed by a co-located pair of ETH group flow termination source (ETHG_FT_So) and sink (ETHG_FT_Sk) functions.

9.2.2.1 ETH group flow termination source function (ETHG_FT_So)

Symbol

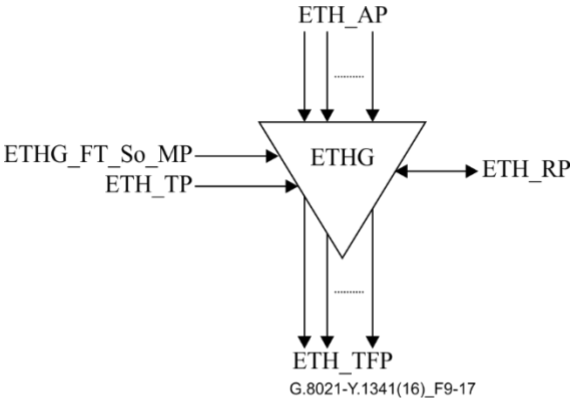


Figure 9-17 – ETHG_FT_So symbol

Interfaces

Table 9-4 – ETHG_FT_So interfaces

Inputs	Outputs
ETH_AP: ETH_AI_D[1...M] ETH_AI_P[1...M] ETH_AI_DE[1...M] **ETH_RP**:	**ETH_TFP**: ETH_CI_D[1...M] ETH_CI_P[1...M] ETH_CI_DE[1...M] **ETH_RP**:

Table 9-4 – ETHG_FT_So interfaces

Inputs	Outputs
ETH_RI_CC_RxFCl ETH_RI_CC_TxFCf ETH_RI_CC_RDI ETH_RI_CC_Blk ETH_RI_LMM(OAM,P,DE) ETH_RI_LMR(rSA,TxFCf,RxFCf,TxFCb,RxFCl) [1...M_{LM}] ETH_RI_DMM(OAM,P,DE) ETH_RI_DMR(rSA,TxTimeStampf, RxTimeStampf,TxTimeStampb,RxTimeb, rTestID) [1...M_{DM}] ETH_RI_SLM(OAM,P,DE,TxFCb) ETH_RI_SLR(rMEP_ID,rTest_ID, TxFCf, TxFCb) [1...M_{SL}] **ETH_TP:** ETHG_FT_So_TI_TimeStampl **ETHG_FT_So_MP**: ETHG_FT_So_MI_MEL ETHG_FT_So_MI_MEP_MAC ETHG_FT_So_MI_CC_Enable ETHG_FT_So_MI_LMC_Enable ETHG_FT_So_MI_MEG_ID ETHG_FT_So_MI_MEP_ID ETHG_FT_So_MI_CC_Period ETHG_FT_So_MI_CC_Pri ETHG_FT_So_MI_LML_Enable[1...M_{LM}] ETHG_FT_So_MI_LM_MAC_DA[1...M_{LM}] ETHG_FT_So_MI_LM_Period[1...M_{LM}] ETHG_FT_So_MI_LM_Pri [1...M_{LM}] ETHG_FT_So_MI_DM_Enable [1...M_{DM}] ETHG_FT_So_MI_DM_MAC_DA [1...M_{DM}] ETHG_FT_So_MI_DM_Test_ID [1...M_{DM}] ETHG_FT_So_MI_DM_Length [1...M_{DM}] ETHG_FT_So_MI_DM_Period [1...M_{DM}] ETHG_FT_So_MI_DM_Pri [1...M_{DM}] ETHG_FT_So_MI_1DM_Enable [1...M_{1DM}] ETHG_FT_So_MI_1DM_MAC_DA [1...M_{1DM}] ETHG_FT_So_MI_1DM_Test_ID [1...M_{1DM}] ETHG_FT_So_MI_1DM_Length [1...M_{1DM}] ETHG_FT_So_MI_1DM_Period [1...M_{1DM}] ETHG_FT_So_MI_1DM_Pri [1...M_{1DM}] ETHG_FT_So_MI_SL_Enable [1...M_{SL}] ETHG_FT_So_MI_SL_MAC_DA [1...M_{SL}] ETHG_FT_So_MI_SL_Test_ID [1...M_{SL}]	ETH_RI_LM_Result(N_TF,N_LF,F_TF,F_LF) [1...M_{LM}] ETH_RI_DM_Result(B_FD,F_FD,N_FD) [1...M_{DM}] ETH_RI_SL_Result(N_TF,N_LF,F_TF,F_LF) [1...M_{SL}]

Table 9-4 – ETHG_FT_So interfaces

Inputs	**Outputs**
ETHG_FT_So_MI_SL_Length [1...M_{SL}] ETHG_FT_So_MI_SL_Period [1...M_{SL}] ETHG_FT_So_MI_SL_Pri [1...M_{SL}] ETHG_FT_So_MI_1SL_Enable [1...M_{1SL}] ETHG_FT_So_MI_1SL_MAC_DA [1...M_{1SL}] ETHG_FT_So_MI_1SL_Test_ID [1...M_{1SL}] ETHG_FT_So_MI_1SL_Length [1...M_{1SL}] ETHG_FT_So_MI_1SL_Period [1...M_{1SL}] ETHG_FT_So_MI_1SL_Pri [1...M_{1SL}]	

Processes

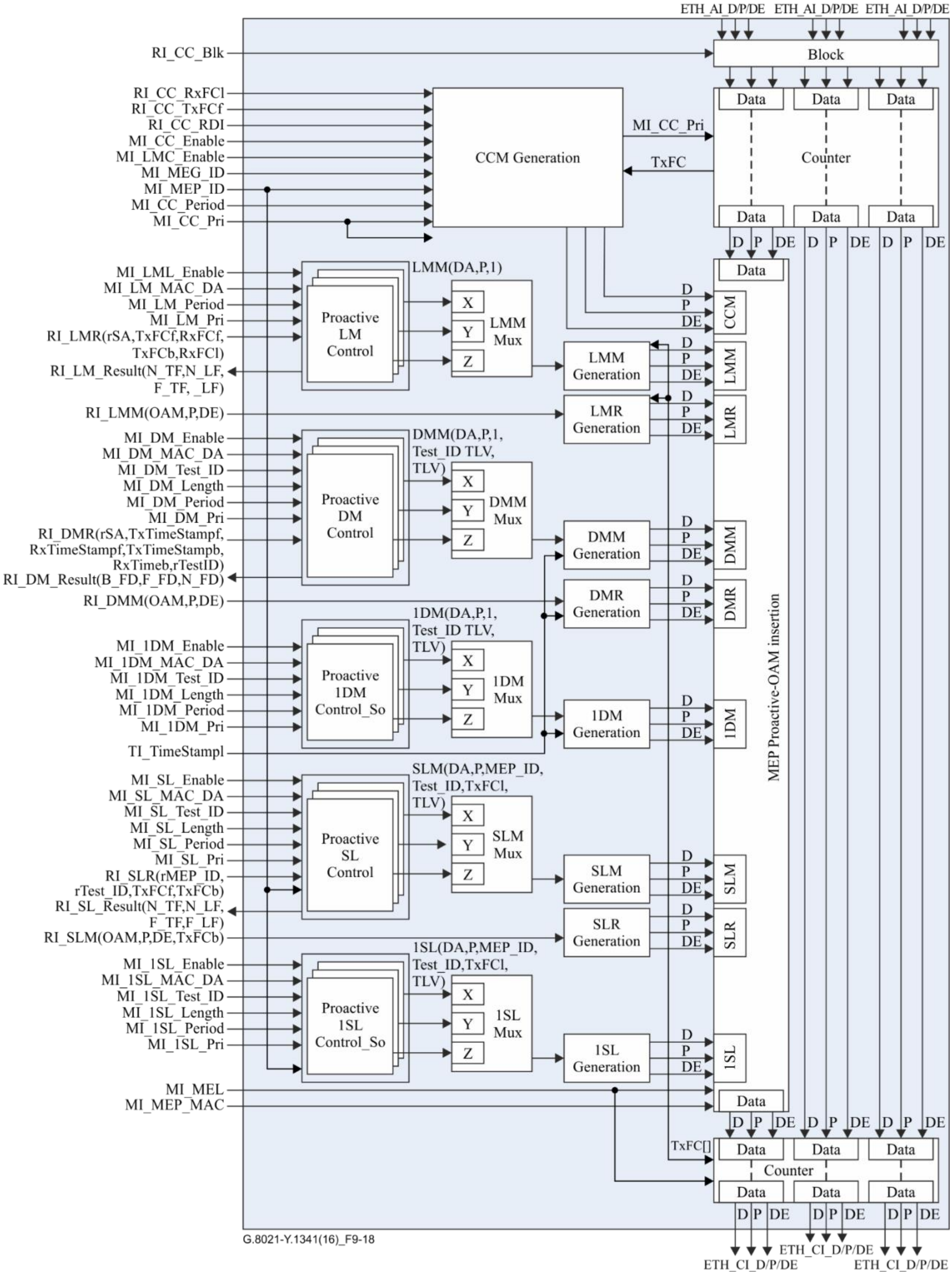


Figure 9-18 – ETHG_FT_So process

MEP proActive OAM insertion process

This process inserts the OAM traffic units in the stream of ETH_CI, sets the MEL field to MI_MEL and sets the SA field to MI_MEP_MAC. This process resides only in the lowest number in the contiguous range of ETH_FPs or a selected ETH_FP within the group of arbitrary ETH_FPs. The detail of the OAM insertion behaviour is described in clause 9.2.1.1.

CCM generation process

This process is defined in clause 8.1.7 where the CC protocol is defined. Clause 8.1.7.2 defines the CCM generation process.

Block process

When RI_CC_Blk is raised, the block process will discard all ETH_CI information within the group of co-located flow points. If RI_CC_Blk is cleared, the received ETH_CI information will be passed to the output port.

Counter process

This process is defined in clauses 8.1.7.4 and 8.1.9.7. It is used to count frames for proactive loss measurements with CCM and proactive LM protocols, respectively.

Proactive LM control

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.2 defines the proactive LM control process.

LMM generation

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.3 defines the LMM generation process.

LMR generation

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.5 defines the LMR generation process.

LMM Mux

The LMM Mux process interleaves the signal sets LMM(DA,P,1) from the input ports (X, Y, Z).

Proactive DM control

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.2 defines the DM control process.

DMM generation

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.3 defines the DMM generation process.

DMR generation

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.5 defines the DMR generation process.

DMM Mux

The DMM Mux process interleaves the signal sets DMM(DA,P,1,Test ID TLV, TLV) from the input ports (X, Y, Z).

Proactive 1DM Control_So

This process is defined in clause 8.1.11 where the 1DM protocol is defined. Clause 8.1.11.2 defines the 1DM Control_So process.

1DM generation

This process is defined in clause 8.1.11 where the 1DM protocol is defined. Clause 8.1.11.3 defines the 1DM generation process.

1DM Mux

The 1DM Mux process interleaves the signal sets 1DM(DA,P,1,Test ID TLV, TLV) from the input ports (X, Y, Z).

Proactive SL control

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.2 defines the SL control process.

SLM generation

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.3 defines the SLM generation process.

SLR generation

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.5 defines the SLR generation process.

SLM Mux

The SLM Mux process interleaves the signal sets SLM(DA,P,MEP_ID,Test_ID,TxFCl,TLV) from the input ports (X, Y, Z).

Proactive 1SL Control_So

This process is defined in clause 8.1.15 where the 1SL protocol is defined. Clause 8.1.15.2 defines the 1SL Control_So process.

1SL generation

This process is defined in clause 8.1.15 where the 1SL protocol is defined. Clause 8.1.15.3 defines the 1SL generation process.

1SL Mux

The 1SL Mux process interleaves the signal sets 1SL(DA,P,Test_ID,MEP_ID,TxFCl, TLV) from the input ports (X, Y, Z).

Defects	None.
Consequent actions	None.
Defect correlations	None.
Performance monitoring	None.

9.2.2.2 ETH group flow termination sink function (ETHG_FT_Sk)

The ETHG_FT_Sk process diagram is shown in Figure 9-19.

Symbol

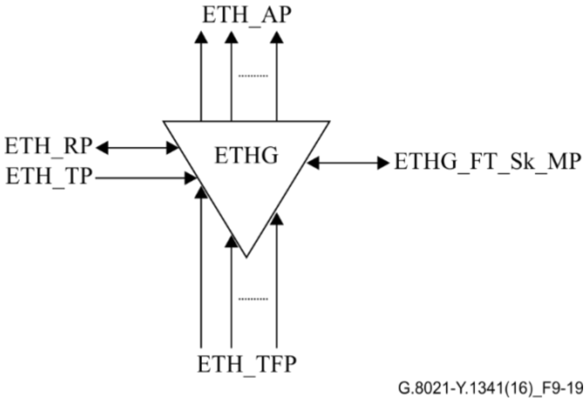


Figure 9-19 – ETHG_FT_Sk symbol

Interfaces

Table 9-5 – ETHG_FT_Sk interfaces

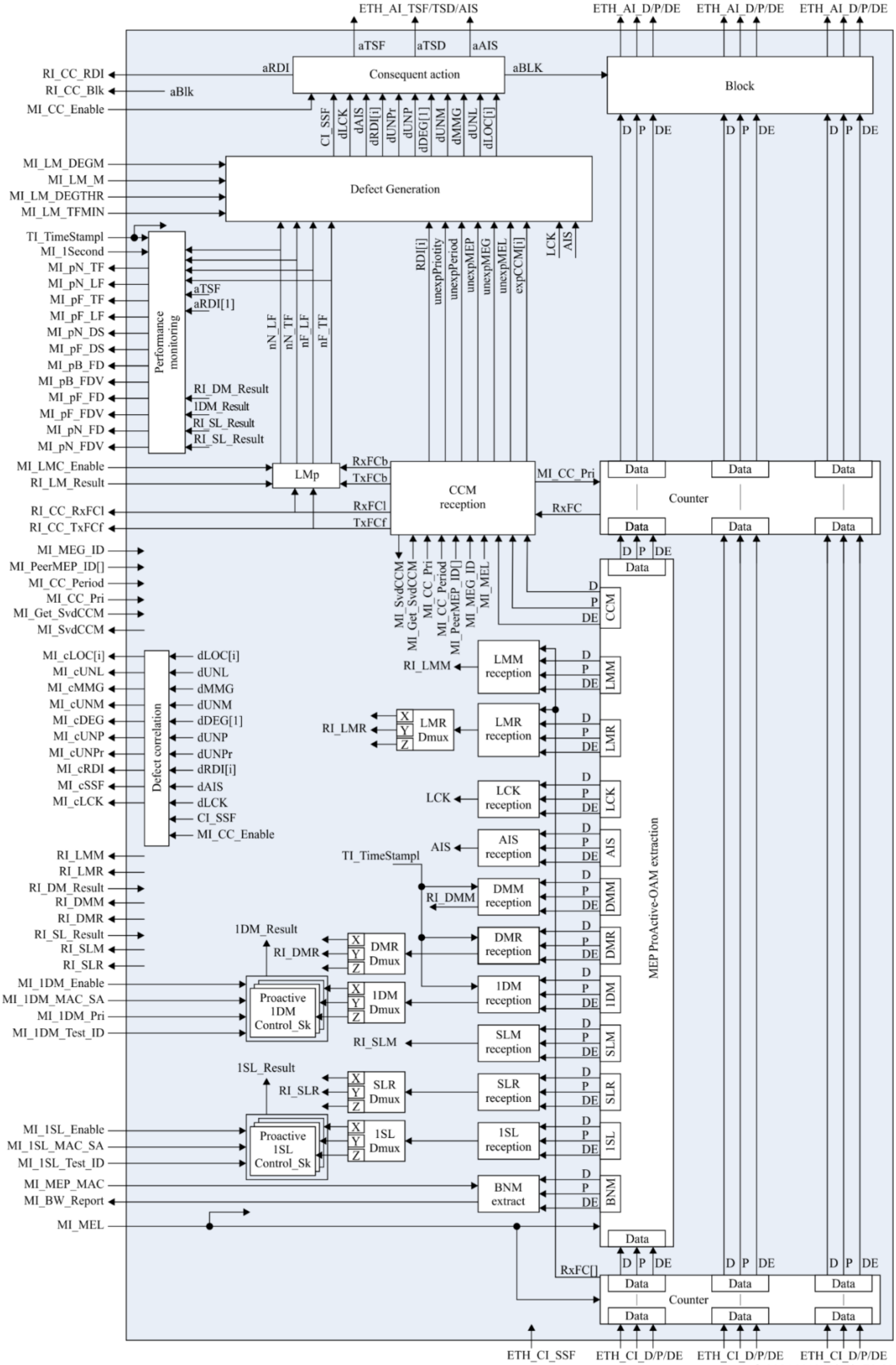
Inputs	Outputs
ETH_TFP: ETH_CI_D[1...M] ETH_CI_P[1...M] ETH_CI_DE[1...M] ETH_CI_SSF **ETH_RP**: ETH_RI_LM_Result(N_TF,N_LF,F_TF,F_LF) [1...M_{LM}] ETH_RI_DM_Result(B_FD,F_FD,N_FD) [1...M_{DM}] ETH_RI_SL_Result(N_TF,N_LF,F_TF,F_LF) [1...M_{SL}] **ETH_TP:** ETHG_FT_Sk_TI_TimeStampl **ETHG_FT_Sk_MP**: ETHG_FT_Sk_MI_CC_Enable ETHG_FT_Sk_MI_LMC_Enable ETHG_FT_Sk_MI_1Second ETHG_FT_Sk_MI_LM_DEGM ETHG_FT_Sk_MI_LM_M ETHG_FT_Sk_MI_LM_DEGTHR ETHG_FT_Sk_MI_LM_TFMIN ETHG_FT_Sk_MI_MEL ETHG_FT_Sk_MI_MEG_ID ETHG_FT_Sk_MI_PeerMEP_ID[i] ETHG_FT_Sk_MI_CC_Period ETHG_FT_Sk_MI_CC_Pri ETHG_FT_Sk_MI_GetSvdCCM	**ETH_AP**: ETH_AI_D[1...M] ETH_AI_P[1...M] ETH_AI_DE[1...M] ETH_AI_TSF ETH_AI_TSD ETH_AI_AIS **ETH_RP**: ETH_RI_CC_RxFCl ETH_RI_CC_TxFCf ETH_RI_CC_RDI ETH_RI_CC_Blk ETH_RI_LMM(OAM,P,DE) ETH_RI_LMR(rSA,TxFCf,RxFCf,TxFCb,RxFCl) [1...M_{LM}] ETH_RI_DMM(OAM,P,DE) ETH_RI_DMR(rSA,TxTimeStampf, RxTimeStampf,TxTimeStampb,RxTimeb, rTestID) [1...M_{DM}] ETH_RI_SLM(OAM,P,DE,TxFCb) ETH_RI_SLR(rMEP_ID,rTest_ID,TxFCf, TxFCb) [1...M_{SL}] **ETHG_FT_Sk_MP**: ETHG_FT_Sk_MI_cLOC[i] ETHG_FT_Sk_MI_cUNL ETHG_FT_Sk_MI_cMMG ETHG_FT_Sk_MI_cUNM ETHG_FT_Sk_MI_cDEG ETHG_FT_Sk_MI_cUNP

Table 9-5 – ETHG_FT_Sk interfaces

Inputs	Outputs
ETHG_FT_Sk_MI_1DM_Enable [1...M_{1DM}] ETHG_FT_Sk_MI_1DM_MAC_SA [1...M_{1DM}] ETHG_FT_Sk_MI_1DM_Pri [1...M_{1DM}] ETHG_FT_Sk_MI_1DM_Test_ID [1...M_{1DM}] ETHG_FT_Sk_MI_1SL_Enable [1...M_{1SL}] ETHG_FT_Sk_MI_1SL_MAC_SA [1...M_{1SL}] ETHG_FT_Sk_MI_1SL_Test_ID [1...M_{1SL}] ETHG_FT_Sk_MI_MEP_MAC	ETHG_FT_Sk_MI_cUNPr ETHG_FT_Sk_MI_cRDI ETHG_FT_Sk_MI_cSSF ETHG_FT_Sk_MI_cLCK ETHG_FT_Sk_MI_pN_TF ETHG_FT_Sk_MI_pN_LF ETHG_FT_Sk_MI_pF_TF ETHG_FT_Sk_MI_pF_LF ETHG_FT_Sk_MI_pF_DS ETHG_FT_Sk_MI_pN_DS ETHG_FT_Sk_MI_pB_FD ETHG_FT_Sk_MI_pB_FDV ETHG_FT_Sk_MI_pF_FD ETHG_FT_Sk_MI_pF_FDV ETHG_FT_Sk_MI_pN_FD ETHG_FT_Sk_MI_pN_FDV ETHG_FT_Sk_MI_SvdCCM ETHG_FT_Sk_MI_BW_Report(SA, PortID, NominalBW, CurrentBW)

NOTE – If the delay measurement message rate is smaller than one second, there will be more than one set of primitive values (i.e., pB_FD, pB_FDV, pF_FD, pF_FDV, pN_FD, pN_FDV) for some 1-second period. If the delay measurement message rate is larger than one second, there will be no set of primitive values for some 1-second period.

Processes



G.8021-Y.1341(16)_F9-20

Figure 9-20 – ETHG_FT_Sk process

MEP proactive OAM extraction process

The MEP proactive OAM extraction process extracts OAM traffic units that are processed in the ETHx_FT_Sk process from the stream of traffic units. This process resides only in the lowest number in the contiguous range of ETH_FPs or a selected ETH_FP within the group of arbitrary ETH_FPs (AIS reception, LCK reception, LMp, and defect generation processes as well). The details of this process are described in clause 9.2.1.2.

AIS reception process

This process generates the AIS event upon receipt of the AIS traffic unit from the OAM MEP extraction process.

LCK reception process

This process generates the LCK event upon receipt of the LCK traffic unit from the OAM MEP extraction process.

LMM reception

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.4 defines the LMM reception process.

LMR reception

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.6 defines the LMR reception process.

LMR Demux

The LMR Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P signal can be used for the selection of the port.

DMM reception

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.4 defines the DMM reception process.

DMR reception

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.6 defines the DMR reception process.

DMR Demux

The DMR Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P and/or Test_ID signal can be used for the selection of the port.

1DM reception

This process is defined in clause 8.1.11 where the 1DM protocol is defined. Clause 8.1.11.4 defines the 1DM reception process.

1DM Demux

The 1DM Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P and/or Test_ID signal can be used for the selection of the port.

Proactive 1DM Control_Sk

This process is defined in clause 8.1.11 where the 1DM protocol is defined. Clause 8.1.11.5 defines the 1DM Control_Sk process.

SLM reception

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.4 defines the SLM reception process.

SLR reception

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.6 defines the SLR reception process.

SLR Demux

The SLR Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P and/or Test_ID signal can be used for the selection of the port.

1SL reception

This process is defined in clause 8.1.15 where the 1SL protocol is defined. Clause 8.1.15.4 defines the 1SL reception process.

1SL Demux

The 1SL Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P and/or Test_ID signal can be used for the selection of the port.

Proactive 1SL Control_Sk

This process is defined in clause 8.1.15 where the 1SL protocol is defined. Clause 8.1.15.5 defines the 1SL Control_Sk process.

Block process

When aBlk is raised, the block process will discard all ETH_CI information within the group of co-located flow points. If aBLK is cleared, the received ETH_CI information will be passed to the output port.

LMp process

This process is defined in clause 8.1.7.4.

Defect generation process

This process detects and clears the defects (dLOC[i], dUNL, dMMG, dUNM, dDEG, dUNP, dUNPr, dRDI[i], dAIS, dLCK) as defined in clause 6, where [i] = maintenance entity.

CCM reception process

This process is defined in clause 8.1.7.3.

Counter process

This process is defined in clauses 8.1.7.4 and 8.1.9.7. It is used to count frames for proactive loss measurements with CCM and proactive LM protocols, respectively.

BNM Extract process

This process is defined in clause 8.1.19.

Defects	See clause 9.2.1.2.
Consequent actions	See clause 9.2.1.2.
Defect correlations	See clause 9.2.1.2.
Performance monitoring	See clause 9.2.1.2.

9.3 ETH adaptation functions

9.3.1 ETH to client adaptation functions (ETH/<client>_A)

For further study.

9.3.2 ETH to ETH adaptation functions (ETHx/ETH_A)

9.3.2.1 ETH to ETH adaptation source function (ETHx/ETH_A_So)

This function maps client ETH_CI traffic units into server ETH_AI traffic units.

Symbol

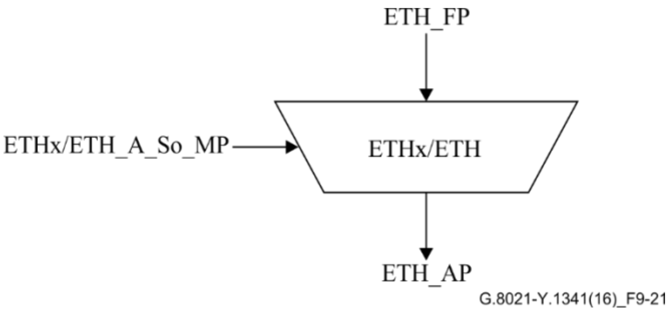


Figure 9-21 – ETHx/ETH_A_So symbol

Interfaces

Table 9-6 – ETHx/ETH_A_So interfaces

Inputs	Outputs
ETH_FP: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_APS ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi **ETHx/ETH_A_So_MP**: ETHx/ETH_A_So_MI_MEP_MAC ETHx/ETH_A_So_MI_Client_MEL ETHx/ETH_A_So_MI_LCK_Period ETHx/ETH_A_So_MI_LCK_Pri ETHx/ETH_A_So_MI_Admin_State ETHx/ETH_A_So_MI_MEL ETHx/ETH_A_So_MI_APS_Pri ETHx/ETH_A_So_MI_CSF_Period ETHx/ETH_A_So_MI_CSF_Pri ETHx/ETH_A_So_MI_CSF_Enable ETHx/ETH_A_So_MI_CSFrdifdiEnable ETHx/ETH_A_So_MI_CSFdciEnable	**ETH_AP**: ETH_AI_D ETH_AI_P ETH_AI_DE

Processes

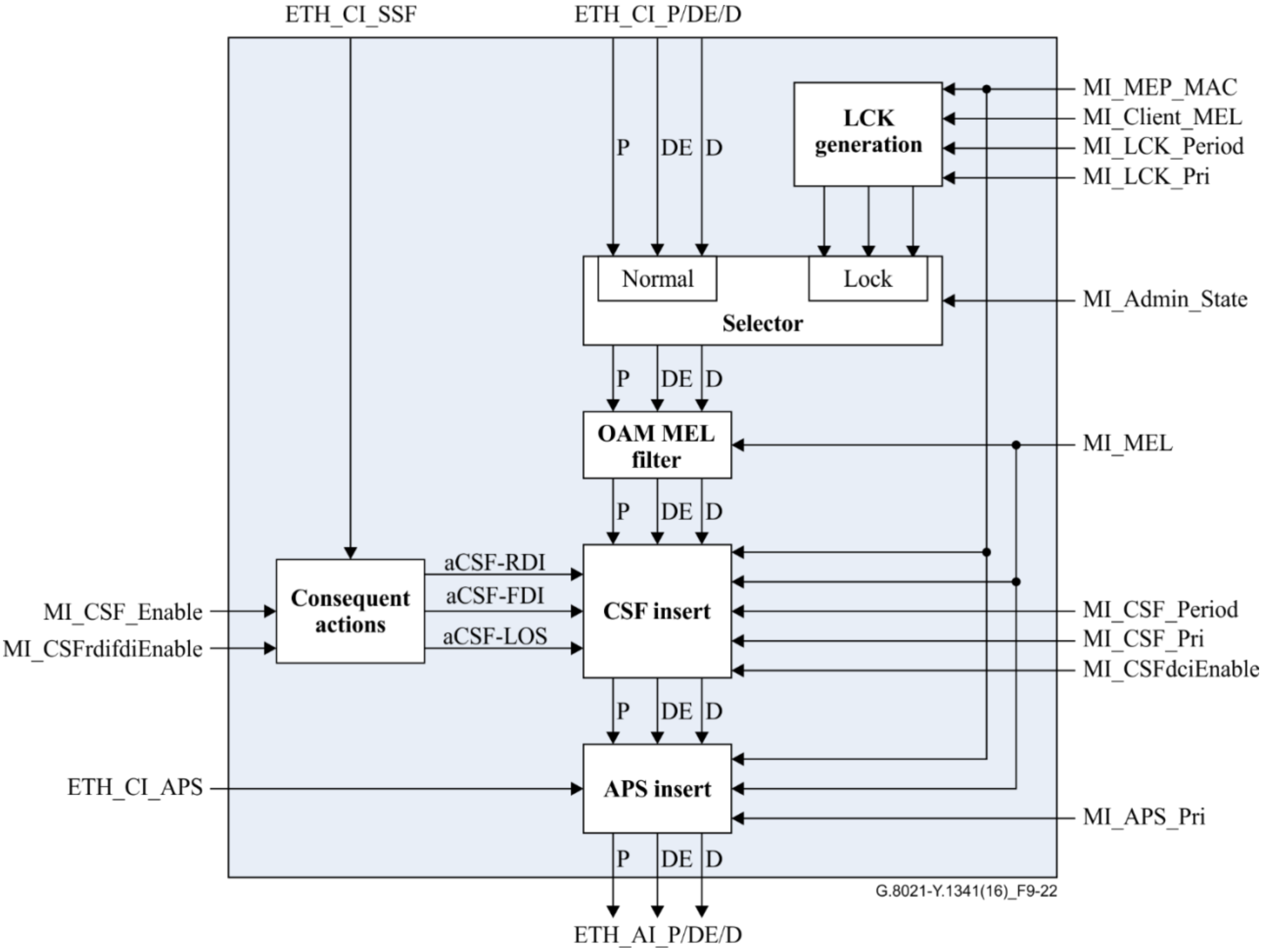


Figure 9-22 – ETHx/ETH_A_So process

LCK generation process

As defined in clause 8.1.2.

Selector process

As defined in clause 8.1.3.

OAM MEL filter process

As defined in clause 8.1.1.

CSF insert process

As defined in clause 8.1.16.

APS insert process

As defined in clause 8.1.5.

When this process is activated, LCK admin state shall be unlocked. See clause 7.5.2.2 of [ITU-T G.8010].

Defects None.

Consequent actions

aCSF-LOS ← CI_SSF and MI_CSFEnable

aCSF-RDI ← CI_SSFrdi and MI_CSFrdifdiEnable and MI_CSFEnable

aCSF-FDI ← CI_SSFfdi and MI_CSFrdifdiEnable and MI_CSFEnable

Defect correlations None.

Performance monitoring None.

9.3.2.2 ETH to ETH adaptation sink function (ETHx/ETH_A_Sk)

This function retrieves client ETH_CI traffic units from server ETH_AI traffic units.

Symbol

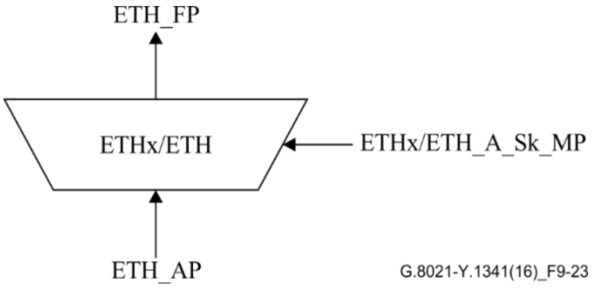


Figure 9-23 – ETHx/ETH_A_Sk symbol

Interfaces

Table 9-7 – ETHx/ETH_A_Sk interfaces

Inputs	Outputs
ETH_AP: ETH_AI_D ETH_AI_P ETH_AI_DE ETH_AI_TSF ETH_AI_TSD ETH_AI_AIS **ETHx/ETH_A_Sk_MP**: ETHx/ETH_A_Sk_MI_MEP_MAC ETHx/ETH_A_Sk_MI_Client_MEL ETHx/ETH_A_Sk_MI_LCK_Period ETHx/ETH_A_Sk_MI_LCK_Pri ETHx/ETH_A_Sk_MI_Admin_State ETHx/ETH_A_Sk_MI_AIS_Period ETHx/ETH_A_Sk_MI_AIS_Pri ETHx/ETH_A_Sk_MI_MEL ETHx/ETH_A_Sk_MI_CSF_Reported ETHx/ETH_A_Sk_MI_CSFrdifdiEnable	**ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_APS ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi ETH_CI_SSD **ETHx/ETH_A_Sk_MP**: ETHx/ETH_A_Sk_MI_cCSF

Processes

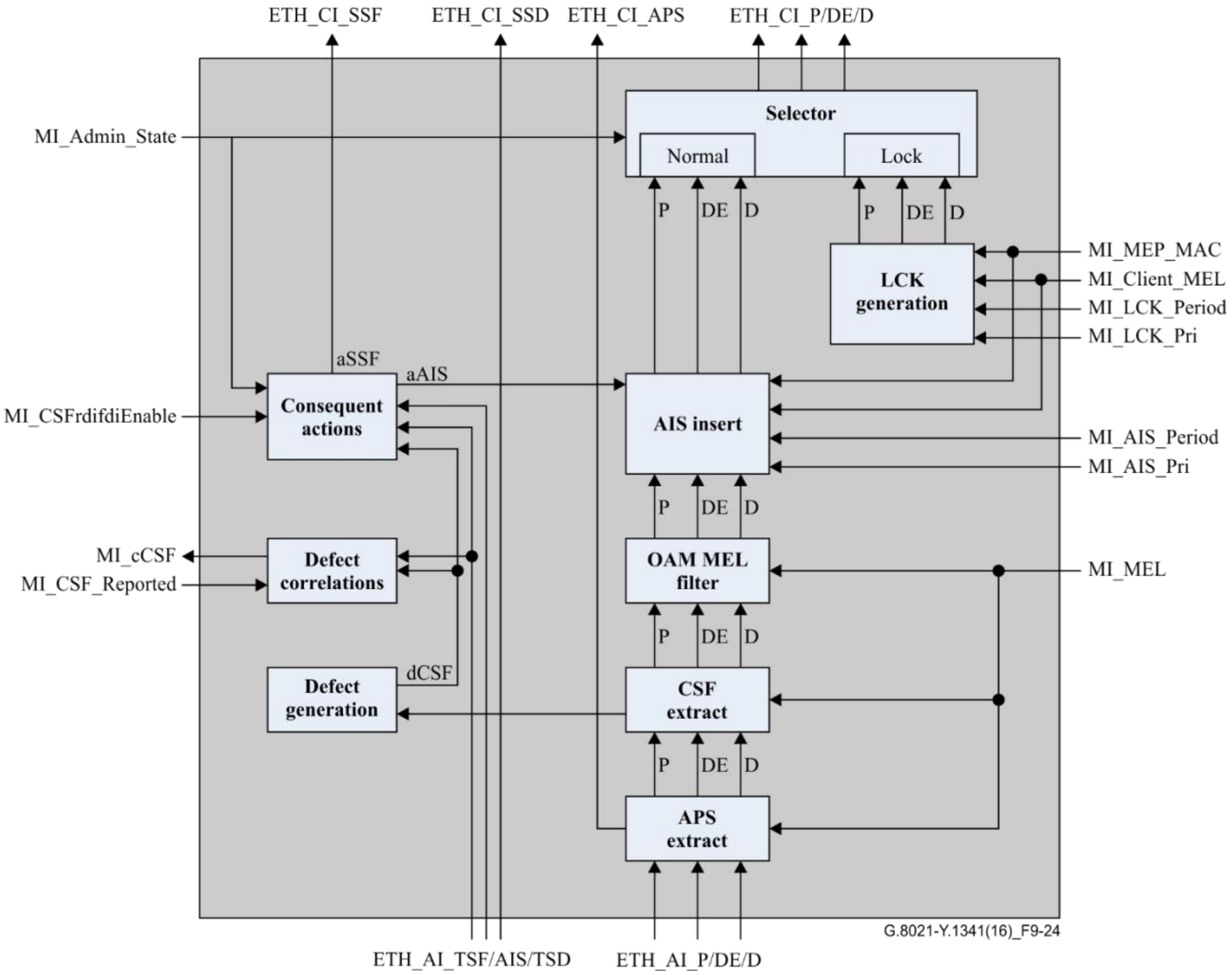


Figure 9-24 – ETHx/ETH_A_Sk process

APS extract process

As defined in clause 8.1.6.

CSF extract process

As defined in clause 8.1.17.

OAM MEL filter process

As defined in clause 8.1.1.

AIS insert process

As defined in clause 8.1.4.

LCK generation process

As defined in clause 8.1.2.

Selector process

As defined in clause 8.1.3.

Defects

dCSF-LOS – See clause 6.1.5.4.

dCSF-RDI – See clause 6.1.5.4.

dCSF-FDI – See clause 6.1.5.4.

Consequent actions

aSSF ← (AI_TSF or dCSF-LOS) and (not MI_Admin_State == Locked)

aSSFrdi ← dCSF-RDI and MI_CSFrdifdiEnable

aSSFfdi ← dCSF-FDI and MI_CSFrdifdiEnable

aAIS ← AI_AIS

Defect correlations

cCSF ← (dCSF-LOS or dCSF-RDI or dCSF-FDI) and (not AI_TSF) and MI_CSF_Reported

Performance monitoring None.

9.3.3 ETH to ETH multiplexing adaptation functions (ETHx/ETH-m_A)

This adaptation function multiplexes different ETH_CI streams into a single ETH_AI stream in the source direction and demultiplexes the ETH_AI stream into individual ETH_CI streams.

Symbol

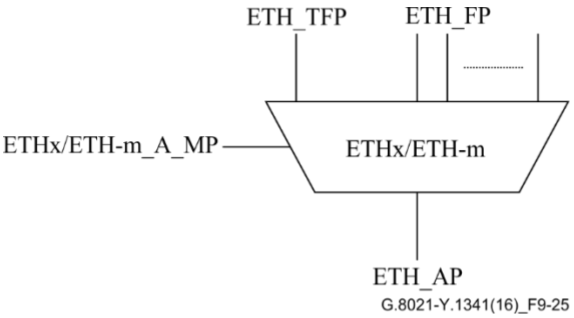


Figure 9-25 – ETHx/ETH-m_A symbol

The ETHx/ETH-m_A (Figure 9-25) function is further decomposed into separate source and sink adaptation functions that are interconnected as shown in Figure 9-26.

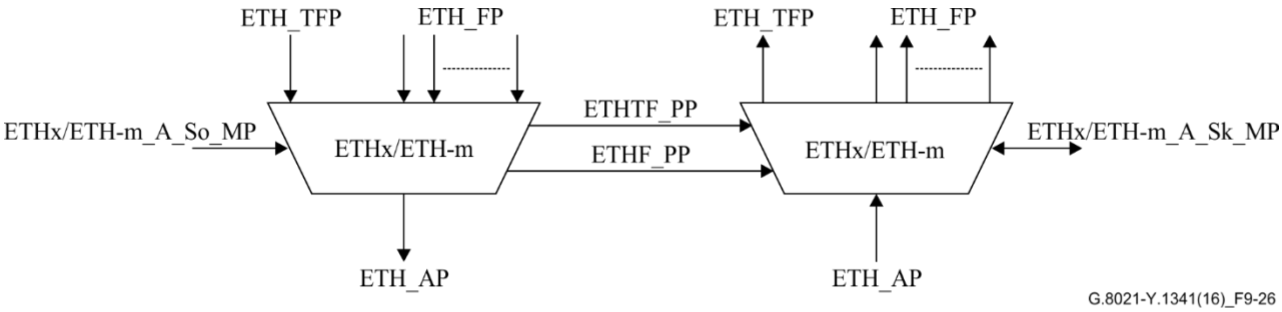


Figure 9-26 – ETHx/ETH-m_A source and sink symbols

9.3.3.1 ETH to ETH multiplexing adaptation source function (ETHx/ETH-m_A_So)

This function multiplexes individual ETH_CI streams into a single ETH_AI stream.

Symbol

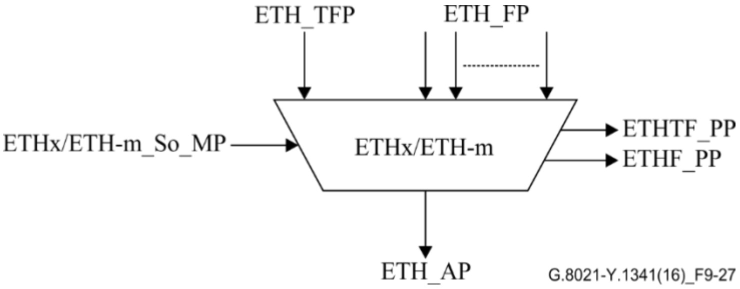


Figure 9-27 – ETHx/ETH-m_A_So symbol

Interfaces

Table 9-8 – ETHx/ETH-m_A_So interfaces

Inputs	Outputs
ETH_FP: ETH_CI_D[1...M] ETH_CI_P[1...M] ETH_CI_DE[1...M] ETH_CI_SSF[1] ETH_CI_SSFrdi[1] ETH_CI_SSFfdi[1] **ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE **ETHx/ETH-m_A_So_MP:** ETHx/ETH-m_A_So_MI_MEP_MAC ETHx/ETH-m_A_So_MI_Client_MEL[1…M] ETHx/ETH-m_A_So_MI_LCK_Period[1…M] ETHx/ETH-m_A_So_MI_LCK_Pri[1…M] ETHx/ETH-m_A_So_MI_Admin_State ETHx/ETH-m_A_So_MI_VLAN_Config[1...M] ETHx/ETH-m_A_So_MI_Etype ETHx/ETH-m_A_So_MI_PCP_Config ETHx/ETH-m_A_So_MI_MEL ETHx/ETH-m_A_So_MI_CSF_Period ETHx/ETH-m_A_So_MI_CSF_Pri ETHx/ETH-m_A_So_MI_CSF_Enable ETHx/ETH-m_A_So_MI_CSFrdifdiEnable ETHx/ETH-m_A_So_MI_CSFdciEnable	**ETH_AP**: ETH_AI_D ETH_AI_P ETH_AI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE

Processes

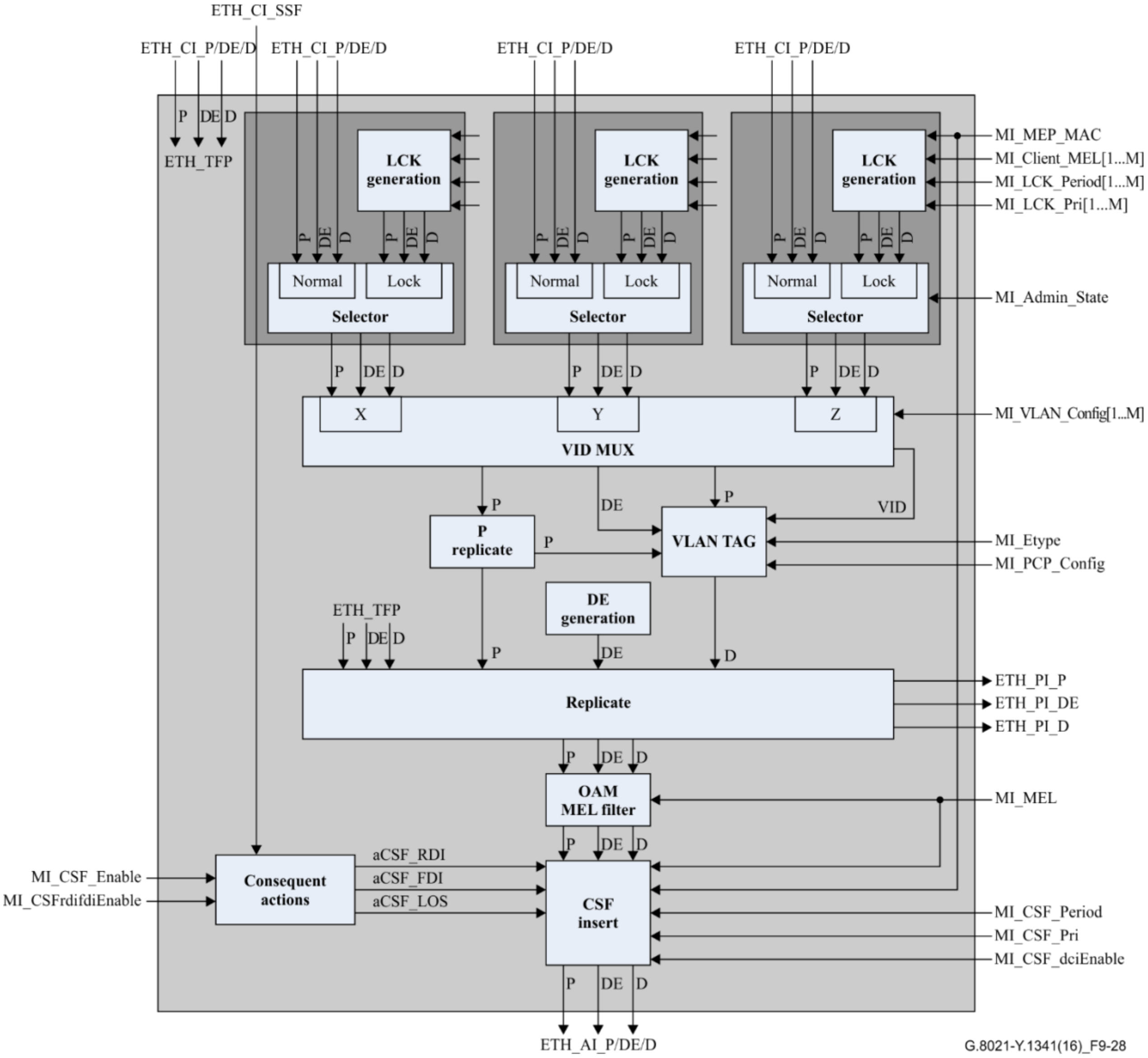


Figure 9-28 – ETHx/ETH-m_A_So process

LCK generation process

As defined in clause 8.1.2. Each FP has its LCK generation process.

Selector process

As defined in clause 8.1.3. The normal CI is blocked if Admin_State = LOCKED.

VID Mux process

The VID MUX process interleaves the signal sets (P, D, DE) from the input ports (X, Y, Z). For each incoming signal set on forwarding the signal set, a VID signal is generated. The value of the VID signal is based on the port on which the signal set is received and the configuration from the MI_VLAN_Config input parameter.

The MI_VLAN_Config input parameter determines for every input port the associated VID value. The allowed values for the VID signal are untagged, priority tagged and 1-4094. The following restriction applies to the allowed MI_VLAN_Config values:

- every VID value is only used once.

Note that IEEE 802.1 standards do not allow IEEE bridges to generate priority tagged frames. Priority tagged frames are only generated by end stations. However a C-VLAN bridge may create S-VLAN priority tagged frames.

VLAN tag process

This process inserts a VLAN tag into the M_SDU field of the incoming D signal. The EtherType used is determined by the value of the MI_Etype input parameter. The MI_PCP_Config signal determines the encoding of the P and DE signals in the VLAN tag. This parameter defines a mapping from P value to PCP value in the case of C-VLAN tags, and from P value to PCP and DEI value in the case of S-VLAN tags.

The VID signal determines the VID value in the VLAN tag. If the VID signal equals priority tagged, the VID value used is 0. If the VID signal equals untagged, no VLAN tag is inserted in the M_SDU field.

P replicate process

The P replicate process replicates the incoming P signal to both output ports without changing the value of the signal.

DE generation process

The DE generation process generates a DE signal with the value drop ineligible.

Replicate process

As defined in clause 8.4.

OAM MEL filter process

As defined in clause 8.1.1.

CSF insert process

As defined in clause 8.1.16. The ETHx/ETH-m adaptation function generates a single OAM flow while it can accommodate multiple ETH APs. In the case of using multiple APs, the CSF signal is supported at only a representative OAM flow.

Defects None.

Consequent actions

aCSF-LOS ← CI_SSF and MI_CSFEnable

aCSF-RDI ← CI_SSFrdi and MI_CSFrdifdiEnable and MI_CSFEnable

aCSF-FDI ← CI_SSFfdi and MI_CSFrdifdiEnable and MI_CSFEnable

Defect correlations None.

Performance monitoring None.

9.3.3.2 ETH to ETH multiplexing adaptation sink function (ETHx/ETH-m_A_Sk)

Symbol

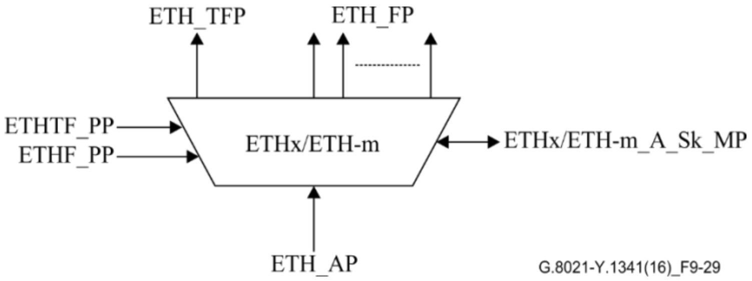


Figure 9-29 – ETHx/ETH-m_A_Sk symbol

Interfaces

Table 9-9 – ETHx/ETH-m_A_Sk interfaces

Inputs	Outputs
ETH_AP: ETH_AI_D ETH_AI_P ETH_AI_DE ETH_AI_TSF ETH_AI_AIS **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHx/ETH-m_A_Sk_MP**: ETHx/ETH-m_A_Sk_MI_MEP_MAC ETHx/ETH-m_A_Sk_MI_Client_MEL[1…M] ETHx/ETH-m _A_Sk_MI_LCK_Period[1…M] ETHx/ETH-m _A_Sk_MI_LCK_Pri[1…M] ETHx/ETH-m_A_Sk_MI_Admin_State ETHx/ETH-m_A_Sk_MI_AIS_Period[1…M] ETHx/ETH-m_A_Sk_MI_AIS_Pri[1…M] ETHx/ETH-m_A_Sk_MI_VLAN_Config[1...M] ETHx/ETH-m_A_Sk_MI_P_Regenerate ETHx/ETH-m_A_Sk_MI_PVID ETHx/ETH-m_A_Sk_MI_PCP_Config ETHx/ETH-m_A_Sk_MI_Etype ETHx/ETH-m_A_Sk_MI_MEL ETHx/ETH-m_A_Sk_MI_CSF_Reported ETHx/ETH-m_A_Sk_MI_CSFrdifdiEnable ETHx/ETH-m_A_Sk_MI_Frametype_Config ETHx/ETH-m_A_Sk_MI_Filter_Config	**ETH_FP**: ETH_CI_D[1...M] ETH_CI_P[1...M] ETH_CI_DE[1...M] ETH_CI_SSF[1...M] ETH_CI_SSFrdi[1] ETH_CI_SSFfdi[1] **ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE **ETHx/ETH-m_A_Sk_MP**: ETHx/ETH-m_A_Sk_MI_cCSF

Processes

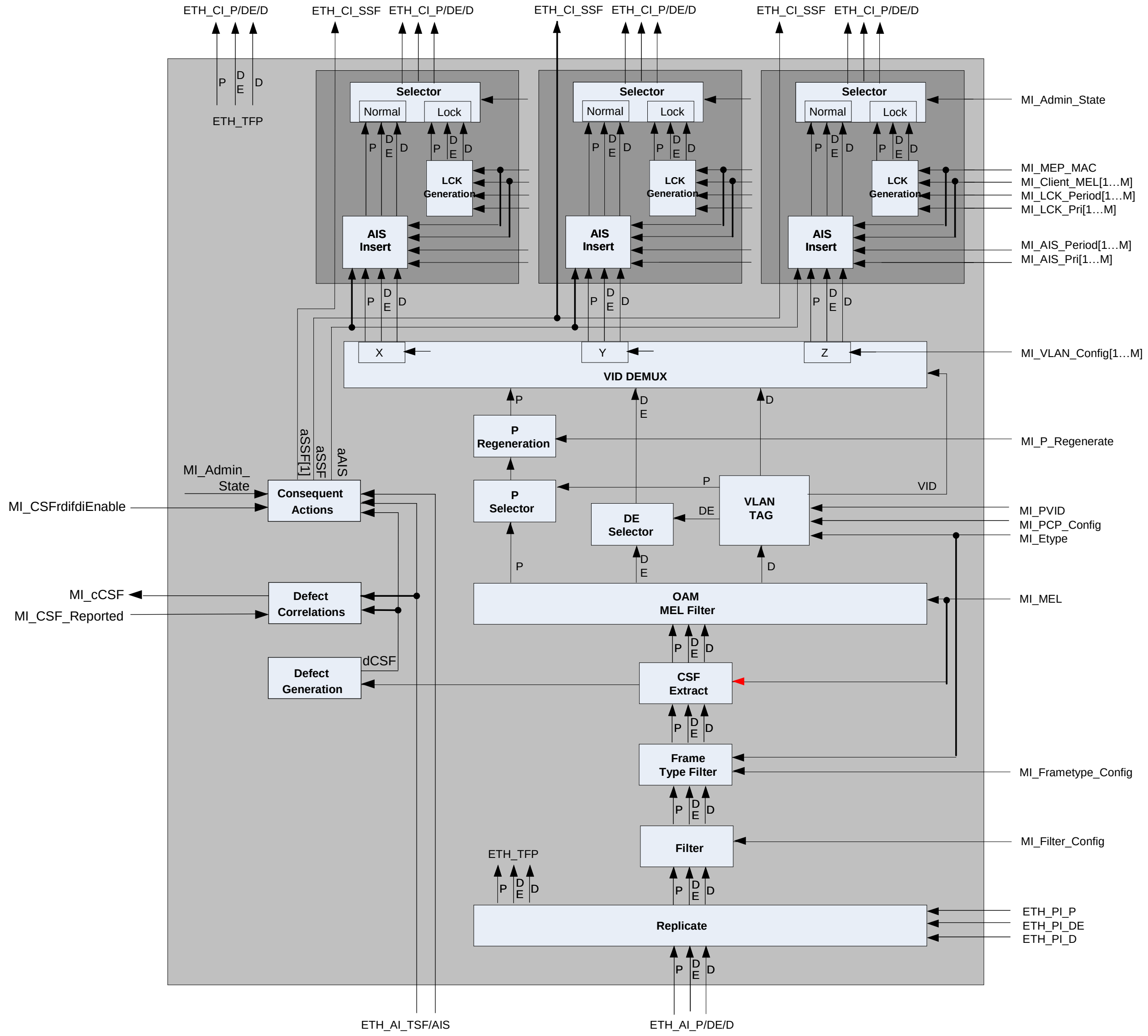


Figure 9-30 – ETHx/ETH-m_A_Sk process

Replicate process

As defined in clause 8.4.

Filter process

As defined in clause 8.3.

Frame type filter process

The frame type filter process filters the ETH_CI depending on the value of the MI_frametype_Config input parameter. There are three possible values for this parameter:

- All Frames
- Only VLAN Tagged
- Only Untagged and Priority Tagged.

If the value of MI_frametype_Config equals "All Frames", all ETH_CI is passed through. For the other two values, the process inspects the M_SDU field of the ETH_CI_D signal. It inspects the length/type field and, if applicable, the VID field.

If MI_frametype_Config is set to "Only Untagged and Priority Tagged", all frames with L/T equals MI_Etype and VID in the range 1…4094 are filtered.

If MI_frametype_Config is set to "Only VLAN Tagged", all frames with L/T not equal to MI_Etype and all frames with L/T equal to MI_Etype and VID equal to zero are filtered.

CSF extract process

As defined in clause 8.1.17. The ETHx/ETH-m adaptation function generates a single OAM flow while it can accommodate multiple ETH APs. In the case of using multiple APs, the CSF signal is supported at only a representative OAM flow.

OAM MEL filter process

As defined in clause 8.1.1.

VLAN tag process

The VLAN tag process inspects the incoming D signal; if the value in the L/T field is equal to the value provisioned by the MI_Etype input parameter a VLAN tag is present in the D signal.

If there is no VLAN tag present the VID signal gets the value presented by the MI_PVID input parameter.

If there is a VLAN tag present the VLAN tag process extracts the P, DE and VID information from this VLAN tag. The VID value is taken from the VID field in the VLAN tag. The P and DE values are decoded from the PCP field of the VLAN tag (C-VLAN) or from the PCP and DEI fields of the VLAN tag (S-VLAN), using the decoding information presented via the MI_PCP_Config input parameter. The P value is presented to the P selector process and the DE value is presented to the DE selector process.

DE selector process

This process forwards the incoming DE signal. If there is no incoming DE signal present, it generates a DE signal with the value drop ineligible.

P selector process

This process forwards the P signal coming from the VLAN tag process. If this signal is not present, the P signal coming from the OAM MEL process is forwarded.

P regeneration process

This process regenerates the incoming P signal, based on the MI_P_Regenerate input signal. The MI_P_Regenerate signal specifies a mapping table from P value to P value.

VID Demux process

The VID Demux process de-interleaves the incoming signal set (DE, P, D) to the different ports (X, Y, Z in Figure 9-30). The VID signal determines the port to be selected, based on the MI_Vlan_Config input parameter.

The MI_Vlan_Config parameter specifies the possible VID values for the ports to be used. If there is no port assigned to a specific VID value and this VID value is used, the VID Demux process will filter the incoming signal set.

Disabling the ingress VID filtering is modelled by setting MI_Vlan_Config [1…4094]. Refer to Appendix VIII.

AIS insert process

As defined in clause 8.1.4.

LCK generation process

As defined in clause 8.1.2. Each FP has its own LCK generation process.

Selector process

As defined in clause 8.1.3. The normal CI is blocked if Admin_State = LOCKED.

Defects

dCSF-LOS – See clause 6.1.5.4.

dCSF-RDI – See clause 6.1.5.4.

dCSF-FDI – See clause 6.1.5.4.

Consequent actions

aSSF[1] ← (AI_TSF or dCSF_LOS) and (not MI_Admin_State == Locked)

aSSFrdi[1] ← dCSF-RDI and MI_CSFrdifdiEnable

aSSFfdi[1] ← dCSF-FDI and MI_CSFrdifdiEnable

aSSF[2…M] ← AI_TSF and (not MI_Admin_State == Locked)

aAIS ← AI_AIS

Defect correlations

cCSF ← (dCSF-LOS or dCSF-RDI or dCSF-FDI) and (not AI_TSF) and MI_CSF_Reported

Performance monitoring None.

9.3.4 ETH group to ETH adaptation functions (ETHG/ETH_A)

9.3.4.1 ETH group to ETH adaptation source function (ETHG/ETH_A_So)

Symbol

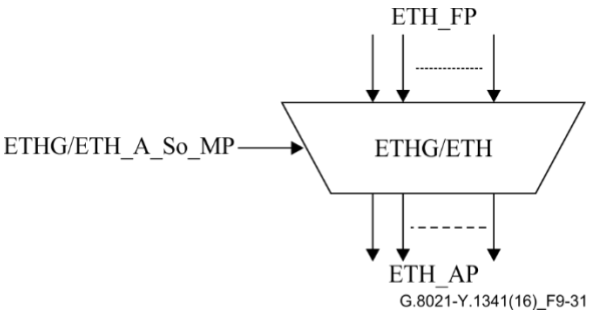


Figure 9-31 – ETHG/ETH_A_So symbol

Interfaces

Table 9-10 – ETHG/ETH_A_So interfaces

Inputs	Outputs
ETH_FP: ETH_CI_D[1…M] ETH_CI_P[1…M] ETH_CI_DE[1…M] ETH_CI_APS ETH_CI_SSF[1] ETH_CI_SSFrdi[1] ETH_CI_SSFfdi[1] **ETHG/ETH_A_So_MP**: ETHG/ETH_A_So_MI_MEP_MAC ETHG/ETH_A_So_MI_Client_MEL[1..M] ETHG/ETH_A_So_MI_LCK_Period[1…M] ETHG/ETH_A_So_MI_LCK_Pri[1…M] ETHG/ETH_A_So_MI_Admin_State ETHG/ETH_A_So_MI_MEL ETHG/ETH_A_So_MI_APS_Pri ETHG/ETH_A_So_MI_CSF_Period ETHG/ETH_A_So_MI_CSF_Pri ETHG/ETH_A_So_MI_CSF_Enable ETHG/ETH_A_So_MI_CSFrdifdiEnable ETHG/ETH_A_So_MI_CSFdciEnable	**ETH_AP**: ETH_AI_D[1...M] ETH_AI_P[1...M] ETH_AI_DE[1...M]

Processes

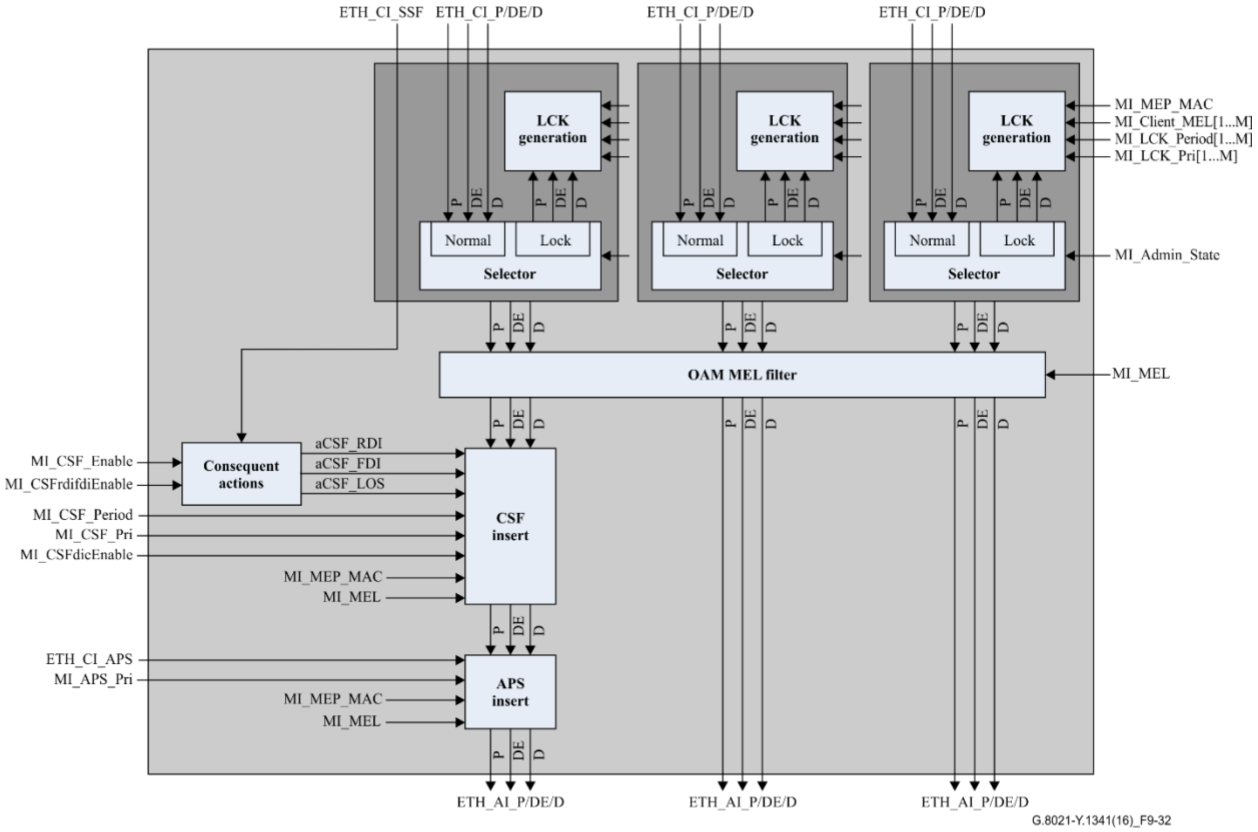


Figure 9-32 – ETHG/ETH_A_So process

LCK generation process

As defined in clause 8.1.2. There is a single LCK generation process for each ETH.

Selector process

As defined in clause 8.1.3. The normal CI of each input is blocked if Admin_State = LOCKED.

OAM MEL filter process

As defined in clause 8.1.1.

APS insert process

As defined in clause 8.1.5.

CSF insert process

As defined in clause 8.1.16.

Defects None.

Consequent actions

aCSF-LOS ← CI_SSF and MI_CSFEnable

aCSF-RDI ← CI_SSFrdi and MI_CSFrdifdiEnable and MI_CSFEnable

aCSF-FDI ← CI_SSFfdi and MI_CSFrdifdiEnable and MI_CSFEnable

Defect correlations None.

Performance Monitoring None.

9.3.4.2 ETH group to ETH adaptation sink function (ETHG/ETH_A_Sk)

Symbol

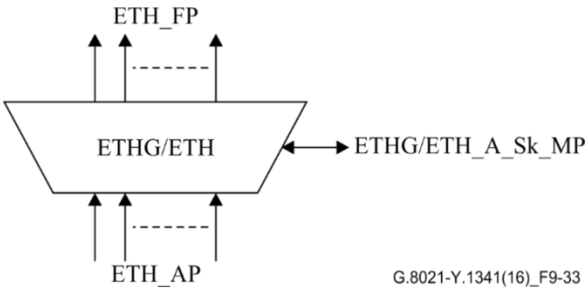


Figure 9-33 – ETHG/ETH_A_Sk symbol

Interfaces

Table 9-11 – ETHG/ETH_A_Sk interfaces

Inputs	Outputs
ETH_AP: ETH_AI_D[1...M] ETH_AI_P[1...M] ETH_AI_DE[1...M] ETH_AI_TSF ETH_AI_TSD ETH_AI_AIS **ETHG/ETH_A_Sk_MP**: ETHG/ETH_A_Sk_MI_MEP_MAC ETHG/ETH_A_Sk_MI_Client_MEL[1…M] ETHG/ETH_A_Sk_MI_LCK_Period[1…M] ETHG/ETH_A_Sk_MI_LCK_Pri[1…M] ETHG/ETH_A_Sk_MI_Admin_State ETHG/ETH_A_Sk_MI_AIS_Period[1…M] ETHG/ETH_A_Sk_MI_AIS_Pri[1…M] ETHG/ETH_A_Sk_MI_MEL ETHG/ETH_A_Sk_MI_CSF_Reported ETHG/ETH_A_Sk_MI_CSFrdifdiEnable	**ETH_FP**: ETH_CI_D[1…M] ETH_CI_P[1…M] ETH_CI_DE[1…M] ETH_CI_APS ETH_CI_SSF[1…M] ETH_CI_SSD ETH_CI_SSFrdi[1] ETH_CI_SSFfdi[1] **ETHG/ETH_A_Sk_MP**: ETHG/ETH_A_Sk_MI_cCSF

Processes

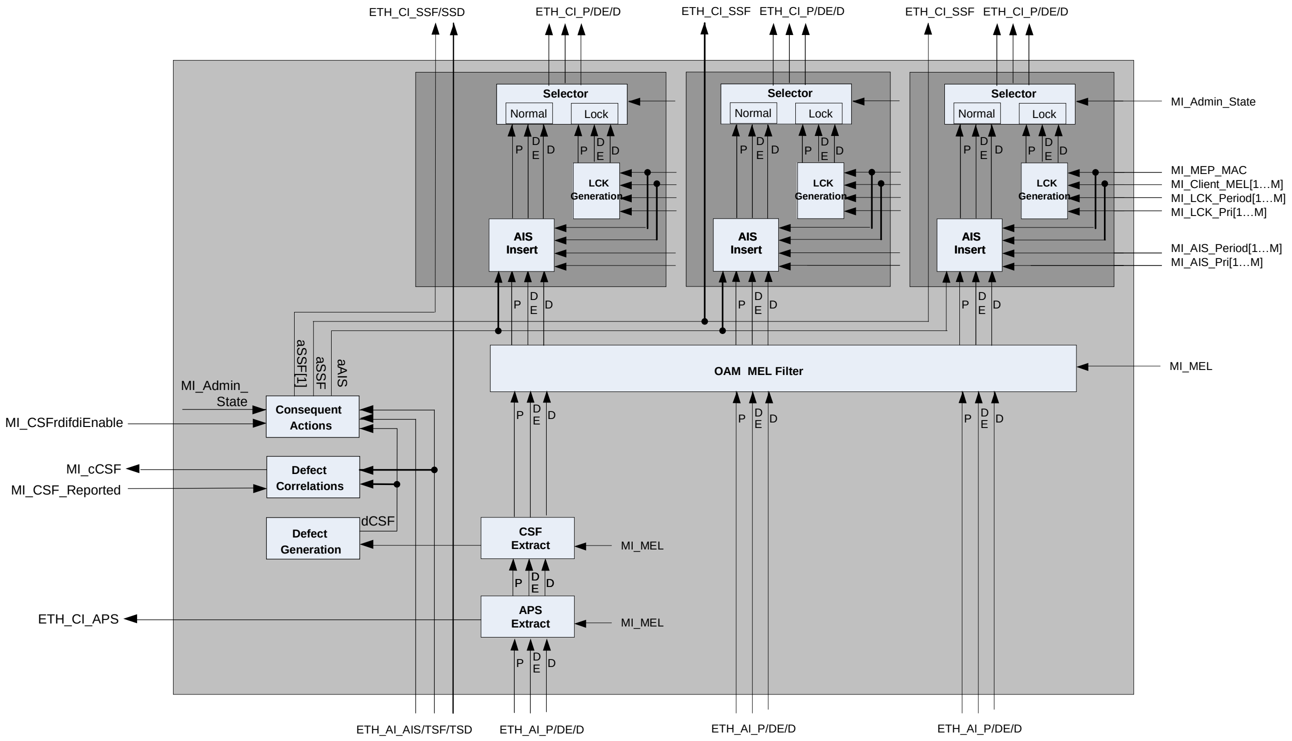


Figure 9-34 – ETHG/ETH_A_Sk process

APS extract process

As defined in clause 8.1.6.

CSF extract process

As defined in clause 8.1.17.

OAM MEL filter process

As defined in clause 8.1.1.

AIS insert process

As defined in clause 8.1.4. There is a single AIS insert process for each ETH.

LCK generation process

As defined in clause 8.1.2. There is a single LCK generation process for each ETH.

Selector process

As defined in clause 8.1.3. The normal CI of each input is blocked if Admin_State = LOCKED.

Defects

dCSF-LOS – See clause 6.1.5.4.

dCSF-RDI – See clause 6.1.5.4.

dCSF-FDI – See clause 6.1.5.4.

Consequent actions

aSSF[1] ← (AI_TSF or dCSF_LOS) and (not MI_Admin_State == Locked)

aSSFrdi[1] ← dCSF-RDI and MI_CSFrdifdiEnable

aSSFfdi[1] ← dCSF-FDI and MI_CSFrdifdiEnable

aSSF[2…M] ← AI_TSF and (not MI_Admin_State == Locked)

aAIS ← AI_AIS

Defect correlations

cCSF ← (dCSF-LOS or dCSF-RDI or dCSF-FDI) and (not AI_TSF) and MI_CSF_Reported

Performance monitoring None.

9.3.5 ETHx to ETH group adaptation functions (ETHx/ETHG_A)

This adaptation function multiplexes different ETH_CI streams in the ETH group into a single ETH_AI stream and demultiplexes the ETH_AI stream into individual ETH_CI streams.

Symbol

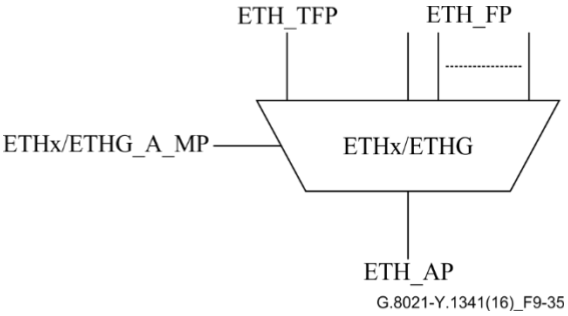


Figure 9-35 – ETHx/ETHG_A symbol

The ETHx/ETHG_A (Figure 9-35) function is further decomposed into separate source and sink adaptation functions that are interconnected as shown in Figure 9-36.

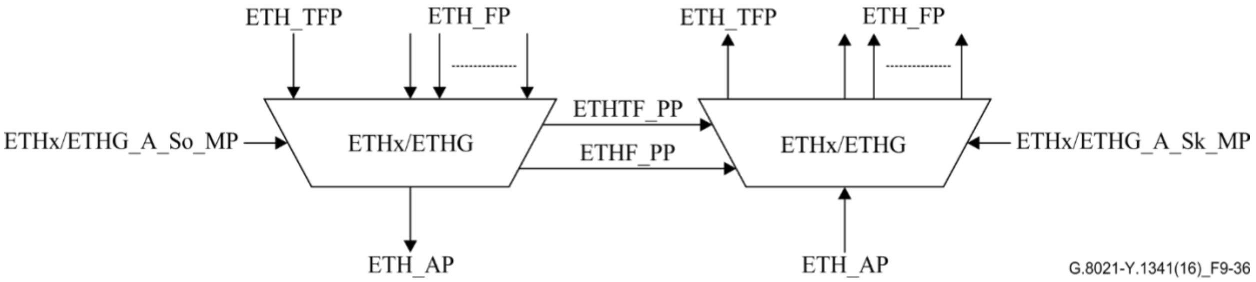


Figure 9-36 – ETHx/ETHG_A source and sink symbols

9.3.5.1 ETHx to ETH group adaptation source function (ETHx/ETHG_A_So)

This function multiplexes individuals ETH_CI streams in the ETH group into a single ETH_AI stream.

Symbol

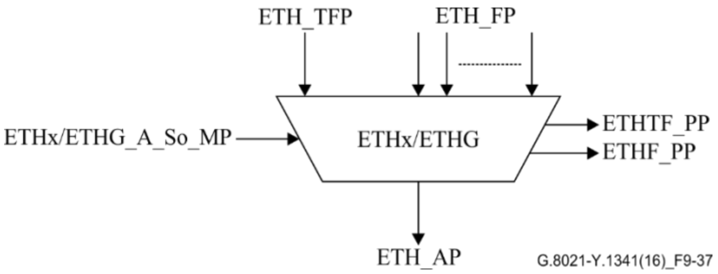


Figure 9-37 – ETHx/ETHG_A_So symbol

Interfaces

Table 9-12 – ETHx/ETHG_A_So interfaces

Inputs	Outputs
ETH_FP: ETH_CI_D[1...M] ETH_CI_P[1...M] ETH_CI_DE[1...M] **ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE **ETHx/ETHG_A_So_MP**: ETHx/ETHG_A_So_MI_MEP_MAC ETHx/ETHG_A_So_MI_Client_MEL[1…M] ETHx/ETHG_A_So_MI_LCK_Period[1…M] ETHx/ETHG_A_So_MI_LCK_Pri[1…M] ETHx/ETHG_A_So_MI_Admin_State ETHx/ETHG_A_So_MI_VLAN_Config[1...M] ETHx/ETHG_A_So_MI_Etype ETHx/ETHG_A_So_MI_PCP_Config ETHx/ETHG_A_So_MI_MEL	**ETH_AP**: ETH_AI_D ETH_AI_P ETH_AI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE

Processes

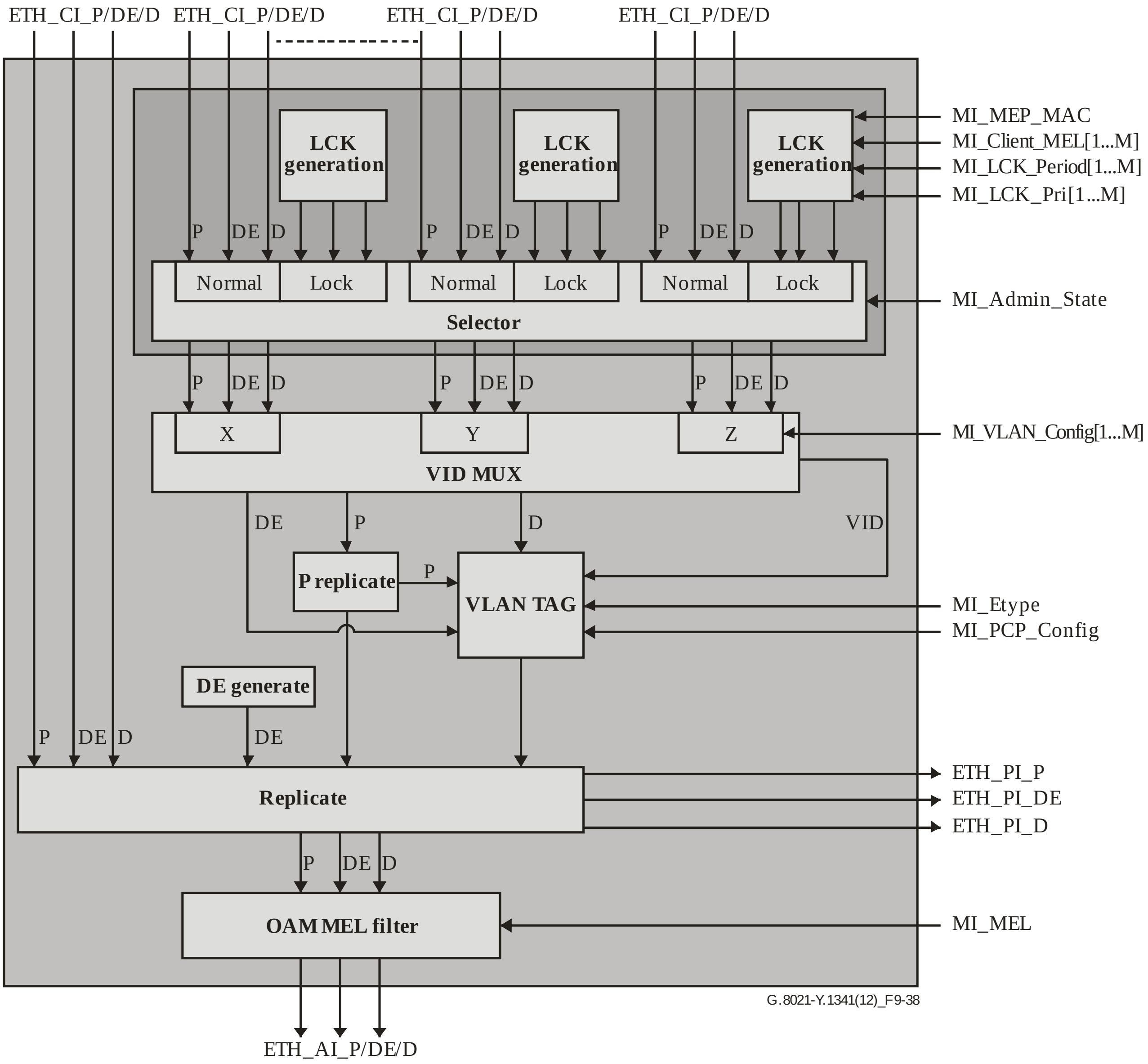


Figure 9-38 – ETHx/ETHG_A_So process

LCK generation process

As defined in clause 8.1.2. Each FP has its LCK generation process.

Selector process

As defined in clause 8.1.3. The normal CI is blocked if Admin_State = LOCKED.

VID Mux process

The VID MUX process interleaves the signal sets (P, D, DE) from the input ports (X, Y, Z). The detail of this process is described in clause 9.3.3.1.

VLAN tag process

This process inserts a VLAN tag into the M_SDU field of the incoming D signal. The detail of this process is described in clause 9.3.3.1.

P replicate process

The P replicate process replicates the incoming P signal to both output ports without changing the value of the signal.

DE generation process

The DE generation process generates a DE signal with the value drop ineligible.

Replicate process

As defined in clause 8.4.

OAM MEL filter process

As defined in clause 8.1.1.

Defects	None.
Consequent actions	None.
Defect correlations	None.
Performance monitoring	None.

9.3.5.2 ETHx to ETH group adaptation sink function (ETHx/ETHG_A_Sk)

Symbol

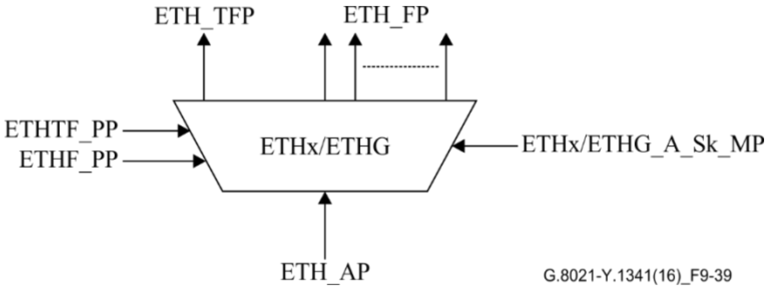


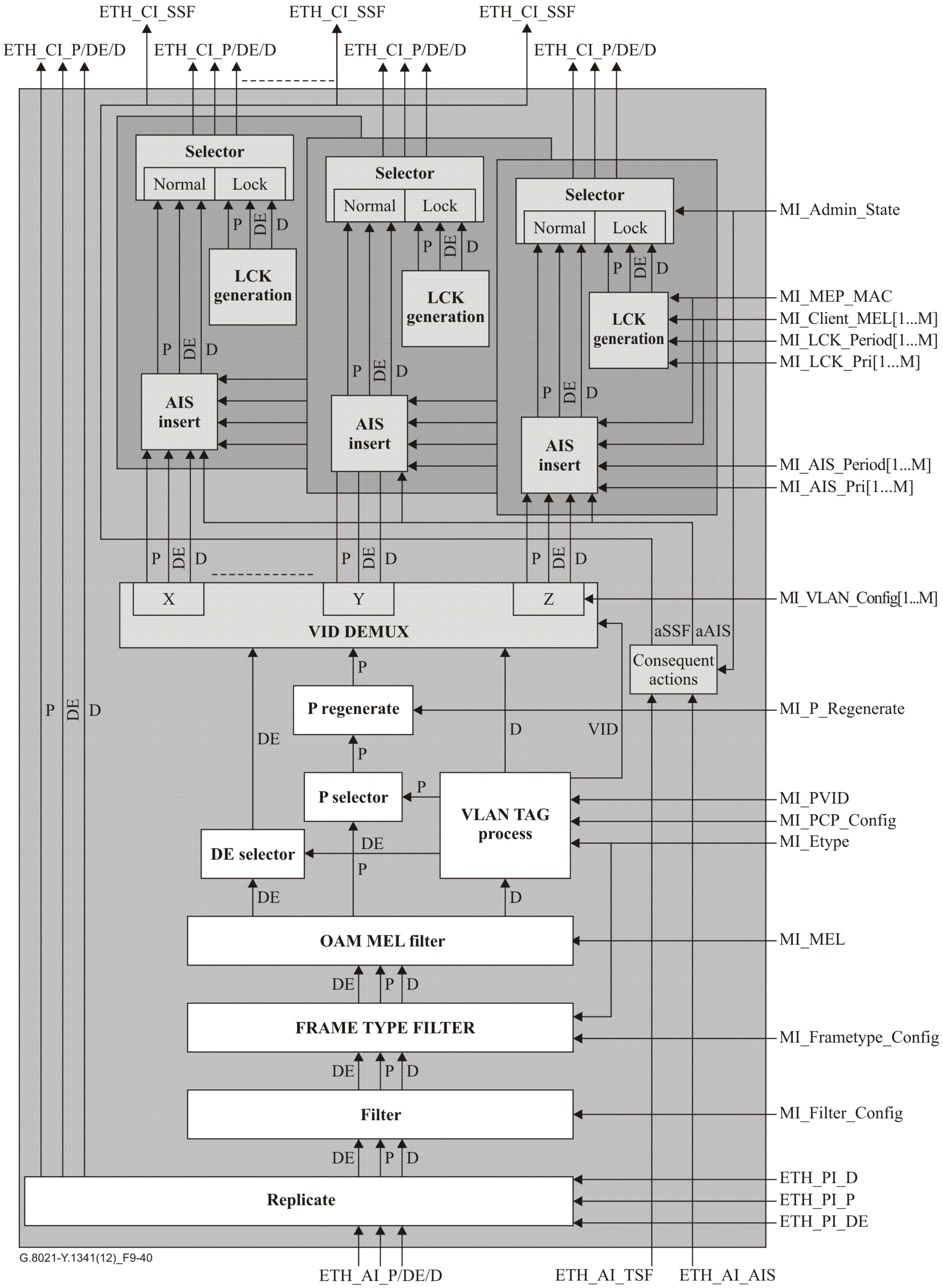
Figure 9-39 – ETHx/ETHG_A_Sk symbol

Interfaces

Table 9-13 – ETHx/ETHG_A_Sk interfaces

Inputs	Outputs
ETH_AP: ETH_AI_D ETH_AI_P ETH_AI_DE ETH_AI_TSF ETH_AI_AIS **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHx/ETHG_A_Sk_MP**: ETHx/ETHG_A_Sk_MI_MEP_MAC ETHx/ETHG_A_Sk_MI_Client_MEL[1…M] ETHx/ETHG_A_Sk_MI_LCK_Period[1…M] ETHx/ETHG_A_Sk_MI_LCK_Pri[1…M] ETHx/ETHG_A_Sk_MI_Admin_State ETHx/ETHG_A_Sk_MI_AIS_Period[1…M] ETHx/ETHG_A_Sk_MI_AIS_Pri[1…M] ETHx/ETHG_A_Sk_MI_VLAN_Config[1...M] ETHx/ETHG_A_Sk_MI_P_Regenerate ETHx/ETHG_A_Sk_MI_PVID ETHx/ETHG_A_Sk_MI_PCP_Config ETHx/ETHG_A_Sk_MI_Etype ETHx/ETHG_A_Sk_MI_MEL ETHx/ETHG_A_Sk_MI_Frametype_Config ETHx/ETHG_A_Sk_MI_Filter_Config	**ETH_FP**: ETH_CI_D[1...M] ETH_CI_P[1...M] ETH_CI_DE[1...M] ETH_CI_SSF[1...M] **ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE

Processes



G.8021-Y.1341(12)_F9-40

Figure 9-40 – ETHx/ETHG_A_Sk process

Replicate process

As defined in clause 8.4.

Filter Process

As defined in clause 8.3.

Frame type filter process

The frame type filter process filters the ETH_CI depending on the value of the MI_frametype_Config input parameter. The details of this process is described in clause 9.3.3.2.

OAM MEL filter process

As defined in clause 8.1.1.

VLAN tag process

The VLAN tag process inspects the incoming D signal. The detail of this process is described in clause 9.3.3.1.

DE selector process

This process forwards the incoming DE signal. If there is no incoming DE signal present, it generates a DE signal with the value drop ineligible.

P selector process

This process forwards the P signal coming from the VLAN tag process. If this signal is not present, the P signal coming from the OAM MEL process is forwarded.

P regeneration process

This process regenerates the incoming P signal, based on the MI_P_Regenerate input signal. The MI_P_Regenerate signal specifies a mapping table from P value to P value.

VID Demux process

The VID Demux process de-interleaves the incoming signal set (DE, P, D) to the different ports (X, Y, Z in Figure 9-40). The detail of this process is described in clause 9.3.3.1.

AIS insert process

As defined in clause 8.1.4.

LCK generation process

As defined in clause 8.1.2. Each FP has its own LCK generation process.

Selector process

As defined in clause 8.1.3. The normal CI is blocked if Admin_State = LOCKED.

Defects	None.
Consequent actions	aSSF ← AI_TSF and (not MI_Admin_State == Locked) aAIS ← AI_AIS
Defect correlations	None.
Performance monitoring	None.

9.3.6 ETH to MCC adaptation functions (ETHx/MCC_A)

9.3.6.1 ETH to MCC adaptation source function (ETHx/MCC_A_So)

This function maps MCC traffic units into server ETH_AI traffic units. It also provides a maintenance communication channel for EMF via a management reference point.

Symbol

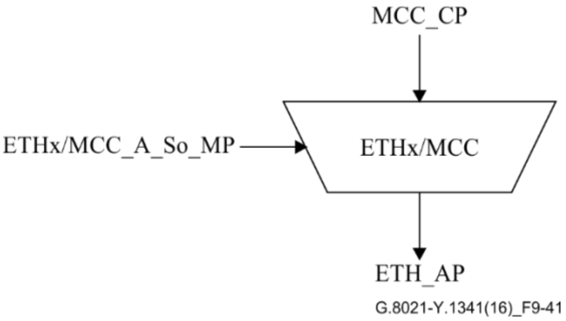


Figure 9-41 – ETHx/MCC_A_So symbol

Interfaces

Table 9-14 – ETHx/MCC_A_So interfaces

Inputs	Outputs
MCC_CP: MCC_CI_D **ETHx/MCC_A_So_MP**: ETHx/MCC_A_So_MI_MEL ETHx/MCC_A_So_MI_MEP_MAC ETHx/MCC_A_So_MI_MCC_Pri ETHx/MCC_A_So_MI_MEP_ID ETHx/MCC_A_So_MI_EDM_Enable ETHx/MCC_A_So_MI_EDM_Period ETHx/MCC_A_So_MI_EDM_Duration	**ETH_AP**: ETH_AI_D ETH_AI_P ETH_AI_DE

Processes

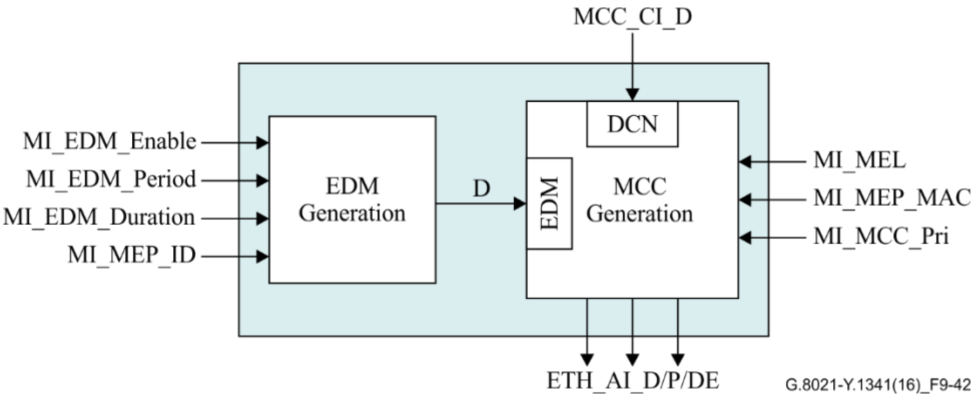


Figure 9-42 – ETHx/MCC_A_So process

MCC generation process

MCC generation process generates MCC traffic units based on the data signals from MCC_connection point or EDM generation process. The data signals from MCC connection point are received at DCN (Data Communication Network) port, and the signals from EDM generation are received at EDM port.

This process builds an MCC traffic unit from the received data signals, MI_MEL for MAC DA and MEG level, MI_MEP_MAC for MAC SA and MI_MCC_Pri signals. Figure 9-43 describes the behaviour of MCC traffic unit generation.

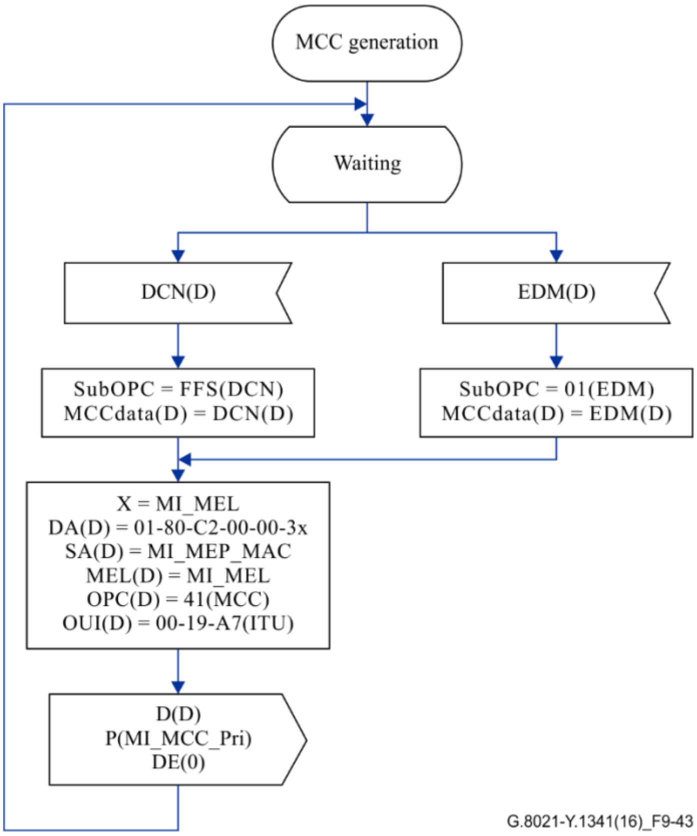


Figure 9-43 – MCC generation behaviour

NOTE – The OUI value for ITU is 00-19-A7; the SubOPC value for EDM is assigned in [ITU-T G.8013]. The SubOPC value for DCN is for further study.

EDM generation process

As defined in clause 8.1.20.1.

Defects	None.
Consequent actions	None.
Defect correlations	None.
Performance monitoring	None.

9.3.6.2 ETH to MCC adaptation sink function (ETHx/MCC_A_Sk)

This function retrieves MCC_CI traffic units from server ETH_AI traffic units. It also provides a maintenance communication channel for EMF via a management reference point.

Symbol

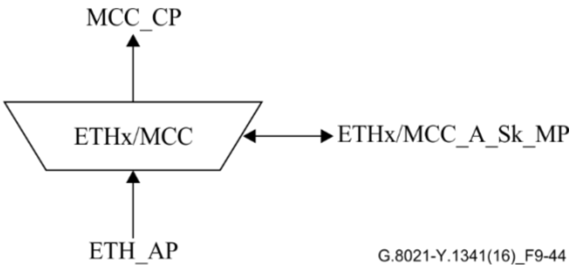


Figure 9-44 – ETHx/MCC_A_Sk symbol

Interfaces

Table 9-15 – ETHx/MCC_A_Sk interfaces

Inputs	Outputs
ETH_AP: ETH_AI_D ETH_AI_P ETH_AI_DE **ETHx/MCC_A_Sk_MP**: ETHx/MCC_A_Sk_MI_MEP_MAC ETHx/MCC_A_Sk_MI_MEL	**MCC_CP**: MCC_CI_D **ETHx/MCC_A_Sk_MP**: ETHx/MCC_A_Sk_MI_EDM_Received (MEP_ID, Duration)

Processes

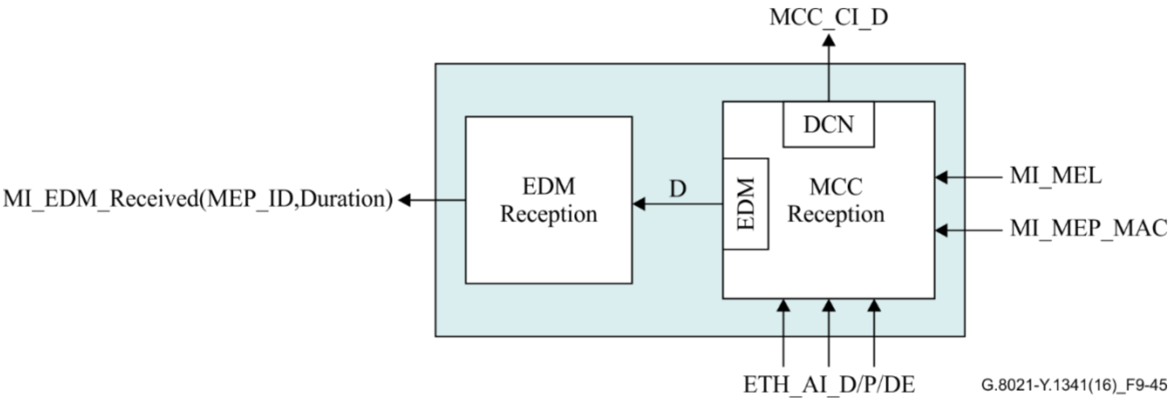


Figure 9-45 – ETHx/MCC_A_Sk process

MCC reception process

This process extracts MCC traffic units that are processed in the ETHx/MCC_A_Sk process according to the following pseudo code:

```
if (TYPE=<ETHOAM>) and (MEL=MI_MEL) and ((DA=Class 1) or (DA=MI_MEP_MAC))
and (OPC=MCC)then
  switch(OUI) {
    case <ITU>: {
```

```
      switch(SubOPC) {
        case <DCN>: extract ETH-MCC OAM traffic unit and forward to DCN Port
        case <EDM>: extract ETH-MCC OAM traffic unit and forward to EDM Port
        default   : discard the traffic unit
      }
    default: outside the scope of this Recommendation
  }
else
  discard the traffic unit
endif
```

NOTE – The OUI value for ITU is 00-19-A7; the SubOPC value for EDM is assigned in [ITU-T G.8013]. The SubOPC value for DCN is for further study.

EDM reception process

As defined in clause 8.1.20.2.

Defects	None.
Consequent actions	None.
Defect correlations	None.
Performance monitoring	None.

9.4 ETH diagnostic functions

9.4.1 ETH diagnostic flow termination functions for MEPs (ETHDe_FT)

The bidirectional ETHDe flow termination (ETHDe_FT) function is performed by a co-located pair of ETHDe flow termination source (ETHDe_FT_So) and sink (ETHDe_FT_Sk) functions.

9.4.1.1 ETH diagnostic flow termination source function for MEPs (ETHDe_FT_So)

The ETHDe_FT_So process diagram is shown in Figure 9-46.

Symbol

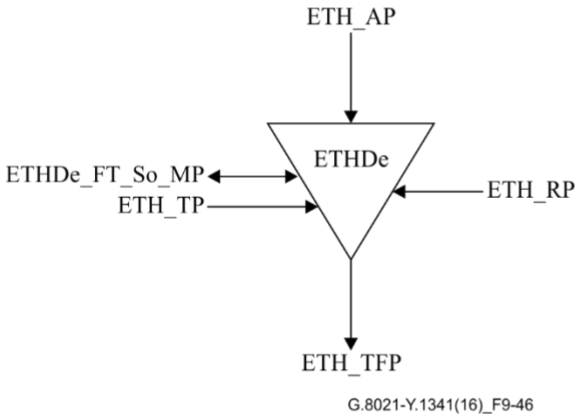


Figure 9-46 – ETHDe_FT_So symbol

Interfaces

Table 9-16 – ETHDe_FT_So interfaces

Inputs	Outputs
ETH_AP: ETH_AI_D ETH_AI_P ETH_AI_DE **ETH_RP**: ETH_RI_LMM(D,P,DE) ETH_RI_LMR(rSA,TxFCf,RxFCf,TxFCb,RxFCl) ETH_RI_LBM(D,P,DE) ETH_RI_LBR(SA,rTLV,TID) ETH_RI_DMM(D,P,DE) ETH_RI_DMR(rSA,TxTimeStampf,RxTimeStampf, TxTimeStampb,RxTimeb,rTestID) ETH_RI_LTM(D,P,DE) ETH_RI_LTR(SA,TTL,TID,TLV) ETH_RI_SLM(OAM,P,DE,TxFCb) ETH_RI_SLR(rMEP_ID,rTest_ID,TxFCf,TxFCb) **ETH_TP:** ETHDe_FT_So_TI_TimeStampl **ETHDe_FT_So_MP**: ETHDe_FT_So_MI_LM_Start(DA,P,Period) ETHDe_FT_So_MI_LM_Intermediate_Request ETHDe_FT_So_MI_LM_Terminate ETHDe_FT_So_MI_LB_Discover(P) ETHDe_FT_So_MI_LB_Series(DA,DE,P,N, Length, Period) ETHDe_FT_So_MI_LB_Test (DA,DE,P,Pattern, Length, Period) ETHDe_FT_So_MI_LB_Test_Terminate ETHDe_FT_So_MI_DM_Start(DA,P,Test ID,Length,Period) ETHDe_FT_So_MI_DM_Intermediate_Request ETHDe_FT_So_MI_DM_Terminate ETHDe_FT_So_MI_1DM_Start(DA,P,Test ID,Length,Period) ETHDe_FT_So_MI_1DM_Terminate ETHDe_FT_So_MI_TST(DA,DE,P,Pattern, Length, Period) ETHDe_FT_So_MI_TST_Terminate ETHDe_FT_So_MI_LT(TA,TTL.P) ETHDe_FT_So_MI_MEP_MAC ETHDe_FT_So_MI_MEL ETHDe_FT_So_MI_MEP_ID ETHDe_FT_So_MI_SL_Start(DA,P,Test_ID,Length, Period) ETHDe_FT_So_MI_SL_Intermediate_Request ETHDe_FT_So_MI_SL_Terminate ETHDe_FT_So_MI_1SL_Start(DA,P,Test_ID,Length,Period) ETHDe_FT_So_MI_1SL_Terminate	**ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE **ETHDe_FT_So_MP**: ETHDe_FT_So_MI_LM_Result(N_TF, N_LF, F_TF, F_LF) ETHDe_FT_So_MI_LB_Discover_Result(MACs) ETHDe_FT_So_MI_LB_Series_Result(REC,ERR,OO) ETHDe_FT_So_MI_LB_Test_Result (Sent, REC, CRC, BER, OO) ETHDe_FT_So_MI_DM_Result(count,B_FD[],F_FD[],N_FD[]) ETHDe_FT_So_MI_TST_Result(Sent) ETHDe_FT_So_MI_LT_Results(Results) ETHDe_FT_So_MI_SL_Result(N_TF,N_LF,F_TF,F_LF)

Processes

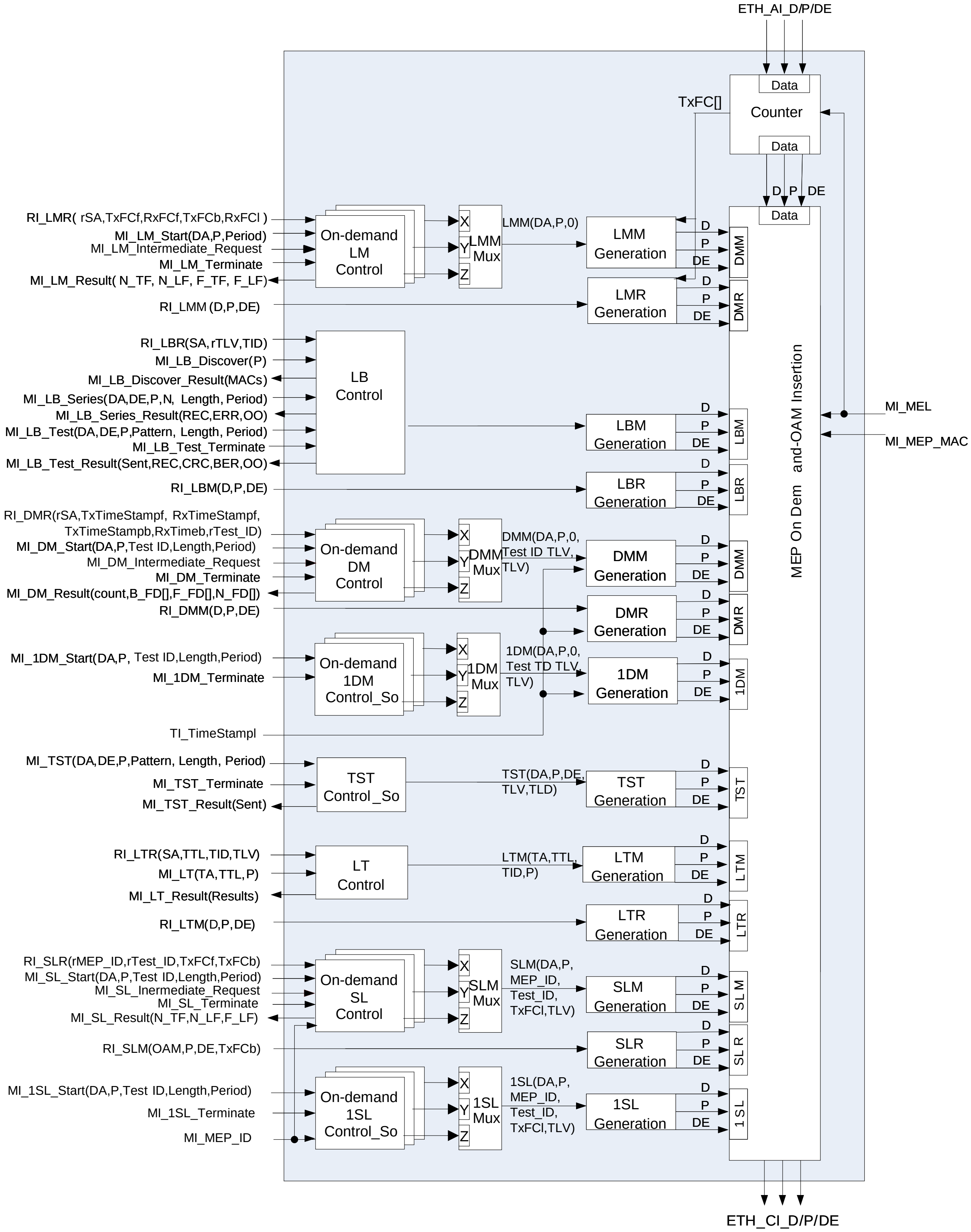


Figure 9-47 – ETHDe_FT_So process

MEP on-demand OAM insertion process

The MEP on-demand OAM insertion process inserts OAM traffic units that are generated in the ETHDe_FT_So process into the stream of traffic units.

For all ETH_CI_D received on any but the data input port, the SA field is overwritten with the MI_MEP_MAC value. In the M_SDU field, the MEL field is overwritten with the MI_MEL value.

If the DA of the OAM traffic unit is a class 1 or class 2 multicast DA the OAM insertion process updates the DA to reflect the right MEL.

This ensures that every generated OAM field has the correct SA, DA and MEL.

LB control

This process is defined in clause 8.1.8 where the LB protocol is defined. Clause 8.1.8.2 defines the LB control process.

LBM generation

This process is defined in clause 8.1.8 where the LB protocol is defined. Clause 8.1.8.3 defines the LBM generation process.

LBR generation

This process is defined in clause 8.1.8 where the LB protocol is defined. Clause 8.1.8.6 defines the LBR generation process.

On-demand LM control

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.2 defines the on-demand LM control process.

LMM generation

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.3 defines the LMM generation process.

LMR generation

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.5 defines the LMR generation process.

LMM Mux

The LMM Mux process interleaves the signal sets LMM(DA,P,0) from the input ports (X, Y, Z).

Counter process

This process is defined in clause 8.1.9.7 and used to count frames for on-demand loss measurements with the on-demand LM protocol.

On-demand DM control

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.2 defines the DM control process.

DMM generation

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.3 defines the DMM generation process.

DMR generation

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.5 defines the DMR generation process.

DMM Mux

The DMM Mux process interleaves the signal sets DMM(DA,P,0,Test ID TLV, TLV) from the input ports (X, Y, Z).

On-demand 1DM Control_So

This process is defined in clause 8.1.11 where the 1DM protocol is defined. Clause 8.1.11.2 defines the 1DM Control_So process.

1DM generation

This process is defined in clause 8.1.11 where the 1DM protocol is defined. Clause 8.1.11.3 defines the 1DM generation process.

1DM Mux

The 1DM Mux process interleaves the signal sets 1DM(DA,P,0,Test ID TLV, TLV) from the input ports (X, Y, Z).

TST Control_So

This process is defined in clause 8.1.12 where the TST protocol is defined. Clause 8.1.12.2 defines the TST control process.

TST generation

This process is defined in clause 8.1.12 where the TST protocol is defined. Clause 8.1.12.3 defines the TST generation process.

LT control

This process is defined in clause 8.1.13 where the LT protocol is defined. Clause 8.1.13.2 defines the LT control process.

LTM generation

This process is defined in clause 8.1.13 where the LT protocol is defined. Clause 8.1.13.3 defines the LTM generation process.

LTR generation

This process is defined in clause 8.1.13 where the LT protocol is defined. Clause 8.1.13.6 defines the LTR generation process.

On-demand SL control

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.2 defines the SL control process.

SLM generation

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.3 defines the SLM generation process.

SLR generation

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.5 defines the SLR generation process.

SLM Mux

The SLM Mux process interleaves the signal sets SLM(DA,P,MEP_ID,Test_ID,TxFCl,TLV) from the input ports (X, Y, Z).

On-demand 1SL Control_So

This process is defined in clause 8.1.15 where the 1SL protocol is defined. Clause 8.1.15.2 defines the 1SL Control_So process.

1SL generation

This process is defined in clause 8.1.15 where the 1SL protocol is defined. Clause 8.1.15.3 defines the 1SL generation process.

1SL Mux

The 1SL Mux process interleaves the signal sets 1SL(DA,P, MEP_ID,Test_ID, TxFCl, TLV) from the input ports (X, Y, Z).

Defects None.

Consequent actions None.

Defect correlations None.

Performance monitoring None.

9.4.1.2 ETH diagnostic flow termination sink function for MEPs (ETHDe_FT_Sk)

The ETHDe_FT_Sk process diagram is shown in Figure 9-48.

Symbol

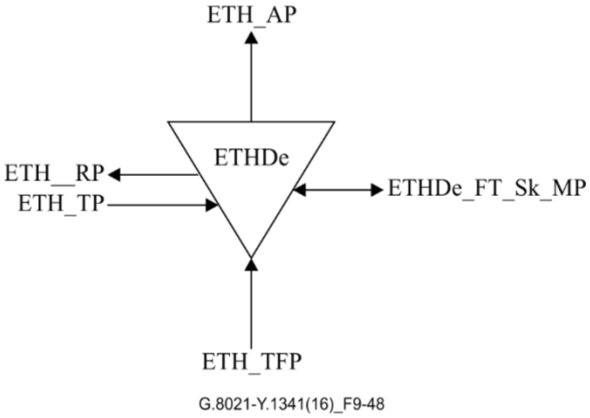


Figure 9-48 – ETHDe_FT_Sk symbol

Interfaces

Table 9-17 – ETHDe_FT_Sk interfaces

Inputs	Outputs
ETH_TFP: ETH_CI_D ETH_CI_P ETH_CI_DE	**ETH_AP**: ETH_AI_D ETH_AI_P ETH_AI_DE
ETHDe_FT_Sk_MP: ETHDe_FT_Sk_MI_MEL ETHDe_FT_Sk_MI_MEP_MAC ETHDe_FT_Sk_MI_1DM_Start(SA,P,Test ID) ETHDe_FT_Sk_MI_1DM_Intermediate_Request ETHDe_FT_Sk_MI_1DM_Terminate ETHDe_FT_Sk_MI_TST_Start(SA,Pattern) ETHDe_FT_Sk_MI_1SL_Intermediate_Request ETHDe_FT_Sk_MI_TST_Terminate ETHDe_FT_Sk_MI_1SL_Start(SA,MEP_ID, Test_ID) ETHDe_FT_Sk_MI_1SL_Terminate	**ETH _RP**: ETH_RI_LMM(D,P,DE) ETH_RI_LMR(TxFCf,RxFCb,TxFCb,RxFCl) ETH_RI_LMR(rSA,TxFCf,RxFCf,TxFCb,RxFCl) ETH_RI_LBM(D,P,DE) ETH_RI_LBR(SA,rTLV,TID) ETH_RI_DMM(D,P,DE) ETH_RI_DMR(rSA,TxTimestampf,RxTimeStampf, TxTimeStampb,RxTimeb,rTest ID) ETH_RI_LTM(D,P,DE) ETH_RI_LTR(SA,TTL,TID,TLV) ETH_RI_SLM(OAM,P,DE,TxFCb) ETH_RI_SLR(rMEP_ID,rTest_ID,TxFCf,TxFCb)
ETH_TP: ETHDe_FT_Sk_TI_TimeStampl	**ETHDe_FT_Sk_MP**: ETHDe_FT_Sk_MI_1DM_Result(count,N_FD[]) ETHDe_FT_Sk_MI_TST_Result(REC,CRC,BER,OO) ETHDe_FT_Sk_MI_1SL_Result(N_TF,N_LF)

Processes

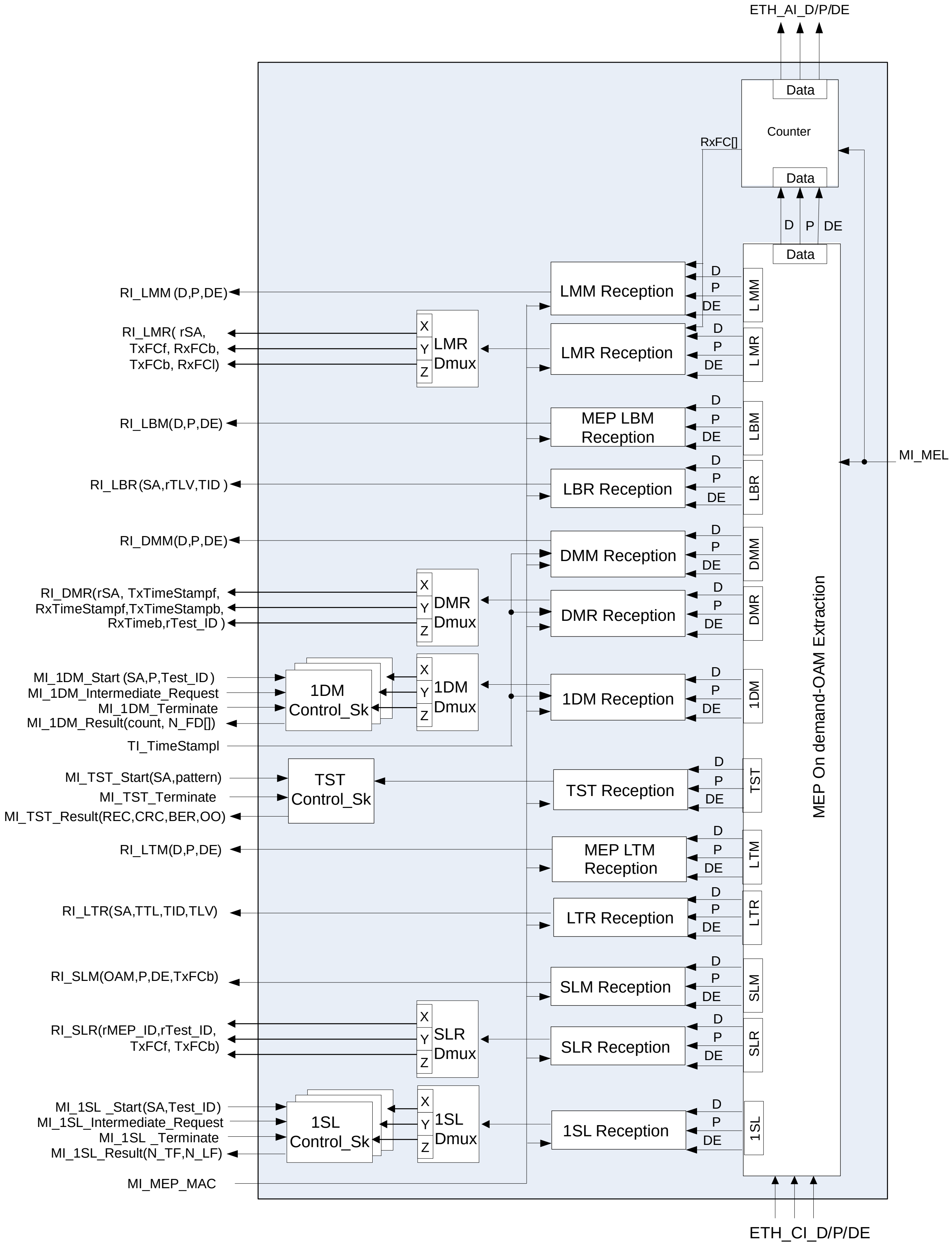


Figure 9-49 – ETHDe_FT_Sk processes

MEP on-demand OAM extraction process

The MEP on-demand OAM extraction process extracts OAM traffic units that are processed in the ETHDe_FT_Sk process from the stream of traffic units as defined in the following pseudo code:

```
if (TYPE=<ETHOAM>) and (MEL=MI_MEL) then
  switch(OPC) {
  case <LMM>: if (Flag.Type=0) then
                 extract ETH-LMM OAM traffic unit and forward to LMM Port
              endif
  case <LMR>: if (Flag.Type=0) then
                 extract ETH-LMR OAM traffic unit and forward to LMR Port
              endif
  case <DMM>: if (Flag.Type=0) then
              extract ETH-DMM OAM traffic unit and forward to DMM Port
              endif
  case <DMR>: if (Flag.Type=0) then
              extract ETH-DMR OAM traffic unit and forward to DMR Port
              endif
  case <1DM>: extract ETH-1DM OAM traffic unit and forward to 1DM Port
  case <LTM>: extract ETH-LTM OAM traffic unit and forward to LTM Port
  case <LTR>: extract ETH-LTR OAM traffic unit and forward to LTR Port
  case <LBM>: extract ETH-LBM OAM traffic unit and forward to LBM Port
  case <LBR>: extract ETH-LBR OAM traffic unit and forward to LBR Port
  case <TST>: extract ETH-TST OAM traffic unit and forward to TST Port
  case <SLM>: extract ETH-SLM OAM traffic unit and forward to SLM port
  case <SLR>: extract ETH-SLR OAM traffic unit and forward to SLR port
  case <1SL>: extract ETH-1SL OAM traffic unit and forward to 1SL Port
  default: forward ETH_CI traffic unit to Data port
  }
else
    forward ETH_CI_traffic unit to Data Port
endif
```

NOTE 1 – Further filtering of OAM traffic units is performed by the OAM MEL filter process which forms part of the ETH adaptation functions specified in clause 9.3.

NOTE 2 – If both ETHDe_FT and ETHx_FT are involved in synthetic loss measurements, the MEP on-demand OAM extraction process needs to determine which flow termination the received ETH-SLM PDU belongs to. Mechanism details are for further study.

MEP LBM reception

This process is defined in clause 8.1.8 where the LB protocol is defined. Clause 8.1.8.5 defines the LBM MEP reception process.

LBR reception

This process is defined in clause 8.1.8 where the LB protocol is defined. Clause 8.1.8.7 defines the LBR reception process.

LMM reception

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.4 defines the LMM reception process.

LMR reception

This process is defined in clause 8.1.9 where the LM protocol is defined. Clause 8.1.9.6 defines the LMR reception process.

LMR Demux

The LMR Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P signal can be used for the selection of the port.

Counter process

This process is defined in clause 8.1.9.7 and used to count frames for on-demand loss measurements with on-demand LM protocol.

DMM reception

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.4 defines the DMM reception process.

DMR reception

This process is defined in clause 8.1.10 where the DM protocol is defined. Clause 8.1.10.6 defines the DMR reception process.

DMR Demux

The DMR Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P and/or Test_ID signal can be used for the selection of the port.

1DM reception

This process is defined in clause 8.1.11 where the 1DM protocol is defined. Clause 8.1.11.4 defines the 1DM reception process.

1DM Demux

The 1DM Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P and/or Test_ID signal can be used for the selection of the port.

1DM Control_Sk

This process is defined in clause 8.1.11 where the 1DM protocol is defined. Clause 8.1.11.5 defines the 1DM Control_Sk process.

TST reception

This process is defined in clause 8.1.12 where the TST protocol is defined. Clause 8.1.12.4 defines the TST reception process.

TST Control_Sk

This process is defined in clause 8.1.12 where the TST protocol is defined. Clause 8.1.12.5 defines the TST Control_Sk process.

MEP LTM reception

This process is defined in clause 8.1.13 where the LT protocol is defined. Clause 8.1.13.5 defines the MEP LTM reception process.

LTR reception

This process is defined in clause 8.1.13 where the LT protocol is defined. Clause 8.1.13.7 defines the LTR reception process.

SLM reception

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.4 defines the SLM reception process.

SLR reception

This process is defined in clause 8.1.14 where the SL protocol is defined. Clause 8.1.14.6 defines the SLR reception process.

SLR Demux

The SLR Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P and/or Test_ID signal can be used for the selection of the port.

1SL reception

This process is defined in clause 8.1.15 where the 1SL protocol is defined. Clause 8.1.15.4 defines the 1SL reception process.

1SL Demux

The 1DM Demux process de-interleaves the incoming signal set (D,P,DE) to the different output ports (X, Y, Z). P and/or Test_ID signal can be used for the selection of the port.

1SL Control_Sk

This process is defined in clause 8.1.15 where the 1SL protocol is defined. Clause 8.1.15.5 defines the 1SL control_Sk process.

Defects	None.
Consequent actions	None.
Defect correlations	None.
Performance monitoring	None.

9.4.2 ETH diagnostic flow termination functions for MIPs (ETHDi_FT)

9.4.2.1 ETH diagnostic flow termination source function for MIPs (ETHDi_FT_So)

Symbol

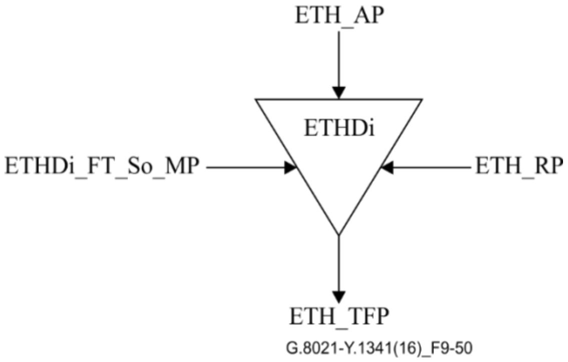


Figure 9-50 – ETHDi_FT_So symbol

Interfaces

Table 9-18 – ETHDi_FT_So interfaces

Inputs	Outputs
ETH_AP: ETH_AI_D ETH_AI_P ETH_AI_DE **ETH_RP**: ETH_RI_LBM(D,P,DE) ETH_RI_LTM(D,P,DE) **ETHDi_FT_So_MP**: ETHDi_FT_So_MI_MEL ETHDi_FT_So_MI_MIP_MAC	**ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE

Processes

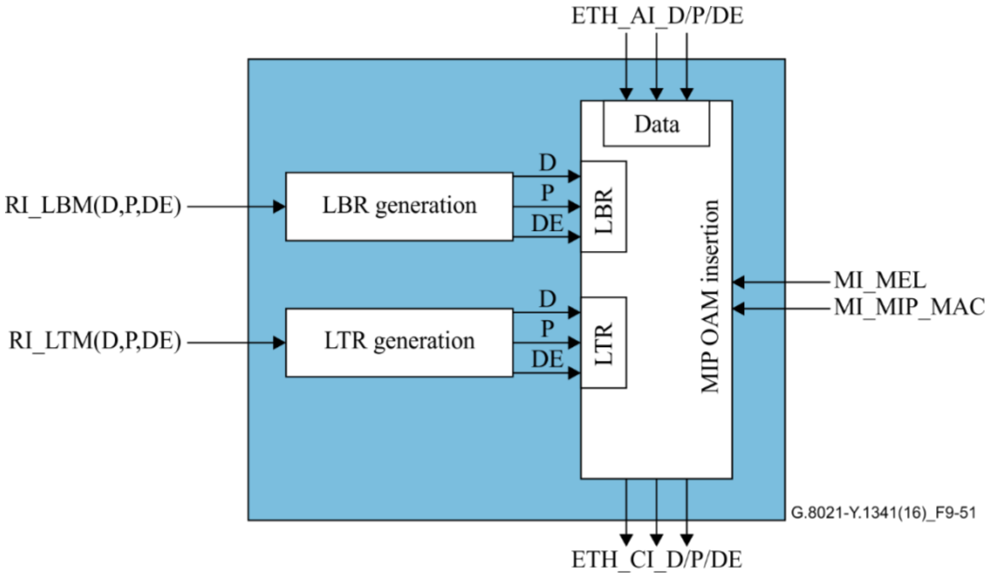


Figure 9-51 – ETHDi_FT_So process

MIP OAM insertion

The MIP OAM insertion process inserts OAM traffic units that are generated in the ETHDi_FT_So process into the stream of traffic units.

For all ETH_CI_D received on any but the data input port, the SA field is overwritten with the MI_MIP_MAC value. In the M_SDU field the EtherType value is overwritten with the OAM EtherType value (89-02) and the MEL field is overwritten with the MI_MEL value.

This ensures that every generated OAM field has the correct SA, EtherType and MEL.

LBR generation

This process is defined in clause 8.1.8 where the LB protocol is defined. Clause 8.1.8.6 defines the LBR generation process.

LTR generation

This process is defined in clause 8.1.13 where the LT protocol is defined. Clause 8.1.13.6 defines the LTR generation process. This process may be regarded as the LT responder which is located outside of this MIP independently, however, the process itself is the same.

Defects	None.
Consequent actions	None.
Defect correlations	None.
Performance monitoring	None.

9.4.2.2 ETH diagnostic flow termination sink function for MIPs (ETHDi_FT_Sk)

Symbol

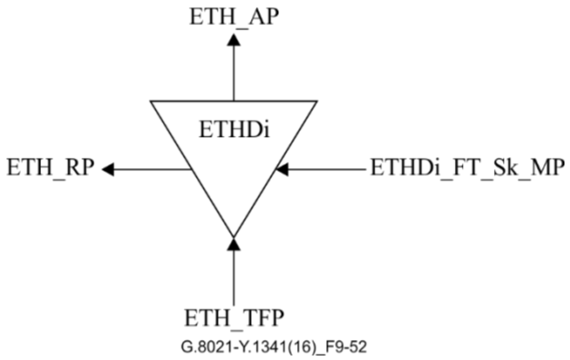


Figure 9-52 – ETHDi_FT_Sk symbol

Interfaces

Table 9-19 – ETHDi_FT_Sk interfaces

Inputs	Outputs
ETH_TFP: ETH_CI_D ETH_CI_P ETH_CI_DE **ETHDi_FT_Sk_MP**: ETHDi_FT_Sk_MI_MEL ETHDi_FT_Sk_MI_MIP_MAC	**ETH_AP**: ETH_AI_D ETH_AI_P ETH_AI_DE **ETH_RP**: ETH_RI_LBM(D,P,DE) ETH_RI_LTM(D,P,DE)

Processes

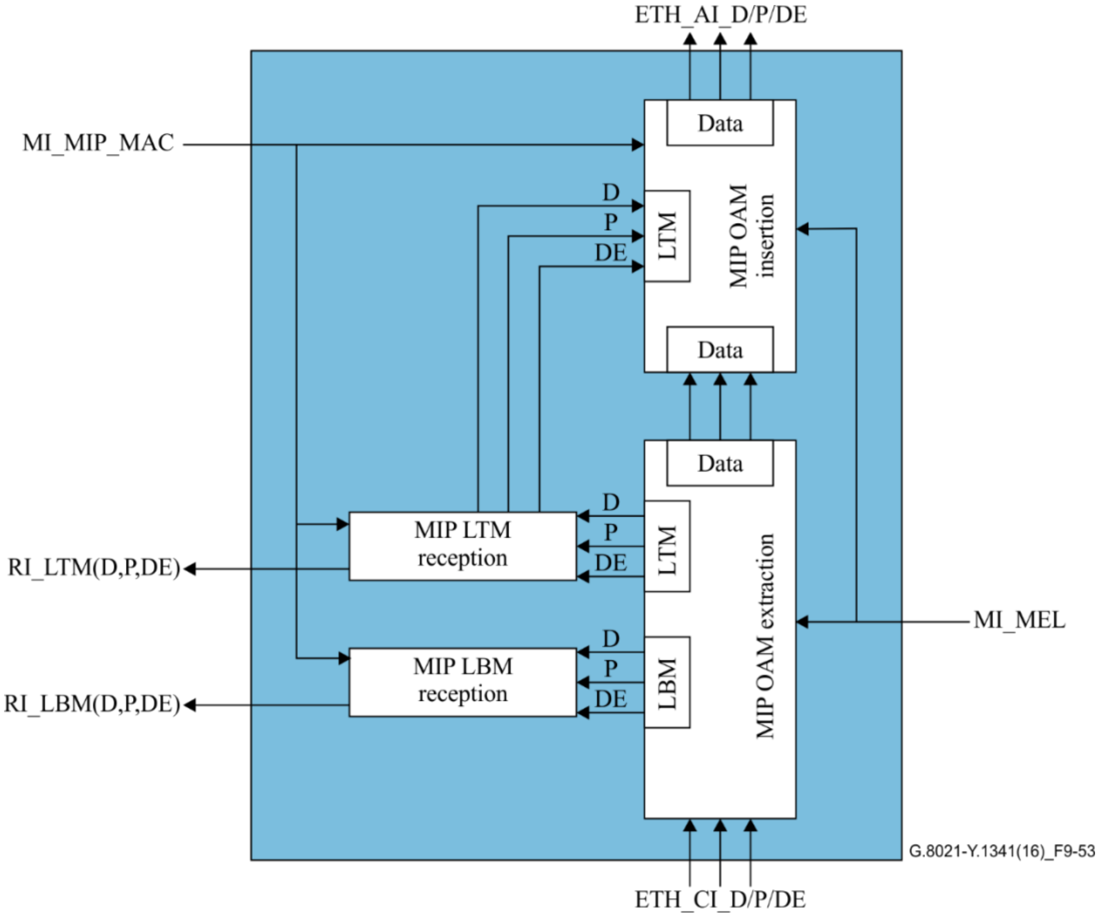


Figure 9-53 – ETHDi_FT_Sk process

MIP OAM extraction process

The MIP OAM extraction process extracts OAM traffic units that are processed in the ETHDi_FT_Sk process from the stream of traffic units as defined in the following pseudo code:

```
if (TYPE=<ETHOAM>) and (MEL=MI_MEL) then
  switch(OPC) {
  case <LBM>: extract ETH-LBM OAM traffic unit
              forward one copy of ETH-LBM OAM traffic unit to LBM Port
              forward one copy of ETH-LBM OAM traffic unit to Data Port
  case <LTM>: extract ETH-LTM OAM traffic unit and forward to LTM Port
  default: forward ETH_CI traffic unit to Data port
  }
else
  forward ETH CI traffic unit to Data Port
endif
```

NOTE – Further filtering of OAM traffic units is performed by the OAM MEL filter process which forms part of the ETH adaptation functions specified in clause 9.3.

MIP OAM insertion process

The MIP OAM insertion process inserts OAM traffic units that are generated in the ETHDi_FT_Sk process into the stream of traffic units.

For all ETH_CI_D received on any but the data input port, the SA field is overwritten with the MI_MIP_MAC value. In the M_SDU field the EtherType value is overwritten with the OAM EtherType value (89-02) and the MEL field is overwritten with the MI_MEL value.

This ensures that every generated OAM field has the correct SA, EtherType and MEL.

MIP LBM reception process

This process is defined in clause 8.1.8 where the LB protocol is defined. Clause 8.1.8.4 defines the LBM MIP reception process.

MIP LTM reception process

This process is defined in clause 8.1.13 where the LT protocol is defined. Clause 8.1.13.4 defines the MIP LTM reception process. This process may be regarded as the LT responder which is located outside of this MIP independently, however, the process itself is the same.

Defects	None.
Consequent actions	None.
Defect correlations	None.
Performance monitoring	None.

9.4.3 ETHD to ETH adaptation functions (ETHD/ETH_A)

The ETHD/ETH adaptation function is an empty function; it is included to satisfy the modelling rules.

The bidirectional ETHD/ETH adaptation function is performed by a co-located pair of ETHD/ETH adaptation source (ETHD/ETH_A_So) and sink (ETHD/ETH_A_Sk) functions.

9.4.3.1 ETHD to ETH adaptation source function (ETHD/ETH_A_So)

The ETHD/ETH_A_So function symbol is shown in Figure 9-54 and the process in Figure 9-55.

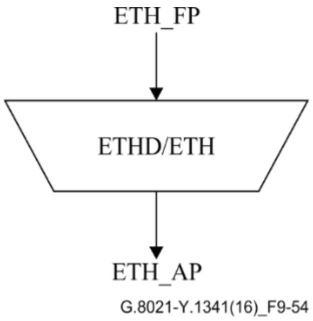


Figure 9-54 – ETHD/ETH_A_So symbol

Interfaces

Table 9-20 – ETHD/ETH_A_So interfaces

Inputs	Outputs
ETH_FP: ETH_CI_D ETH_CI_P ETH_CI_DE *See specific OAM process for additional inputs*	**ETH_AP**: ETHD_AI_D ETHD_AI_P ETHD_AI_DE *See specific OAM process for additional inputs*

Processes

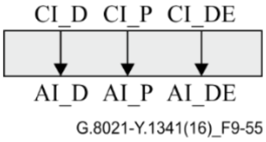


Figure 9-55 – ETHD/ETH_A_So process

Defects	None.
Consequent actions	None.
Defect correlations	None.
Performance monitoring	None.

9.4.3.2 ETHD to ETH adaptation sink function (ETHD/ETH_A_Sk)

The ETHD/ETH_A_Sk function symbol is shown in Figure 9-56 and the process in Figure 9-57.

Symbol

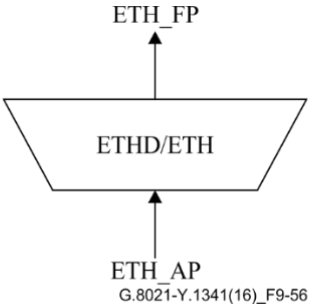


Figure 9-56 – ETHD/ETH_A_Sk symbol

Interfaces

Table 9-21 – ETHD/ETH_A_Sk interfaces

Inputs	Outputs
ETH_AP: ETHD_AI_D ETHD_AI_P ETHD_AI_DE	**ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE

Processes

The ETHD/ETH_A_Sk process diagram is shown in Figure 9-57.

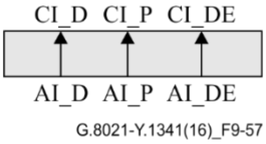


Figure 9-57 – ETHD/ETH_A_Sk process

9.4.4 ETHDi to ETH adaptation functions (ETHDi/ETH_A)

The ETHDi/ETH inserts and extracts the R-APS information into or from the stream of ETH_CI.

9.4.4.1 ETHDi to ETH adaptation source function (ETHDi/ETH_A_So)

This function allows the insertion of R-APS information into a stream of ETH_CI.

Symbol

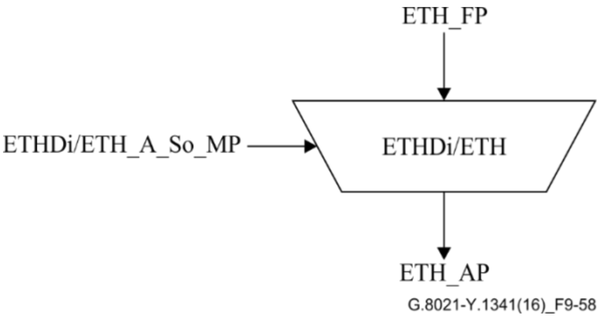


Figure 9-58 – ETHDi/ETH_A_So symbol

Interfaces

Table 9-22 – ETHDi/ETH_A_So interfaces

Inputs	Outputs
ETH_FP: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_RAPS **ETHDi/ETH_A_So_MP**: ETHDi/ETH_A_So_MI_MEL ETHDi/ETH_A_So_MI_RAPS_Pri ETHDi/ETH_A_So_MI_MIP_MAC	**ETH_AP**: ETH_AI_D ETH_AI_P ETH_AI_DE

Processes

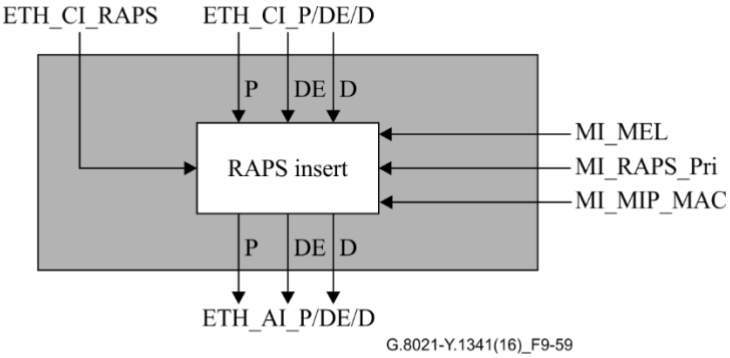


Figure 9-59 – ETHDi/ETH_A_So process

RAPS insert

The RAPS insert process encodes the ETH_CI_RAPS signal into the ETH_CI_D signal of an ETH_CI traffic unit; the resulting RAPS traffic unit is inserted into the stream of incoming traffic units, i.e., the outgoing stream consist of the incoming traffic units and the inserted RAPS traffic units. The ETH_CI_RAPS signal contains the RAPS specific information as defined in [ITU-T G.8032].

The ETH_CI_D signal contains a source and destination address field and an M_SDU field. The format of the M_SDU field for RAPS traffic units is determined by the ETH_CI_RAPS signal. The MEL in the M_SDU field is determined by the MI_MEL input parameter.

> The values of the source and destination address fields in the ETH_CI_D signal are determined by the local MAC address of the maintenance entity group intermediate point (MIP) (MI_MIP_MAC) and the ring multicast address as described in [ITU-T G.8032]. The value of the ring multicast MAC address is 01-19-A7-00-00-01. The value of MI_ MIP_MAC should be a valid unicast MAC address.

The value of the ETH_CI_P signal associated with the generated RAPS traffic units is determined by the MI_RAPS_Pri input parameter.

The value of the ETH_CI_DE signal associated with the generated RAPS traffic units is set to drop ineligible.

9.4.4.2 ETHDi to ETH adaptation sink function (ETHDi/ETH_A_Sk)

This function extracts the RAPS information from the RAPS traffic units without filtering the traffic unit.

Symbol

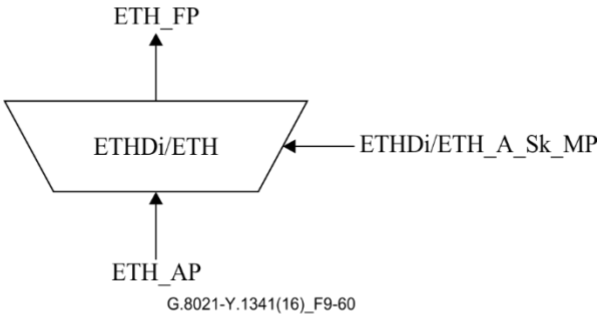


Figure 9-60 – ETHDi/ETH_A_Sk symbol

Interfaces

Table 9-23 – ETHDi/ETH_A_Sk interfaces

Inputs	Outputs
ETH_AP: ETH_AI_D ETH_AI_P ETH_AI_DE ETH_AI_TSF **ETHDi/ETH_A_Sk_MP**: ETHDi/ETH_A_Sk_MI_MEL	**ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_RAPS ETH_CI_SSF

NOTE – Currently in this Recommendation, for the ETHDi_FT_Sk, no consequent action for the ETH_CI_SSF input has been defined. However the consequent action should be ETH_AI_TSF output to propagate the failure information.

Processes

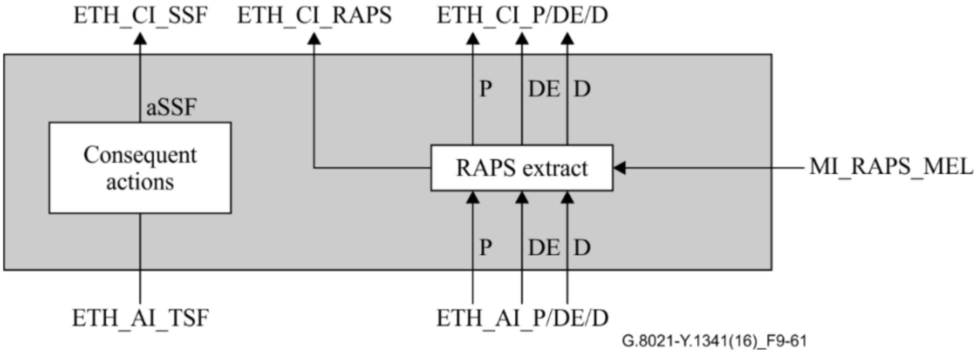


Figure 9-61 – ETHDi/ETH_A_Sk process

RAPS extract

The RAPS extract process extracts ETH_CI_RAPS signals from the incoming stream of ETH_CI traffic units without filtering the RAPS traffic unit. ETH_CI_RAPS signals are only extracted if they belong to the MEL as defined by the MI_MEL input parameter.

If an incoming traffic unit is an RAPS traffic unit belonging to the MEL defined by MI_MEL, the traffic unit will be duplicated. The original RAPS traffic unit will be transparently forwarded and the ETH_CI_RAPS signal will be extracted from the duplicate. The ETH_CI_RAPS is the RAPS specific information contained in the received traffic unit. All other traffic units will be transparently forwarded without being duplicated. The encoding of the ETH_CI_D signal for RAPS frames is defined in clause 9.10 of [ITU-T G.8013].

The criteria for filtering are based on the values of the fields within the M_SDU field of the ETH_CI_D signal:

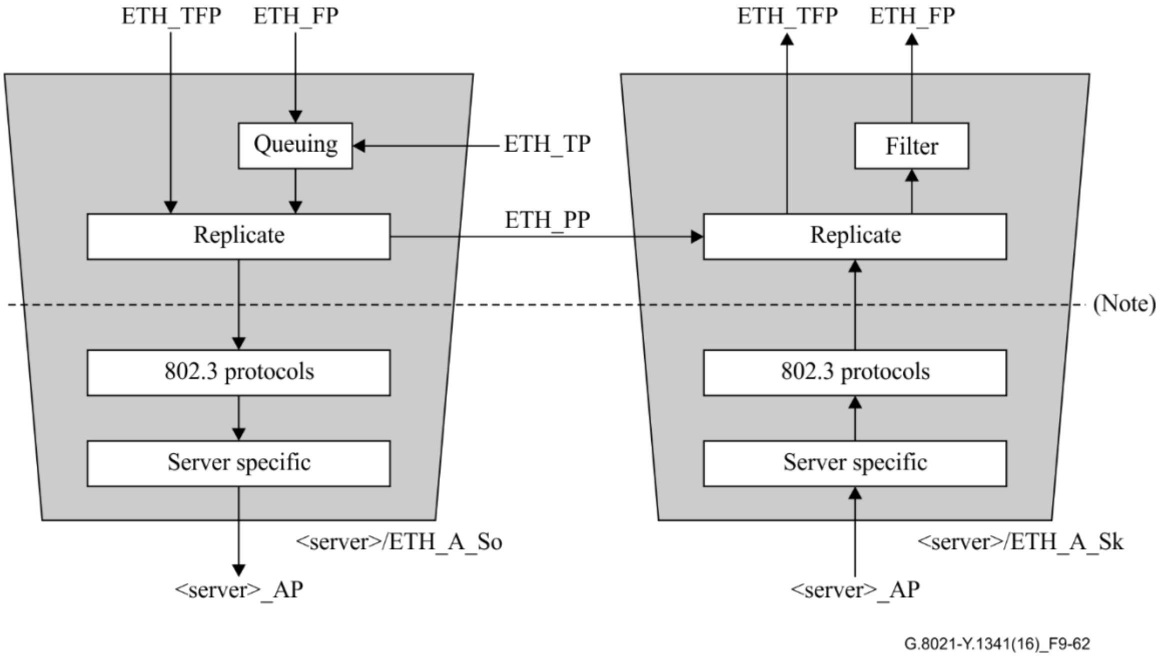
- length/type field equals the OAM EtherType (89-02)
- MEL field equals MI_MEL
- OAM type equals RAPS (40), as defined in clause 9.1 of [ITU-T G.8013].

Defects	None.
Consequent actions	aSSF ← AI_TSF
Defect correlations	None.
Performance monitoring	None.

9.5 Server to ETH adaptation functions (<server>/ETH_A)

There are two basic types of Server to ETH adaptation functions defined in [ITU-T G.8010]: single ETH flow point (<server>/ETH_A) and multiple ETH flow points (<server>/ETH-m_A).

Figure 9-62 presents a high level view of the processes that are present in a generic server to ETH adaptation function (<server>/ETH). The information crossing the <server>/ETH termination flow point (ETH_TFP) is referred to as the ETH characteristic information (ETH_CI). The information crossing the server layer access point (<server>_AP) is referred to as the server-specific adapted information (<server>_AI). Note that for some server signals not all processes need to be present, as defined in the server specific adaptation functions.



NOTE – This interface is shown for reference only. It corresponds to the ISS interface in the IEEE 802 model.

Figure 9-62 – Server to ETH adaptation functions

The following generic processes are specified: "Filter" in clause 8.3, "Queues" in clause 8.2, "Replicate" in clause 8.4, and "802.3 Protocols" in clause 8.5. Server-specific processes are specified in server-specific clauses.

The <server>/ETH-m_A is defined as a compound function of the <server>/ETH_A atomic function, an optional NCP MEP compound function (see clause 9.8.1) and the ETHx/ETH-m_A atomic function (see clause 9.3.3).

NOTE 1 – Filtering in the <server>/ETH_A sink adaptation function is not applied to frames forwarded to the ETH_TFP. The processes connected to this ETH_TFP should filter ETH_CI or process it.

NOTE 2 – Queueing of frames in the source direction is also not applied for frames from the ETH_TFP. If queueing of frames in the sink direction is required when traffic conditioning is applied, this will be included in the traffic conditioning function.

NOTE 3 – For the EPL service defined in [ITU-T G.8001] ETH_TFP is unconnected. For services supporting ETH_TFP in the source direction, prioritization of frames received across the ETH_FP and ETH_TFP interfaces will be required. Such prioritization is for further study.

NOTE 4 – Server to ETH adaptation functions may have the processes of AIS insert (see clause 8.1.4) and LCK generation (see clause 8.1.2), and BNM insert (see clause 8.1.18). Note that Figure 9-62 and related figures in clauses 9.7, 10 and 11 do not explicitly depict those features to avoid introducing the description complexity.

9.6 ETH traffic conditioning and shaping functions (ETH_TCS)

9.6.1 ETH traffic conditioning and shaping functions (ETH_TCS)

9.6.1.1 ETH traffic shaping function (ETH_TCS_So)

Symbol

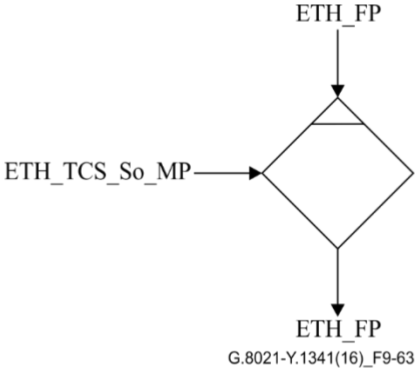


Figure 9-63 – ETH_TCS_So symbol

Interfaces

Table 9-24 – ETH_TCS_So interfaces

Inputs	Outputs
ETH_FP: ETH_CI_D ETH_CI_P ETH_CI_DE **ETH_TCS_So_MP**: ETH_TCS_So_MI_Prio_Config ETH_TCS_So_MI_Queue_Config[] ETH_TCS_So_MI_Sched_Config	**ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE

Processes

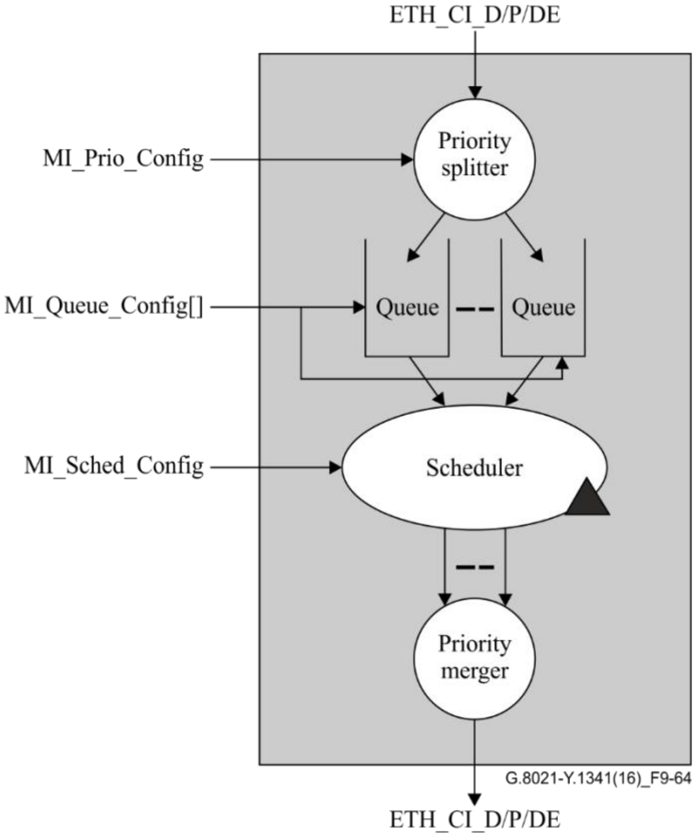


Figure 9-64 – ETH_TCS_So process

Priority splitter

As defined in clause 8.9.2.

Queue

As defined in clause 8.9.1.

Scheduler

As defined in clause 8.9.5.

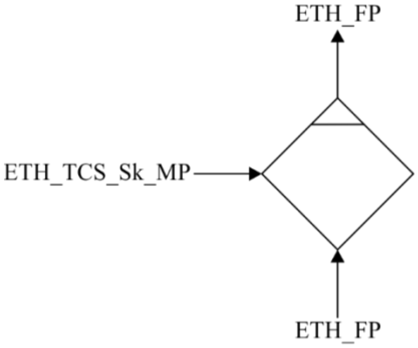
Priority merger

As defined in clause 8.9.3.

Defects	None.
Consequent actions	None.
Defect correlations	None.
Performance monitoring	None.

9.6.1.2 ETH traffic conditioning function (ETH_TCS_Sk)

Symbol



G.8021-Y.1341(16)_F9-65

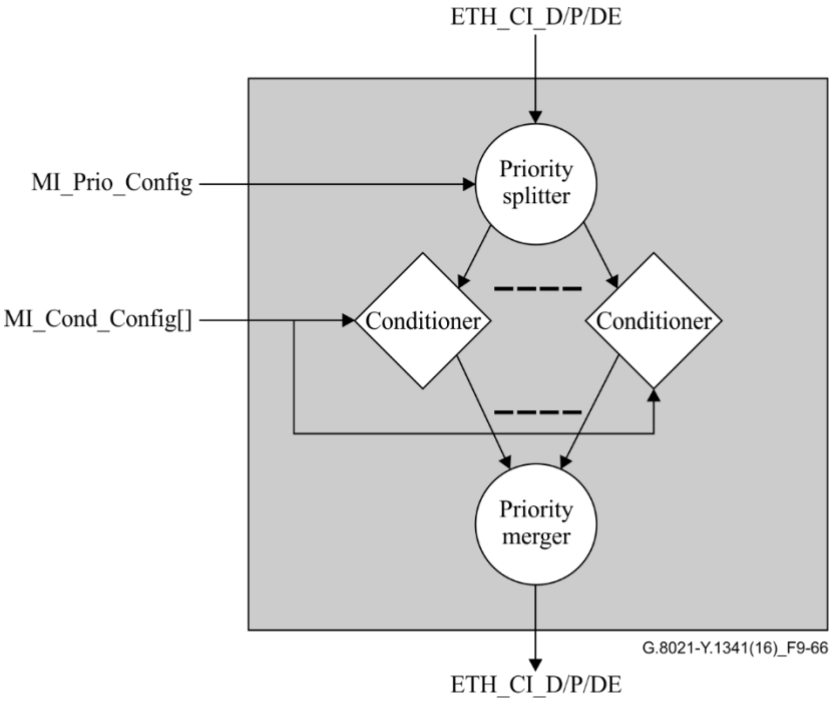
Figure 9-65 – ETH_TCS_Sk symbol

Interfaces

Table 9-25 – ETH_TCS_Sk interfaces

Inputs	Outputs
ETH_FP: ETH_CI_D ETH_CI_P ETH_CI_DE **ETH_TCS_Sk_MP**: ETH_TCS_Sk_MI_Prio_Config ETH_TCS_Sk_MI_Cond_Config[]	**ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE

Processes



G.8021-Y.1341(16)_F9-66

Figure 9-66 – ETH_TCS_Sk processes

Priority splitter

As defined in clause 8.9.2.

Conditioner

As defined in clause 8.9.4.

Priority merger

As defined in clause 8.9.3.

Defects	None.
Consequent actions	None.
Defect correlations	None.
Performance monitoring	None.

9.6.2 ETH group traffic conditioning and shaping functions (ETH_GTCS)

9.6.2.1 ETH group traffic shaping function (ETH_GTCS_So)

Symbol

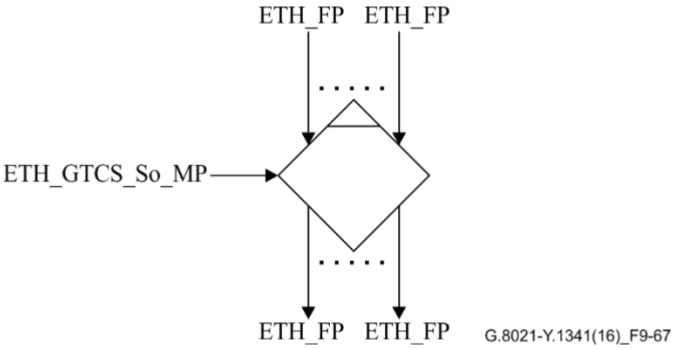


Figure 9-67 – ETH_GTCS_So symbol

Interfaces

Table 9-26 – ETH_GTCS_So interfaces

Inputs	Outputs
ETH_FP: ETH_CI_D ETH_CI_P ETH_CI_DE **ETH_GTCS_So_MP**: ETH_GTCS_So_MI_Prio_Config[] ETH_GTCS_So_MI_Queue_Config[][] ETH_GTCS_So_MI_Sched_Config	**ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE

Processes

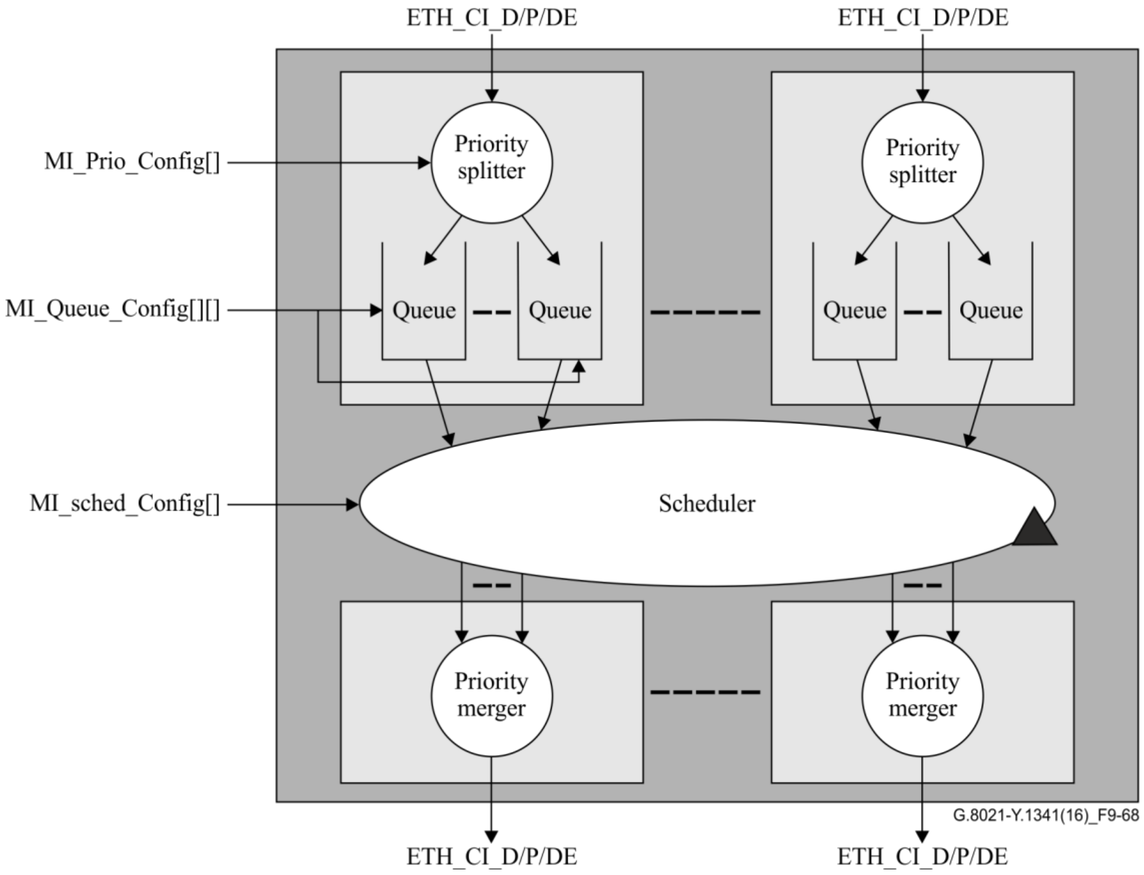


Figure 9-68 – ETH_GTCS_So processes

Priority splitter

As defined in clause 8.9.2.

Queue

As defined in clause 8.9.1.

Scheduler

As defined in clause 8.9.5.

Priority merger

As defined in clause 8.9.3.

Defects None.

Consequent actions None.

Defect correlations None.

Performance monitoring None.

9.6.2.2 ETH group traffic conditioning function (ETH_GTCS_Sk)

For ETH group traffic, the traffic conditioning process is performed per flow point but there is no correlation between each process. Threfore, an ETH_GTCS_Sk function can be modelled by multiple ETH_TCS_Sk functions and no specific function is defined in this Recommendation.

9.7 ETH link aggregation functions

The ETH link aggregation functions model the link aggregation functionality as described in [IEEE 802.1AX].

The generic model used is shown in Figures 9-69 and 9-70. Figure 9-69 shows the simplified model for the case of one single aggregator, while Figure 9-70 shows the generic model for the case of several aggregators. Np denotes the number of members in the LAG, while Na is the number of aggregators.

NOTE – Figures 9-69 and 9-70 simply decompose the internal processes of ETH-LAG-Np-Na_FT function from a modelling standpoint. This flow termination function only includes the ETHx_FT functions of the optional NCM MEPs defined in clause 9.8.

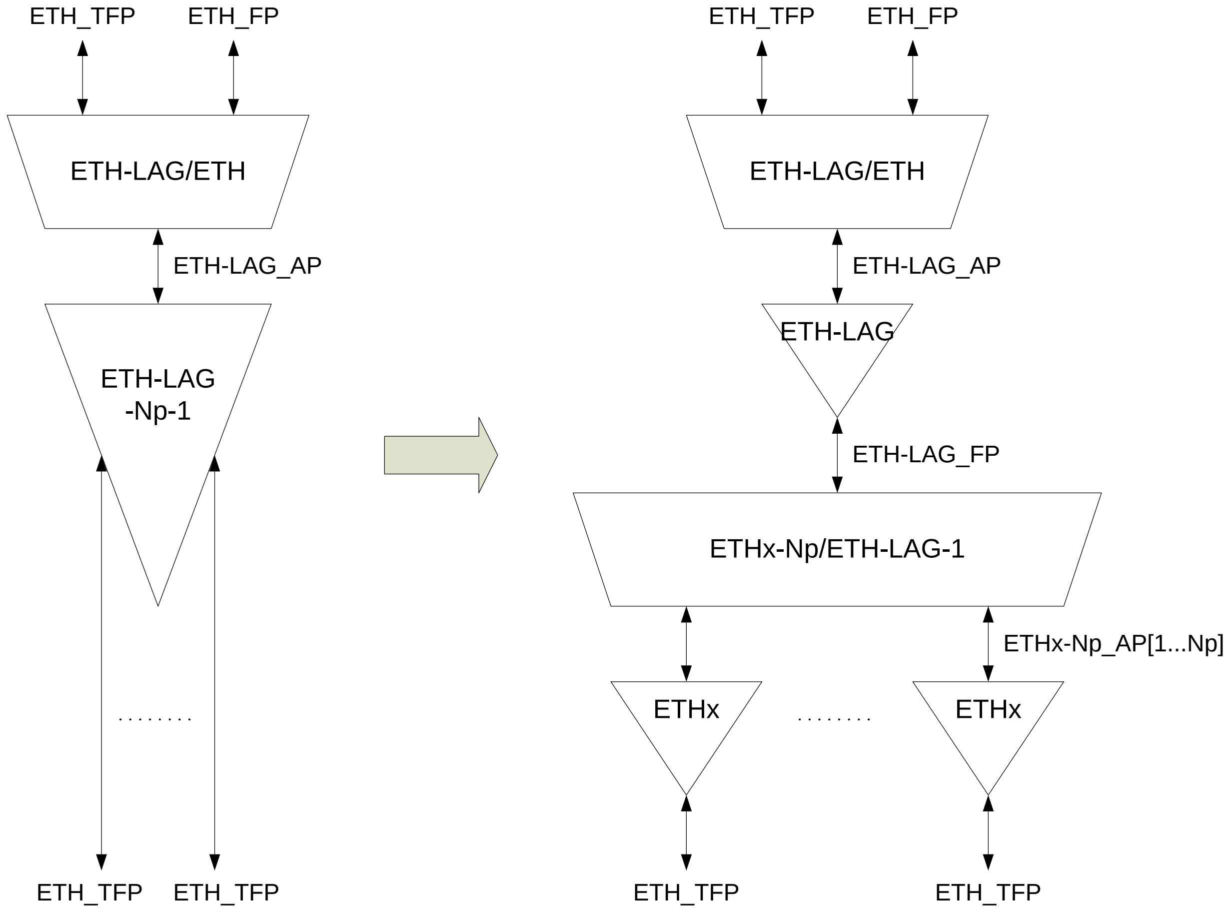


Figure 9-69 – Simplified model of Ethernet link aggregation with decomposition of ETH-LAG-Np-Na_FT function for Na=1

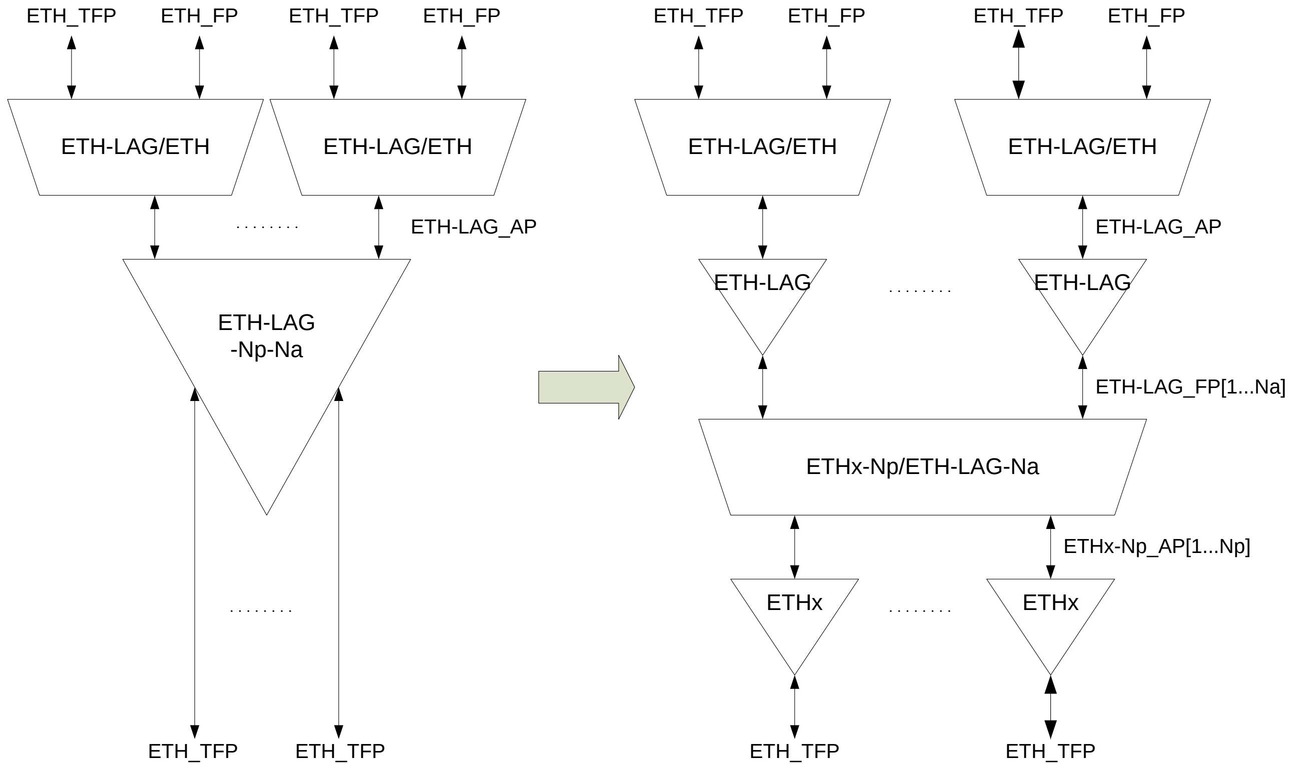


Figure 9-70 – Generic model of Ethernet link aggregation with decomposition of ETH-LAG-Np-Na_FT function

9.7.1 ETH link aggregation layer flow termination function (ETH-LAG-Np-Na_FT)

The ETH-LAG-Np-Na_FT function is decomposed as shown in Figures 9-69 and 9-70. The ETHx_FT function is described in clause 9.2.1.

NOTE – ETH-LAG-Np-Na_FT functions always consist of a pair of identically-sized source and sink functions (i.e., a source function with certain values of Na/Np and a sink function with the same Na/Np values), as per [IEEE 802.3].

NOTE – In principle the ETH_TFP can be connected to any adaptation function that supports an ETH_TFP. In practice, this will normally be an xTSi[G]/ETH_A function (see clause 6.1 of [ITU-T G.8023]), representing an [IEEE 802.3] PHY, but this is not strictly required.

9.7.1.1 ETH link aggregation adaptation source function (ETHx-Np/ETH-LAG-Na_A_So)

Symbol

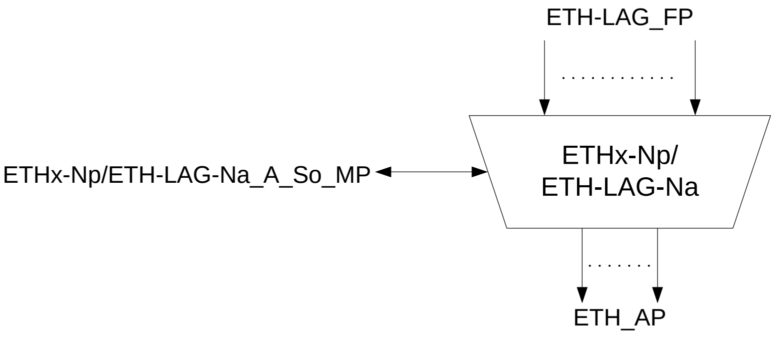


Figure 9-71 – ETHx-Np/ETH-LAG-Na_A_So symbol

Interfaces

Table 9-27 – ETHx-Np/ETH-LAG-Na_A_So interfaces

Inputs	Outputs
ETH-LAG_FP: ETH-LAG-Na_CI_D = ETH-LAG_CI[1..Na]_D ETH-LAG-Na_CI_P = ETH-LAG_CI[1..Na]_P ETH-LAG-Na_CI_DE = ETH-LAG_CI[1..Na]_DE **ETHx-Np/ETH-LAG-Na _A_So_MP**: ETHx-Np/ETH-LAG-Na_A_So_ MI_Agg[1..Na]_AP_List ETHx-Np/ETH-LAG-Na_A_So_ MI_AggPort[1..Np]_ ActorAdmin_State	**ETH_AP**: ETH-Np_AI_D = ETH_AI[1..Np]_D ETH-Np_AI_P = ETH_AI[1..Np]_P ETH-Np_AI_DE = ETH_AI[1..Np]_DE **ETHx-Np/ETH-LAG-Na _A_So_MP**: ETHx-Np/ETH-LAG-Na_A_So_ MI_Agg[1..Na]_ ActorSystemID ActorSystemPriority ActorOperKey PartnerSystemID PartnerSystemPriority PartnerOperKey DataRate CollectorMaxDelay ETHx-Np/ETH-LAG-Na_A_So_ MI_AggPort[1..Np]_ ActorOperKey PartnerOperSystemPriority PartnerOperSystemID PartnerOperKey ActorPort ActorPortPriority PartnerOperPort PartnerOperPortPriority ActorOperState PartnerOperState ETHx-Np/ETH-LAG-Na_A_So_ MI_pAggOctetsTxOK[1..Na] ETHx-Np/ETH-LAG-Na_A_So_ MI_pAggFramesTxOK[1..Na]

NOTE 1 – The signals MI_Agg[1..Na]_… and MI_AggPort[1..Np]_… represent the attributes of the "Aggregator" and "Aggregator Port" objects of the same name in the model in clause 6.7 of [IEEE 802.1AX]. As an example, the output MI_Agg[k]_PartnerSystemID corresponds to the IEEE read-only attribute aAggPartnerSystemID for aggregator object #k.

NOTE 2 – For the purposes of Ethernet transport equipment, the above table contains the minimum set of aggregator and aggregator port inputs and outputs to be supported. This set is a subset of the [IEEE 802.1AX] model, of which some attributes have been omitted because they are specific to the IEEE management philosophy or for simplification in transport equipment. All parameters not explicitly settable per the table above take their default values as per [IEEE 802.1AX].

NOTE 3 – This is the minimum set of common requirements that transport equipment must fulfil.

Processes

A process diagram of this function is shown in Figure 9-72.

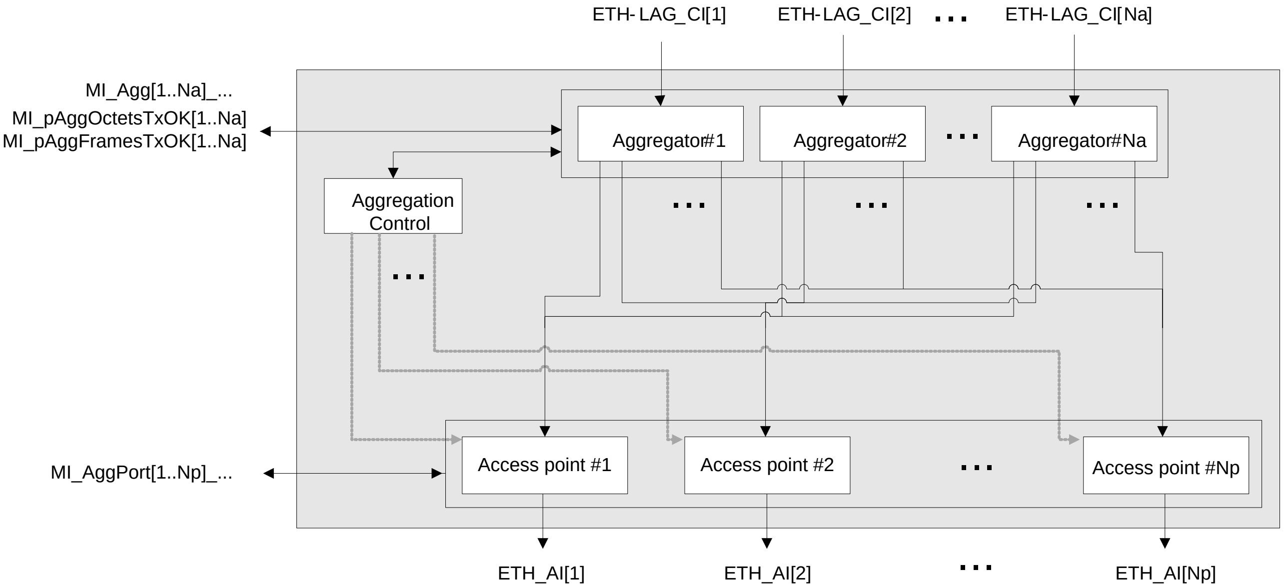


Figure 9-72 – ETHx-Np/ETH-LAG-Na_A_So processes

The input MI_Agg[1..Na]_AP_List defines for each aggregator, which ports (access points) are provisioned to be assigned to it. The AP_List attributes for all aggregators are disjunct lists.

The system shall assign a unique value for the parameter aAggActorAdminKey for each aggregator in the system. The system shall also assign the value used for each aggregator to the parameter aAggPortActorAdminKey of all ports in its assigned port list (AP_List).

NOTE 4 – This automated AdminKey assignment is a simplification of the IEEE provisioning model where the keys are provisioned explicitly for each port and aggregator.

NOTE 5 – Automated assignment of PartnerAdminKey attributes is for further study.

Access point

This represents the individual LAG member within the process diagram, including the source part of the control parser/multiplexer process as specified in [IEEE 802.1AX]. The MAC layer processes that are specific to a member are performed by the function that is attached to the ETH_AP for that member.

NOTE 6 – The control parser/multiplexer process is a single process shared between the source and the sink of a pair of source/sink adaptation functions.

Aggregation control

This process is the source part of the process of the same name in [IEEE 802.1AX].

NOTE 7 – The aggregation control process is a single process shared between the source and the sink of a pair of source/sink adaptation functions.

NOTE 8 – As per the IEEE model and given the automated key assignment, only ports from each aggregator's AP_List will be eligible to be selected by that aggregator.

Aggregator

This process is the source part of the process of the same name in [IEEE 802.1AX]. A coupled mux state machine model is used.

NOTE 9 – Each "Aggregator #k" process is a single process shared between the source and the sink of a pair of source/sink adaptation functions.

Defects None.

Consequent actions None.

Defect correlations None.

Performance monitoring

For each aggregator:

MI_pAggOctetsTxOK[1..Na] as per clause 7 of [IEEE 802.1AX].

MI_pAggFramesTxOK[1..Na] as per clause 7 of [IEEE 802.1AX].

9.7.1.2 ETH link aggregation adaptation sink function (ETHx-Np/ETH-LAG-Na_A_Sk)

Symbol

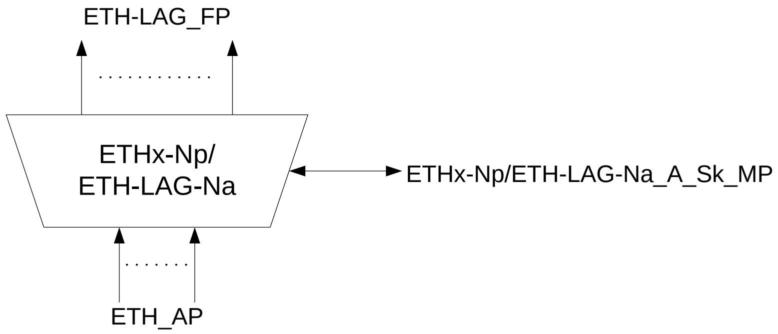


Figure 9-73 – ETHx-Np/ETH-LAG-Na_A_Sk symbol

Interfaces

Table 9-28 – ETHx-Np/ETH-LAG-Na_A_Sk interfaces

Inputs	Outputs
ETH_AP: ETH-Np_AI_D= ETH_AI[1..Np]_D ETH-Np_AI_P= ETH_AI[1..Np]_P ETH-Np_AI_DE= ETH_AI[1..Np]_DE **ETHx-Np/ETH-LAG-Na _A_Sk_MP**: ETHx-Np/ETH-LAG-Na_A_Sk_ MI_PLLThr[1..Na]	**ETH-LAG_FP**: ETH-LAG-Na_CI_D= ETH-LAG_CI[1..Na]_D ETH-LAG-Na_CI_P= ETH-LAG_CI[1..Na]_P ETH-LAG-Na_CI_DE= ETH-LAG_CI[1..Na]_DE ETH- LAG-Na_CI_aSSF= ETH-LAG_CI[1..Na]_aSSF **ETHx-Np/ETH-LAG-Na _A_Sk_MP**: ETHx-Np/ETH-LAG-Na_A_Sk_ MI_cPLL[1..Na] ETHx-Np/ETH-LAG-Na_A_Sk_ MI_cTLL[1..Na] ETHx-Np/ETH-LAG-Na_A_Sk_ MI_pAggOctetsRxOK[1..Na] ETHx-Np/ETH-LAG-Na_A_Sk_ MI_pAggFramesRxOK[1..Na]

Processes

A process diagram of this function is shown in Figure 9-74.

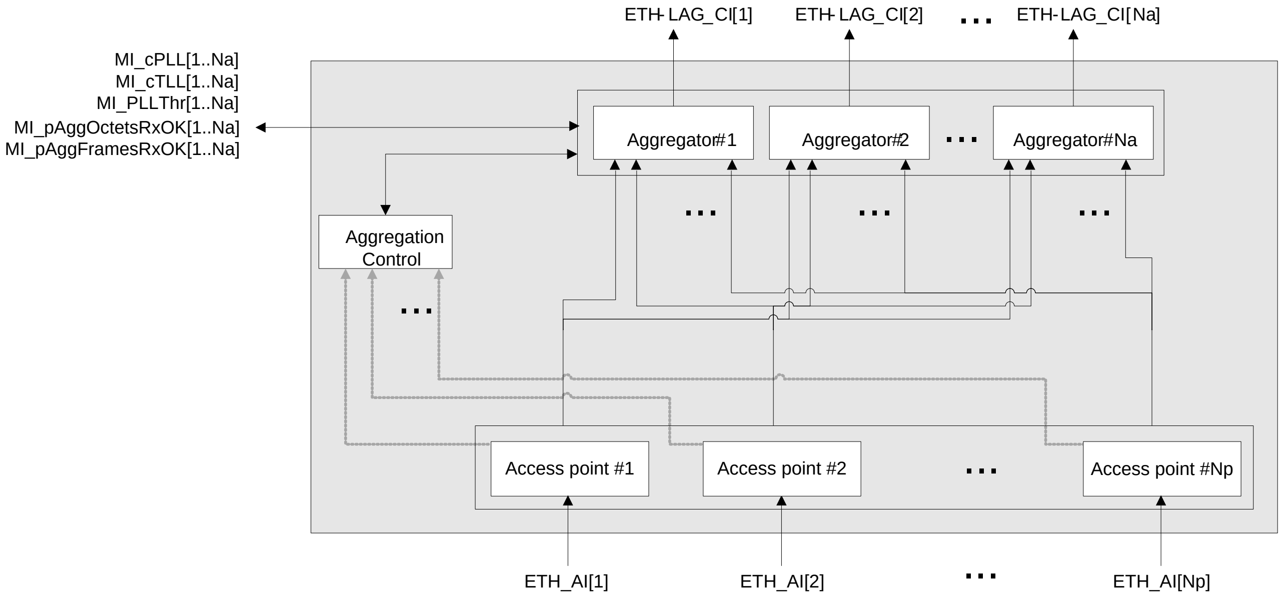


Figure 9-74 – ETHx-Np/ETH-LAG-Na_A_Sk process

Access point

This represents the individual LAG member within the process diagram, including the sink part of the control parser/multiplexer process as specified in [IEEE 802.1AX]. The MAC layer processes that are specific to a member are performed by the function that is attached to the ETH_AP for that member.

NOTE 1 – The control parser/multiplexer process is a single process shared between the source and the sink of a pair of source/sink adaptation functions.

Aggregation control

This process is the sink part of the process of the same name in [IEEE 802.1AX].

NOTE 2 – The aggregation control process is a single process shared between the source and the sink of a pair of source/sink adaptation functions. The parameters used by this bidirectional process are defined in the interface section of the source adaptation function.

Aggregator

This process is the sink part of the process of the same name in [IEEE 802.1AX]. A coupled mux state machine model is used.

NOTE 3 – Each "Aggregator #k" process is a single process shared between the source and the sink of a pair of source/sink adaptation functions. The parameters used by this bidirectional process are defined in the interface section of the source adaptation function.

Defects

dMNCD[j] (Member j not Collecting/Distributing): The defect shall be raised if an access point (port) in an aggregator's AP_List stays outside of the COLLECTING_DISTRIBUTING state for longer than X_{raise} seconds. The defect shall be cleared if the port enters the COLLECTING_DISTRIBUTING state and stays there for X_{clear} seconds.

$$X_{raise} = X_{clear} = 1 \text{ second.}$$

Consequent actions

$$\mathrm{ETH\text{-}LAG\_CI[k]\_aSSF} \leftarrow \prod_{j \in \mathrm{MI\_AP\_List}[k]} \mathrm{dMNCD}[j]$$

NOTE 4 – In other words, aSSF will be raised at the output ETH-LAG_CI[k] of an aggregator if all ports in its assigned port list (AP_List[k]) have the dMNCD defect active.

Defect correlations

Defining

$$\mathrm{mAP\_Active}[k] = \sum_{j \in \mathrm{MI\_AP\_List}[k]} (\mathrm{not\ dMNCD}[j])$$

i.e., the number of active (no-defect) ports among those in an aggregator's AP_List,

then:

$$ETH - LAG\_cTLL[k] \leftarrow mAP\_Active[k] = 0$$

$$ETH - LAG\_cPLL[k] \leftarrow (0 < mAP\_Active[k]) and (mAP\_Active[k] < MI\_PLLThr[k])$$

NOTE 5 – In other words, a cTLL (total link loss) fault cause will be raised if no ports are active for an aggregator. A cPLL (partial link loss) fault cause shall be raised if the number of active ports is less than the provisioned threshold.

Performance monitoring

For each aggregator:

MI_pAggOctetsRxOK[1..Na] as per clause 7 of [IEEE 802.1AX].

MI_pAggFramesRxOK[1..Na] as per clause 7 of [IEEE 802.1AX].

9.7.1.3 ETH link aggregation flow termination source function (ETH-LAG_FT_So)

Symbol

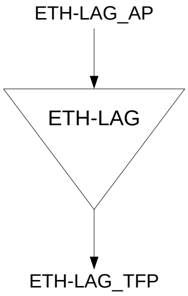


Figure 9-75 – ETH-LAG_FT_So symbol

Interfaces

Table 9-29 – ETH-LAG_FT_So interfaces

Inputs	Outputs
ETH-LAG_AP: ETH-LAG_AI_D ETH-LAG_AI_P ETH-LAG_AI_DE	**ETH-LAG_TFP**: ETH-LAG_CI_D ETH-LAG_CI_P ETH-LAG_CI_DE

Processes

This function just forwards the ETH-LAG_AP information onto the ETH-LAG_FP without manipulation.

Defects None.

Consequent actions None.

Defect correlations None.

Performance monitoring None.

9.7.1.4 ETH link aggregation flow termination sink function (ETH-LAG_FT_Sk)

Symbol

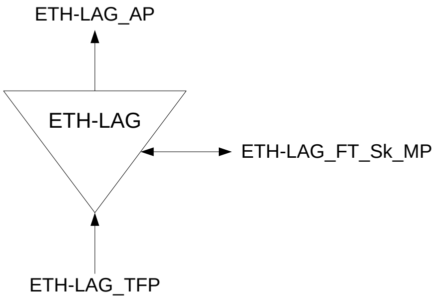


Figure 9-76 – ETH-LAG_FT_Sk symbol

Interfaces

Table 9-30 – ETH-LAG_FT_Sk interfaces

Inputs	Outputs
ETH-LAG_TFP: ETH-LAG_CI_D ETH-LAG_CI_P ETH-LAG_CI_DE ETH-LAG_CI_SSF **ETH-LAG_FT_Sk_MP**: ETH-LAG_FT_Sk_MI_SSF_Reported	**ETH-LAG_AP**: ETH-LAG_AI_D ETH-LAG_AI_P ETH-LAG_AI_DE ETH-LAG_AI_TSF ETH-LAG_AI_AIS **ETH-LAG_FT_Sk_MP**: ETH-LAG_FT_Sk_MI_cSSF

Processes

This function just forwards the ETH-LAG_FP information onto the ETH-LAG_AP without manipulation.

Defects: None.

Consequent actions aTSF ← CI_SSF

Defect correlations cSSF ← CI_SSF and SSF_Reported

Performance monitoring None.

9.7.2 ETH-LAG to ETH adaptation function (ETH-LAG/ETH_A)

9.7.2.1 ETH-LAG to ETH adaptation source function (ETH-LAG/ETH_A_So)

Symbol

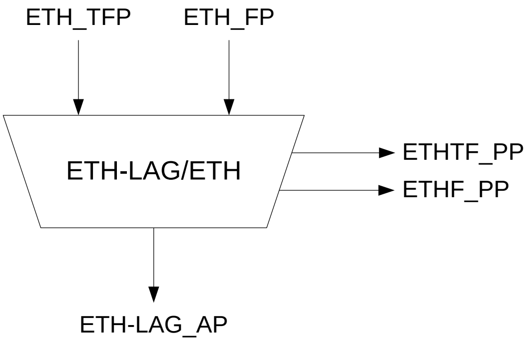


Figure 9-77 – ETH-LAG/ETH_A_So symbol

Interfaces

Table 9-31 – ETH-LAG/ETH_A_So interfaces

Inputs	Outputs
ETH_TFP: ETH_CI_D ETH_CI_P ETH_CI_DE **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE	**ETH-LAG_AP**: ETH-LAG_AI_D ETH-LAG_AI_P ETH-LAG_AI_DE **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE

Processes

A process diagram of this function is shown in Figure 9-78.

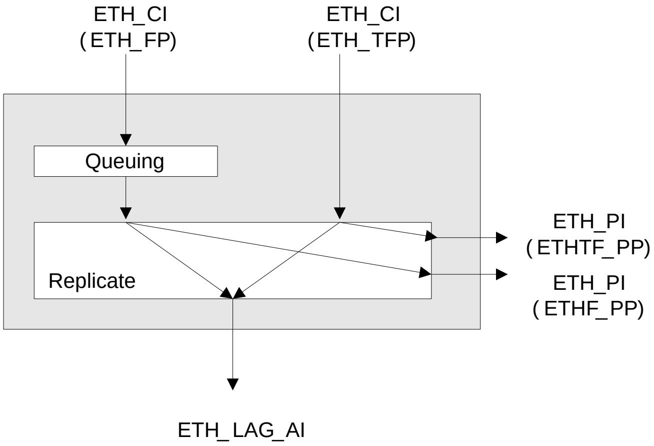


Figure 9-78 – ETH-LAG/ETH_A_So process

See "Queueing" in clause 8.2 and "Replicate" in clause 8.4.

Defects None.

Consequent actions None.

Defect correlations None.

Performance monitoring None.

9.7.2.2 ETH-LAG to ETH adaptation sink function (ETH-LAG/ETH_A_Sk)

Symbol

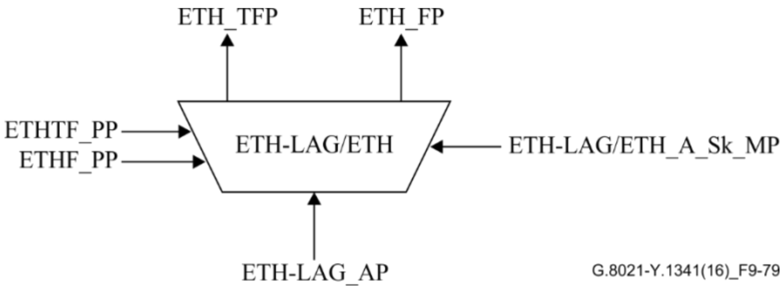


Figure 9-79 – ETH-LAG/ETH_A_Sk symbol

Interfaces

Table 9-32 – ETH-LAG/ETH_A_Sk interfaces

Inputs	Outputs
ETH-LAG_AP: ETH-LAG_AI_D ETH-LAG_AI_P ETH-LAG_AI_DE ETH-LAG-AI_TSF ETH-LAG-AI_AIS **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETH-LAG/ETH_A_Sk_MP**: ETH-LAG/ETH_A_Sk_MI_FilterConfig	**ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF

Processes

A process diagram of this function is shown in Figure 9-80.

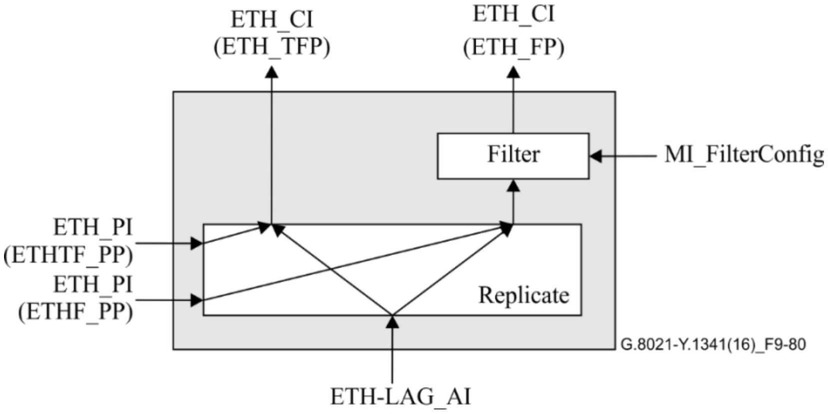


Figure 9-80 – ETH-LAG/ETH_A_Sk process

See "Filter" in clause 8.3 and "Replicate" in clause 8.4.

Defects	None.
Consequent actions	None.
Defect correlations	None.
Performance monitoring	None.

9.8 ETH MEP and MIP functions

MEP and MIP compound functions are defined in [ITU-T G.806]. This clause specifies the composition of those functions with ETH flow termination, adaptation and diagnostic atomic functions described in clauses 9.2, 9.3 and 9.4, respectively.

9.8.1 ETH NCM MEP function

An ETH NCM (network connection monitoring) MEP function is capable of originating, filtering and terminating proactive ETH OAM signals and originating, responding to and terminating diagnostic ETH OAM signals at the NCM MEG levels. The NCM MEP is composed of ETHx_FT, ETHD/ETH_A and ETHDe_FT atomic functions. This MEP is located at the ETH (sub)layer boundary and connected with ETHx/client_A or ETHx/ETH-m_A. Application with other adaptation functions and the model for multiple access points are for further study.

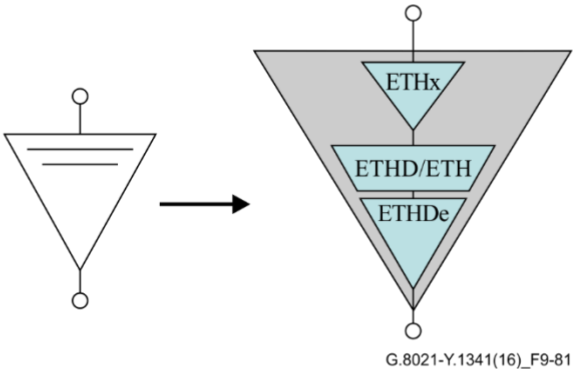


Figure 9-81 – ETH NCM MEP compound functions

9.8.2 ETH TCM MEP function

An ETH TCM (tandem connection monitoring) MEP function is capable of originating, filtering and terminating proactive ETH OAM signals and originating, responding to and terminating diagnostic

ETH OAM signals at one of the TCM MEG levels. The TCM MEP is composed of ETHx/ETH_A, ETHx_FT, ETHD/ETH_A and ETHDe_FT atomic functions. In addition, it can be composed of ETHG/ETH_A, ETHG_FT, ETHD/ETH_A and ETDe_FT if ETH group MEG is configured and multiple access point pools are accommodated. This MEP is located within an ETH (sub)layer.

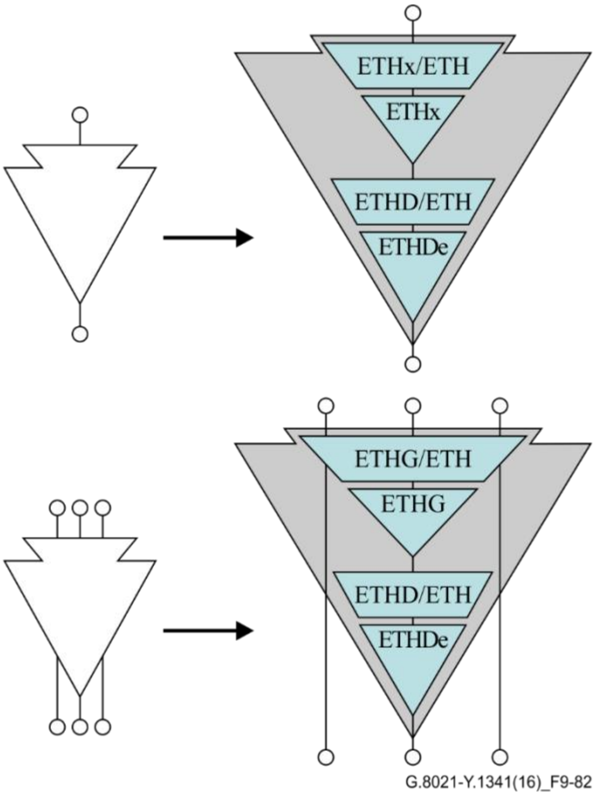


Figure 9-82 – ETH TCM MEP compound functions

9.8.3 ETH MIP function

An ETH MIP function is capable of responding to on-demand ETH OAM signals at one of the MEG levels in both directions. The MIP combines two back-to-back half-MIP functions. It consists of two pairs of ETHD/ETH_A and ETHDi_FT atomic functions, each facing opposite directions. The model for multiple flow points is for further study.

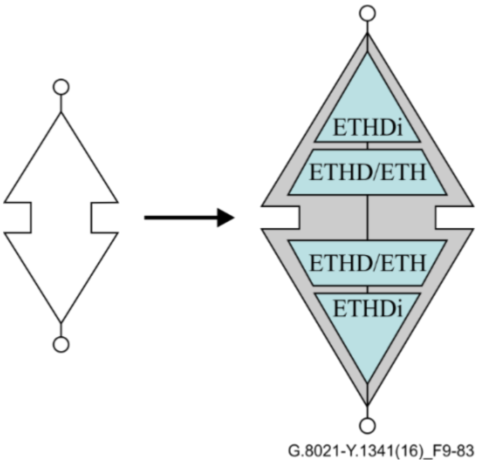


Figure 9-83 – ETH MIP compound functions

9.8.4 ETH half MIP function

An ETH half MIP function is capable of responding to on-demand ETH OAM signals at one of the MEG levels in a single direction. The half MIP is composed of a pair of ETHD/ETH_A and ETHDi_FT atomic functions. The model for multiple flow points is for further study.

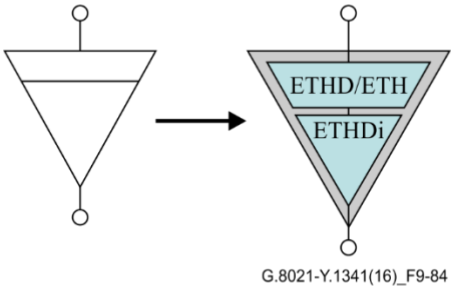


Figure 9-84 – ETH MIP compound functions

10 Ethernet server to ETH adaptation functions

[ITU-T G.8023] defines the atomic functions for the Ethernet physical layer, as defined in [IEEE 802.3], and for the Flex Ethernet interfaces, as defined in [OIF FLEXE IA].

NOTE – The atomic functions defined in clause 10 of [b-ITU-T G.8021-2016] and older versions of this Recommendation have been superseded by the atomic functions defined in [ITU-T G.8023]. See Appendix X for some mapping guidelines between the atomic functions defined in clause 10 of [b-ITU-T G.8021-2016] and the new atomic functions defined in [ITU-T G.8023].

The xTSi[G]/ETH_A functions, defined in clause 6.1 of [ITU-T G.8023], provide the adaptation between the ETH layer network and the Ethernet physical layer defined in [IEEE 802.3].

The FlexEC/ETH_A functions, defined in clause 8.2 of [ITU-T G.8023], provide the adaptation between the ETH layer network and the FlexE client interfaces defined in [OIF FLEXE IA].

11 Non-Ethernet server to ETH adaptation functions

11.1 SDH to ETH adaptation functions (S/ETH_A)

11.1.1 VC-n to ETH adaptation functions (Sn/ETH_A; n = 3, 3-X, 4, 4-X)

This covers non-concatenated, contiguously concatenated, and non-LCAS VCAT. See clause 11.1.2 for LCAS-capable VC-n-Xv/ETH adaptation functions.

11.1.1.1 VC-n to ETH adaptation source function (Sn/ETH_A_So)

This function maps ETH_CI information onto an Sn_AI signal (n = 3, 3-X, 4, 4-X).

Data at the Sn_AP is a VC-n (n = 3, 3-X, 4, 4-X), having a payload as described in [ITU-T G.707], but with indeterminate POH bytes: J1, B3, G1.

Symbol

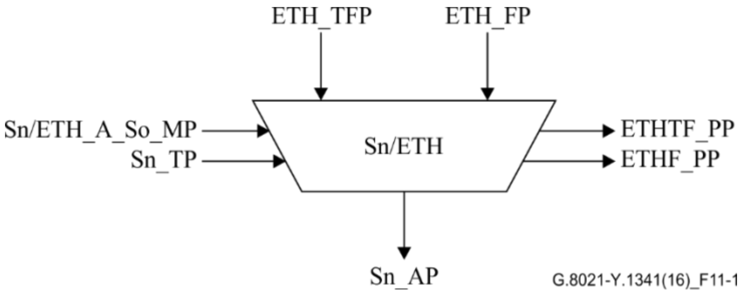


Figure 11-1 – Sn/ETH_A_So symbol

Interfaces

Table 11-1 – Sn/ETH_A_So interfaces

Inputs	Outputs
ETH_TFP: ETH_CI_D ETH_CI_P ETH_CI_DE **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi **Sn_TP**: Sn_TI_Clock Sn_TI_FrameStart **Sn/ETH_A_So_MP**: Sn/ETH_A_So_MI_CSFEnable Sn/ETH_A_So_MI_CSFrdifdiEnable	**Sn_AP**: Sn_AI_Data Sn_AI_ClocK Sn_AI_FrameStart **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE

Processes

A process diagram of this function is shown in Figure 11-2.

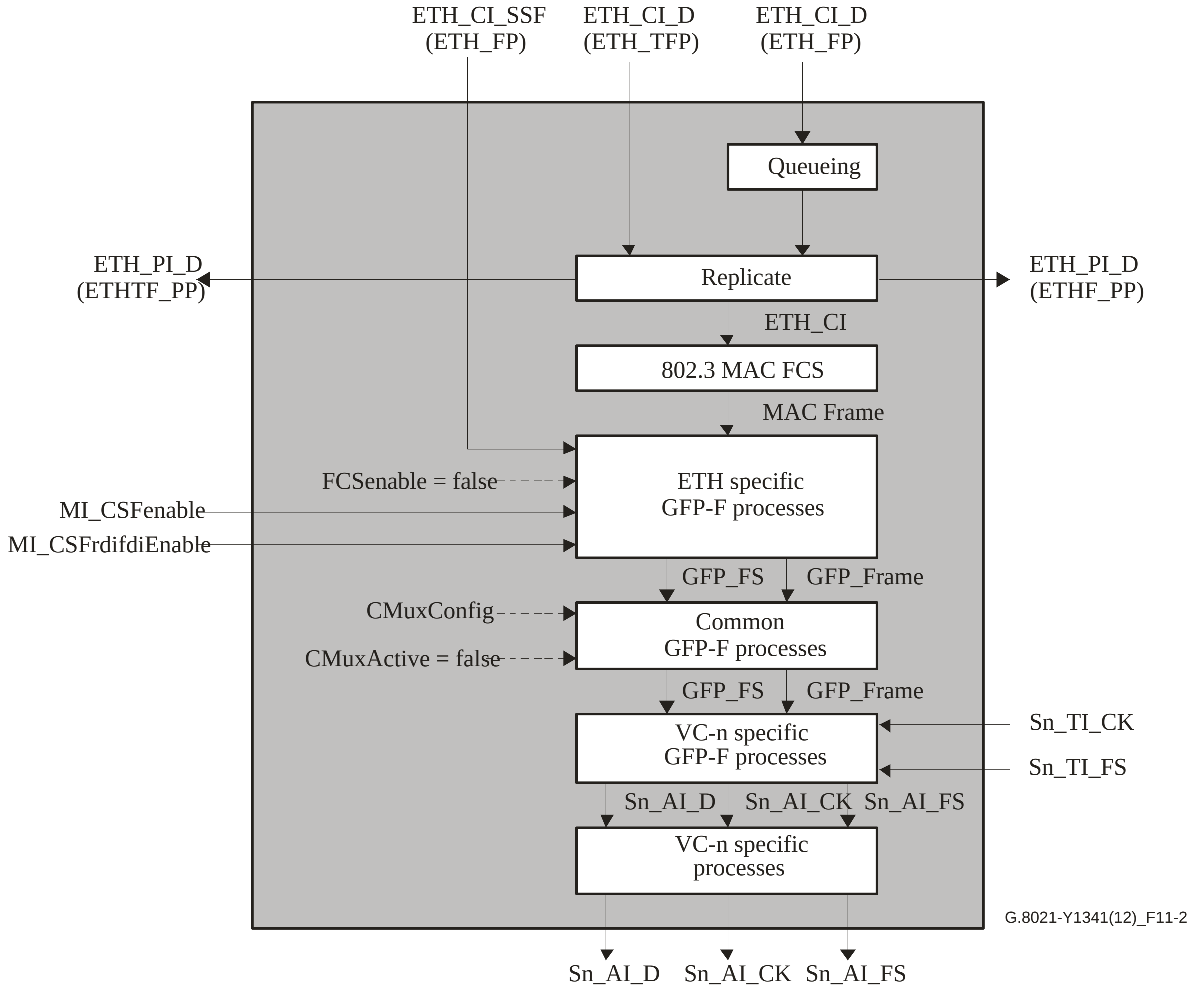


Figure 11-2 – Sn/ETH_A_So process

Queueing process

See clause 8.2.

Replicate" process

See clause 8.4.

802.3 MAC FCS generation

See clause 8.8.1.

Ethernet specific GFP-F source process

See clause 8.8.6.1.

Common GFP source process

See clause 8.5.3.1 of [ITU-T G.806]. GFP channel multiplexing is not supported (CMuxActive=false).

VC-n specific GFP source process

See clause 8.5.2.1 of [ITU-T G.806]. The GFP frames are mapped into the VC-n payload area according to clause 10.6 of [ITU-T G.707].

VC-n specific source process

C2: Signal label information is derived directly from the adaptation function type. The value for GFP mapping in Table 9-11 of [ITU-T G.707] is placed in the C2 byte position.

H4: For Sn/ETH_A_So with n = 3, 4, the H4 byte is sourced as all-zeros.

NOTE 1 – For Sn/ETH_A_So with n = 3-X, 4-X, the H4 byte is undefined at the Sn-X_AP output of this function (as per clause 12 of [ITU-T G.783]).

NOTE 2 – For Sn/ETH_A_So with n = 3, 4, 3-X, 4-X, the K3, F2, F3 bytes are undefined at the Sn-X_AP output of this function (as per clause 12 of [ITU-T G.783]).

Counter processes

For further study.

Defects None.

Consequent actions

aCSF-RDI ← CI_SSFrdi and CSFrdifdiEnable and CSFEnable

aCSF-FDI ← CI_SSFfdi and CSFrdifdiEnable and CSFEnable

aCSF-LOS ← CI_SSF and CSFEnable

Defect correlations None.

Performance monitoring For further study.

11.1.1.2 VC-n to ETH adaptation sink function (Sn/ETH_A_Sk)

This function extracts ETH_CI information from the Sn_AI signal (n = 3, 3-X, 4, 4-X), delivering ETH_CI to ETH_TFP and ETH_FP.

Data at the Sn_AP is as described in [ITU-T G.707].

Symbol

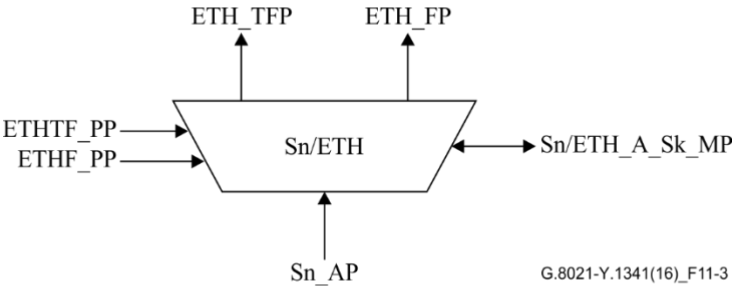


Figure 11-3 – Sn/ETH_A_Sk symbol

Interfaces

Table 11-2 – Sn/ETH_A_Sk interfaces

Inputs	Outputs
Sn_AP: Sn_AI_Data Sn_AI_ClocK Sn_AI_FrameStart Sn_AI_TSF **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **Sn/ETH_A_Sk_MP**: Sn/ETH_A_Sk_MI_FilterConfig Sn/ETH_A_Sk_MI_CSF_Reported Sn/ETH_A_Sk_MI_MAC_Length Sn/ETH_A_Sk_MI_CSFrdifdiEnable	**ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi **Sn/ETH_A_Sk_MP**: Sn/ETH_A_Sk_MI_AcSL Sn/ETH_A_Sk_MI_AcEXI Sn/ETH_A_Sk_MI_AcUPI Sn/ETH_A_Sk_MI_cPLM Sn/ETH_A_Sk_MI_cLFD Sn/ETH_A_Sk_MI_cUPM Sn/ETH_A_Sk_MI_cEXM Sn/ETH_A_Sk_MI_cCSF Sn/ETH_A_Sk_MI_pFCSError

Processes

A process diagram of this function is shown in Figure 11-4.

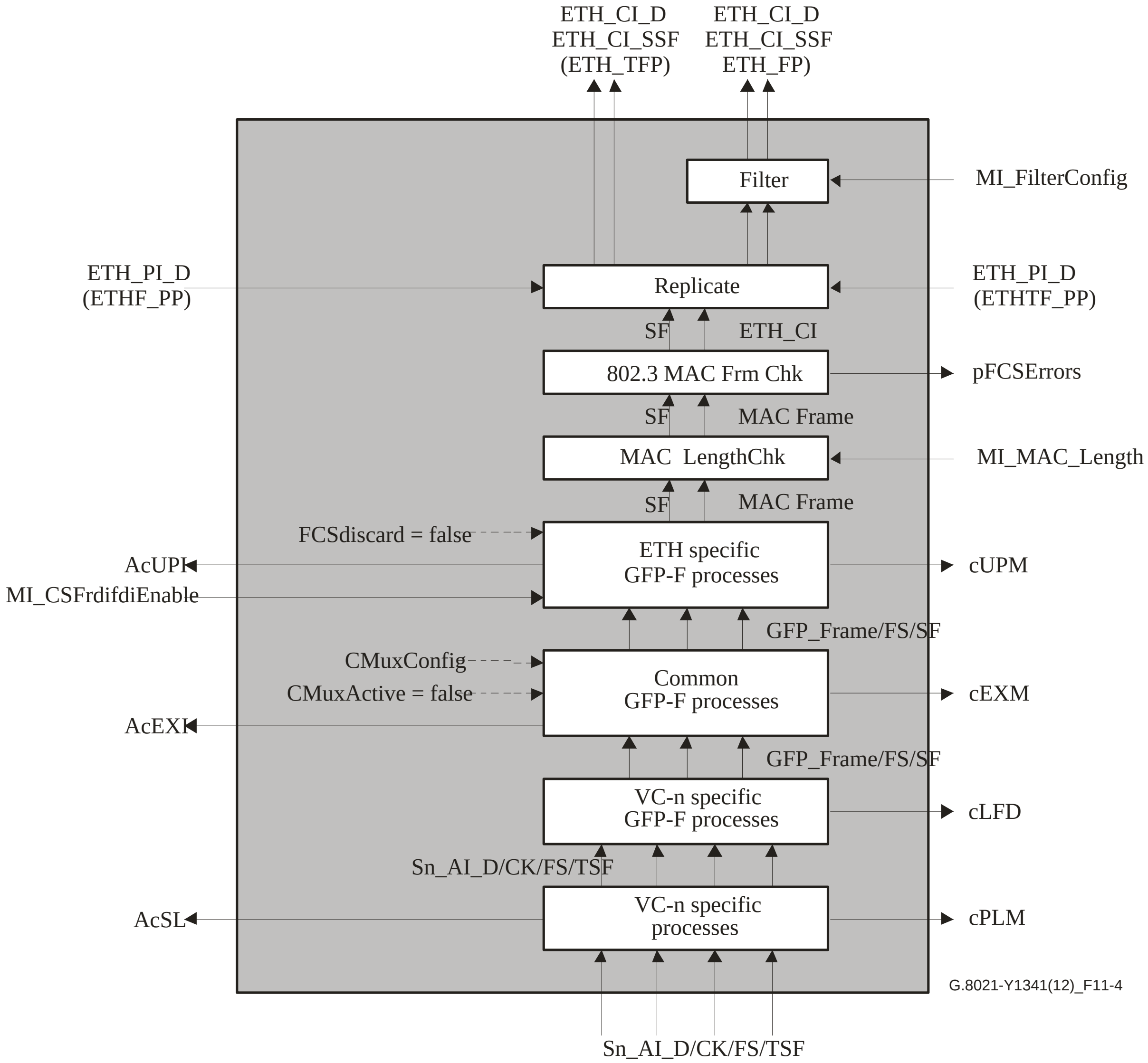


Figure 11-4 – Sn/ETH_A_Sk process

Filter process

See clause 8.3.

Replicate process

See clause 8.4.

802.3 MAC FCS check process

See clause 8.8.2.

Ethernet specific GFP-F sink process

See clause 8.8.6.2.

Common GFP sink process

See clause 8.5.3.2 of [ITU-T G.806]. GFP channel multiplexing is not supported (MI_CMuxActive=false).

VC-n specific GFP sink process

See clause 8.5.2.2 of [ITU-T G.806]. The GFP frames are demapped from the VC-n payload area according to clause 10.6 of [ITU-T G.707].

VC-n specific sink process

C2: The signal label is recovered from the C2 byte as per clause 6.2.4.2 of [ITU-T G.806]. The signal label for GFP mapping in Table 9-11 of [ITU-T G.707] shall be expected. The accepted value of the signal label is also available at the Sn/ETH_A_Sk_MP.

Defects

dPLM – See clause 6.2.4.2 of [ITU-T G.806].

dLFD – See clause 6.2.5.2 of [ITU-T G.806].

dUPM – See clause 6.2.4.3 of [ITU-T G.806].

dEXM – See clause 6.2.4.4 of [ITU-T G.806].

dCSF-LOS – See clause 8.8.6.2.

dCSF-RDI – See clause 8.8.6.2.

dCSF-FDI – See clause 8.8.6.2.

Consequent actions

The function shall perform the following consequent actions:

aSSF ← AI_TSF or dPLM or dLFD or dUPM or dEXM or dCSF-LOS

aSSFrdi ← dCSF-RDI and CSFrdifdiEnable

aSSFfdi ← dCSF-FDI and CSFrdifdiEnable

Defect correlations

The function shall perform the following defect correlations to determine the most probable fault cause (see clause 6.4 of [ITU-T G.806]). This fault cause shall be reported to the EMF.

cPLM ← dPLM and (not AI_TSF);

cLFD ← dLFD and (not dPLM) and (not AI_TSF);

cUPM ← dUPM and (not dEXM) and (not dPLM) and (not dLFD) and (not AI_TSF);

cEXM ← dEXM and (not dPLM) and (not dLFD) and (not AI_TSF);

cCSF ← (dCSF-LOS or dCSF-RDI or dCSF-FDI) and (not dEXM) and (not dUPM) and (not dPLM) and (not dLFD) and (not AI_TSF) and CSF_Reported.

Performance monitoring

The function shall perform the following performance monitoring primitives processing. The performance monitoring primitives shall be reported to the EMF.

pFCSErrors: count of FrameCheckSequenceErrors per second.

NOTE – This primitive is calculated by the MAC FCS Check process.

11.1.2 LCAS-capable VC-n-Xv to ETH adaptation functions (Sn-X-L/ETH_A; n = 3, 4)

11.1.2.1 LCAS-capable VC-n-Xv to ETH adaptation source function (Sn-X-L/ETH_A_So)

This function maps ETH_CI information onto an Sn-X-L_AI signal (n = 3 or 4).

Data at the Sn-X-L_AP is a VC-n-X (n = 3 or 4), having a payload as described in [ITU-T G.707], but with indeterminate POH bytes: J1, B3, G1.

Symbol

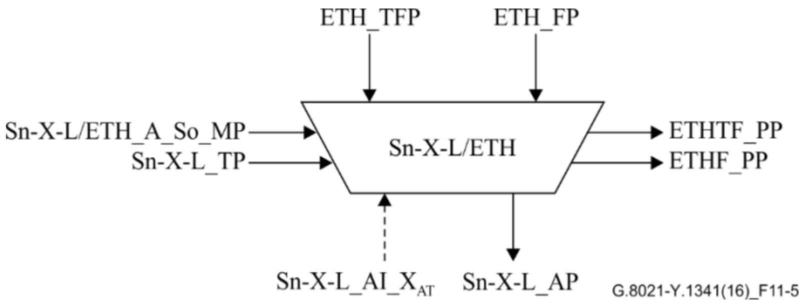


Figure 11-5 – Sn-X-L/ETH_A_So symbol

Interfaces

Table 11-3 – Sn-X-L/ETH_A_So interfaces

Inputs	Outputs
ETH_TFP: ETH_CI_D ETH_CI_P ETH_CI_DE **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi **Sn-X-L_AP**: Sn-X-L_AI_XAT **Sn-X-L_TP**: Sn-X-L_TI_ClocK Sn-X-L_TI_FrameStart **Sn-X-L/ETH_A_So_MP**: Sn-X-L/ETH_A_So_MI_CSFEnable Sn-X-L/ETH_A_So_MI_CSFrdifdiEnable	**Sn-X-L_AP**: Sn-X-L_AI_Data Sn-X-L_AI_ClocK Sn-X-L_AI_FrameStart **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE

Processes

A process diagram of this function is shown in Figure 11-6.

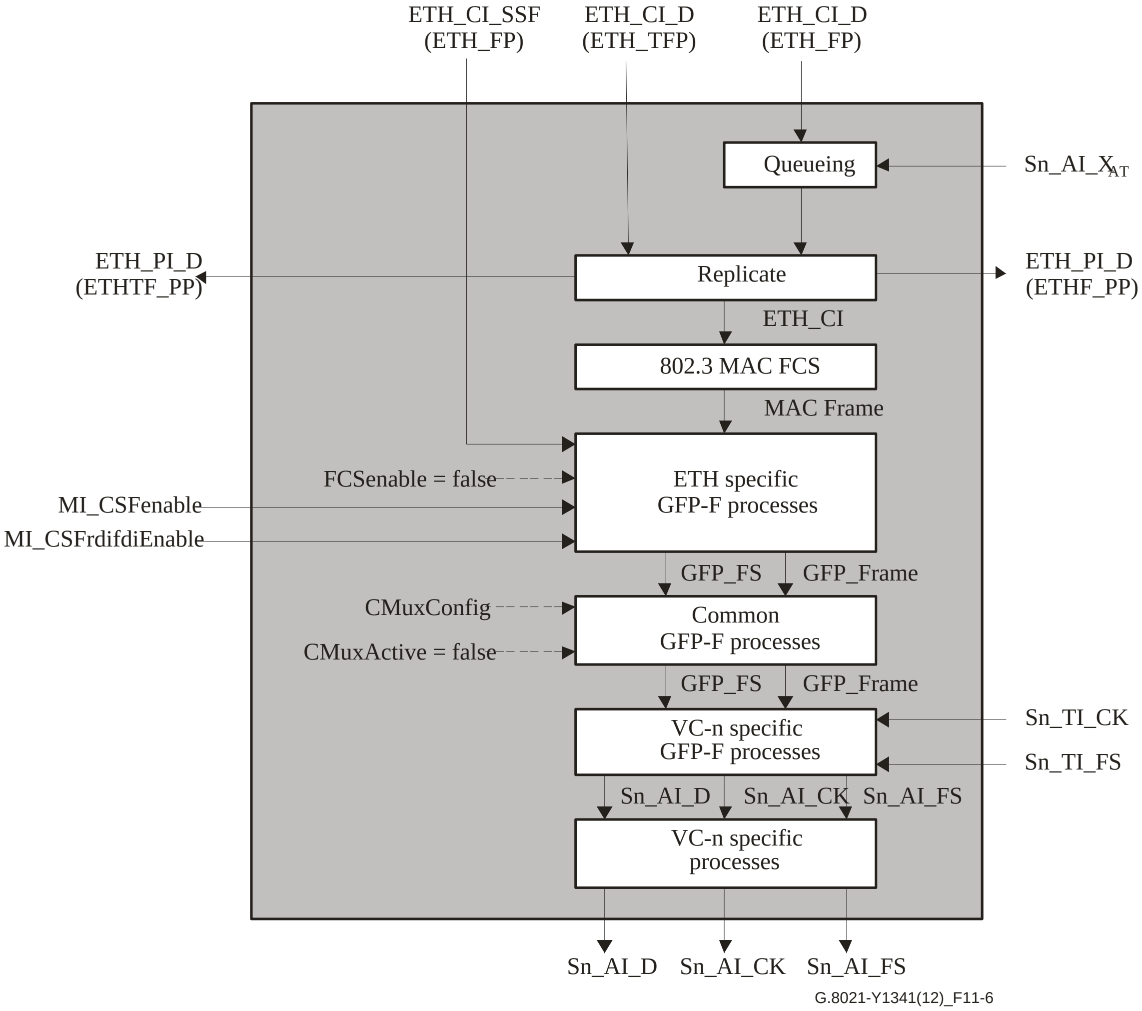


Figure 11-6 – Sn-X-L/ETH_A_So process

See clause 11.1.1.1 for a description of Sn-X-L/ETH_A processes.

Defects None.

Consequent actions

aCSF-RDI ← CI_SSFrdi and CSFrdifdiEnable and CSFEnable

aCSF-FDI ← CI_SSFfdi and CSFrdifdiEnable and CSFEnable

aCSF-LOS ← CI_SSF and CSFEnable

Defect correlations None.

Performance monitoring For further study.

11.1.2.2 LCAS-capable VC-n-Xv to ETH adaptation sink function (Sn-X-L/ETH_A_Sk)

This function extracts ETH_CI information from a VC-n-Xv server signal (n = 3 or 4), delivering ETH_CI to ETH_TFP and ETH_FP.

Data at the Sn-X-L_AP is a VC-n-Xv (n = 3 or 4), having a payload as described in [ITU-T G.707].

Symbol

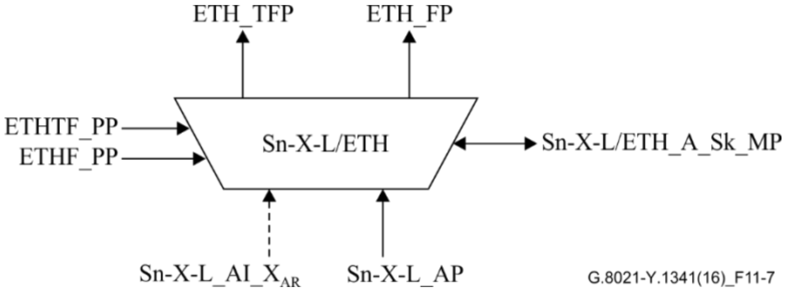


Figure 11-7 – Sn-X-L/ETH_A_Sk symbol

Interfaces

Table 11-4 – Sn-X-L/ETH_A_Sk interfaces

Inputs	Outputs
Sn-X-L_AP: Sn-X-L_AI_Data Sn-X-L_AI_ClocK Sn-X-L_AI_FrameStart Sn-X-L_AI_TSF Sn-X-L_AI_XAR **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **Sn-X-L/ETH_A_Sk_MP**: Sn-X-L/ETH_A_Sk_MI_FilterConfig Sn-X-L/ETH_A_Sk_MI_CSF_Reported Sn-X-L/ETH_A_Sk_MI_MAC_Length Sn-X-L/ETH_A_Sk_MI_CSFrdifdiEnable	**ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi **Sn-X-L/ETH_A_Sk_MP**: Sn-X-L/ETH_A_Sk_MI_AcSL Sn-X-L/ETH_A_Sk_MI_AcEXI Sn-X-L/ETH_A_Sk_MI_AcUPI Sn-X-L/ETH_A_Sk_MI_cPLM Sn-X-L/ETH_A_Sk_MI_cLFD Sn-X-L/ETH_A_Sk_MI_cUPM Sn-X-L/ETH_A_Sk_MI_cEXM Sn-X-L/ETH_A_Sk_MI_cCSF Sn-X-L/ETH_A_Sk_MI_pFCSError

Processes

See process diagram and process description in clause 11.1.1.2. The additional Sn-X-L_AI_X$_{AR}$ interface is not connected to any of the internal processes.

Defects

dPLM – See clause 6.2.4.2 of [ITU-T G.806].

dLFD – See clause 6.2.5.2 of [ITU-T G.806].

dUPM – See clause 6.2.4.3 of [ITU-T G.806].

dEXM – See clause 6.2.4.4 of [ITU-T G.806].

dCSF-LOS – See clause 8.8.6.2.

dCSF-RDI – See clause 8.8.6.2.

dCSF-FDI – See clause 8.8.6.2.

Consequent actions

The function shall perform the following consequent actions:

aSSF ← AI_TSF or dPLM or dLFD or dUPM or dEXM or dCSF-LOS

aSSFrdi ← dCSF-RDI and CSFrdifdiEnable

aSSFfdi ← dCSF-FDI and CSFrdifdiEnable

NOTE 1 – XAR = 0 results in AI_TSF being asserted, so there is no need to include it as an additional contributor to aSSF.

Defect correlations

The function shall perform the following defect correlations to determine the most probable fault cause (see clause 6.4 of [ITU-T G.806]). This fault cause shall be reported to the EMF.

cPLM ← dPLM and (not AI_TSF)

cLFD ← dLFD and (not dPLM) and (not AI_TSF)

cUPM ← dUPM and (not dEXM) and (not dPLM) and (not dLFD) and (not AI_TSF)

cEXM ← dEXM and (not dPLM) and (not dLFD) and (not AI_TSF)

cCSF ← (dCSF-LOS or dCSF-RDI or dCSF-FDI) and (not dEXM) and (not dUPM) and (not dPLM) and (not dLFD) and (not AI_TSF) and CSF_Reported

Performance monitoring

The function shall perform the following performance monitoring primitives processing. The performance monitoring primitives shall be reported to the EMF.

pFCSError: count of FrameCheckSequenceErrors per second.

NOTE 2 – This primitive is calculated by the MAC FCS check process.

11.1.3 VC-m to ETH adaptation functions (Sm/ETH_A; m = 11, 11-Xv, 12, 12-Xv, 2)

11.1.3.1 VC-m to ETH adaptation source function (Sm/ETH_A_So)

This function maps ETH_CI information onto a VC-m server signal (m = 11, 11-X, 12, 12-X, 2) and sources the Sm_AP signal.

Data at the Sm_AP is a VC-m (m = 11, 11-X, 12, 12-X, 2), it has a payload as described in [ITU-T G.707], but with indeterminate POH bytes: J2, V5[1-4], V5[8].

Symbol

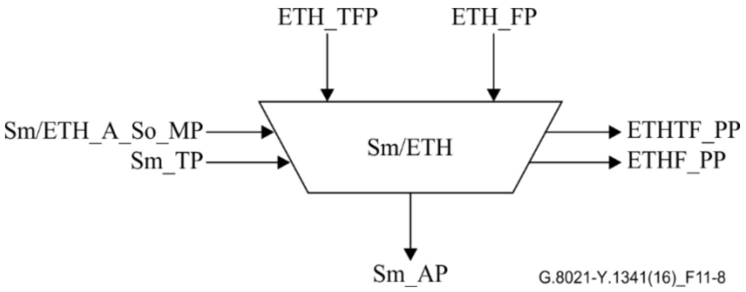


Figure 11-8 – Sm/ETH_A_So symbol

Interfaces

Table 11-5 – Sm/ETH_A_So interfaces

Inputs	Outputs
ETH_TFP: ETH_CI_D ETH_CI_P ETH_CI_DE **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi **Sm_AP**: Sm_AI_XAT **Sm_TP**: Sm_TI_ClocK Sm_TI_FrameStart **Sm/ETH_A_So_MP**: Sm/ETH_A_So_MI_CSFEnable Sm/ETH_A_So_MI_CSFrdifdiEnable	**Sm_AP**: Sm_AI_Data Sm_AI_ClocK Sm_AI_FrameStart **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE

Processes

A process diagram of this function is shown in Figure 11-9.

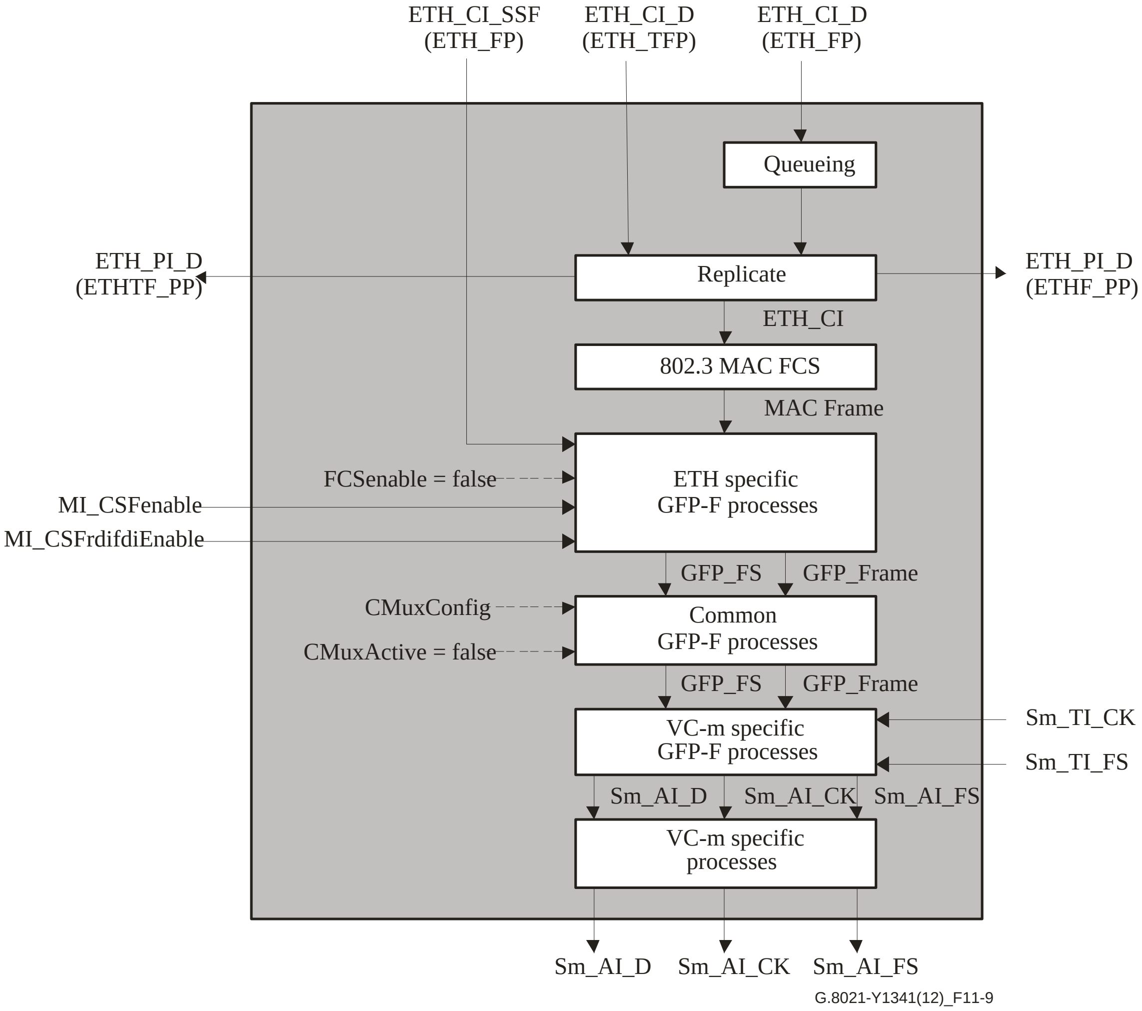


Figure 11-9 – Sm/ETH_A_So process

Queueing process

See clause 8.2.

Replicate process

See clause 8.4.

802.3 MAC FCS generation

See clause 8.8.1.

Ethernet specific GFP-F source process

See clause 8.5.4.1.1 of [ITU-T G.806]. GFP pFCS generation is disabled (FCSenable=false). The UPI value for frame-mapped Ethernet shall be inserted (Table 6-3 of [ITU-T G.7041]). The Ethernet frames are inserted into the client payload information field of the GFP-F frames according to clause 7.1 of [ITU-T G.7041].

Response to ETH_CI_SSF asserted is for further study.

Common GFP source process

See clause 8.5.3.1 of [ITU-T G.806]. GFP channel multiplexing is not supported (CMuxActive=false).

VC-m specific GFP source process

See clause 8.5.2.1 of [ITU-T G.806]. The GFP frames are mapped into the VC-m payload area according to clause 10.6 of [ITU-T G.707].

VC-m specific source process

V5[5-7] and K4[1]: Signal label information is derived directly from the adaptation function type. The value for GFP mapping in Table 9-13 of [ITU-T G.707] is placed in the K4[1] extended signal label field as described in clause 8.2.3.2 of [ITU-T G.783].

K4[2]: For Sm/ETH_A_So with m = 11, 12, 2, the K4[2] bit is sourced as all-zeros.

NOTE 1 – For Sm/ETH_A_So with m = 11-X, 12-X, the K4[2] bit is undefined at the Sm-X_AP output of this function (as per clause 13 of [ITU-T G.783]).

NOTE 2 – For Sm/ETH_A_So with m = 11, 11-X, 12, 12-X, 2, the K4[3-8], V5[1-4] and V5[8] bits are undefined at the Sm-X_AP output of this function (as per clause 13 of [ITU-T G.783]).

Defects None.

Consequent actions

aCSF-RDI ← CI_SSFrdi and CSFrdifdiEnable and CSFEnable

aCSF-FDI ← CI_SSFfdi and CSFrdifdiEnable and CSFEnable

aCSF-LOS ← CI_SSF and CSFEnable

Defect correlations None.

Performance monitoring For further study.

11.1.3.2 VC-m to ETH adaptation sink function (Sm/ETH_A_Sk)

This function extracts ETH_CI information from the Sm_AI signal (m = 11, 11-X, 12, 12-X, 2), delivering ETH_CI to ETH_TFP and ETH_FP.

Data at the Sm_AP is as described in [ITU-T G.707].

Symbol

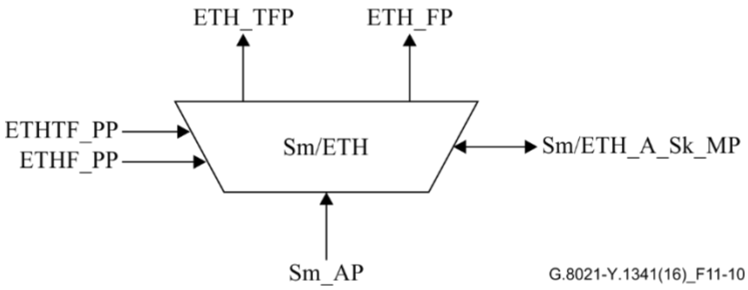


Figure 11-10 – Sm/ETH_A_Sk symbol

Interfaces

Table 11-6 – Sm/ETH_A_Sk interfaces

Inputs	Outputs
Sm_AP: Sm_AI_Data Sm_AI_ClocK Sm_AI_FrameStart Sm_AI_TSF **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **Sm/ETH_A_Sk_MP**: Sm/ETH_A_Sk_MI_FilterConfig Sm/ETH_A_Sk_MI_CSF_Reported Sm/ETH_A_Sk_MI_MAC_Length Sm/ETH_A_Sk_MI_CSFrdifdiEnable	**ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi **Sm/ETH_A_Sk_MP**: Sm/ETH_A_Sk_MI_AcSL Sm/ETH_A_Sk_MI_AcEXI Sm/ETH_A_Sk_MI_AcUPI Sm/ETH_A_Sk_MI_cPLM Sm/ETH_A_Sk_MI_cLFD Sm/ETH_A_Sk_MI_cUPM Sm/ETH_A_Sk_MI_cEXM Sm/ETH_A_Sk_MI_cCSF Sm/ETH_A_Sk_MI_pFCSError

Processes

A process diagram of this function is shown in Figure 11-11.

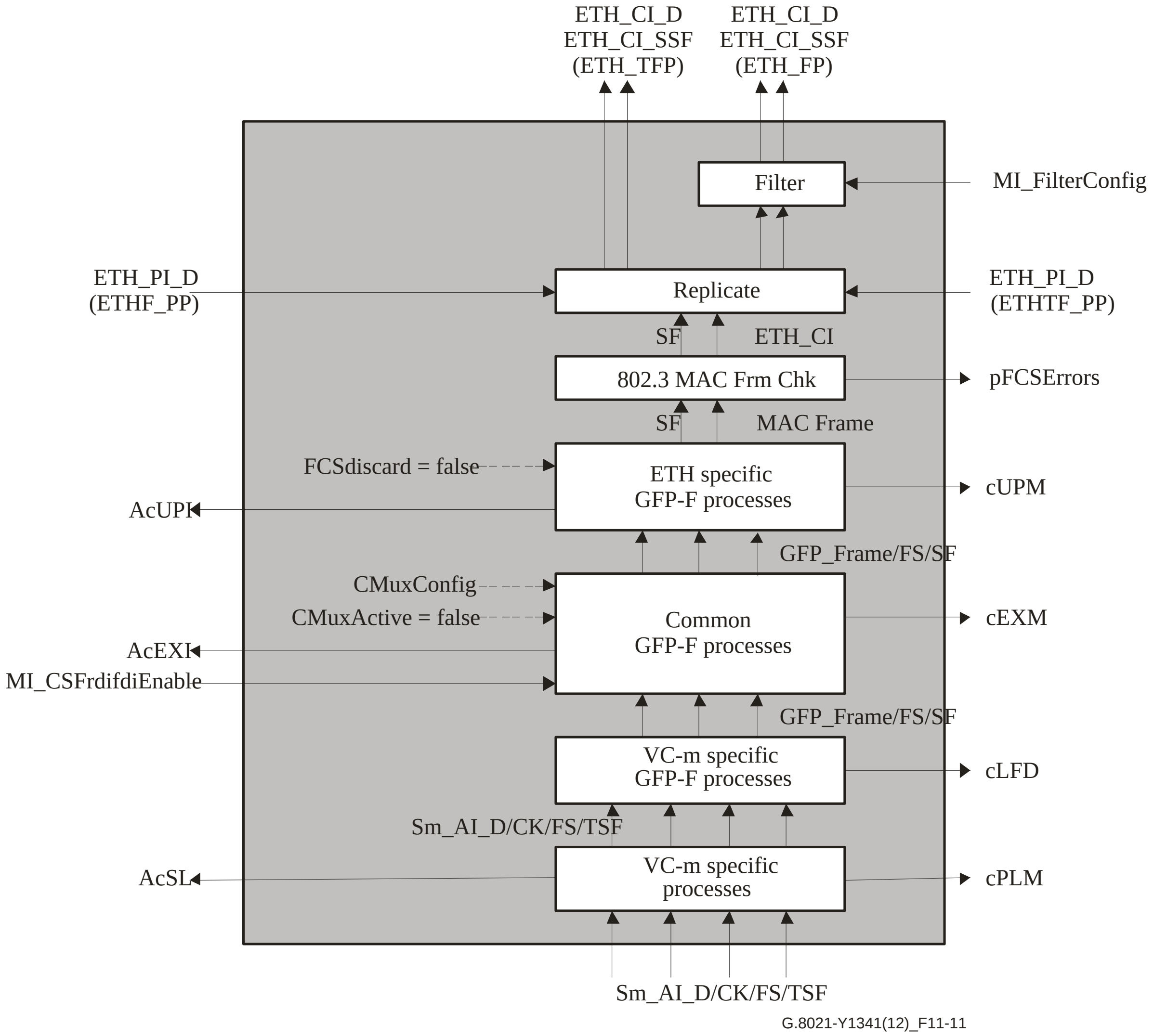


Figure 11-11 – Sm/ETH_A_Sk process

Filter process

See clause 8.3.

Replicate process

See clause 8.4.

802.3 MAC FCS check process

See clause 8.8.2.

Ethernet specific GFP-F sink process

See clause 8.5.4.1.2 of [ITU-T G.806]. GFP pFCS checking, GFP p_FCSError, p_FDis are not supported (FCSdiscard=false). The UPI value for frame-mapped Ethernet shall be expected (Table 6-3 of [ITU-T G.7041]). The Ethernet frames are extracted from the client payload information field of the GFP-F frames according to clause 7.1 of [ITU-T G.7041].

Common GFP sink process

See clause 8.5.3.2 of [ITU-T G.806]. GFP channel multiplexing is not supported (CMuxActive=false).

VC-m specific GFP sink process

See clause 8.5.2.2 of [ITU-T G.806]. The GFP frames are demapped from the VC-m payload area according to clause 10.6 of [ITU-T G.707].

VC-m specific sink process

V5[5-7] and K4[1]: The signal label is recovered from the extended signal label position as described in clause 8.2.3.2 of [ITU-T G.783] and clause 6.2.4.2 of [ITU-T G.806]. The signal label for GFP mapping in Table 9-13 of [ITU-T G.707] shall be expected. The accepted value of the signal label is also available at the Sm/ETH_A_Sk_MP.

Defects

dPLM – See clause 6.2.4.2 of [ITU-T G.806].

dLFD – See clause 6.2.5.2 of [ITU-T G.806].

dUPM – See clause 6.2.4.3 of [ITU-T G.806].

dEXM – See clause 6.2.4.4 of [ITU-T G.806].

dCSF-LOS – See clause 8.8.6.2.

dCSF-RDI – See clause 8.8.6.2.

dCSF-FDI – See clause 8.8.6.2.

Consequent actions

The function shall perform the following consequent actions:

aSSF ← AI_TSF or dPLM or dLFD or dUPM or dEXM or dCSF-LOS

aSSFrdi ← dCSF-RDI and CSFrdifdiEnable

aSSFfdi ← dCSF-FDI and CSFrdifdiEnable

Defect correlations

The function shall perform the following defect correlations to determine the most probable fault cause (see clause 6.4 of [ITU-T G.806]). This fault cause shall be reported to the EMF.

cPLM ← dPLM and (not AI_TSF)

cLFD ← dLFD and (not dPLM) and (not AI_TSF)

cUPM ← dUPM and (not dEXM) and (not dPLM) and (not dLFD) and (not AI_TSF)

cEXM ← dEXM and (not dPLM) and (not dLFD) and (not AI_TSF)

cCSF ← (dCSF-LOS or dCSF-RDI or dCSF-FDI) and (not dEXM) and (not dUPM) and (not dPLM) and (not dLFD) and (not AI_TSF) and CSF_Reported

Performance monitoring

The function shall perform the following performance monitoring primitives processing. The performance monitoring primitives shall be reported to the EMF.

pFCSError: count of FrameCheckSequenceErrors per second.

NOTE – This primitive is calculated by the MAC FCS check process.

11.1.4 LCAS-capable VC-m-Xv to ETH adaptation functions (Sm-X-L/ETH_A; m = 11, 12)

11.1.4.1 LCAS-capable VC-m-Xv to ETH adaptation source function (Sm-X-L/ETH_A_So)

This function maps ETH_CI information onto an Sm-X-L_AI signal (m = 11 or 12).

Data at the Sm-X-L_AP is a VC-m-X (m = 11 or 12), it has a payload as described in [ITU-T G.707], but with indeterminate POH bytes: J2, V5[1-4], V5[8].

Symbol

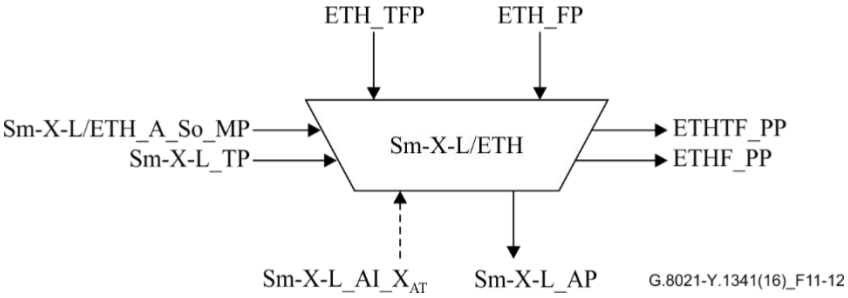


Figure 11-12 – Sm-X-L/ETH_A_So symbol

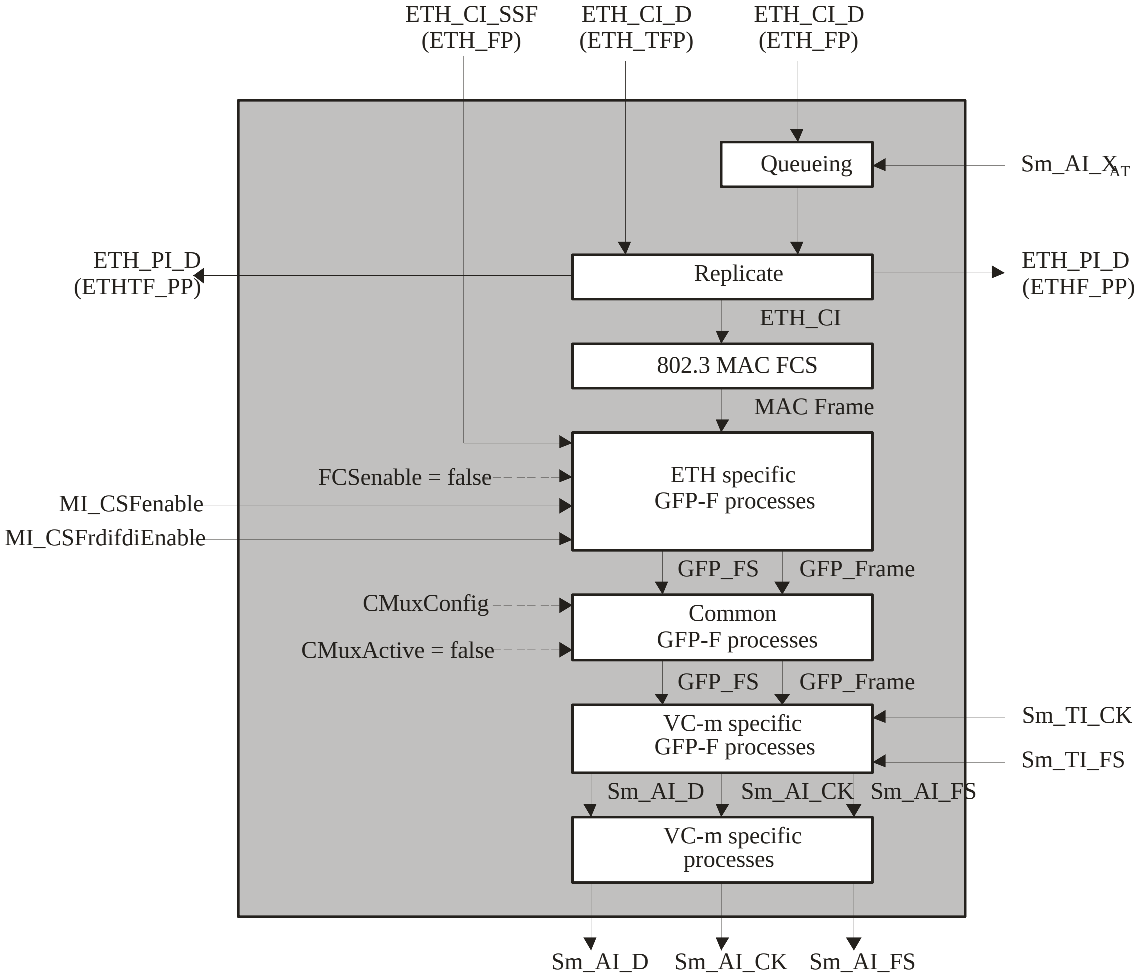
Interfaces

Table 11-7 – Sm-X-L/ETH_A_So interfaces

Inputs	Outputs
ETH_TFP: ETH_CI_D ETH_CI_P ETH_CI_DE **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi **Sm-X-L_AP**: Sm-X-L_AI_XAT **Sm _TP**: Sm_TI_ClocK Sm_TI_FrameStart **Sm-X-L/ETH_A_So_MP**: Sm-X-L/ETH_A_So_MI_CSFEnable Sm-X-L/ETH_A_So_MI_CSFrdifdiEnable	**Sm-X-L_AP**: Sm-X-L_AI_Data Sm-X-L_AI_ClocK Sm-X-L_AI_FrameStart **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE

Processes

A process diagram of this function is shown in Figure 11-13.



G.8021-Y1341(12)_F11-13

Figure 11-13 – Sm-X-L/ETH_A_So process

See clause 11.1.3.1 for a description of Sm-X-L/ETH_A processes.

Defects None.

Consequent actions

aCSF-RDI ← CI_SSFrdi and CSFrdifdiEnable and CSFEnable

aCSF-FDI ← CI_SSFfdi and CSFrdifdiEnable and CSFEnable

aCSF-LOS ← CI_SSF and CSFEnable

Defect correlations None.

Performance monitoring For further study.

11.1.4.2 LCAS-capable VC-m-Xv to ETH adaptation sink function (Sm-X-L/ETH_A_Sk)

This function extracts ETH_CI information from the Sm-X-L_AI signal (m = 11 or 12), delivering ETH_CI to ETH_TFP and ETH_FP.

Data at the Sm_AP is as described in [ITU-T G.707].

Symbol

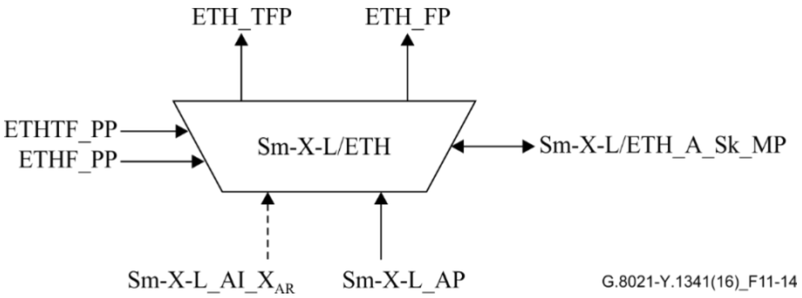


Figure 11-14 – Sm-X-L/ETH_A_Sk symbol

Interfaces

Table 11-8 – Sm-X-L/ETH_A_Sk interfaces

Inputs	Outputs
Sm-X-L_AP: Sm-X-L_AI_Data Sm-X-L_AI_ClocK Sm-X-L_AI_FrameStart Sm-X-L_AI_TSF Sm-X-L_AI_XAR **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **Sm-X-L/ETH_A_Sk_MP**: Sm-X-L/ETH_A_Sk_MI_FilterConfig Sm-X-L/ETH_A_Sk_MI_CSF_Reported Sm-X-L/ETH_A_Sk_MI_MAC_Length Sm-X-L/ETH_A_Sk_MI_CSFrdifdiEnable	**ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi **Sm-X-L/ETH_A_Sk_MP**: Sm-X-L/ETH_A_Sk_MI_AcSL Sm-X-L/ETH_A_Sk_MI_AcEXI Sm-X-L/ETH_A_Sk_MI_AcUPI Sm-X-L/ETH_A_Sk_MI_cPLM Sm-X-L/ETH_A_Sk_MI_cLFD Sm-X-L/ETH_A_Sk_MI_cUPM Sm-X-L/ETH_A_Sk_MI_cEXM Sm-X-L/ETH_A_Sk_MI_cCSF Sm-X-L/ETH_A_Sk_MI_pFCSError

Processes

See the process diagram and process description in clause 11.1.1.2. The additional Sm-X-L_AI_X$_{AR}$ interface is not connected to any of the internal processes.

Defects

dPLM – See clause 6.2.4.2 of [ITU-T G.806].

dLFD – See clause 6.2.5.2 of [ITU-T G.806].

dUPM – See clause 6.2.4.3 of [ITU-T G.806].

dEXM – See clause 6.2.4.4 of [ITU-T G.806].

dCSF-LOS – See clause 8.8.6.2.

dCSF-RDI – See clause 8.8.6.2.

dCSF-FDI – See clause 8.8.6.2.

Consequent actions

The function shall perform the following consequent actions:

aSSF ← AI_TSF or dPLM or dLFD or dUPM or dEXM or dCSF-LOS

aSSFrdi ← dCSF-RDI and CSFrdifdiEnable

aSSFfdi ← dCSF-FDI and CSFrdifdiEnable

NOTE 1 – X_{AR} = 0 results in AI_TSF being asserted, so there is no need to include it as an additional contributor to aSSF.

Defect correlations

The function shall perform the following defect correlations to determine the most probable fault cause (see clause 6.4 of [ITU-T G.806]). This fault cause shall be reported to the EMF.

cPLM ← dPLM and (not AI_TSF)

cLFD ← dLFD and (not dPLM) and (not AI_TSF)

cUPM ← dUPM and (not dEXM) and (not dPLM) and (not dLFD) and (not AI_TSF)

cEXM ← dEXM and (not dPLM) and (not dLFD) and (not AI_TSF)

cCSF ← (dCSF-LOS or dCSF-RDI or dCSF-FDI) and (not dEXM) and (not dUPM) and (not dPLM) and (not dLFD) and (not AI_TSF) and CSF_Reported

Performance monitoring

The function shall perform the following performance monitoring primitives processing. The performance monitoring primitives shall be reported to the EMF.

pFCSError: count of FrameCheckSequenceErrors per second.

NOTE 2 – This primitive is calculated by the MAC FCS process.

11.2 SDH to ETC adaptation functions (Sn-X/ETC3_A)

NOTE – The abbreviation ETC3 is superseded and replaced by ETC1000X in [ITU-T G.8023]

11.2.1 VC-n-X to ETC3 adaptation source function (Sn-X/ETC3_A_So)

This function maps ETC_CI information from an ETC3 onto an Sn-X_AI signal (n=3, 4). This mapping is currently only defined for X=7 for VC-4 and X=22 for VC-3.

Data at the Sn-X_AP is a VC-n-Xv, it has a payload as described in [ITU-T G.707], but with indeterminate POH bytes: J1, B3, G1.

Symbol

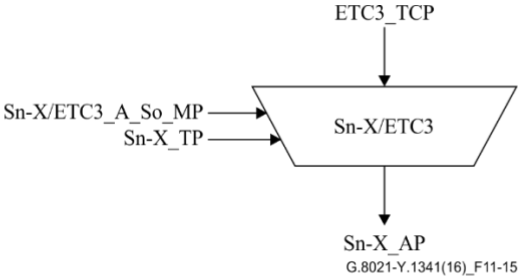


Figure 11-15 – Sn-X/ETC3_A_So symbol

Interfaces

Table 11-9 – Sn-X/ETC3_A_So interfaces

Inputs	Outputs
ETC3_TCP: ETC3_CI_Data_Control ETC3_CI_ClocK ETC3_CI_Control_Ind ETC3_CI_SSF **Sn-X_TP**: Sn-X _TI_ClocK Sn-X _TI_FrameStart **Sn-X/ETC3_A_So_MP**: Sn-X/ETC3_A_So_MI_CSFEnable	**Sn-X_AP**: Sn-X_AI_Data Sn-X_AI_ClocK Sn-X_AI_FrameStart

Processes

A process diagram of this function is shown in Figure 11-16.

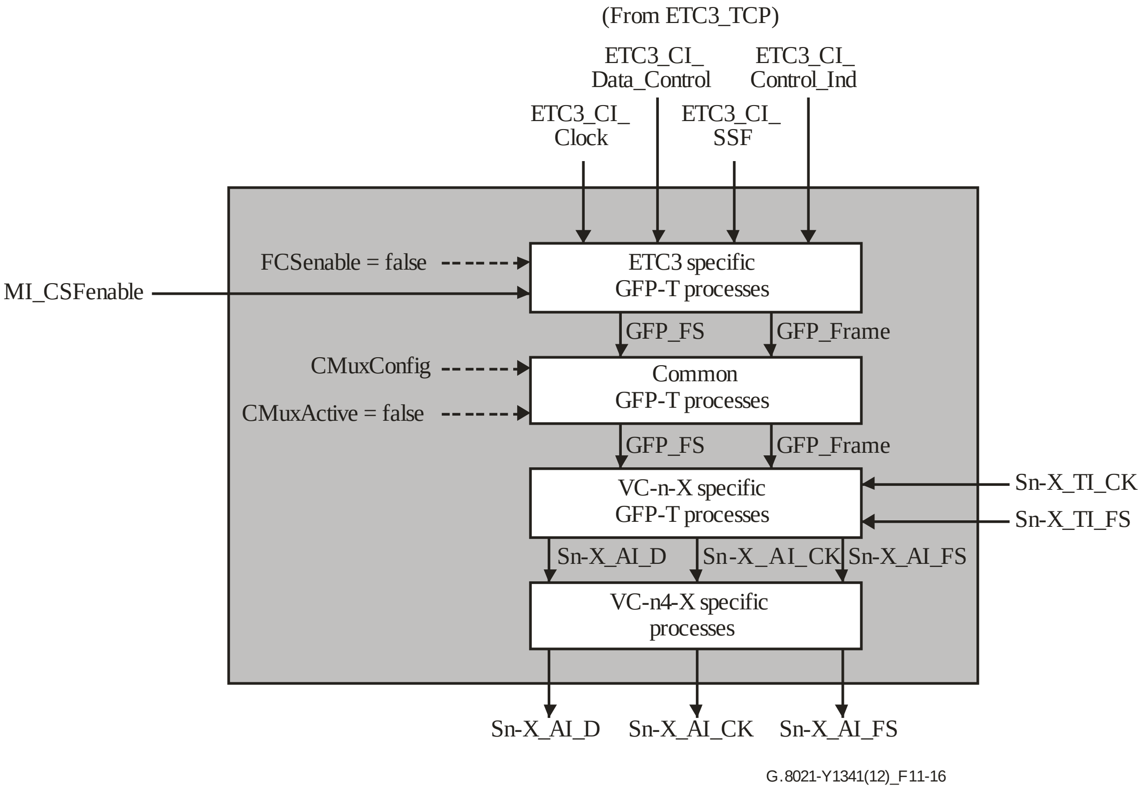


Figure 11-16 – Sn-X/ETC3_A_So process

Ethernet specific GFP-T source process

See clause 8.5.4.2.1 of [ITU-T G.806]. GFP pFCS generation is disabled (FCSenable=false). The UPI value for transparent Gb Ethernet shall be inserted (Table 6-3 of [ITU-T G.7041]). The Ethernet codeword information is inserted into the client payload information field of the GFP-T frames according to clause 8 of [ITU-T G.7041]. 65B rate adaptation is enabled (RAdisable=false).

NOTE 1 – Equipment designed prior to this amendment may not support configuration of RAdisable; in such equipment the use of 65B rate adaptation is implicitly enabled.

Response to ETC3_CI_SSF is according to the principles in clauses 8.3 and 8.3.4 of [ITU-T G.7041] and Appendix VIII of [ITU-T G.806]. Details are for further study.

Common GFP source process

See clause 8.5.3.1 of [ITU-T G.806]. GFP channel multiplexing is not supported (CMuxActive=false).

VC-n-X specific GFP source process

See clause 8.5.2.1 of [ITU-T G.806]. The GFP frames are mapped into the VC-n-X (n=3,4) payload area according to clause 10.6 of [ITU-T G.707].

VC-n-X specific source process

C2: Signal label information is derived directly from the adaptation function type. The value for GFP mapping in Table 9-11 of [ITU-T G.707] is placed in the C2 byte position.

NOTE 2 – For Sn-X/ETC3_A_So, the H4, K3, F2, and F3 bytes are undefined at the Sn-X_AP output of this function (as per clause 12 of [ITU-T G.783]).

Defects None.

Consequent actions

aCSF-RDI ← CI_SSFrdi and CSFrdifdiEnable and CSFEnable

aCSF-FDI ← CI_SSFfdi and CSFrdifdiEnable and CSFEnable

aCSF-LOS ← CI_SSF and CSFEnable

Defect correlations None.

Performance monitoring For further study.

11.2.2 VC-n-X to ETC3 adaptation sink function (Sn-X/ETC3_A_Sk)

This function extracts ETC3_CI information from the Sn-X_AI signal (n=3, 4), delivering ETC3_CI to the ETC3_TCP.

Data at the Sn-X_AP is as described in [ITU-T G.707]. This mapping is currently only defined for X=7 for VC-4 and X=22 for VC-3.

Symbol

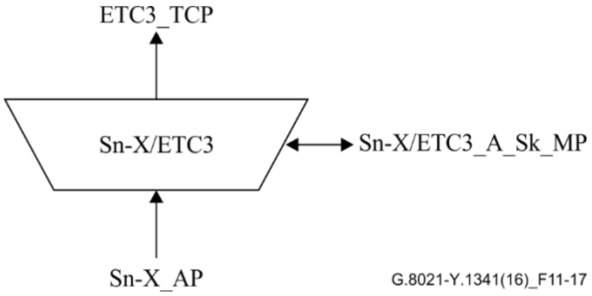


Figure 11-17 – Sn-X/ETC3_A_Sk symbol

Interfaces

Table 11-10 – Sn-X/ETC3_A_Sk interfaces

Inputs	Outputs
Sn-X_AP: Sn-X_AI_Data Sn-X_AI_ClocK Sn-X_AI_FrameStart Sn-X_AI_TSF **Sn-X/ETC3_A_Sk_MP**: Sn-X/ETC3_A_Sk_MI_CSF_Reported	**ETC3_TCP**: ETC3_CI_Data_Control ETC3_CI_ClocK ETC3_CI_Control_Ind ETC3_CI_SSF **Sn-X / ETC3_A_Sk_MP**: Sn-X / ETC3_A_Sk_MI_AcSL Sn-X / ETC3_A_Sk_MI_AcEXI Sn-X / ETC3_A_Sk_MI_AcPFI Sn-X / ETC3_A_Sk_MI_AcUPI Sn-X / ETC3_A_Sk_MI_cPLM Sn-X / ETC3_A_Sk_MI_cLFD Sn-X / ETC3_A_Sk_MI_cUPM Sn-X / ETC3_A_Sk_MI_cEXM Sn-X / ETC3_A_Sk_MI_cCSF Sn-X / ETC3_A_Sk_MI_pCRC16Errors

Processes

A process diagram of this function is shown in Figure 11-18.

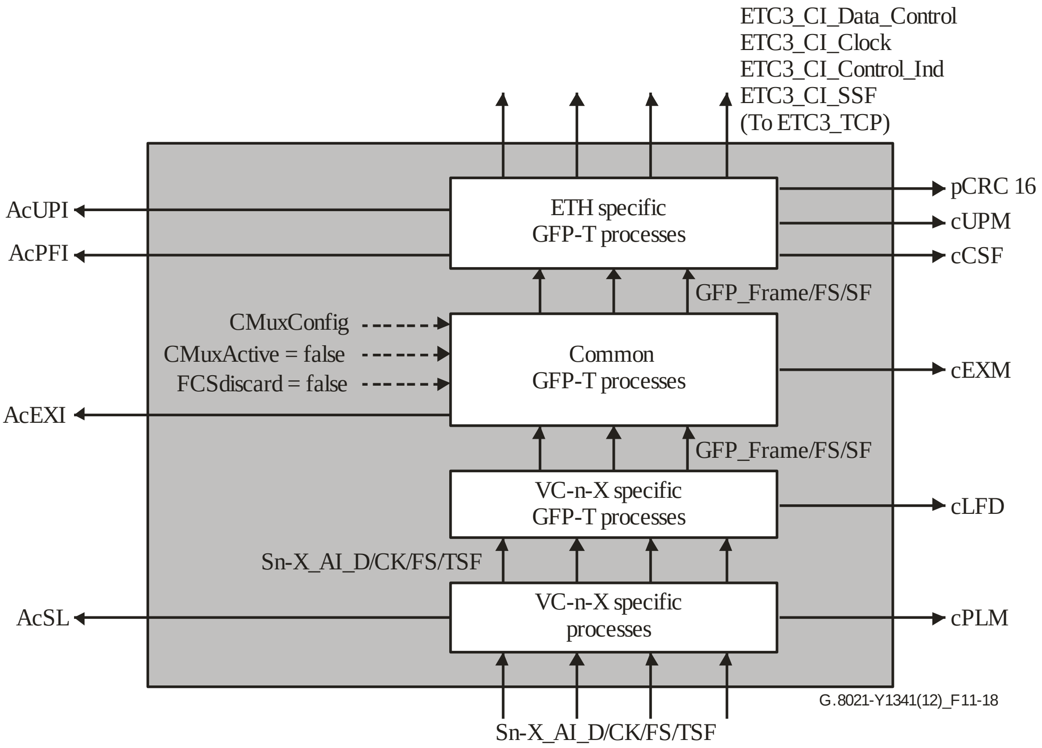


Figure 11-18 – Sn-X/ETC3_A_Sk process

Ethernet specific GFP-T sink process

See clause 8.5.4.2.2 of [ITU-T G.806]. GFP pFCS checking and GFP p_FCSError are not supported (FCSdiscard=false). The UPI value for transparent Gb Ethernet shall be expected (Table 6-3 of [ITU-T G.7041]). Frames discarded due to incorrect PFI or UPI values shall be counted in _pFDis. Errors detected in a received superblock are reported as a _pCRC16Error. If ECenable=true, then single transmission channel errors in the superblock shall be corrected using the superblock CRC-16. The Ethernet codeword information is extracted from the client payload information field of the GFP-F frames according to clause 8 of [ITU-T G.7041].

Common GFP sink process

See clause 8.5.3.2 of [ITU-T G.806]. GFP channel multiplexing is not supported (MI_CMuxActive=false). Frames discarded due to EXI mismatch or errors detected by the tHEC shall be counted in _pFDis.

VC-n-X specific GFP sink process

See clause 8.5.2.2 of [ITU-T G.806]. The GFP frames are demapped from the VC-n-X payload area according to clause 10.6 of [ITU-T G.707].

VC-n-X specific sink process

C2: The signal label is recovered from the C2 byte as per clause 6.2.4.2 of [ITU-T G.806]. The signal label for GFP mapping in Table 9-11 of [ITU-T G.707] shall be expected. The accepted value of the signal label is also available at the Sn-X/ETC3_A_Sk_MP.

Defects

dPLM – See clause 6.2.4.2 of [ITU-T G.806].

dLFD – See clause 6.2.5.2 of [ITU-T G.806].

dUPM – See clause 6.2.4.3 of [ITU-T G.806].

dEXM – See clause 6.2.4.4 of [ITU-T G.806].

dCSF – See clause 6.2.6.4 of [ITU-T G.806].

Consequent actions

The function shall perform the following consequent actions:

aSSF ← AI_TSF or dPLM or dLFD or dUPM or dEXM or dCSF

Defect correlations

The function shall perform the following defect correlations to determine the most probable fault cause (see clause 6.4 of [ITU-T G.806]). This fault cause shall be reported to the EMF.

cPLM ← dPLM and (not AI_TSF)

cLFD ← dLFD and (not dPLM) and (not AI_TSF)

cUPM ← dUPM and (not dEXM) and (not dPLM) and (not dLFD) and (not AI_TSF)

cEXM ← dEXM and (not dPLM) and (not dLFD) and (not AI_TSF)

cCSF as per clause 8.5.4.2.2 of [ITU-T G.806].

Performance monitoring

The function shall perform the following performance monitoring primitives processing. The performance monitoring primitives shall be reported to the EMF.

pCRC16Errors: count of superblock CRC-16 errors per second

_pFDis = sum (n_FDis_tHEC + n_FDis_eHEC_EXI + n_FDis_PTI_UPI)

11.3 S4-64c to ETH-w adaptation functions

This covers 64B/66B-encoded mapping of Ethernet frames into VC-4-64c.

For further study.

11.4 PDH to ETH adaptation functions (P/ETH_A)

11.4.1 Pq to ETH adaptation functions (Pq/ETH_A; q = 11s, 12s, 31s, 32e)

11.4.1.1 Pq to ETH adaptation source function (Pq/ETH_A_So)

This function maps ETH_CI information onto a Pq_AI signal (q = 11s, 12s, 31s, 32e).

Data at the Pq_AP is a Pq (q = 11s, 12s, 31s, 32e), it has a payload as described in [ITU-T G.7043] with a value of N=1. The VLI byte is reserved and not used for payload data.

Symbol

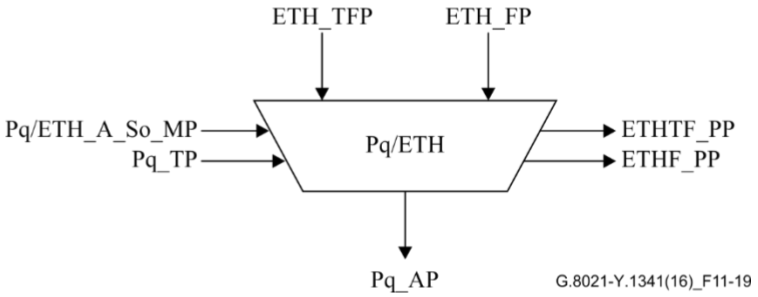


Figure 11-19 – Pq/ETH_A_So symbol

Interfaces

Table 11-11 – Pq/ETH_A_So interfaces

Inputs	Outputs
ETH_TFP: ETH_CI_D ETH_CI_P ETH_CI_DE **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi **Pq_TP**: Pq_TI_ClocK Pq_TI_FrameStart **Pq/ETH_A_So_MP**: Pq/ETH_A_So_MI_CSFEnable Pq/ETH_A_So_MI_CSFrdifdiEnable	**Pq_AP**: Pq_AI_Data Pq_AI_ClocK Pq_AI_FrameStart **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE

Processes

A process diagram of this function is shown in Figure 11-20.

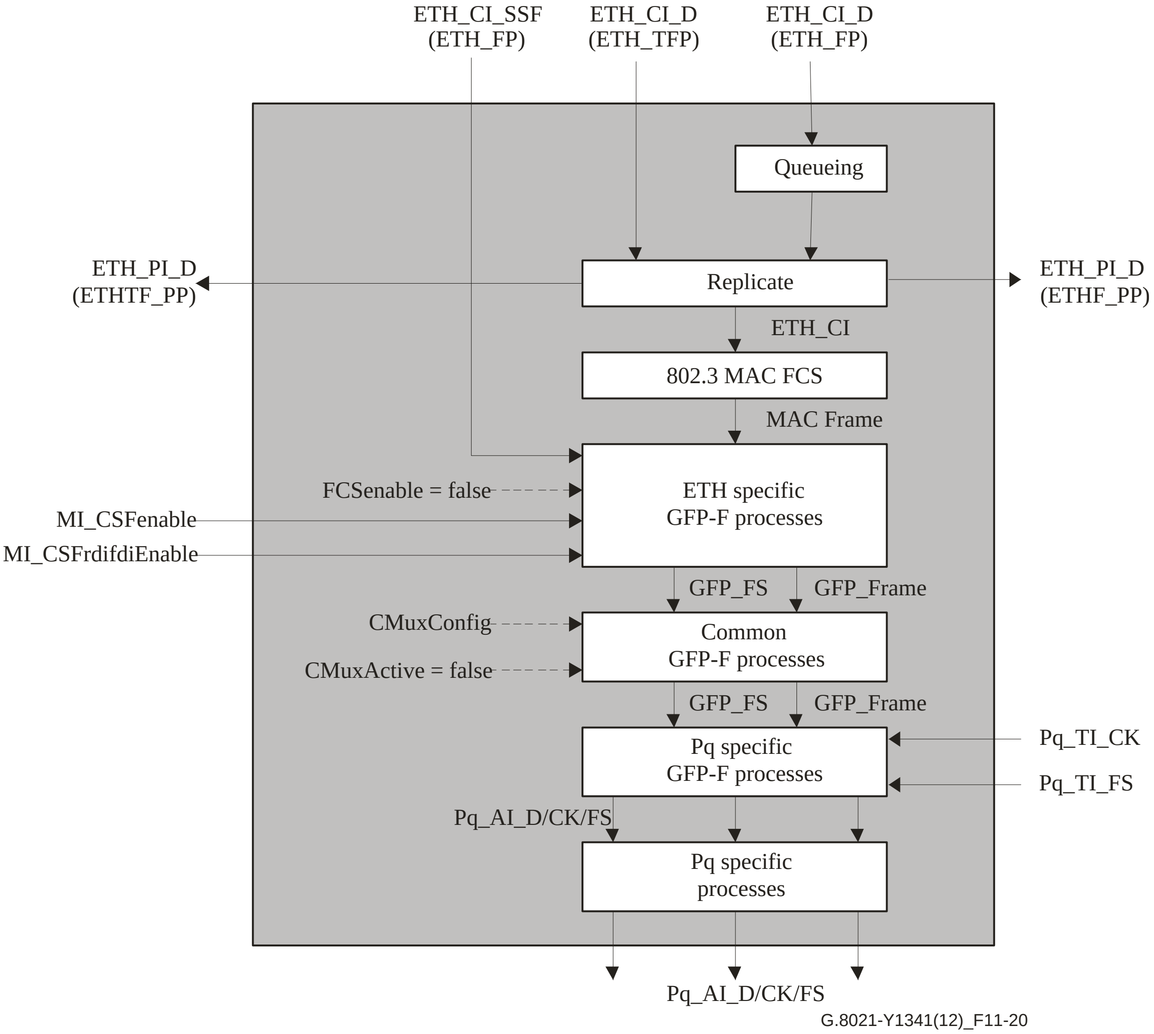


Figure 11-20 – Pq/ETH_A_So process

Queueing process

See clause 8.2.

Replicate process

See clause 8.4.

802.3 MAC FCS generation

See clause 8.8.1.

Ethernet specific GFP-F source process

See clause 8.5.4.1.1 of [ITU-T G.806]. GFP pFCS generation is disabled (FCSenable=false). The UPI value for frame-mapped Ethernet shall be inserted (Table 6-3 of [ITU-T G.7041]). The Ethernet frames are inserted into the client payload information field of the GFP-F frames according to clause 7.1 of [ITU-T G.7041].

Response to ETH_CI_SSF asserted is for further study.

Common GFP source process

See clause 8.5.3.1 of [ITU-T G.806]. GFP channel multiplexing is not supported (CMuxActive=false).

Pq specific GFP source process

See clause 8.5.2.1 of [ITU-T G.806]. The GFP frames are mapped into the Pq payload area according to [ITU-T G.8040].

Pq specific source process

NOTE – The VLI byte is fixed stuff equal to 0x00 at the Pq_AP output of this function.

P31s specific

MA: Signal label information is derived directly from the adaptation function type. The value for GFP mapping in clause 2.1 of [ITU-T G.832] is placed in the payload type field of the MA byte.

Defects None.

Consequent actions

aCSF-RDI ← CI_SSFrdi and CSFrdifdiEnable and CSFEnable

aCSF-FDI ← CI_SSFfdi and CSFrdifdiEnable and CSFEnable

aCSF-LOS ← CI_SSF and CSFEnable

Defect correlations None.

Performance monitoring For further study.

11.4.1.2 Pq to ETH adaptation sink function (Pq/ETH_A_Sk)

This function extracts ETH_CI information from a Pq_AI signal (q = 11s, 12s, 31s, 32e), delivering ETH_CI to ETH_TFP and ETH_FP.

Data at the Pq_AP is a Pq (q = 11s, 12s, 31s, 32e), it has a payload as described in [ITU-T G.7043] with a value of N=1. The VLI byte is reserved and not used for payload data.

Symbol

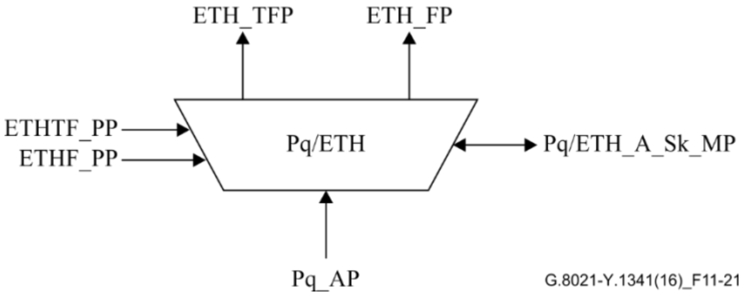


Figure 11-21 – Pq/ETH_A_Sk symbol

Interfaces

Table 11-12 – Pq/ETH_A_Sk interfaces

Inputs	Outputs
Pq_AP: Pq_AI_Data Pq_AI_ClocK Pq_AI_FrameStart Pq_AI_TSF **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **Pq/ETH_A_Sk_MP**: Pq/ETH_A_Sk_MI_FilterConfig Pq/ETH_A_Sk_MI_CSF_Reported Pq/ETH_A_Sk_MI_MAC_Length Pq/ETH_A_Sk_MI_CSFrdifdiEnable	**ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi **Pq/ETH_A_Sk_MP**: Pq/ETH_A_Sk_MI_AcSL Pq/ETH_A_Sk_MI_AcEXI Pq/ETH_A_Sk_MI_AcUPI Pq/ETH_A_Sk_MI_cPLM Pq/ETH_A_Sk_MI_cLFD Pq/ETH_A_Sk_MI_cUPM Pq/ETH_A_Sk_MI_cEXM Pq/ETH_A_Sk_MI_cCSF Pq/ETH_A_Sk_MI_pFCSError

Processes

A process diagram of this function is shown in Figure 11-22.

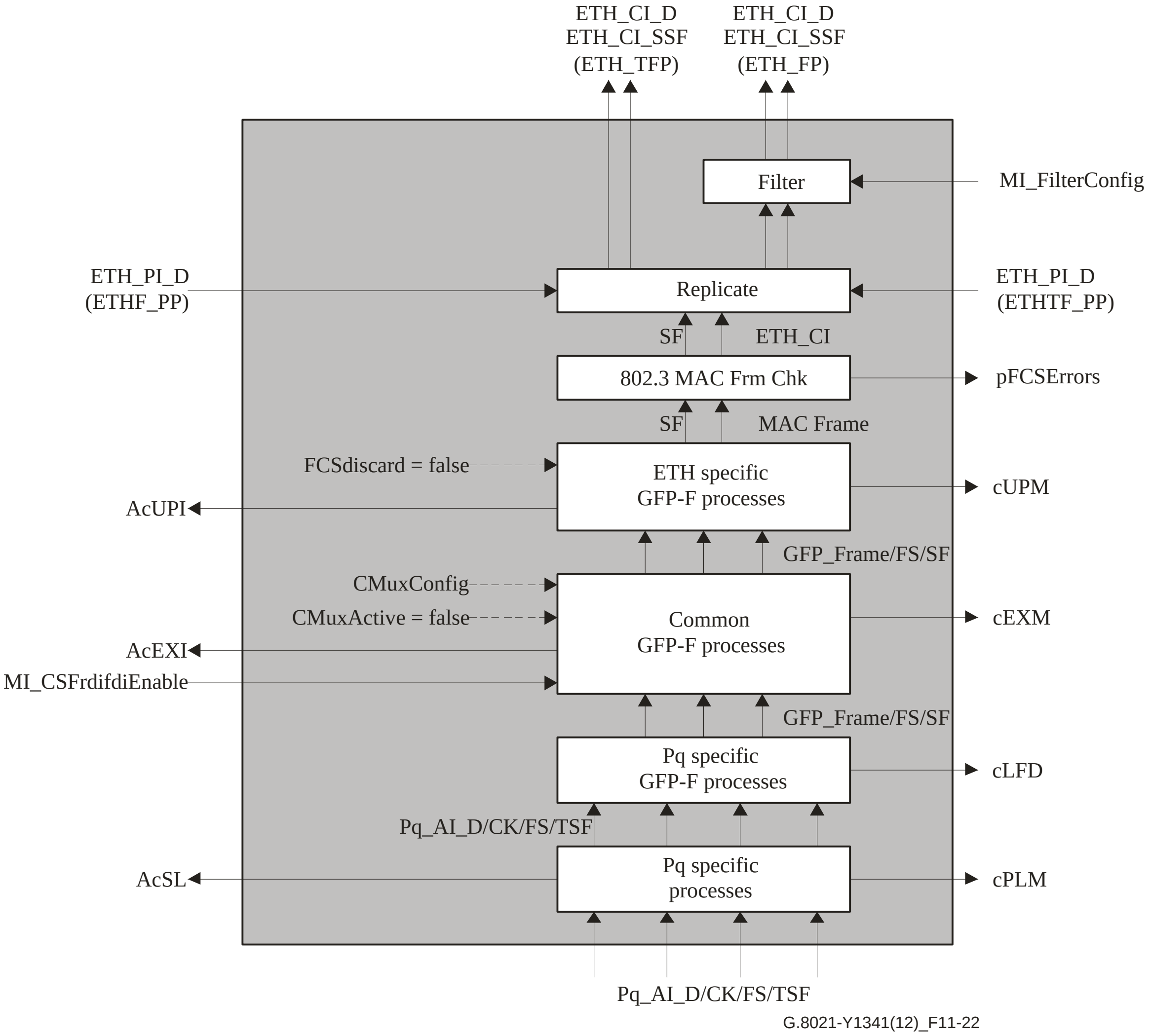


Figure 11-22 – Pq/ETH_A_Sk process

Filter process

See clause 8.3.

Replicate process

See clause 8.4.

802.3 MAC FCS check process

See clause 8.8.2.

Ethernet specific GFP-F sink process

See clause 8.5.4.1.2 of [ITU-T G.806]. GFP pFCS checking, GFP p_FCSError, p_FDis are not supported (FCSdiscard=false). The UPI value for frame-mapped Ethernet shall be expected (Table 6-3 of [ITU-T G.7041]). The Ethernet frames are extracted from the client payload information field of the GFP-F frames according to clause 7.1 of [ITU-T G.7041].

Common GFP sink process

See clause 8.5.3.2 of [ITU-T G.806]. GFP channel multiplexing is not supported (MI_CMuxActive=false).

Pq specific GFP sink process

See clause 8.5.2.2 of [ITU-T G.806]. The GFP frames are demapped from the Pq payload area according to [ITU-T G.8040].

Pq specific sink process

NOTE 1 – The VLI byte at the Pq_AP input of this function is ignored.

P31s specific

MA: The signal label is recovered from the payload type field in the MA byte as per clause 6.2.4.2 of [ITU-T G.806]. The signal label for GFP mapping in clause 2.1 of [ITU-T G.832] shall be expected. The accepted value of the signal label is also available at the P31s/ETH_A_Sk_MP.

Defects

dPLM – See clause 6.2.4.2 of [ITU-T G.806].

dLFD – See clause 6.2.5.2 of [ITU-T G.806].

dUPM – See clause 6.2.4.3 of [ITU-T G.806].

dEXM – See clause 6.2.4.4 of [ITU-T G.806].

dCSF-LOS – See clause 8.8.6.2.

dCSF-RDI – See clause 8.8.6.2.

dCSF-FDI – See clause 8.8.6.2.

NOTE 2 – dPLM is only defined for q = 31s. dPLM is assumed to be false for q = 11s, 12s, 32e.

Consequent actions

The function shall perform the following consequent actions:

aSSF ← AI_TSF or dPLM or dLFD or dUPM or dEXM or dCSF-LOS

aSSFrdi ← dCSF-RDI and CSFrdifdiEnable

aSSFfdi ← dCSF-FDI and CSFrdifdiEnable

Defect correlations

The function shall perform the following defect correlations to determine the most probable fault cause (see clause 6.4 of [ITU-T G.806]). This fault cause shall be reported to the EMF.

cPLM ← dPLM and (not AI_TSF)

cLFD ← dLFD and (not dPLM) and (not AI_TSF)

cUPM ← dUPM and (not dEXM) and (not dPLM) and (not dLFD) and (not AI_TSF);

cEXM ← dEXM and (not dPLM) and (not dLFD) and (not AI_TSF)

cCSF ← (dCSF-LOS or dCSF-RDI or dCSF-FDI) and (not dEXM) and (not dUPM) and (not dPLM) and (not dLFD) and (not AI_TSF) and CSF_Reported

Performance monitoring

The function shall perform the following performance monitoring primitives processing. The performance monitoring primitives shall be reported to the EMF.

pFCSError: count of FrameCheckSequenceErrors per second.

NOTE 3 – This primitive is calculated by the MAC FCS check process.

11.4.2 LCAS-capable Pq-Xv to ETH adaptation functions (Pq-X-L/ETH_A; q = 11s, 12s, 31s, 32e)

11.4.2.1 LCAS-capable Pq-Xv to ETH adaptation source function (Pq-X-L/ETH_A_So)

This function maps ETH_CI information onto a Pq-X-L_AI signal (q = 11s, 12s, 31s, 32e).

Data at the Pq-X-L_AP is a Pq-X-L (q = 11s, 12s, 31s, 32e), it has a payload as described in [ITU-T G.7043].

Symbol

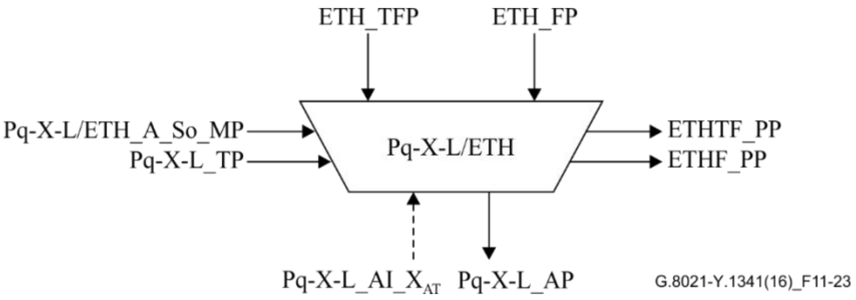


Figure 11-23 – Pq-X-L/ETH_A_So symbol

Interfaces

Table 11-13 – Pq-X-L/ETH_A_So interfaces

Inputs	Outputs
ETH_TFP: ETH_CI_D ETH_CI_P ETH_CI_DE **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi **Pq-X-L_AP**: Pq-X-L_AI_XAT **Pq-X-L_TP**: Pq-X-L_TI_ClocK Pq-X-L_TI_FrameStart **Pq-X-L/ETH_A_So_MP**: Pq-X-L/ETH_A_So_MI_CSFEnable Pq-X-L/ETH_A_So_MI_CSFrdifdiEnable	**Pq-X-L_AP**: Pq-X-L_AI_Data Pq-X-L_AI_ClocK Pq-X-L_AI_FrameStart **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE

Processes

A process diagram of this function is shown in Figure 11-24.

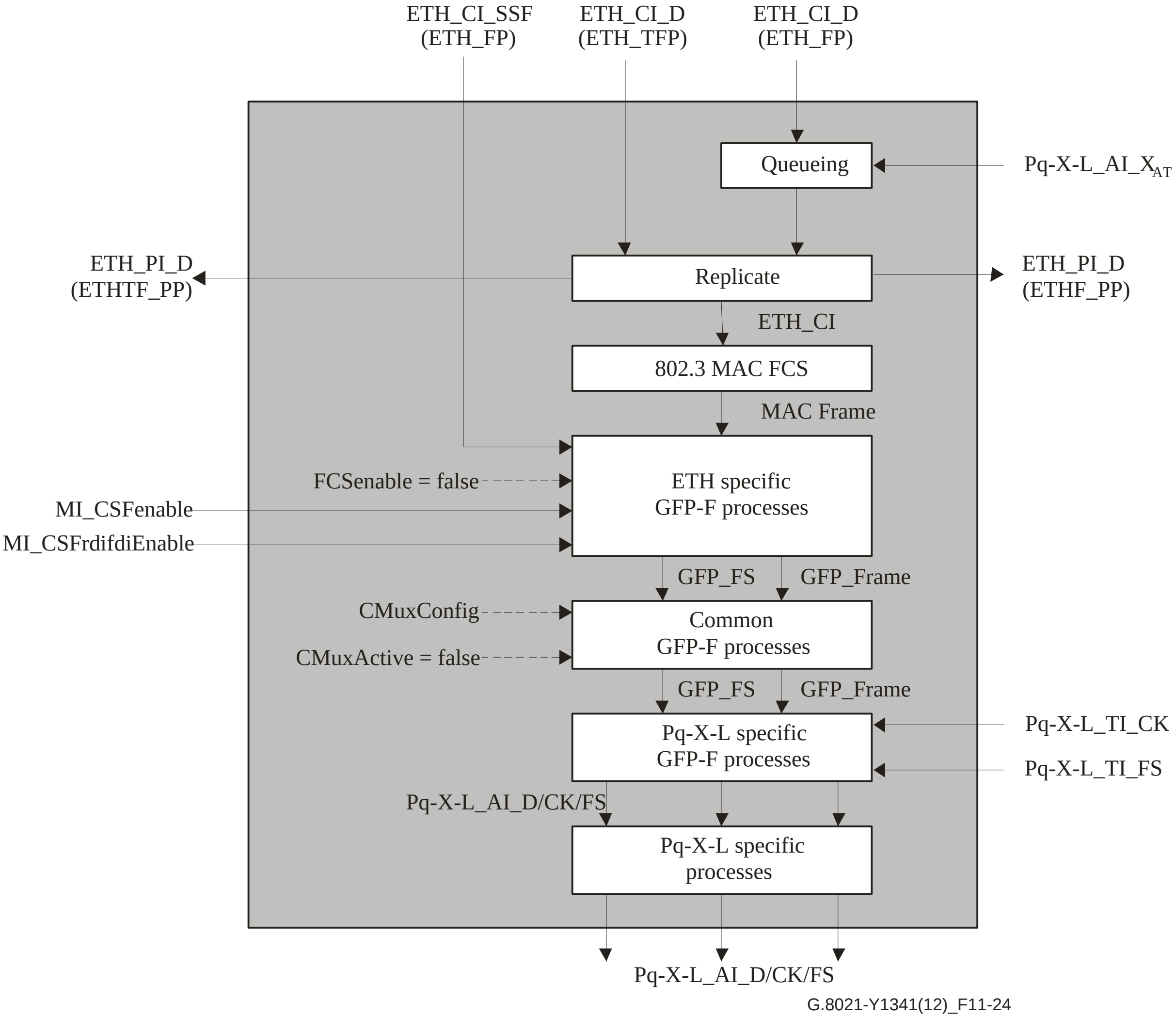


Figure 11-24 – Pq-X-L/ETH_A_So process

Queueing process

See clause 8.2.

Replicate process

See clause 8.4.

802.3 MAC FCS generation

See clause 8.8.1.

Ethernet specific GFP-F source process

See clause 8.5.4.1.1 of [ITU-T G.806]. GFP pFCS generation is disabled (FCSenable=false). The UPI value for frame-mapped Ethernet shall be inserted (Table 6-3 of [ITU-T G.7041]). The Ethernet frames are inserted into the client payload information field of the GFP-F frames according to clause 7.1 of [ITU-T G.7041].

Response to ETH_CI_SSF asserted is for further study.

Common GFP source process

See clause 8.5.3.1 of [ITU-T G.806]. GFP channel multiplexing is not supported (CMuxActive=false).

Pq-X-L specific GFP source process

See clause 8.5.2.1 of [ITU-T G.806]. The GFP frames are mapped into the Pq-X-L payload area according to [ITU-T G.8040].

Pq-X-L specific source process

P31s-X-L specific

MA: Signal label information is derived directly from the adaptation function type. The value for GFP mapping in clause 2.1 of [ITU-T G.832] is placed in the payload type field of the MA byte.

NOTE – The VLI byte is undefined at the Pq-X-L_AP output of this function.

Defects None.

Consequent actions

aCSF-RDI ← CI_SSFrdi and CSFrdifdiEnable and CSFEnable

aCSF-FDI ← CI_SSFfdi and CSFrdifdiEnable and CSFEnable

aCSF-LOS ← CI_SSF and CSFEnable

Defect correlations None.

Performance monitoring For further study.

11.4.2.2 LCAS-capable Pq-Xv to ETH adaptation sink function (Pq-X-L/ETH_A_Sk)

This function extracts ETH_CI information from a Pq-X-L_AI signal (q = 11s, 12s, 31s, 32e), delivering ETH_CI to ETH_TFP and ETH_FP.

Data at the Pq-X-L_AP is a Pq-X-L (q = 11s, 12s, 31s, 32e), it has a payload as described in [ITU-T G.7043].

Symbol

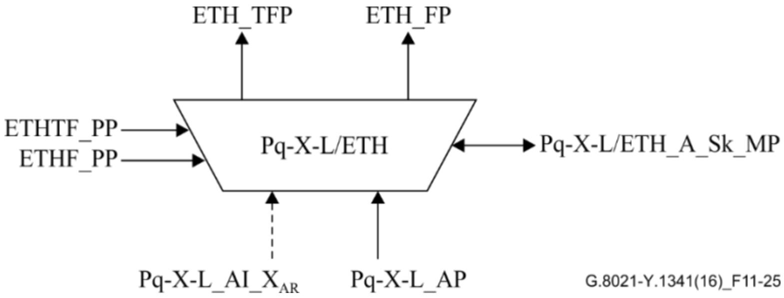


Figure 11-25 – Pq-X-L/ETH_A_Sk symbol

Interfaces

Table 11-14 – Pq-X-L/ETH_A_Sk interfaces

Inputs	Outputs
Pq-X-L_AP: Pq-X-L_AI_Data Pq-X-L_AI_ClocK Pq-X-L_AI_FrameStart Pq-X-L_AI_TSF Pq-X-L_AI_XAR **ETHTF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **ETHF_PP**: ETH_PI_D ETH_PI_P ETH_PI_DE **Pq-X-L/ETH_A_Sk_MP**: Pq-X-L/ETH_A_Sk_MI_FilterConfig Pq-X-L/ETH_A_Sk_MI_CSF_Reported Pq-X-L/ETH_A_Sk_MI_MAC_Length Pq-X-L/ETH_A_Sk_MI_CSFrdifdiEnable	**ETH_TFP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF **ETH_FP**: ETH_CI_D ETH_CI_P ETH_CI_DE ETH_CI_SSF ETH_CI_SSFrdi ETH_CI_SSFfdi **Pq-X-L/ETH_A_Sk_MP**: Pq-X-L/ETH_A_Sk_MI_AcSL Pq-X-L/ETH_A_Sk_MI_AcEXI Pq-X-L/ETH_A_Sk_MI_AcUPI Pq-X-L/ETH_A_Sk_MI_cPLM Pq-X-L/ETH_A_Sk_MI_cLFD Pq-X-L/ETH_A_Sk_MI_cUPM Pq-X-L/ETH_A_Sk_MI_cEXM Pq-X-L/ETH_A_Sk_MI_cCSF Pq-X-L/ETH_A_Sk_MI_pFCSError

Processes

A process diagram of this function is shown in Figure 11-26.

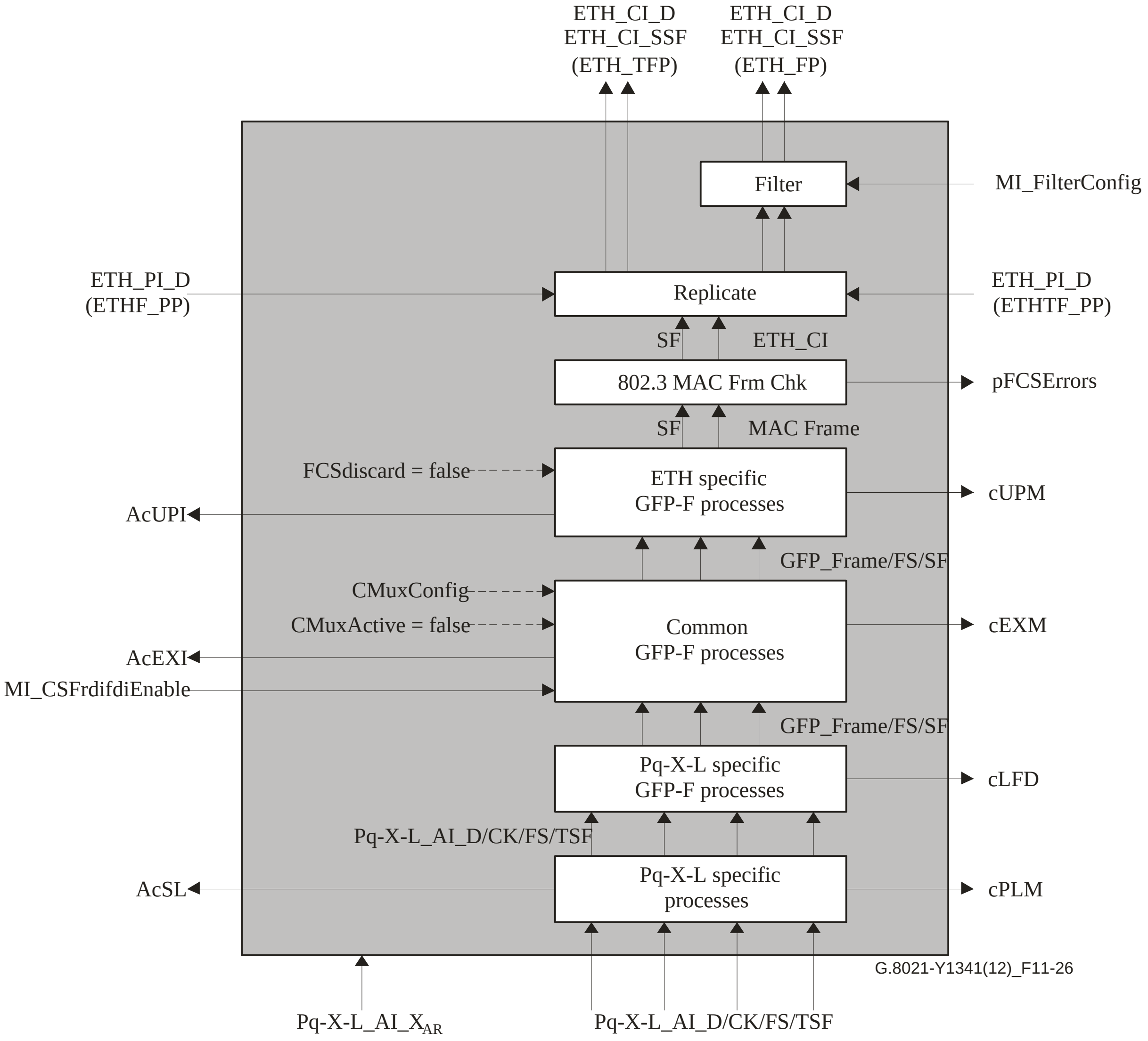


Figure 11-26 – Pq-X-L/ETH_A_Sk process

Filter process

See clause 8.3.

Replicate process

See clause 8.4.

802.3 MAC FCS check process

See clause 8.8.2.

Ethernet specific GFP-F sink process

See clause 8.5.4.1.2 of [ITU-T G.806]. GFP pFCS checking, GFP p_FCSError, p_FDis are not supported (FCSdiscard=false). The UPI value for frame-mapped Ethernet shall be expected (Table 6-3 of [ITU-T G.7041]). The Ethernet frames are extracted from the client payload information field of the GFP-F frames according to clause 7.1 of [ITU-T G.7041].

Common GFP sink process

See clause 8.5.3.2 of [ITU-T G.806]. GFP channel multiplexing is not supported (MI_CMuxActive=false).

Pq-X-L specific GFP sink process

See clause 8.5.2.2 of [ITU-T G.806]. The GFP frames are demapped from the Pq-X-L payload area according to [ITU-T G.8040].

Pq-X-L specific sink process

P31s-X-L specific

MA: The signal label is recovered from the payload type field in the MA byte as per clause 6.2.4.2 of [ITU-T G.806]. The signal label for GFP mapping in clause 2.1 of [ITU-T G.832] shall be expected. The accepted value of the signal label is also available at the P31s-X-L/ETH_A_Sk_MP.

NOTE 1 – The Pq-X-L_AI_X_{AR} interface is not connected to any of the internal processes.

Defects

dPLM – See clause 6.2.4.2 of [ITU-T G.806].

dLFD – See clause 6.2.5.2 of [ITU-T G.806].

dUPM – See clause 6.2.4.3 of [ITU-T G.806].

dEXM – See clause 6.2.4.4 of [ITU-T G.806].

dCSF-LOS – See clause 8.8.6.2.

dCSF-RDI – See clause 8.8.6.2.

dCSF-FDI – See clause 8.8.6.2.

NOTE 2 – dPLM is only defined for q = 31s. dPLM is assumed to be false for q = 11s, 12s, 32e.

Consequent actions

The function shall perform the following consequent actions:

aSSF ← AI_TSF or dPLM or dLFD or dUPM or dEXM or dCSF-LOS

aSSFrdi ← dCSF-RDI and CSFrdifdiEnable

aSSFfdi ← dCSF-FDI and CSFrdifdiEnable

NOTE 3 – X_{AR}=0 results in AI_TSF being asserted, so there is no need to include it as an additional contributor to aSSF.

Defect correlations

The function shall perform the following defect correlations to determine the most probable fault cause (see clause 6.4 of [ITU-T G.806]). This fault cause shall be reported to the EMF.

cPLM ← dPLM and (not AI_TSF)

cLFD ← dLFD and (not dPLM) and (not AI_TSF)

cUPM ← dUPM and (not dEXM) and (not dPLM) and (not dLFD) and (not AI_TSF)

cEXM ← dEXM and (not dPLM) and (not dLFD) and (not AI_TSF)

cCSF ← (dCSF-LOS or dCSF-RDI or dCSF-FDI) and (not dEXM) and (not dUPM) and (not dPLM) and (not dLFD) and (not AI_TSF) and CSF_Reported

Performance monitoring

The function shall perform the following performance monitoring primitives processing. The performance monitoring primitives shall be reported to the EMF.

pFCSError: count of FrameCheckSequenceErrors per second.

NOTE 4 – This primitive is calculated by the MAC FCS check process.

11.5 OTH to ETH adaptation functions (O/ETH_A)

11.5.1 ODUk to ETH adaptation functions (ODUkP/ETH_A)

The ODUkP to Ethernet adaptation function supporting GFP-F mapping of Ethernet over ODUk is given in clause 14.3.11 of [ITU-T G.798].

11.5.2 ODU2P to Ethernet Reconciliation sublayer adaptation functions

The ODU2P to Ethernet Reconciliation sublayer adaptation function for supporting the transport of the preamble and ordered set information of the 10GBASE-R signals over extended OPU2 payload area is given in clause 14.3.3 of [ITU-T G.798].

11.5.3 ODU0P to 1 GbE client adaptation functions (ODU0P/CBRx_A)

The adaptation function that supports the transport of 1GbE signals in the OTN is the ODU0P to the client adaptation function (ODU0P/CBRx_A) ($0 \leq x \leq 1.25G$) described in [ITU-T G.798]. When the client is 1 GbE, the CBRx and ETC1000X signals are equivalent.

11.6 MPLS to ETH adaptation functions (MPLS/ETH_A)

For further study.

11.7 ATM VC to ETH adaptation functions (VC/ETH_A)

For further study.

Appendix I

Applications and functional diagrams

(This appendix does not form an integral part of this Recommendation.)

Figure I.1 presents the set of atomic functions associated with the Ethernet signal transport, shown in several example applications.

- Ethernet UNI/NNI interface port on EoT equipment.
- Ethernet over SDH NNI interface port on EoT equipment.
- Ethernet UNI interface port supporting multiplexed access on EoT equipment.

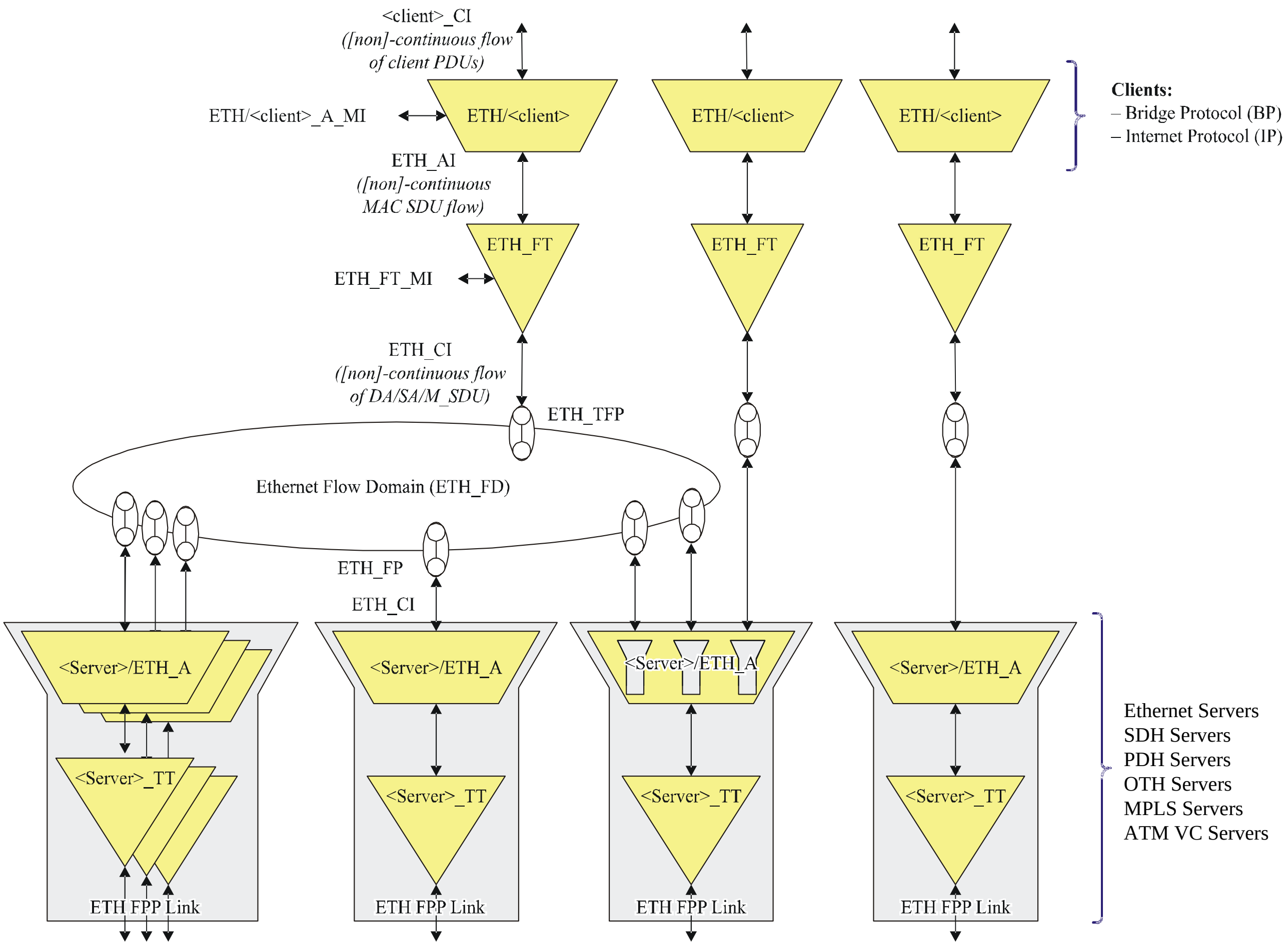


Figure I.1 – Ethernet atomic functions in several possible applications

Appendix II

AIS/RDI mechanism for an Ethernet private line over a single SDH or OTH server layer

(This appendix does not form an integral part of this Recommendation.)

In order to address fault notification for failures in either the access links or within the SDH / OTH server layer, the following functionality is required:

a) Convey fault notification for an access link failure from one side of the network to the other.

b) Convey fault notification for an SDH / OTH server layer failure to the access links.

[ITU-T G.7041] defines client management frames (CMFs) for conveying information about the client signal from an ingress edge NE to the egress edge NE. Defined CMF signals are client signal fail (CSF), client forward defect indication (FDI) and client reverse defect indication (RDI) implementing the remote fail indication mechanism.

[ITU-T G.806] defines the equipment functional details of the CSF and RFI mechanisms.

This Recommendation defines the Ethernet specific equipment functional details for the CSF and RFI mechanisms.

The combination of the above three Recommendations provides the functionality required by (a) and (b).

In addition, this basic functionality can be further enhanced to support fault notification for the Ethernet client by using Ethernet physical layer defect signals shown in Appendix VI of [ITU-T G.7041] by means of Ethernet OAM. For example, use of the link fault flag defined in clause 57 of [IEEE 802.3] (EFM OAM), in conjunction with the GFP-F CMF CSF and RFI indications. This is illustrated below.

A simplifying assumption can be made regarding the conditioning of the Ethernet access links on either side of the SDH / OTH transport network. For an EPL application, the access link is specific to a single service and since an Ethernet service is bidirectional, a fault in either direction should result in the access link being conditioned as "failed".

The following fault scenarios and accompanying figures illustrate this example of interworking of the EFM OAM link fault flag with the GFP-F CMF CSF and RFI indications to appropriately condition the Ethernet access links. Only unidirectional faults are considered, the scenarios can be combined as per the superposition principle to describe bidirectional faults. Furthermore, only an SDH server layer is shown in the examples. CE = Customer edge. PE = Provider edge.

Scenario 1

In Figure II.1, a unidirectional fault occurs on the east access link on ingress to the carrier network.

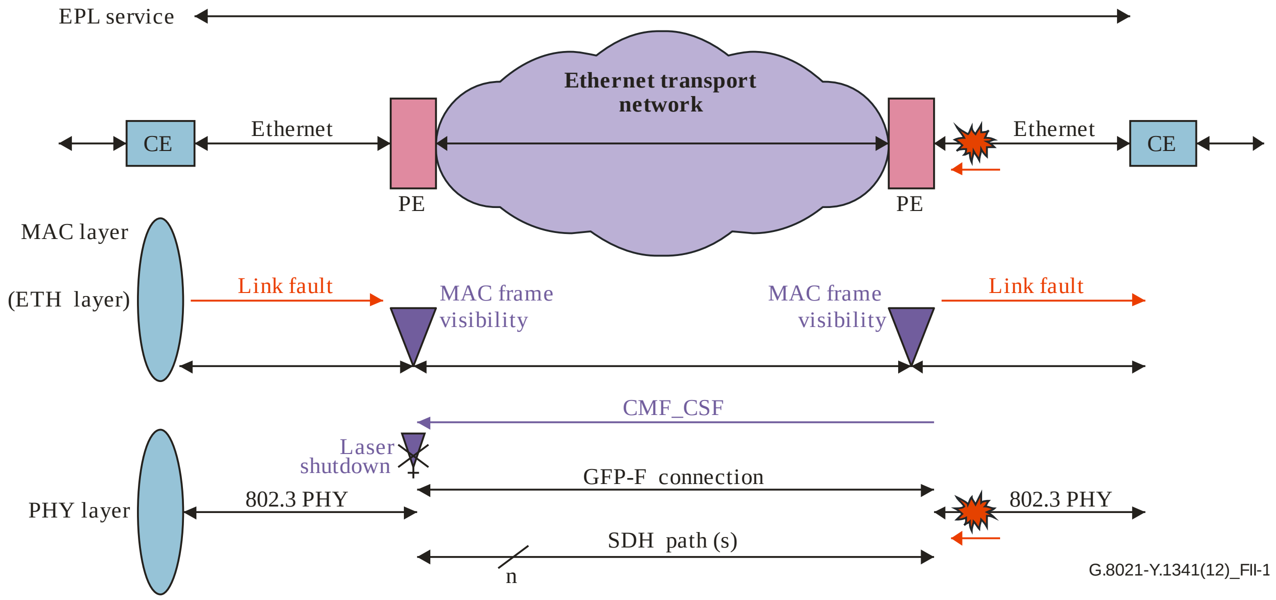


Figure II.1 – Fault on ingress

– The east PE detects a loss of signal on the ingress access link:
  - 802.3 EFM OAM sends "Link fault" upstream, interspersed with Idles;
  - GFP-F CMF CSF indication is sent into the network.

– The east CE detects "Link fault":
  - Idles are sent towards the network and towards the enterprise.

– The west PE detects the GFP-F CMF CSF indication:
  - If there is no network_ETH_AIS indication available, the laser (or electrical driver) is shut down.

– The west CE detects a loss of signal:
  - 802.3 EFM OAM sends "Link fault" upstream, interspersed with Idles;
  - Idles are sent towards the enterprise.

Scenario 2

In Figure II.2, a unidirectional fault occurs westbound on the server layer within the carrier network.

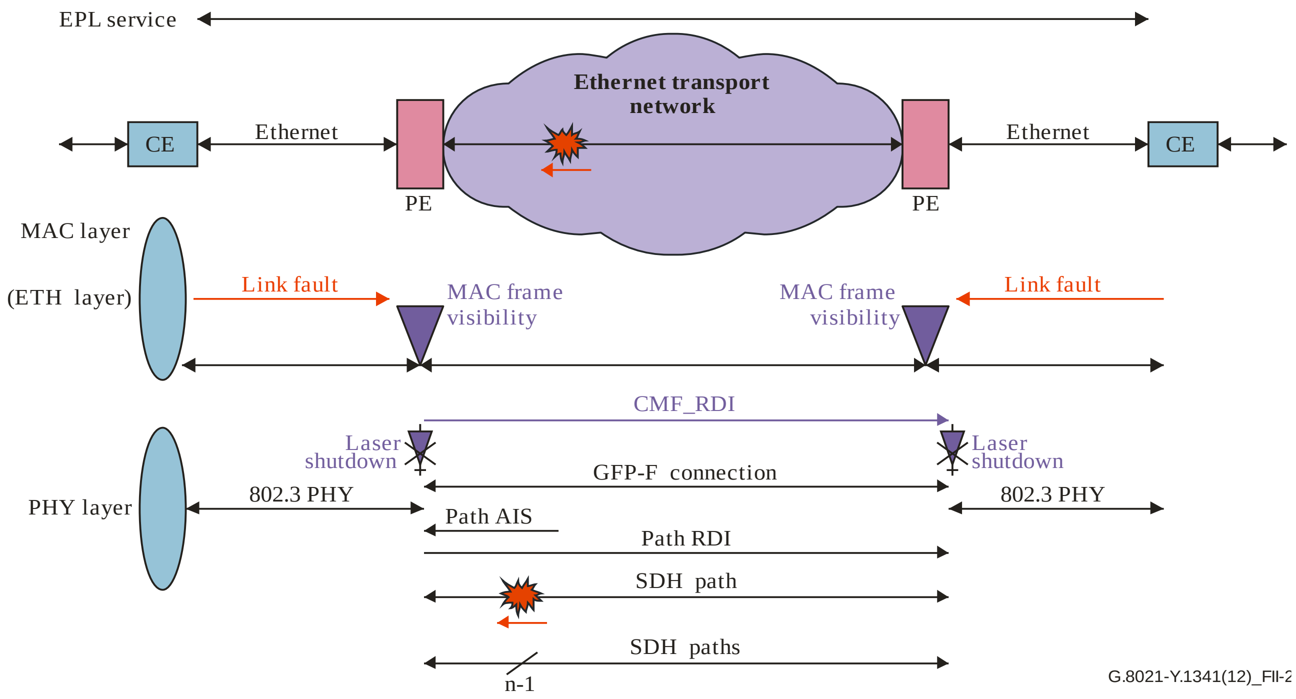


Figure II.2 – Fault within a carrier network

- An NE in the carrier network detects the failure of one of the member paths of a VCAT group:
  - SDH path AIS is generated downstream on the affected path.
- The west PE detects SDH path AIS:
  - SDH path RDI is generated back into the network on the associated path;
  - GFP-F CMF RDI is generated back into the network;
  - if there is no network_ETH_AIS indication available, the laser (or electrical driver) is shut down.
- The west CE detects a loss of signal:
  - 802.3 EFM OAM sends "Link fault" upstream, interspersed with Idles;
  - Idles are sent towards the enterprise.
- The east PE detects the GFP-F CMF RDI indication:
  - If there is no network_ETH_RDI indication available, the laser (or electrical driver) is shut down.
- The east CE detects a loss of signal:
  - 802.3 EFM OAM sends "Link fault" upstream, interspersed with Idles;
  - Idles are sent towards the enterprise.

Note that for a network failure affecting all member paths of a VCAT group (where LCAS is not supported) the same steps above apply, with the addition of SDH path AIS and RDI being sent on all the member paths.

Scenario 3

In Figure II.3, a unidirectional fault occurs on the west access link towards the enterprise network.

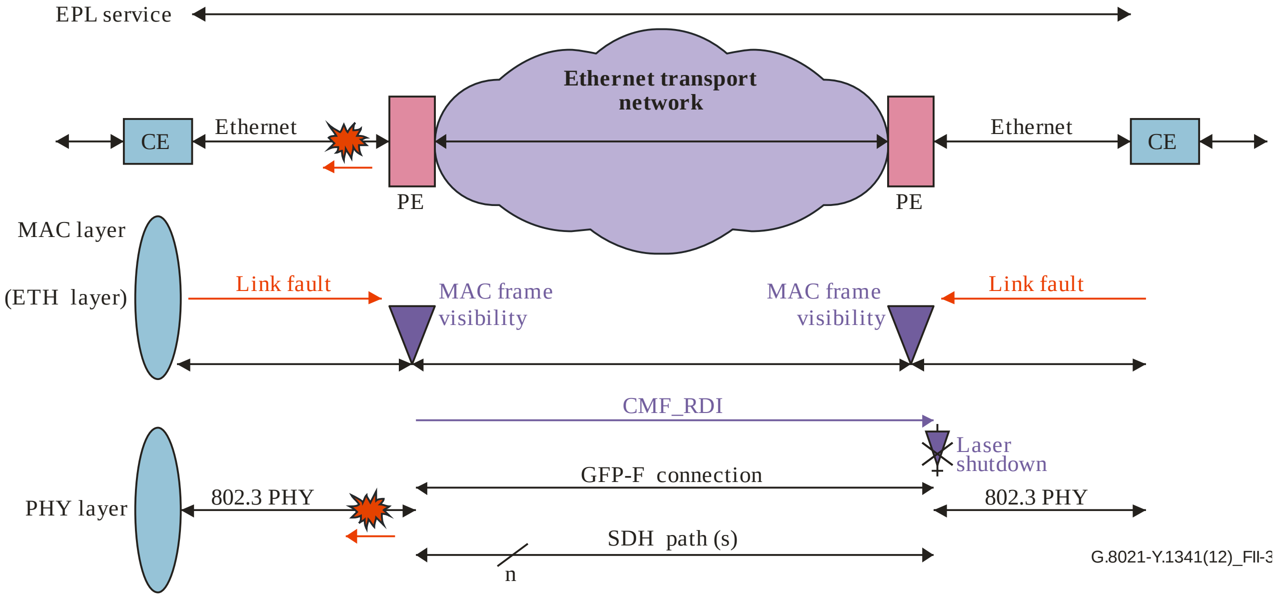


Figure II.3 – Fault on egress

- The west CE detects a loss of signal:
  - 802.3 EFM OAM sends "Link fault" upstream, interspersed with Idles;
  - Idles are sent towards the enterprise.
- The west PE detects the link fault indication:
  - GFP-F CMF RDI indication is sent into the network;
  - Idles are sent towards the CE.
- The east PE detects the GFP-F CMF RDI indication:
  - If there is no network_ETH_RDI indication available, the laser (or electrical driver) is shut down.
- The east CE detects a loss of signal:
  - 802.3 EFM OAM sends "Link fault" upstream, interspersed with Idles;
  - Idles are sent towards the enterprise.

Note that a PE only reacts to the reception of a link fault indication when there are no other conditioning alarms (i.e., the PE takes no further conditioning action when it receives a link fault indication in response to having shut down its own egress laser).

Appendix III

Compound functions

(This appendix does not form an integral part of this Recommendation.)

ETH MEP and MIP compound functions are defined in clause 9.8.

Appendix IV

Startup conditions

(This appendix does not form an integral part of this Recommendation.)

The set of interconnected ETH_FF processes must be loop-free, otherwise the integrity of the network may be compromised. This requirement implies that one can only include ports of the ETH_FF process in the ETH_C function if it is known that this will not create a loop.

In [b-IEEE 802.1D] and [IEEE 802.1Q] this is secured by starting in a state without connectivity, except for the exchange of BPDUs. Consequently, the spanning tree protocol extends the connectivity while making sure that this does not create any loops.

This means that the ETH_C function as defined in this Recommendation, on startup of the equipment may not contain an ETH_FF that includes more than one port of its enclosing ETH_FF process. After startup, ports may be added to the ETH_FF process under the control of the spanning tree protocol. Alternatively, this may be done under control of a management system, as long as the management system can guarantee that there are no loops created.

Appendix V

SDL descriptions

(This appendix does not form an integral part of this Recommendation.)

In this Recommendation, detailed characteristics of equipment functional blocks are described with SDL diagrams specified in [ITU-T Z.101]. The SDL diagrams use the following conventions.

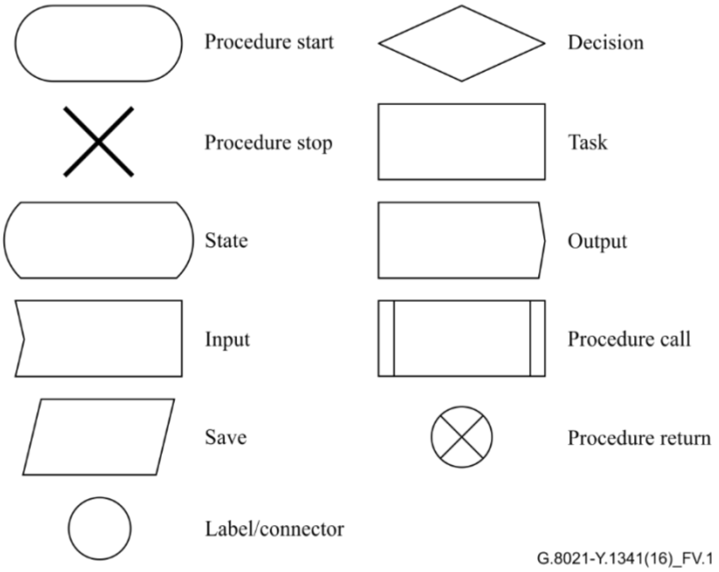


Figure V.1 – SDL symbols

Appendix VI

Calculation methods for frame loss measurement

(This appendix does not form an integral part of this Recommendation.)

Frame loss measurement is performed by the collection of counter values for ingress and egress service frames and the exchange of OAM frames with the local counter value between a pair of MEPs. In this Recommendation two different mechanisms are defined for frame loss measurement and both mechanisms have different calculation methods.

VI.1 Dual-ended loss measurement

This is performed by proactive OAM and both MEPs send dual-ended CCM frames to its peer MEP periodically. The calculation method specified in the proactive loss measurement process is depicted as shown in Figure VI.1.

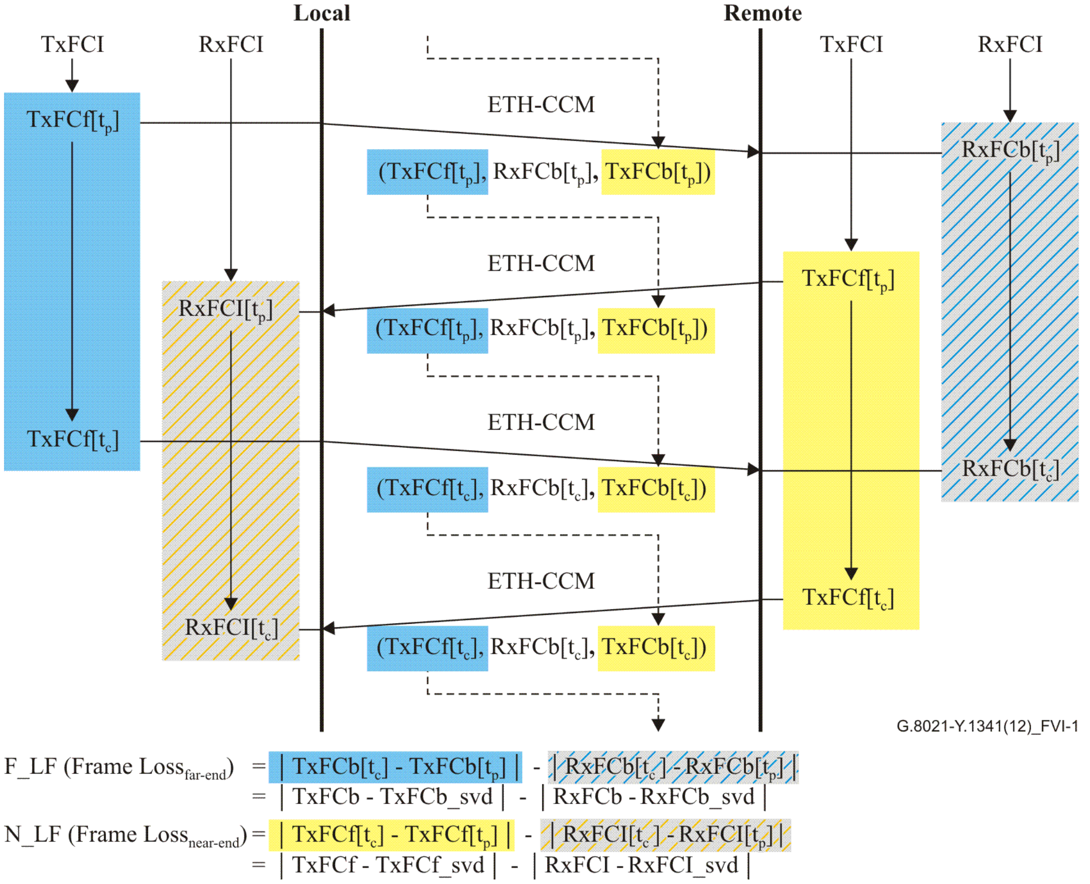
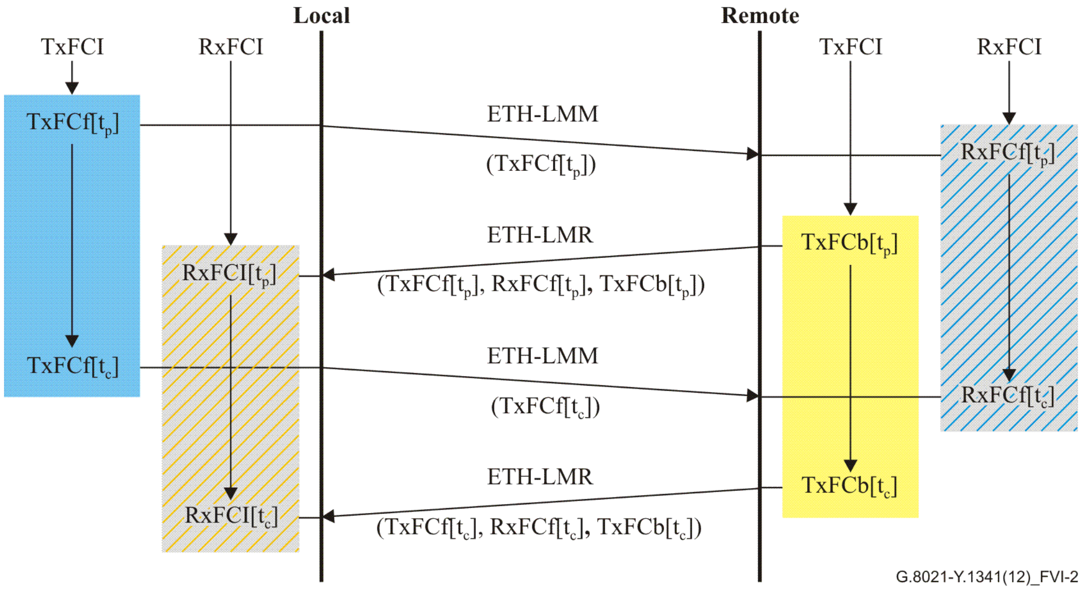


Figure VI.1 – Dual-ended ETH LM

VI.2 Single-ended loss measurement

This is performed by the on-demand OAM and an MEP which sends LMM frames to its peer MEP and receives LMR frames from its peer MEP. The calculation method specified in the LM control process is depicted as shown in Figure VI.2.



$$\text{F\_LF (Frame Loss}_{\text{far-end}}) = |\text{TxFCf}[t_c] - \text{TxFCf}[t_p]| - |\text{RxFCf}[t_c] - \text{RxFCf}[t_p]|$$

$$= |\text{TxFCf} - \text{TxFCf\_svd}| - |\text{RxFCf} - \text{RxFCf\_svd}|$$

$$\text{N\_LF (Frame Loss}_{\text{near-end}}) = |\text{TxFCb}[t_c] - \text{TxFCb}[t_p]| - |\text{RxFCl}[t_c] - \text{RxFCl}[t_p]|$$

$$= |\text{TxFCb} - \text{TxFCb\_svd}| - |\text{RxFCl} - \text{RxFCl\_svd}|$$

Figure VI.2 – Single-ended ETH LM

Appendix VII

Considerations of the support of a rooted multipoint EVC service

(This appendix does not form an integral part of this Recommendation.)

This appendix considers the support of a rooted multipoint service defined in [ITU-T G.8011]. Connectivity of a rooted multipoint service is established between one or more rooted points and zero or more leaf points. Each leaf point can only exchange data with the root point, while a root point can exchange data with each leaf point and other root points. Consequently, some extended mechanisms on the ETH layer is required to disable the connectivity between particular pairs of points.

Two potential models are introduced in this appendix. The first model is achieved by the enhancement of a port group functionality to the ETH flow forwarding function. The second model is composed of the usage of asymmetric VLANs configuration described in clause B.1.3 of [IEEE 802.1Q]. The subclasses below describe a principle of the operation for each model.

NOTE 1 – The asymmetric VLAN model will be included in the main body of a later version of this Recommendation after the development of the functional modelling and the study of interworking between the asymmetric VLAN model and the port group model.

NOTE 2 – Both the port group and the asymmetric VLAN models are also applicable to other network scenarios such as the multipoint-to-multipoint service defined in [ITU-T G.8011] while this appendix addresses the rooted multipoint service only. Examples of application to other scenarios will be considered in a later version of this Recommendation.

VII.1 Port group function

The port group function is achieved by the enhancement to the ETH flow forwarding function defined in clause 9.1.1. Figure VII.1 shows a principle of the operation for the port group function. A port group is configured to the ports {A, B, C} for which the split horizon behaviour are applied in an ETH flow forwarding function. Frames arriving via an input port in the port group can be forwarded to one or more output ports with the exception of the output ports that are members of the port group. Frames arriving on an input port that is not a member of the port group can be forwarded to any output ports with the exception of the port over which the frame arrived. As a result, the direct communication between members of the port group can be disabled.

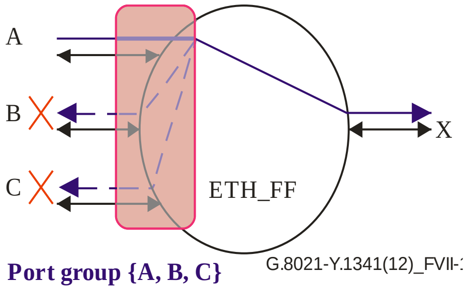


Figure VII.1 – Principle of the port group function

Figure VII.2 shows an example of the port group function composing a rooted-multipoint EVC. The node X in this figure is configured to disable the forwarding ETH_CI traffic signal between members of the port group {X2, X3, X4}.

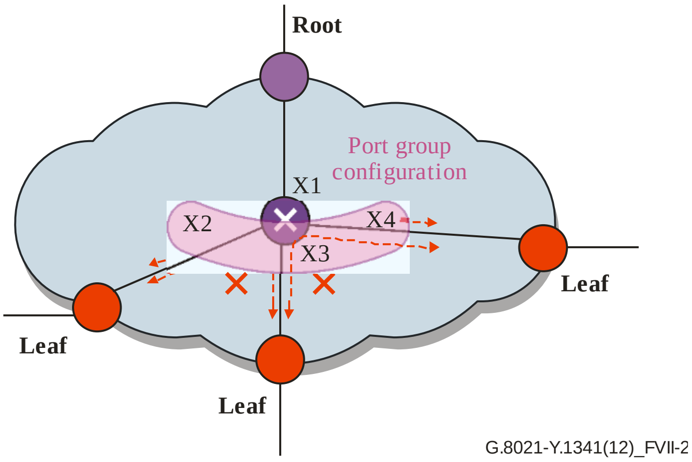


Figure VII.2 – Application example of the port group function

VII.2 Configuration of asymmetric VLANs

Clause B.1.3 of [IEEE 802.1Q] describes a configuration example of asymmetric VLANs to support a rooted multipoint service. The configuration allocates two different VLANs to the traffic generated by a root and a leaf (leaves) respectively. As a result, it can disable the direct communication between any pair of leaves. To facilitate an appropriate MAC learning over the different VLANs, this configuration uses a shared VLAN learning (SVL) mode described in clause 9.1.1.

Figure VII.3 shows an example of the operation. In this figure, the ports A, B, and C are attached to leaf nodes while the port X is attached to a root node. The VID M allocated to the traffic from the root node to leaf nodes is configured on ports A, B and C. The VID N allocated to the traffic from the leaf nodes to a root node is configured on port X only. As a result, asymmetric VLANs are configured and the appropriate connectivity between ports A, B, C and X is established.

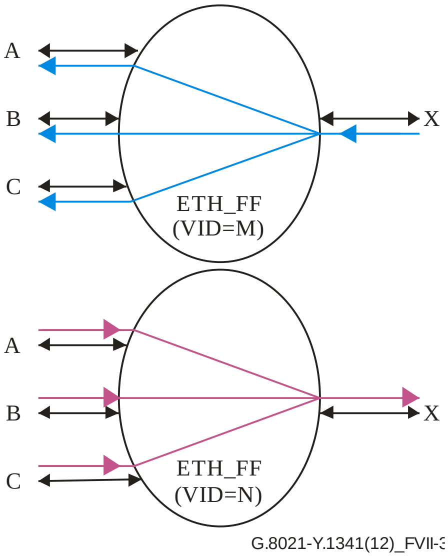


Figure VII.3 – Principle of the asymmetric VLANs

Figure VII.4 shows an application example of the asymmetric VLANs to a rooted multipoint service. Note that both a root node and leaf nodes can use the single VID or untagged frames on the client ports (depicted as yellow bidirectional arrows in this figure), while multiple VIDs are required within the EVC. This VID configuration on the client ports can be achieved by the VID translation and/or untagging on the output interfaces.

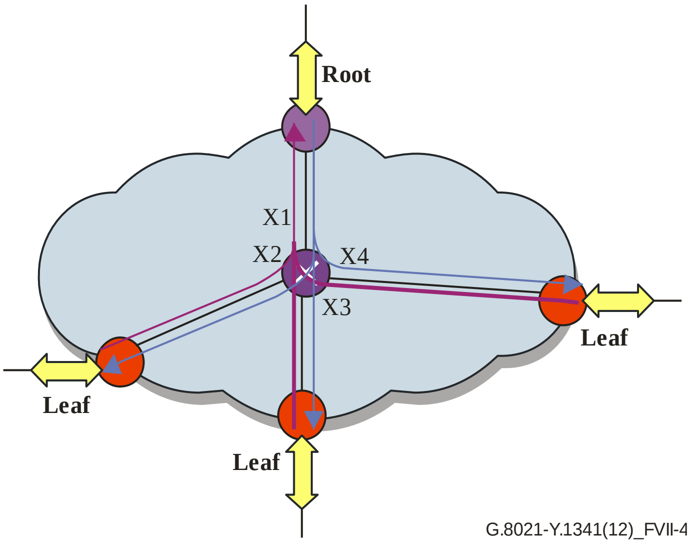


Figure VII.4 – Application example of the asymmetric VLANs

NOTE – This appendix only describes a scenario of the single rooted multipoint environment as a basic example. However, the asymmetric VLAN model can also support multiple root nodes and/or grouping of leaf nodes as advanced rooted multipoint scenarios.

Appendix VIII

Configurations for ingress VID filtering

(This appendix does not form an integral part of this Recommendation.)

This appendix describes an example of the configuration for ingress VID filtering described in [IEEE 802.1Q].

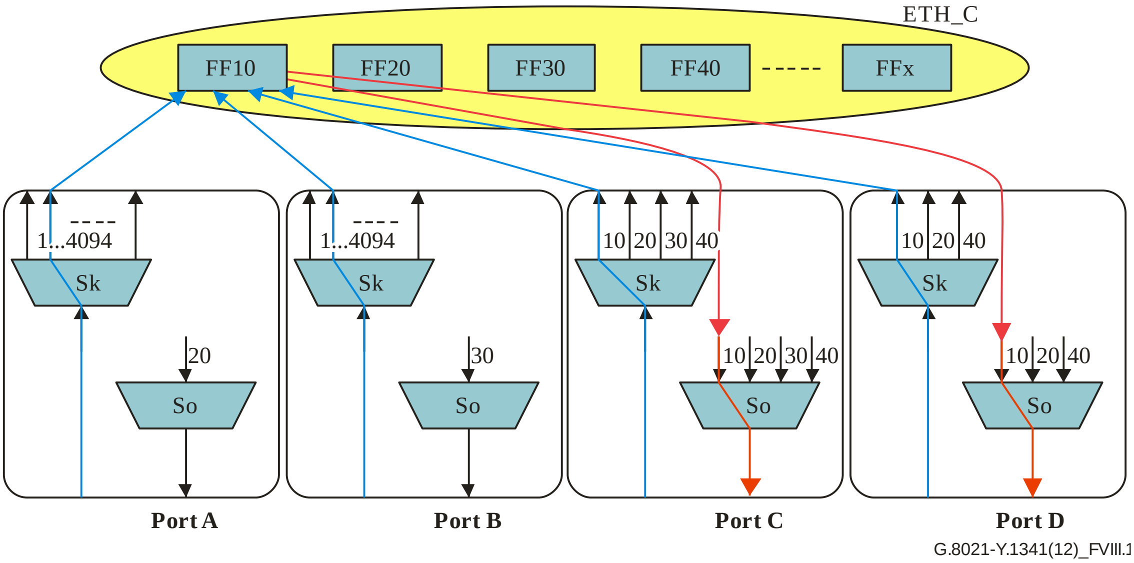


Figure VIII.1 – Example of configuration for ingress VID filtering

Table VIII.1 – VID configuration

VID	Port A		Port B		Port C		Port D	
	Ingress	Egress	Ingress	Egress	Ingress	Egress	Ingress	Egress
10	✓		✓		✓	✓	✓	✓
20	✓	✓	✓		✓	✓	✓	✓
30	✓		✓	✓	✓	✓		
40	✓		✓		✓	✓	✓	✓
Others	✓		✓					

Figure VIII.1 and Table VIII.1 show an example of the configuration. For the ingress configuration, MI_Vlan_Config[] signal is set to ETHx/ETH-m_A_Sk function and ETH_CI signals corresponding VIDs are connected to ETH_FF processes in ETH_C function. For the egress configuration, MI_Vlan_Config[] signal is set to ETHx/ETH-m_A_So function and ETH_CI signals corresponding VIDs are connected to ETH_FF processes in ETH_C function.

On ports A and B in this example, MI_Vlan_Config[1…4094] are set to ETHx/ETH-m_A_Sk in order to disable the ingress VID filtering. In this case, all incoming VIDs traffic is forwarded once to ETH_C. Since ETH_FF is connected to configured ingress and egress ports only, the traffic is forwarded to the appropriate ports.

Appendix IX

Handling of Expected Defects

(This appendix does not form an integral part of this Recommendation.)

This appendix describes how the Expected Defect messages (EDMs) can be used to avoid spurious Loss of Continuity defects, and provides some recommendations for how the Element Management function (EMF) should control the associated management information (in particular, ETHx_FT_Sk_MI_CC_Enable).

There are two primary use cases for handling of expected defects:

- Interruption events, such as in-service software upgrade, where transmission of CCMs is interrupted but there is no impact on the flow of data frames.
- Service activation, in particular adding a new end-point to an existing multipoint service.

These are discussed further below, followed by some additional considerations.

IX.1 Interruption events

In implementations where the OAM Generation functions execute in different hardware to that used for traffic forwarding – typically either a dedicated hardware chip designed for OAM, or in software on a general-purpose CPU – it is possible that the OAM traffic may be interrupted without affecting the data traffic flow, as shown in Figure IX.1. The desirable behaviour in this case may be for any peer devices to ignore the loss-of-continuity of the OAM traffic (since there is no interruption in the data traffic). If the interruption is due to a failure and is hence unexpected, that may not be possible; however, if it is due to an administrative action, then there is the possibility of notifying the peer in advance of the event. Examples of such intentional events include software or firmware upgrades, or manual recovery from earlier failure conditions.

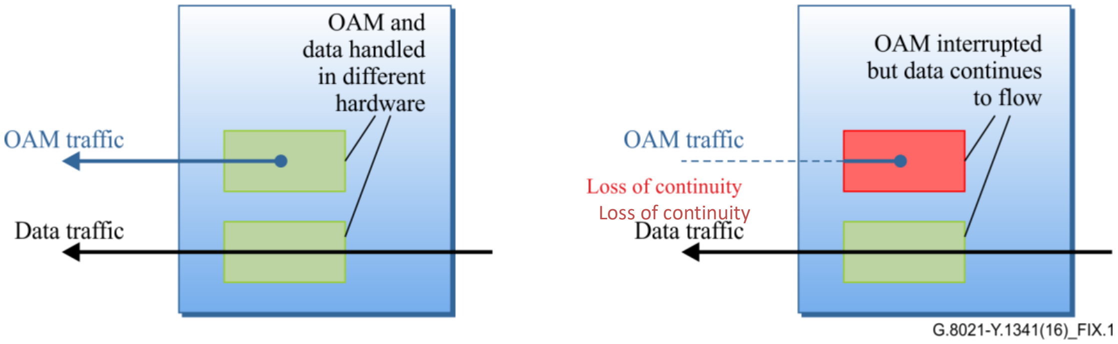


Figure IX.1 – Example where OAM functions and data traffic forwarding are in different hardware.

A mechanism to notify the peer MEPs in advance is particularly useful when OAM is used across multiple administrative domains (e.g., across a UNI or ENNI), as in these cases it may not be possible to correlate the event with the loss of continuity at the management layer. The Expected Defect message provides such a mechanism, by indicating that a loss of continuity is expected for a specified duration. It is triggered by setting MI_EDM_Enable to true, and MI_EDM_Duration to the expected duration for which CCM transmission will be interrupted.

On receiving an EDM, the peer MEP relays the information to the EMF. If configured to do so, the EMF can then unset MI_CC_Enable in the flow termination sink function of the MEP (ETHx_FT_Sk or ETHG_FT_Sk), to disable receipt of CCMs for the duration specified in the EDM. When receipt of CCMs is disabled, loss of continuity (dLOC) does not result in either alarms (cLOC) or consequent actions (aTSF). After the specified duration, the EMF re-enables MI_CC_Enable; if CCMs have resumed, then dLOC will no longer be detected. If CCMs are still not being received, then dLOC will still be detected and this will result in alarms (cLOC) and consequent actions (aTSF).

An example showing the use of EDM in this case is shown in Figure IX.2.

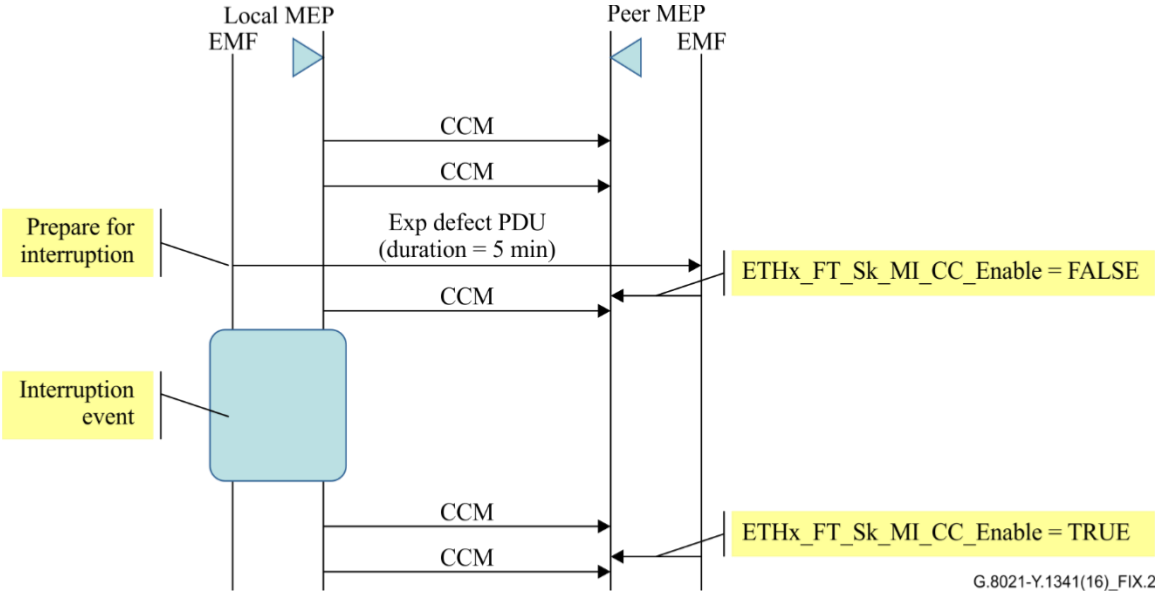


Figure IX.2 – Example showing EDM used to handle an interruption event

IX.2 Service activation

To enable correct operation of Continuity Checks in G.8013-based Ethernet OAM, it is necessary to configure each MEP in a MEG with its own unique MEP ID (via MI_MEP_ID), and in addition with the MEP IDs of all of its peer MEPs (via MI_PeerMEP_ID[]).

This can cause difficulties when adding a new end-point (and hence a new MEP) to an existing service: to avoid spurious defects and alarms, the configuration must be changed on all of the existing MEPs simultaneously with enabling CCMs on the new MEP. Again this is particularly challenging when the MEPs are in different administrative domains.

The spurious alarms can be avoided in this case using the Expected Defect message, as follows. When the new MEP is added, before enabling CCM transmission, Expected Defect messages are sent. As in the above case, on receiving these, the existing MEPs in the MEG relay the information to their corresponding EMFs, and if so configured, the EMFs disabled CCM reception. The EMFs at the existing MEPs can then add the MEP ID of the new MEP to the list of Peer MEP IDs (MI_PeerMEP_ID[]) without triggering any Loss of Continuity alarms or consequent actions for the new MEP, even though CCMs are not yet being received. Once this is done, CCM transmission can be enabled at the new MEP, and this will not trigger Unexpected MEP defects (dUNM) at the existing MEPs, as the new MEP ID has already been added to their lists of Peer MEPs. Finally once the duration in the EDM has passed, the EMFs at the existing MEPs re-enable CCM reception.

An example showing this sequence can be seen in Figure IX.3.

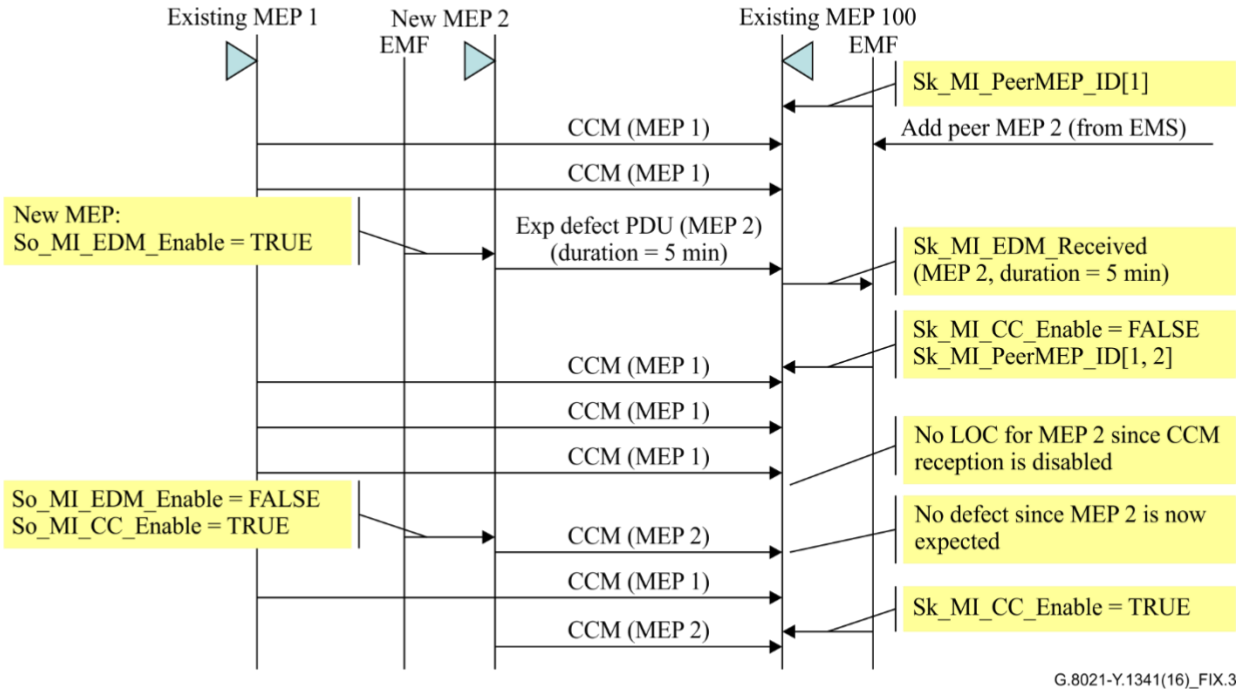


Figure IX.3 – Example showing EDM used to handle a new MEP

IX.3 Additional considerations

It should not be possible – maliciously or accidentally – to circumvent normal fault monitoring by continuously sending Expected Defect notifications for an extended period of time. This can be prevented in a number of ways:

– Implementations should limit the maximum value of MI_EDM _Duration that the user can specify. In some cases, the EMF may be able to derive the value without input from the end user; for example, in the case of an in-service software upgrade, the EMF can determine how long this will take, and hence for how long normal CCM transmission will be interrupted. It can then set MI_EDM_Enable and MI_EDM _Duration accordingly.

– The Expected Defect signal (MI_EDM_Received) should be ignored by the EMF unless processing is explicitly enabled by the user. The EMF should allow the user control over when this is enabled; for example, the user may wish to only allow Expected Defect notifications to be processed during a maintenance window. Even when enabled by the user, the EMF should temporarily disable the handling in some cases as described below.

– The EMF should allow the user to specify the maximum duration of an Expected Defect notification that will be handled. If an EDM is received indicating a longer duration than this, the duration is truncated to this value.

– The EMF should limit the number of times an Expected Defect notification is processed in a given period of time, for example to 3 times in a month. Note that the limit applies to each series of consecutive EDMs (from the same peer MEP), not to the number of individual EDM frames received.

– To prevent multiple uses of the Expected Defect notification in quick succession, the EMF should disable processing for a short time after the end of each expected defect condition.

– Whenever an Expected Defect notification is received, this should be logged by the EMF, so that any long-term trends can be analysed and misuse can be detected.

Appendix X

Mapping guidelines between the atomic functions defined in [b-ITU-T G.8021-2016] and those defined in [ITU-T G.8023]

(This appendix does not form an integral part of this Recommendation.)

Clause 10 of [b-ITU-T G.8021-2016] and older versions of this Recommendation were defining the ETY layer network to model the Ethernet physical layer defined in [IEEE 802.3]. These ETY-related atomic functions have been superseded by the atomic functions defined in [ITU-T G.8023]: clause 10 of this Recommendation has been aligned with [ITU-T G.8023]. This appendix provides some informative guidelines to map between the atomic functions defined in [b-ITU-T G.8021-2016] and those defined in [ITU-T G.8023].

Ethernet interfaces providing the adaptation between the ETH layer network and the Ethernet physical layer defined in [IEEE 802.3] are modelled in [ITU-T G.8023] by the xTSi[G]/ETH_A atomic functions, defined in clause 6.1 of [ITU-T G.8023], that supersede both the ETYn_TT and ETYn/ETH_A atomic functions, defined in clauses 10.2 and 10.3 of [b-ITU-T G.8021-2016].

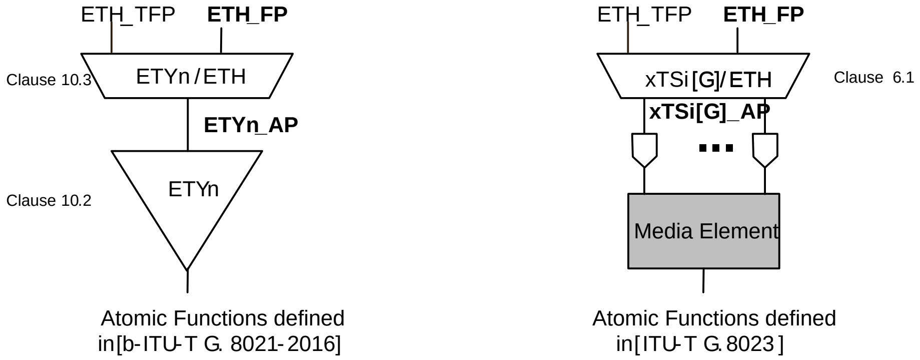


Figure X.1 – Mapping between ETYn/ETH and xTSi[G]/ETH adaptation functions

Ethernet interfaces providing the adaptations between the 8B/10B codewords of the 1000BASE-X PCS and the Ethernet physical layer defined in [IEEE 802.3] are modelled in [ITU-T G.8023] by the OTSi/ETC1000X_A atomic functions, defined in clause 6.3 of [ITU-T G.8023], that supersede both the ETY3_TT atomic functions, which were not defined in [b-ITU-T G.8021-2016] and the ETY3/ETC3_A atomic functions, defined in clauses 10.4 of [b-ITU-T G.8021-2016].

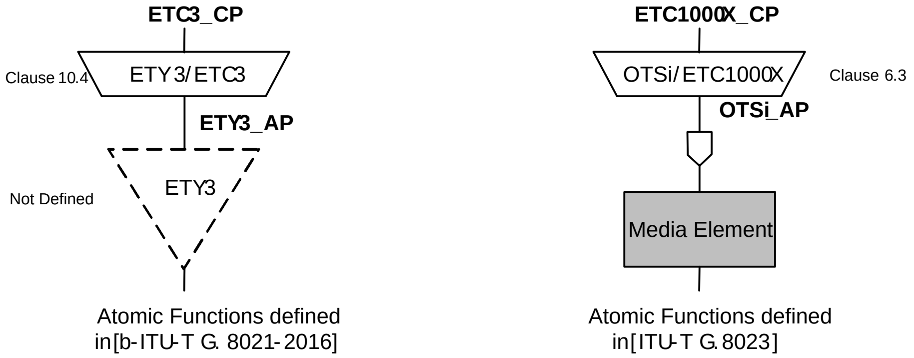


Figure X.2 – Mapping between ETY3/ETC3 and OTSi/ETC1000X adaptation functions

NOTE – The ETY3_TT atomic functions in this diagram that would have been used as the server layer of the ETY3/ETC3_A atomic functions should have been different from the ETYn_TT (n=3) atomic functions, defined in clause 10.2 of [b-ITU-T G.8021-2016] because the ETY3/ETC3_A function operates on 8B/10B codewords instead of MAC frames. This inconsistency has been resolved with the new model defined in [ITU-T G.8023].

Ethernet interfaces providing the adaptations between the preamble and ordered set information of the 10G Ethernet Reconciliation sub-layer and the Ethernet physical layer defined in [IEEE 802.3] are modelled in [ITU-T G.8023] by the OTSi/ERS10G_A atomic functions, defined in clause 6.2 of [ITU-T G.8023], that supersede both the ETY4_TT atomic functions, which were not defined in [b-ITU-T G.8021-2016] and ETY4/ETHPP-OS_A atomic functions, defined in clauses 10.7 of [b-ITU-T G.8021-2016].

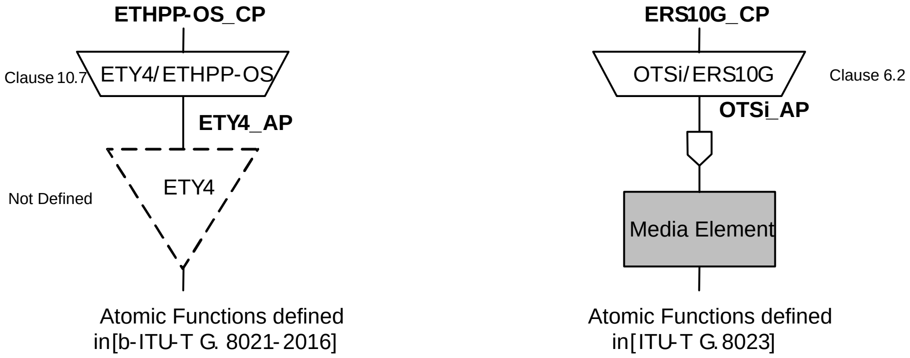


Figure X.3 – Mapping between ETY4/ETHPP-OS and OTSi/ERS10G adaptation functions

NOTE – The ETY4_TT atomic functions in this diagram that would have been used as the server layer of the ETY4/ETHPP-OS_A atomic functions should have been different from the ETYn_TT (n=4) atomic functions, defined in clause 10.2 of [b-ITU-T G.8021-2016] because the ETY4/ETHPP-OS_A function preserves preamble and ordered set information. This inconsistency has been resolved with the new model defined in [ITU-T G.8023].

Bibliography

[b-ITU-T G.704] Recommendation ITU-T G.704 (1998), *Synchronous frame structures used at 1544, 6312, 2048, 8448 and 44 736 kbit/s hierarchical levels.*

[b-ITU-T G.8021-2016] Recommendation ITU-T G.8021/Y.1341 (2016), *Characteristics of Ethernet transport network equipment functional blocks.*

[b-ITU-T I.732] Recommendation ITU-T I.732 (2000), *Functional characteristics of ATM equipment.*

[b-ITU-T M.3208.1] Recommendation ITU-T M.3208.1 (1997), *TMN management services for dedicated and reconfigurable circuits network: Leased circuit services.*

[b-IEEE 802.1D] IEEE 802.1D (2004), *IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges*.

ITU-T Y-SERIES RECOMMENDATIONS

GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES

GLOBAL INFORMATION INFRASTRUCTURE	
General	Y.100–Y.199
Services, applications and middleware	Y.200–Y.299
Network aspects	Y.300–Y.399
Interfaces and protocols	Y.400–Y.499
Numbering, addressing and naming	Y.500–Y.599
Operation, administration and maintenance	Y.600–Y.699
Security	Y.700–Y.799
Performances	Y.800–Y.899
INTERNET PROTOCOL ASPECTS	
General	Y.1000–Y.1099
Services and applications	Y.1100–Y.1199
Architecture, access, network capabilities and resource management	Y.1200–Y.1299
Transport	**Y.1300–Y.1399**
Interworking	Y.1400–Y.1499
Quality of service and network performance	Y.1500–Y.1599
Signalling	Y.1600–Y.1699
Operation, administration and maintenance	Y.1700–Y.1799
Charging	Y.1800–Y.1899
IPTV over NGN	Y.1900–Y.1999
NEXT GENERATION NETWORKS	
Frameworks and functional architecture models	Y.2000–Y.2099
Quality of Service and performance	Y.2100–Y.2199
Service aspects: Service capabilities and service architecture	Y.2200–Y.2249
Service aspects: Interoperability of services and networks in NGN	Y.2250–Y.2299
Enhancements to NGN	Y.2300–Y.2399
Network management	Y.2400–Y.2499
Network control architectures and protocols	Y.2500–Y.2599
Packet-based Networks	Y.2600–Y.2699
Security	Y.2700–Y.2799
Generalized mobility	Y.2800–Y.2899
Carrier grade open environment	Y.2900–Y.2999
FUTURE NETWORKS	Y.3000–Y.3499
CLOUD COMPUTING	Y.3500–Y.3999
INTERNET OF THINGS AND SMART CITIES AND COMMUNITIES	
General	Y.4000–Y.4049
Definitions and terminologies	Y.4050–Y.4099
Requirements and use cases	Y.4100–Y.4249
Infrastructure, connectivity and networks	Y.4250–Y.4399
Frameworks, architectures and protocols	Y.4400–Y.4549
Services, applications, computation and data processing	Y.4550–Y.4699
Management, control and performance	Y.4700–Y.4799
Identification and security	Y.4800–Y.4899
Evaluation and assessment	Y.4900–Y.4999

For further details, please refer to the list of ITU-T Recommendations.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	**Transmission systems and media, digital systems and networks**
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities
Series Z	Languages and general software aspects for telecommunication systems