IEC 61158-6-3

Edition 4.0 2019-06

INTERNATIONAL STANDARD

Industrial communication networks – Fieldbus specifications – Part 6-3: Application layer protocol specification – Type 3 elements

IEC 61158-6-3:2019-06(en)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 000 terminological entries in English and French, with equivalent terms in 16 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary
67 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC 61158-6-3

Edition 4.0 2019-06

INTERNATIONAL STANDARD

Industrial communication networks – Fieldbus specifications – Part 6-3: Application layer protocol specification – Type 3 elements

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-7008-0

Warning! Make sure that you obtained this publication from an authorized distributor.

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-3: Application layer protocol specification – Type 3 elements

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

Attention is drawn to the fact that the use of the associated protocol type is restricted by its intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a layer protocol type to be used with other layer protocols of the same type, or in other type combinations explicitly authorized by its intellectual-property-right holders.

NOTE Combinations of protocol types are specified in IEC 61784-1 and IEC 61784-2.

International Standard IEC 61158-6-3 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This fourth edition cancels and replaces the third edition published in 2014. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

- corrected substitutions in Table 4;
- corrections in 5.3.14;
- corrections in 5.5.6;
- corrections in 5.17.15;
- corrections in 5.17.16.2;
- spelling and grammar.

The text of this International standard is based on the following documents:

FDIS	Report on voting
65C/948/FDIS	65C/956/RVD

Full information on the voting for the approval of this International Standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts of the IEC 61158 series, published under the general title *Industrial communication networks – Fieldbus specifications*, can be found on the IEC web site.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under <http://webstore.iec.ch> in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

A bilingual version of this publication may be issued at a later date.

INTRODUCTION

This part of IEC 61158 is one of a series produced to facilitate the interconnection of automation system components. It is related to other standards in the set as defined by the "three-layer" fieldbus reference model described in IEC 61158-1.

The application protocol provides the application service by making use of the services available from the data-link or other immediately lower layer. The primary aim of this document is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer application entities (AEs) at the time of communication. These rules for communication are intended to provide a sound basis for development in order to serve a variety of purposes:

- as a guide for implementors and designers;
- for use in the testing and procurement of equipment;
- as part of an agreement for the admittance of systems into the open systems environment;
- as a refinement to the understanding of time-critical communications within OSI.

This document is concerned, in particular, with the communication and interworking of sensors, effectors and other automation devices. By using this document together with other standards positioned within the OSI or fieldbus reference models, otherwise incompatible systems may work together in any combination.

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-3: Application layer protocol specification – Type 3 elements

1 Scope

1.1 General

The Fieldbus Application Layer (FAL) provides user programs with a means to access the fieldbus communication environment. In this respect, the FAL can be viewed as a “window between corresponding application programs.”

This part of IEC 61158 provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 3 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This International Standard defines in an abstract way the externally visible behavior provided by the Type 3 fieldbus application layer in terms of

a) the abstract syntax defining the application layer protocol data units conveyed between communicating application entities,
b) the transfer syntax defining the application layer protocol data units conveyed between communicating application entities,
c) the application context state machine defining the application service behavior visible between communicating application entities; and
d) the application relationship state machines defining the communication behavior visible between communicating application entities.

The purpose of this document is to define the protocol provided to

a) define the wire-representation of the service primitives specified in IEC 61158-5-3, and
b) define the externally visible behavior associated with their transfer.

This document specifies the protocol of the Type 3 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498-1) and the OSI Application Layer Structure (ISO/IEC 9545).

FAL services and protocols are provided by FAL application-entities (AE) contained within the application processes. The FAL AE is composed of a set of object-oriented Application Service Elements (ASEs) and a Layer Management Entity (LME) that manages the AE. The ASEs provide communication services that operate on a set of related application process object (APO) classes. One of the FAL ASEs is a management ASE that provides a common set of services for the management of the instances of FAL classes.

Although these services specify, from the perspective of applications, how request and responses are issued and delivered, they do not include a specification of what the requesting and responding applications are to do with them. That is, the behavioral aspects of the applications are not specified; only a definition of what requests and responses they can

send/receive is specified. This permits greater flexibility to the FAL users in standardizing such object behavior. In addition to these services, some supporting services are also defined in this document to provide access to the FAL to control certain aspects of its operation.

1.2 Specifications

The principal objective of this document is to specify the syntax and behavior of the application layer protocol that conveys the application layer services defined in IEC 61158-5-3.

A secondary objective is to provide migration paths from previously-existing industrial communications protocols. It is this latter objective which gives rise to the diversity of protocols standardized in parts of the IEC 61158-6 subparts.

1.3 Conformance

This does not specify individual implementations or products, nor does it constrain the implementations of application layer entities within industrial automation systems.

There is no conformance of equipment to the application layer service definition standard. Instead, conformance is achieved through implementation of this application layer protocol specification.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE All parts of the IEC 61158 series, as well as IEC 61784-1 and IEC 61784-2 are maintained simultaneously. Cross-references to these documents within the text therefore refer to the editions as dated in this list of normative references.

IEC 61158-3-3:2014, *Industrial communication networks – Fieldbus specifications – Part 3-3: Data-link layer service definition – Type 3 elements*

IEC 61158-4-3:2019, *Industrial communication networks – Fieldbus specifications – Part 4-3: Data-link layer protocol specification – Type 3 elements*

IEC 61158-5-3:2014, *Industrial communication networks – Fieldbus specifications – Part 5-3: Application layer service definition – Type 3 elements*

IEC 61158-5-10:2019, *Industrial communication networks – Fieldbus specifications – Part 5-10: Application layer service definition – Type 10 elements*

IEC 61158-6-10:2019, *Industrial communication networks – Fieldbus specifications – Part 6-10: Application layer protocol specification – Type 10 elements*

ISO/IEC 7498-1, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ISO/IEC 8822, *Information technology – Open Systems Interconnection – Presentation service definition*

ISO/IEC 8824-1, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 9545, *Information technology – Open Systems Interconnection – Application Layer structure*

ISO/IEC 10731, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

IEEE 754, *IEEE Standard for Floating-Point Arithmetic,* available at <http://www.ieee.org> [viewed 2018-09-10]

3 Terms, definitions, abbreviations, symbols and conventions

For the purposes of this document, the following terms, definitions, abbreviations, symbols and conventions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org
- ISO Online browsing platform: available at <http://www.iso.org/obp>

3.1 Referenced terms and definitions

3.1.1 ISO/IEC 7498-1 terms

For the purposes of this document, the following terms as defined in ISO/IEC 7498-1 apply:

a) application entity
b) application process
c) application protocol data unit
d) application service element
e) application entity invocation
f) application process invocation
g) application transaction
h) real open system
i) transfer syntax

3.1.2 ISO/IEC 8822 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8822 apply:

a) abstract syntax
b) presentation context

3.1.3 ISO/IEC 9545 terms

For the purposes of this document, the following terms as defined in ISO/IEC 9545 apply:

a) application-association
b) application-context
c) application context name
d) application-entity-invocation
e) application-entity-type
f) application-process-invocation
g) application-process-type

h) application-service-element

i) application control service element

3.1.4 ISO/IEC 8824-1 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8824-1 apply:

a) object identifier

b) type

3.1.5 Fieldbus Data Link Layer terms

For the purposes of this document, the following terms as defined in IEC 61158-3-3 and IEC 61158-4-3 apply.

a) DL-Time

b) DL-Scheduling-policy

c) DLCEP

d) DLC

e) DL-connection-oriented mode

f) DLPDU

g) DLSDU

h) DLSAP

i) link

j) network address

k) node address

l) node

m) scheduled

3.2 Additional definitions

3.2.1
access protection
limitation of the usage of an application object to one client

3.2.2
address-assignment-table
mapping of the client's internal I/O-Data object storage to the decentralized input and output data objects

3.2.3
channel
single physical or logical link of an input or output application object of a server to the process

3.2.4
channel related diagnosis
information concerning a specific element of an input or output application object, provided for maintenance purposes

EXAMPLE: validity of data.

3.2.5
configuration check
comparison of the expected I/O-Data object structuring of the client with the real I/O-Data object structuring to the server in the start-up phase

3.2.6
configuration fault
an unacceptable difference between the expected I/O-Data object structuring and the real I/O-Data object structuring, as detected by the server

3.2.7
configuration identifier
representation of a portion of I/O Data of a single input- and/or output-module of a server.

3.2.8
configuration identifier related diagnosis
compromises all module specific data available at the server for maintenance purposes

3.2.9
control commands
action invocations transferred from client to server to clear outputs, freeze inputs and/or synchronize outputs

3.2.10
data consistency
means for coherent transmission and access of the input- or output-data object between and within client and server

3.2.11
Data-eXchange-Broadcast
DXB
service for conveyance of information between the DP-master (Class 1) and the assigned DP-slaves initiating the Publisher and Subscriber functionality in the DP-slaves

3.2.12
default DL-address
value 126 as an initial value for DL-address, which has to be changed (e.g. by assignment of an DL-address via the fieldbus) before operation with a DP-master (class 1)

3.2.13
device related diagnosis
compromises all data related to the whole DP-slave available at the server for maintenance purposes

3.2.14
diagnosis information
all data available at the server for maintenance purposes

3.2.15
diagnosis information collection
system diagnosis information that is assembled at the client side

3.2.16
DP-master (class 1)
a controlling device which controls several DP-slaves (field devices)

Note 1 to entry: This is usually a programmable controller or a distributed control system.

3.2.17
DP-master (class 2)
controlling device which manages configuration data (parameter sets) and diagnosis data of a DP-master (Class 1), and that additionally performs all communication capabilities of a DP-master (Class 1)

3.2.18
DP-slave
field device that can be assigned to one DP-master (Class 1) as a provider for cyclic I/O data exchange, in addition acyclic functions and alarms could be provided

3.2.19
DXB request
cyclic data exchange service request with a specific DL service between the DP-master (Class 1) and the assigned DP-slaves initiating the Publisher functionality in the DP-slaves

3.2.20
DXB response
cyclic data exchange service response not to the individual address of the requester (DP-master (Class 1)) but to the broadcast address (=127)

3.2.21
freeze

function at the DP-slaves for simultaneous data transfer between the input data object and the process

3.2.22
group
set of DP-slaves which perform a Freeze or Sync function

3.2.23
I/O data
object designated to be transferred cyclically for the purpose of processing.

3.2.24
ident number
DP-master (Class 1) or DP-slave device type

3.2.25
index
address of an object within an application process

3.2.26
isochronous mode
special operation mode of a DP system that implies both a constant DP cycle with a fixed schedule of the cyclic and acyclic DP services, and the synchronisation of the application in the DP-master (Class 1) and the DP-slaves with this constant DP cycle

3.2.27
master parameter set
the configuration and parameterization data of all DP-slaves that are assigned to the corresponding DP-master and the bus parameters

3.2.28
module
addressable unit inside the DP-slave

3.2.29
process data
object(s) which are already pre-processed and transferred acyclically for the purpose of information or further processing

3.2.30
publisher
role of a CR endpoint that transmits APDUs onto the Fieldbus for consumption by one or more subscribers

Note 1 to entry: A publisher may not be aware of the identity or the number of subscribers and it may publish its APDUs using a dedicated CR.

3.2.31
real configuration
input and output data structure of a DP-slave, including definition of data consistency

3.2.32
subscriber
role of a CREP in which it receives APDUs produced by a publisher

3.2.33
slot
address of a module within a DP-slave

3.2.34
sync
function at the DP-slaves for simultaneous data transfer between the output data object and the process

3.3 Abbreviations and symbols

Ack	Acknowledgement
Add	Address
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier
AR	Application Relationship
AREP	Application Relationship End Point
ARL	Application Relationship List
ASE	Application Service Element
Cfg	Configuration
cnf	confirmation primitive
CREP	Communication Relationship End Point
CRL	Communication Relationship List
D_	Destination
DLL	Data Link Layer
DLM	Data Link-management
DLSAP	Data Link Service Access Point
DLSDU	Data Link Service Data Unit
DMPM	Data Link Mapping Protocol Machine
DMPMM1	Data Link Mapping Protocol Machine DP-master (Class 1)
DMPMM2	Data Link Mapping Protocol Machine DP-master (Class 2)
DMPMS	Data Link Mapping Protocol Machine DP-slave

DSAP	Destination Service Access Point
DXB	Data eXchange Broadcast
FAL	Fieldbus Application Layer
FIFO	First In First Out
FSPMM1	FAL Service Protocol Machine DP-master (Class 1)
FSPMM2	FAL Service Protocol Machine DP-master (Class 2)
FSPMS	FAL Service Protocol Machine DP-slave
HSA	Highest Station Address
I&M	Identification and Maintenance Profile
ID	Identifier
IEC	International Electrotechnical Commission
ind	indication primitive
Inp	Input
ISO	International Organization For Standardization
IsoM	Isochronous Mode
L_sdu	Link Service Data Unit
lsb	least significant bit
msb	most significant bit
OSI	Open Systems Interconnection
Outp	Output
Param	Parameter
PDU	Protocol Data Unit
RD_	Read
Rem	Remote
Req	Request (used to indicate an APDU type)
req	request primitive
REQ	Request to indicate a Request PDU
RES	Response to indicate a Response PDU
RM	Resource Manager
RPL_UPD	Reply Update
Rsp	Response (used to indicate an APDU type)
rsp	response primitive
S_	Source
SAP	Service Access Point
SCL	Security Level
SD	Start Delimiter
SDN	Send Data with no Acknowledge
SDR	Station Delay Responder
Seq	Sequence
Src	Source

SRD	Send and Request Data with Reply
SSAP	Source Service Access Point
TR	Technical Report
USIF	User Interface
WD	Watchdog

3.4 Conventions

3.4.1 General concept

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate subclause. Each ASE specification is composed of three parts: its class definitions, its services, and its protocol specification. The first two are contained in IEC 61158-5-3. The protocol specification for each of the ASEs is defined in this document.

The class definitions define the attributes of the classes supported by each ASE. The attributes are accessible from instances of the class using the Management ASE services specified in IEC 61158-5-3. The service specification defines the services that are provided by the ASE.

This document uses the descriptive conventions given in ISO/IEC 10731.

3.4.2 Abstract syntax conventions

PDUs are described as octets or groups of octets.

a) Groups of octets separated by a comma appear in the order they are transferred If optional octets are not present the following octets appear without a gap

b) If octets or groups of octets are grouped within "{ }" the order is arbitrary

c) If octets or groups of octets are marked with "*" they may appear more than once. If it is used within a "{ }" section they may appear mixed with other octets or group of octets of this section.

d) Octets can be grouped or values can be assigned within "( )"

e) If octets or groups of octets are grouped within "[ ]" the group can be omitted

f) Complex APDUs may be built by means of substitutions (sub-structures)

g) Exclusive selections of octets or groups of octets are separated by "^"

NOTE 1 The formal PDU example

AP_PDU=Octet1,OctetGroup1,[Octet2],[Octet3],{[OctGroup2*],OctetGroup3^Octet4}
According to this the following variants are valid on the wire (non exhaustive):

Variant 1: Octet1, OctetGroup1, Octet2, Octet3, OctetGroup2,OctetGroup3

Variant 2: Octet1, OctetGroup1, Octet2, Octet3, OctetGroup2, OctetGroup2, OctetGroup2,OctetGroup3

Variant 3: Octet1, OctetGroup1, OctetGroup2, OctetGroup2, OctetGroup2,OctetGroup3, OctetGroup2

Variant4: Octet1, OctetGroup1, OctetGroup2, OctetGroup3, OctetGroup2, OctetGroup2, OctetGroup2, OctetGroup2

Variant 5: Octet1, OctetGroup1, Octet3, Octet4

NOTE 2 The arbitrary order implies that groups of octets are characterized by a special header that is described within the coding rules.

NOTE 3 The APDU syntax implies that according to the maximum DLSDU a APDU shall not exceed 244 octets in total.

3.4.3 Convention for the encoding of reserved bits and octets

The term "reserved" may be used to describe bits in octets or whole octets. All bits or octets that are reserved should be set to zero at the sending side and shall not be tested at the receiving side except it is explicitly stated or if the reserved bits or octets are checked by a state machine.

The term "reserved" may also be used to indicate that certain values within the range of a parameter are reserved for future extensions. In this case the reserved values should not be used at the sending side and shall not be tested at the receiving side except it is explicitly stated or if the reserved values are check by a state machine.

3.4.4 Conventions for the common coding s of specific field octets

APDUs may contain specific fields that carry information in a primitive and condensed way. These fields shall be coded in the order according to Figure 1.

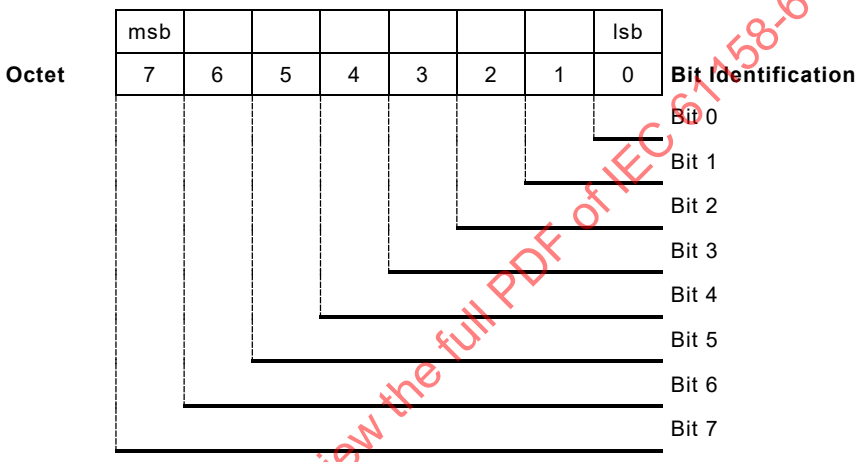


Figure 1 – Common structure of specific fields

– Bits may be grouped as group of bits. Each bit or group of bits shall be addressed by its Bit Identification (e.g. Bit 0, Bit 1 to 4). The position within the octet shall be according to the figure above. Alias names may be used for each bit or group of bits or they may be marked as reserved. The grouping of individual bits shall be in ascending order without gaps. The values for a group of bits may be represented as binary, decimal or hexadecimal values. This value shall only be valid for the grouped bits and can only represent the whole octet if all 8 bits are grouped. Decimal or hexadecimal values shall be transferred in binary values so that the bit with the highest number of the group represents the msb concerning the grouped bits.

EXAMPLE 1: Description and relation for the specific field octet

Bit 0: reserved.

Bit 1-3: Reason_Code The decimal value 2 for the Reason_Code means general error.

Bit 4-7: shall always set to one.

The octet that is constructed according to the description above looks as follows:

(msb) Bit 7 = 1,

Bit 6 = 1,

Bit 5 = 1,

Bit 4 = 1,

Bit 3 = 0,

Bit 2 = 1,

Bit 1 = 0,

(lsb) Bit 0 = 0.

The bit combination "0-1-0" for Bit 1-3 equals the decimal value 2.

3.5 Conventions used in state machines

3.5.1 State machine conventions

The protocol sequences are described by means of State Machines.

In state diagrams states are represented as boxes state transitions are shown as arrows. Names of states and transitions of the state diagram correspond to the names in the textual listing of the state transitions.

The textual listing of the state transitions is structured as follows, see also Table 1.

a) The first row contains the name of the transition.

b) The second row in define the current state.

c) The third row contains an optional event followed by Conditions starting with a "/" as first line character and finally followed by the Actions starting with a "=>" as first line character.

d) The last row contains the next state.

e) If the event occurs and the conditions are fulfilled the transition fires, i.e. the actions are executed and the next state is entered.

The layout of a Machine description is shown in Table 1. The meaning of the elements of a State Machine Description are shown in Table 2.

Table 1 – State machine description elements

#	Current state	Event or condition => action	Next state

Table 2 – Description of state machine elements

Description element	Meaning
Current state Next state	Name of the given states.
#	Name or number of the state transition.
Event	Name or description of the event.
/Condition	Boolean expression. The preceding "\" is not part of the condition.
=> Action	List of assignments and service or function invocations. The preceding "=>" is not part of the action.

The conventions used in the state machines are shown in Table 3.

Table 3 – Conventions used in state machines

Convention	Meaning
:=	Value of an item on the left is replaced by value of an item on the right. If an item on the right is a parameter, it comes from the primitive shown as an input event.
xxx	A parameter name. Example: Identifier:= reason means value of a 'reason' parameter is assigned to a parameter called 'Identifier.'
"xxx"	Indicates fixed value. Example: Identifier:= "abc" means value "abc" is assigned to a parameter named 'Identifier.'
=	A logical condition to indicate an item on the left is equal to an item on the right.
<	A logical condition to indicate an item on the left is less than the item on the right.
>	A logical condition to indicate an item on the left is greater than the item on the right.
<>	A logical condition to indicate an item on the left is not equal to an item on the right.
&&	Logical "AND"
||	Logical "OR"

for (Identifier:= start_value to end_value)
actions
endfor
	This construct allows the execution of a sequence of actions in a loop within one transition. The loop is executed for all values from start_value to end_value.

If (condition)
actions
else
actions
	This construct allows the execution of alternative actions depending on some condition (which might be the value of some identifier or the outcome of a previous action) within one transition. The parts beginning with “else” can be omitted if there is no action if the condition is not fulfilled.

Readers are strongly recommended to refer to the subclauses for the AREP and CREP attribute definitions, the local functions, and the FAL-PDU definitions to understand protocol machines. It is assumed that readers have sufficient knowledge of these definitions and they are used without further explanations.

In addition the following description elements are used:

Wildcard in names

name_XXX: “XXX” is used as wildcard string for all names beginning with “name”.

The typical use of a wildcard is an Event. In this context there are as many state transitions as possible events for this wildcard exists.

Conditional macro

<CONDITIONAL -MACRO-NAME>

<

CONDITION1: macrobody1

CONDITION2: macrobody2

…

>

CONDITION1, CONDITION2, etc. define all possible cases for the conditional macro. The “CONDITIONAL-MACRO-NAME” acts as place holder for the “macrocode” depending on the result of the condition.

Replacement macro

<REPLACEMENT-MACRO-NAME>

<

XXX= Name1: macrobody1

XXX= Name2: macrobody2

…

>

Name1, Name2, etc. define all possible cases for the replacement macro. The “REPLACEMENT-MACRO-NAME” acts as place holder for the “macrocode” depending on the value of the current use of the wildcard XXX.

Example:
<SERVICE_REQ_PARA>
<
XXX=Read: Para1, Para2
XXX=Write: Para1, Para2, Para3
>
when called in
MSAB_XXX.req(<SERVICE_REQ_PARA>)
will result in
MSAB_Read.req(Para1, Para2) or MSAB_Write.req(Para1, Para2, Para3)

4 FAL syntax description

4.1 APDU abstract syntax

Table 4 defines the abstract syntax of the DP Application Layer PDUs referred to as APDUs. The defined order of octets shall be used to convey the APDUs.

Table 4 – APDU syntax

APDU name	APDU structure
DataExchange-REQ-PDU	[Outp_Data*]
DataExchange-RES-PDU	[Inp_Data*]
RD_Input-RES-PDU	[Inp_Data*]
RD_Output-RES-PDU	[Outp_Data*]
Chk_Cfg-REQ-PDU	{ [Identifier_Format*], [Special_Identifier_Format*] ^ [Extended_Special_Identifier_Format*] } The field SI_Flag.DPV1_Data_Types defines whether Special_Identifier_Format or Extended_Special_Identifier_Format applies. The sequence of identifiers shall follow the ascending number of slots/moduls.
Get_Cfg-RES-PDU	{ [Identifier_Format*], [Special_Identifier_Format*] ^ [Extended_Special_Identifier_Format*] }
Set_Prm-REQ-PDU	Station_status, WD_Fact_1, WD_Fact_2, min_TSDR, Ident_Number, Group_Ident, User_Prm_Data
Set_Ext_Prm-REQ-PDU	Structured_Prm_Data*
Diagnosis-RES-PDU	Station_status_1, Station_status_2, Station_status_3, Diag_Master_Add, Ident_Number, { [Device_Related_Diagnosis*] ^ [Alarm], [Status*], [Modul_Status], [DXB_Link_Status], { [Red_Status] ^[PrmCmdAck] }), [Identifier_Related_Diagnosis], [Channel_Related_Diagnosis*], [Revision_Number] } The field Device_Related_Diagnosis may be present if DPV1=FALSE, otherwise if DPV1=TRUE it shall not be present. In this case the fields Alarm, Status, Modul_Status may be present.
Set_Slave_Add-REQ-PDU	New_Slave_Add, Ident_Number, No_Add_Chg, [Rem_Slave_Data*]
Global_Control-REQ-PDU	Control_Command, Group_Select
Clock-Value-PDU	Clock_value_time_event, Clock_value_previous_TE, Clock_value_status1, Clock_value_status2
Initiate-REQ-PDU	Function_Num(0x57), reserved (3 Octets), Send_Timeout, Features_Supported, Profile_Features_Supported, Profile_Ident_Number, Add_Addr_Param
Initiate-RES-PDU	Function_Num(0x57), Max_Len_Data_Unit, Features_Supported, Profile_Features_Supported, Profile_Ident_Number, Add_Addr_Param
Initiate-NRS-PDU	Function_Num(0xD7), Error_Decode, Error_Code_1, Error_Code_2
Abort-REQ-PDU	Function_Num(0x58), Subnet, Instance_Reason_Code
Read-REQ-PDU	Function_Num(0x5E), Slot_Number, Index (0 .. 254), Length
Read-RES-PDU	Function_Num(0x5E), Slot_Number, Index (0 .. 254), Length, [Data*]
Read-NRS-PDU Pull-NRS-PDU	Function_Num(0xDE), Error_Decode, Error_Code_1, Error_Code_2
Write-REQ-PDU	Function_Num(0x5F), Slot_Number, Index (0 .. 254), Length, [Data*]
Write-RES-PDU	Function_Num(0x5F), Slot_Number, Index (0 .. 254), Length
Write-NRS-PDU Initiate_Load_NRS-PDU Push-NRS-PDU Terminate_Load-NRS-PDU Start-NRS-PDU Stop-NRS-PDU Resume-NRS-PDU Reset-NRS-PDU Call-NRS-PDU Get_FI_State-NRS-PDU	Function_Num(0xDF), Error_Decode, Error_Code_1, Error_Code_2
Alarm_Ack-REQ-PDU	Function_Num(0x5C), Slot_Number, Alarm_Type, Alarm_Specifier
Alarm_Ack-RES-PDU	Function_Num(0x5C), Slot_Number, Alarm_Type, Alarm_Specifier
Alarm_Ack-NRS-PDU	Function_Num(0xDC), Error_Decode, Error_Code_1, Error_Code_2
Idle-REQ-PDU	Function_Num(0x48)

APDU name	APDU structure
Idle-RES-PDU	Function_Num(0x48)
Data_Transport-REQ-PDU	Function_Num(0x51), Slot_Number, Index, Length, [Data*]
Data_Transport-RES-PDU	Function_Num(0x51), Slot_Number, Index, Length, [Data*]
Data_Transport-NRS-PDU	Function_Num(0xD1), Error_Decode, Error_Code_1, Error_Code_2
Initiate_Load-REQ-PDU	Function_Num(0x5F), Slot_Number, Index (255), Length, Extended_Function_Num (0x00), Options, LR_Index, LR_Length, Intersegment_Request_Timeout, [User_Specific]
Initiate_Load-RES-PDU	Function_Num(0x5E), Slot_Number, Index (255), Length, Extended_Function_Num (0x00), Max_Segment_Length, LR_Index, LR_Length, Max_Response_Delay
Push-REQ-PDU	Function_Num(0x5F), Slot_Number, Index (255), Length, Extended_Function_Num (0x01), Options, Sequence_Number, LR_Data
Pull-REQ-PDU	Function_Num(0x5E), Slot_Number, Index (255), Length
Pull-RES-PDU	Function_Num(0x5E), Slot_Number, Index (255), Length, Extended_Function_Num (0x02), Options, Sequence_Number, LR_Data
Terminate_Load-REQ-PDU	Function_Num(0x5F), Slot_Number, Index (255), Length, Extended_Function_Num (0x03), reserved (1 octet), reserved (2 Octets),
Start-REQ-PDU	Function_Num(0x5F), Slot_Number, Index (255), Length, Extended_Function_Num (0x04), reserved (1 octet), FI_Index, [Execution_Argument]
Stop-REQ-PDU	Function_Num(0x5F), Slot_Number, Index (255), Length, Extended_Function_Num (0x05), reserved (1 octet), FI_Index
Resume-REQ-PDU	Function_Num(0x5F), Slot_Number, Index (255), Length, Extended_Function_Num (0x06), reserved (1 octet), FI_Index, [Execution_Argument]
Reset-REQ-PDU	Function_Num(0x5F), Slot_Number, Index (255), Length, Extended_Function_Num (0x07), reserved (1 octet), FI_Index
Call-REQ-PDU	Function_Num(0x5F), Slot_Number, Index (255), Length, Extended_Function_Num (0x08), Entity Number, FI_Index (=0…64999), [Execution_Argument] Function_Num(0x5F), Slot_Number, Index (255), Length, Extended_Function_Num (0x08), Entity Number, FI_Index (=65000..65199), [IMData_Execution_Argument]
Call-RES-PDU	Function_Num(0x5E), Slot_Number, Index (255), Length, Extended_Function_Num (0x08), Entity Number, FI_Index (=0…64999), [Result_Argument] Function_Num(0x5E), Slot_Number, Index (255), Length, Extended_Function_Num (0x08), Entity Number, FI_Index (=65000…65199), [IMData_Result_Argument]
Get_FI_State-REQ-PDU	Function_Num(0x5F), Slot_Number, Index (255), Length, Extended_Function_Num (0x09), reserved (1 octet), FI_Index
Get_FI_State-RES-PDU	Function_Num(0x5E), Slot_Number, Index (255), Length, Extended_Function_Num (0x09), reserved (1 octet), FI_Index, FI_State
Push-RES-PDU Terminate_Load-RES-PDU Start-RES-PDU Stop-RES-PDU Resume-RES-PDU Reset-RES-PDU	Function_Num(0x5F), Slot_Number, Index (255), Length
RM-REQ-PDU	Function_Num(0x56), Server_SAP, Send_Timeout
Get_Master_Diag-REQ-PDU	Function_Num(0x41), MDiag_Identifier
Get_Master_Diag-RES-PDU	Function_Num(0x41), MDiag_Identifier(=0..125), Diagnosis-RES-PDU Function_Num(0x41), MDiag_Identifier(=126), System_Diagnosis Function_Num(0x41), MDiag_Identifier(=127), Master_Status Function_Num(0x41), MDiag_Identifier(=128), Data_Transfer_List
Start_Seq-REQ-PDU	Function_Num(0x42), Area_CodeUpDownload, Timeout
Start_Seq-RES-PDU	Function_Num(0x42), Area_CodeUpDownload, Timeout, Max_Len_Data_Unit

APDU name	APDU structure
Download-REQ-PDU	Function_Num(0x43), Area_CodeUpDownload(=0..125), Add_Offset, DP_Slave_Parameter_Set Function_Num(0x43), Area_CodeUpDownload(=127), Add_Offset, Bus_Parameter_Set Function_Num(0x43), Area_CodeUpDownload(=129), Add_Offset, Statistic_Counters Function_Num(0x43), Area_CodeUpDownload(=136 to 139), Add_Offset, Master_Parameter_Set
Download-RES-PDU	Function_Num(0x43), Area_CodeUpDownload, Add_Offset
Upload-REQ-PDU	Function_Num(0x44), Area_CodeUpDownload, Add_Offset, Data_Len
Upload-RES-PDU	Function_Num(0x44), Area_CodeUpDownload(=0..125), Add_Offset, DP_Slave_Parameter_Set Function_Num(0x44), Area_CodeUpDownload(=127), Add_Offset, Bus_Parameter_Set Function_Num(0x44), Area_CodeUpDownload(=129), Add_Offset, Statistic_Counters Function_Num(0x43), Area_CodeUpDownload(=136 to 139), Add_Offset, Master_Parameter_Set
End_Seq-REQ-PDU	Function_Num(0x45)
End_Seq-RES-PDU	Function_Num(0x45)
Act_Para_Brct-REQ-PDU	Function_Num(0x46), Area_CodeActBrct
Act_Param-REQ-PDU	Function_Num(0x47), Area_CodeAct, Activate
Act_Param-RES-PDU	Function_Num(0x47), Area_CodeAct, Activate
ZERO-PDU	Octet1(0)
NULL-PDU	

NOTE 1 If a APDU consists only of user data octets the distinction is made by use of different Data Link Layer service access points (see Protocol Machines).

NOTE 2 A ZERO-PDU contains one octet with all bits set to zero. This APDU is used to indicate diagnosis from a device without inputs.

NOTE 3 A NULL-PDU contains no octets. It is used to submit no DLSDU to data link layer.

Table 5 defines structures for substitutions of elements of the APDU structures shown in Table 4.

Table 5 – Substitutions

Substitution name	Structure
Identifier_Format	Cfg_Identifier
Special_Identifier_Format	Special_Cfg_Identifier, [Length_Octet] , [Length_Octet], [Manufacturer_Specific_Data*] Length_Octet fields shall always start with fields that indicate output data if present.
Extended_Special_Identifier_Format	Special_Cfg_Identifier, [Extended_Length_Octet], [Extended_Length_Octet], [Data_Type*], [Manufacturer_Specific_Data*] Extended_Length_Octet fields shall always start with fields that indicate output data if present. The field Data_Type shall only be omitted in case of an empty slot. Otherwise it shall be present in relation to the length fields.
Device_Related_Diagnosis	Header_Octet, Diagnosis_User_Data*
Identifier_Related_Diagnosis	Header_Octet, Identifier_Diagnosis_Data_Array
Channel_Related_Diagnosis	Identifier_Number, Channel_Number, Type_of_Diagnosis
Alarm	Header_Octet, Alarm_Type, Slot_Number, Alarm_Specifier, [Diagnosis_User_Data*]

Substitution name	Structure
Status	Header_Octet, Status_Type, Slot_Number, Status_Specifier, [Diagnosis_User_Data*]
Modul_Status	Header_Octet, Status_Type(=Modul_Status), Slot_Number(=0), Status_Specifier, Modul_Status_Array
DXB_Link_Status	Header_Octet, Status_Type(=DXB_Link_Status), Slot_Number(=0), Status_Specifier, (Publisher_Address, Publisher_Status)*
PrmCmdAck	Header_Octet, Status_Type(=PrmCmdAck), Slot_Number(=0), Status_Specifier, Function, Red_Status1, Red_Status2, Red_Status3
Red_Status	Header_Octet, Status_Type(=Red_Status), Slot_Number(=0), Status_Specifier, Function, Red_Status1, Red_Status2, Red_Status3
User_Prm_Data	[DPV1_Status_1, DPV1_Status_2, DPV1_Status_3], [Structured_Prm_Data*] ^ [User_Prm_Data_Element*] Special case DPV1=FALSE: [User_Prm_Data_Element*] Special case DPV1=TRUE and DPV1_Status_3.Prm_Structure=FALSE: DPV1_Status_1, DPV1_Status_2, DPV1_Status_3, [User_Prm_Data_Element*] Special case DPV1=TRUE and DPV1_Status_3.Prm_Structure=TRUE: DPV1_Status_1, DPV1_Status_2, DPV1_Status_3, Structured_Prm_Data* The fields DPV1_Status_1, DPV1_Status_2 and DPV1_Status_3 shall be present if DPV1=TRUE. Structured_Prm_Data shall be present if the DPV1_Status_3.Prm_Structure is TRUE, otherwise it shall not be used.
Structured_Prm_Data	Structure_Length, Structure_Type, Slot_Number, reserved (1 octet), User_Prm_Data_Element* Special Case PrmCmd: Structure_Length, Structure_Type(=2), Slot_Number(=0), Specifier, Function, Properties, Output_Hold_Time Special Case DXB LinkTable: Structure_Length, Structure_Type(=3), Slot_Number(=0), reserved (1 octet), Version(=1), (Publisher_Addr, Publisher_Length, Sample_Offset, Sample_Length)* Special Case IsoM Parameter: Structure_Length, Structure_Type(=4), Slot_Number(=0), reserved (1 octet), Version(=1), TBASE_DP, TDP, TMAPC, TBASE_IO, TI, TO, TDX, TPLL_W, TPLL_D Special Case F-Parameter: Structure_Length, Structure_Type(=5), Slot_Number, reserved (1 octet), User_Prm_Data* Special Case DXB-Subscribertable: Structure_Length, Structure_Type(=7), Slot_Number(=0), reserved (1 octet), Version(=1), (Publisher_Addr, Publisher_Length, Sample_Offset, Dest_Slot_Number, Offset_Data_Area, Sample_Length)* Special Case Time AR Parameter: Structure_Length, Structure_Type(=8), Slot_Number(=0), reserved (1 octet), Clock Sync Interval, [CS Delay Time]
Features_Supported	Features_Supported_1, Features_Supported_2
Profile_Features_Supported	Profile_Features_Supported_1, Profile_Features_Supported_2
Add_Addr_Param	S_Type, S_Len, D_Type, D_Len, S_API, S_SCL, [S_Network_Address], [S_MAC_Address], D_API, D_SCL, [D_Network_Address], [D_MAC_Address]
Master_Status	USIF_State, Ident_Number, Hardware_Release_DP, Firmware_Release_DP, Hardware_Release_User, Firmware_Release_User, reserved (9 Octets)
DP_Slave_Parameter_Set	Slave_Para_Len, SI_Flag, Slave_Type, Max_Diag_Data_Len, Max_Alarm_Len, Max_Channel_Data_Length, Diag_Upd_Delay, Alarm_Mode, Add_SI_Flag, MS1_Timeout, reserved (4 Octets), Prm_Data_Len, Prm_Data, Cfg_Data_Len, Cfg_Data, Add_Tab_Len, Add_Tab, Slave_User_Data_Len, Slave_User_Data, Ext_Prm_Data_Len, [Ext_Prm_Data]
Bus_Parameter_Set	Bus_Para_Len, DL_Add, Data_rate, T_{SL}, min T_{SDR}, max T_{SDR}, T_{QUI}, T_{SET}, T_{TR}, G, HSA, max_retry_limit, Bp_Flag, Min_Slave_Interval, Poll_Timeout, Data_Control_Time, Alarm_Max, Max_User_Global_Control, reserved (4 Octets), Master_User_Data_Len, Master_Class2_Name, Master_User_Data*, T_{CT}, maxT_{SH}

Substitution name	Structure
Master_Parameter_Set	Bus_Parameter_Set, DP_Slave_Parameter_Set*
Statistic_Counters	Counter_Group*
Counter_Group	if Add_Offset 0..126: DLPDU_sent_count(Add_Offset), Error_count(Add_Offset) if Add_Offset 127: SD_sent_count, SD_error_count
Add_Tab	[Number_of_Entries], [Add_Tab_Entry_Header, I/O_Data_Length, IO_Config_Address, Host_Address]*
IMData_Execution_Argument	[I&M1] ^ [I&M2] ^ [I&M3] ^ [I&M4] ^ [I&M_Profile_Specific_x] ^ [I&M_Manufacturer_Specific_x]
IMData_Result_Argument	I&M0 ^ [I&M1] ^ [I&M2] ^ [I&M3] ^ [I&M4] ^ [I&M_Profile_Specific_x] ^ [I&M_Manufacturer_Specific_x]
I&M0	IM_Header, IM_Manufacturer_ID, OrderID, IM_Serial_Number, IM_Hardware_Revision, IM_Software_Revision, IM_Revision_Counter, IM_Profile_ID, IM_Profile_Specific_Type, IM_Version, IM_Supported
I&M1	IM_Header, IM_Tag_Function, IM_Tag_Location
I&M2	IM_Header, IM_Date
I&M3	IM_Header, IM_Descriptor
I&M4	IM_Header, IM_Signature
I&M_Profile_Specific_x	IM_Header, IM_Profile_Specific_Data
IM_Version	IM_Version_Major, IM_Version_Minor
IM_Software_Revision	SWRevisionPrefix, IM_SWRevision_Functional_Enhancement, IM_SWRevision_Bug_Fix, IM_SWRevision_Internal_Change

NOTE The DP_Slave_Parameter_Set and the Bus_Parameter_Set may exceed the maximum APDU size to a size up to 64 Koctets. The conveyance of the data is therefore segmented by means of a window addressed with the parameter Add_Offset.

4.2 Data types

4.2.1 Notation for the Boolean type

Boolean::= BOOLEAN -- TRUE if the value is non-zero.

-- FALSE if the value is zero.

4.2.2 Notation for the Integer type

Integer8::= INTEGER (-128..+127) -- range $-2^7 <= I <= 2^7-1$

Integer16::= INTEGER (-32768..+32767) -- range $-2^{15} <= I <= 2^{15}-1$

Integer32::= INTEGER -- range $-2^{31} <= I <= 2^{31}-1$

Integer64::= INTEGER -- range $-2^{63} <= I <= 2^{63}-1$

4.2.3 Notation for the Unsigned type

Unsigned8::= INTEGER (0..255) -- range $0 <= I <= 2^8-1$

Unsigned16::= INTEGER (0..65535) -- range $0 <= I <= 2^{16}-1$

Unsigned32::= INTEGER -- range $0 <= I <= 2^{32}-1$

Unsigned64::= INTEGER -- range $0 <= I <= 2^{64}-1$

4.2.4 Notation for the Floating Point type

Floating32::= BIT STRING SIZE (4) -- IEEE 754 Single precision

Floating64::= BIT STRING SIZE (8) -- IEEE 754 Double precision

4.2.5 Notation for the OctetString type

OctetString::= OCTET STRING -- For generic use

4.2.6 Notation for VisibleString type

VisibleString2::= VISIBLE STRING -- For generic use

4.2.7 Notation for BinaryDate type

BinaryDate::= OctetString7

4.2.8 Notation for TimeOfDay type

TimeOfDay with date indication::= OctetString6

TimeOfDay without date indication::= OctetString4

4.2.9 Notation for TimeDifference type

TimeDifference with date indication::= OctetString6

TimeDifference without date indication::= OctetString4

4.2.10 Notation for Network Time type

Network Time::= OctetString8

4.2.11 Notation for Network Time Difference type

Network Time Difference::= OctetString8

5 Transfer syntax

5.1 Coding of basic data types

5.1.1 Encoding of a Boolean value

a) The encoding of a Boolean value shall be primitive. The ContentsOctets shall consist of a single octet.

b) If the Boolean value is FALSE, the ContentsOctets shall be 0 (zero). If the Boolean value is TRUE, the ContentsOctets shall be 0xff.

5.1.2 Encoding of an Integer value

a) The encoding of a fixed-length Integer value of Integer8, Integer16, and Integer32 types shall be primitive, and the ContentsOctets shall consist of exactly one, two, or four octets, respectively.

b) The ContentsOctets shall be a two's complement binary number equal to the integer value, and consist of bits 7 to 0 of the first octet, followed by bits 7 to 0 of the second octet, followed by bits 7 to 0 of each octet in turn up to and including the last octet of the ContentsOctets.

NOTE The value of a two's complement binary number is derived by numbering the bits in the ContentsOctets, starting with bit 0 of the last octet as bit zero and ending the numbering with bit 7 of the first octet. Each bit is assigned a numerical value of 2^N, where N is its position in the above numbering sequence. The value of the two's complement binary number is obtained by adding the numerical values assigned to each bit for those bits which are set to one, excluding bit 7 of the first octet, and then reducing this value by the numerical value assigned to bit 7 of the first octet if that bit is set to one.

5.1.3 Encoding of an Unsigned value

a) The encoding of a fixed-length Unsigned value of Unsigned8, Unsigned16, and Unsigned32 types shall be primitive, and the ContentsOctets shall consist of exactly one, two, or four octets, respectively.

b) The ContentsOctets shall be a binary number equal to the Unsigned value, and consist of bits 7 to 0 of the first octet, followed by bits 7 to 0 of the second octet, followed by bits 7 to 0 of each octet in turn up to and including the last octet of the ContentsOctets.

NOTE The value of a binary number is derived by numbering the bits in the ContentsOctets, starting with bit 0 of the last octet as bit zero and ending the numbering with bit 7 of the first octet. Each bit is assigned a numerical value of 2^N, where N is its position in the above numbering sequence. The value of the binary number is obtained by adding the numerical values assigned to each bit for those bits which are set to one.

5.1.4 Encoding of a Floating-Point value

a) The encoding of a Floating32 value shall be primitive, and the ContentsOctets shall consist of exactly four octets.

b) The ContentsOctets shall contain floating-point values defined in conformance with IEEE 754. The sign is encoded in bit 7 of the first octet. It is followed by the exponent starting from bit 6 of the first octet, and then the mantissa starting from bit 6 of the second octet for Floating32.

5.1.5 Encoding of a Visible String value

a) The encoding of a variable length VisibleString value shall be primitive.

b) There is no Length field; the length is encoded implicitly.

c) The ContentsOctets shall be a sequence of octets. The leftmost string element is encoded in the first octet, followed by the second octet, followed by each octet in turn up to and including the last octet as rightmost of the ContentsOctets.

5.1.6 Encoding of an Octet String value

a) The encoding of a variable length OctetString value shall be primitive.

b) There is no Length field; the length is encoded implicitly.

c) The ContentsOctets shall be a sequence of octets. The leftmost string element is encoded in the first octet, followed by second octet, followed by each octet in turn up to and including the last octet as rightmost of the ContentsOctets.

5.1.7 Encoding of a BinaryDate value

Coding of BinaryDate within IEC 61158-6-10 applies.

5.1.8 Encoding of a TimeOfDay with and without date indication value

Coding of TimeOfDay with and without date indication within IEC 61158-6-10 applies.

5.1.9 Encoding of a Time Difference with and without date indication value

Coding of Time Difference with and without date indication within IEC 61158-6-10 applies.

5.1.10 Encoding of a Network Time value

Coding of Network Time within IEC 61158-6-10 applies.

5.1.11 Encoding of a Network Time Difference value

Coding of Network Time Difference within IEC 61158-6-10 applies.

5.1.12 Encoding of a Null value

a) The encoding of a NULL value shall be primitive.

b) The ContentsOctet shall be omitted.

5.2 Coding section related to data exchange PDUs

5.2.1 General

IEC 61158-5-10, Clause 5 applies.

NOTE Data types with variable length (Visible String, Octet String, TimeOfDay, Time Difference) for Inp_Data or Outp_Data are only once present in the DataExchange-RES-PDU or DataExchange-REQ-PDU. The user data correspond to the contents of the Configuration-PDU.

5.2.2 Coding of the field Outp_Data

E.g. one of the following data types shall be used for each Outp_Data field:

Boolean, Integer, Unsigned, Floating Point, Visible String, Octet String, Date, TimeOfDay, Time-Difference.

5.2.3 Coding of the field Inp_Data

E.g. one of the following data types shall be used for each Inp_Data field:

Boolean, Integer, Unsigned, Floating Point, Visible String, Octet String, Date, TimeOfDay, Time-Difference.

5.3 Coding section related to slave diagnosis PDUs

5.3.1 Coding of the field Station_status_1

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: Diag.Station_Non_Existent
Shall be set to zero in case of Diagnosis-RES-PDU

Value (0): means FALSE if Get_Master_Diag-RES-PDU

Value (1): means TRUE if Get_Master_Diag-RES-PDU

Bit 1: Diag.Station_Not_Ready
FALSE (0)

TRUE (1)

Bit 2: Diag.Cfg_Fault (Configuration Fault)
FALSE (0)

TRUE (1)

Bit 3: Diag.Ext_Diag (Extended Diagnosis)
FALSE (0): means that the device is NOT in state diagnosis

TRUE (1): means that the device is in state diagnosis

Flag should be set to FALSE if no faulty conditions exist. If set to TRUE, the reason should be reported as Device Related Diagnosis, Identifier Related Diagnosis, or Channel Related Diagnosis.

Bit 4: Diag.Not_Supported
FALSE (0)

TRUE (1)

Bit 5: Diag.Invalid_Slave_Response
Shall be set to zero if Diagnosis-RES-PDU

Value (0): means FALSE if Get_Master_Diag-RES-PDU

Value (1): means TRUE if Get_Master_Diag-RES-PDU

Bit 6: Diag.Prm_Fault (Parameter Fault)
FALSE (0), TRUE (1)

Bit 7: Diag.Master_Lock
Shall be set to zero if Diagnosis-RES-PDU

Value (0): means FALSE if Get_Master_Diag-RES-PDU

Value (1): means TRUE if Get_Master_Diag-RES-PDU

5.3.2 Coding of the field Station_status_2

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: Diag.Prm_Req (Parameterization Requested)
FALSE (0)

TRUE (1)

Bit 1: Diag.Stat_Diag (Static Diagnosis)
FALSE (0)

TRUE (1)

Bit 2: DP (DP Protocol)
Shall always set to ones

Bit 3: Diag.WD_On (Watchdog on)
FALSE (0)

TRUE (1)

Bit 4: Diag.Freeze_Mode
FALSE (0)

TRUE (1)

Bit 5: Diag.Sync_Mode (Synchronisation Mode)
FALSE (0)

TRUE (1)

Bit 6: reserved

Bit 7: Diag.Deactivated
Shall be set to zero if Diagnosis-RES-PDU

Value (0): means FALSE if Get_Master_Diag-RES-PDU

Value (1): means TRUE if Get_Master_Diag-RES-PDU

5.3.3 Coding of the field Station_status_3

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 6: reserved

Bit 7: Diag.Ext_Diag_Overflow (Overflow of extended Diagnosis)
FALSE (0)

TRUE (1)

5.3.4 Coding of the field Diag_Master_Add

This field shall be coded as data type Unsigned8 with the following values:

Decimal (0-125)
means the address of the DP-master (Class 1) which has parameterized this DP-slave

Decimal (255)
means DP-slave has not been parameterized or has got invalid parameterisation.

Decimal (126-254)
not allowed

5.3.5 Coding of the field Ident_Number

This field shall be coded as data type Unsigned16.

5.3.6 Coding of the field Header_Octet

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 5: Block_Length
It shall contain the length of the diagnosis block according to the values of Table 6, Table 7, Table 8, and Table 9. The Header_Octet itself shall always be counted.

Table 6 – Block_Length for Selection:= 0

Value (decimal)	Meaning
2 to 63	Device Related Diagnosis in base diagnosis format
4 to 63	Device Related Diagnosis in DPV1 diagnosis format

Table 7 – Block_Length for Selection:= 1

Value (decimal)	Meaning
2 to 32	Identifier Related Diagnosis

NOTE The Identifier Related Diagnosis is limited to 32 because the identifiers itself are limited to 244.

Table 8 – Block_Length for Selection:= 2

Value (decimal)	Meaning
0 to 63	Content of the field Header_Octet.Block_Length shall be used as Identifier_Number See 5.3.15

NOTE One Channel Related Diagnosis entry contains three octets.

Table 9 – Block_Length for Selection:= 3

Value (decimal)	Meaning
1 to 63	Content of the field Header_Octet.Block_Length shall be used as Revision_Number See 5.3.18

NOTE One Revision_Number entry contains one octet.

Bit 6 to 7: Selection
The Selection shall contain the values according to Table 10.

Table 10 – Selection range

Value (decimal)	Meaning
0	Device Related Diagnosis
1	Identifier Related Diagnosis
2	Channel Related Diagnosis
3	Revision_Number

5.3.7 Coding of the field Alarm_Type

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 6: Alarm_Type
The Alarm_Type shall contain the values according to Table 11.

Table 11 – Alarm_Type range

Value (decimal)	Meaning
0	reserved
1	Diagnostic_Alarm
2	Process_Alarm
3	Pull_Alarm
4	Plug_Alarm
5	Status_Alarm
6	Update_Alarm
7 – 31	reserved
32 – 126	manufacturer-specific
127	reserved

Bit 7: shall be zero (identifying Alarm)
Decimal (0): means Alarm

5.3.8 Coding of the field Status_Type

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 6: Status_Type
The Status_Type shall contain values according to Table 12.

Table 12 – Status_Type value range

Value (decimal)	Meaning
0	reserved
1	Status_Message
2	Modul_Status
3	DXB_Link_Status
4 to 29	reserved
30	PrmCmdAck
31	Red_Status
32 to 126	Manufacturer-specific
127	reserved

Bit 7: shall be one (identifying Status)
Decimal (1): means Status

5.3.9 Coding of the field Slot_Number

This field shall be coded as data type Unsigned8. The value shall be taken from the range 0 to 254. The value 255 is reserved.

5.3.10 Coding of the field Alarm_Specifier

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 1: Alarm_Specifier
This bit group shall contain the alarm specifier with the values according to Table 13.

Table 13 – Alarm_Specifier

Value (decimal)	Meaning	Comment
0	No further differentiation	—
1	Error appears and Slot disturbed	the slot generates an alarm due to an error
2	Error disappears and Slot is okay	the slot generates an alarm and indicates that the slot has no further errors
3	Error disappears and Slot is still disturbed	the slot generates an alarm and indicates that the slot has still further errors

Bit 2: Additional_Acknowledge
This bit group shall contain the Additional_Acknowledge with the values according to Table 14.

Table 14 – Additional_Acknowledge

Value (decimal)	Meaning	Comment
0	No additional acknowledge used	—
1	Additional acknowledge is used	—

Bit 3 to 7: Sequence_Number
The value range shall be from 0 to 31 (decimal).

5.3.11 Coding of the field Status_Specifier

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 1: Status_Specifier
This bit group shall contain the status specifier with the values according to Table 15.

Table 15 – Status_Specifier

Value (decimal)	Meaning	Comment
0	No further differentiation	—
1	Status appears	—
2	Status disappears	—
3	Reserved	—

Bit 2 to 7: reserved

5.3.12 Coding of the field Diagnosis_User_Data

One of the following data types shall be used for each Diagnosis_User_Data field:

Boolean, Integer, Unsigned, Floating Point, Visible String, Octet String, Date, TimeOfDay, Time-Difference.

5.3.13 Coding of the field Modul_Status_Array

The field Modul_Status_Array is an array of octets that shall contain at least one and at most 61 octets referred to as Modul_Status_Octet. A Modul_Status_Octet shall consist of at least one and at most 4 module stati referred to as Modul_Status_Entry_1 to Modul_Status_Entry_4 coded in two bits each. Therefore, the number of Modul_Status_Octet corresponds to the number of configured modules within a device and shall be calculated as follows

a) N = 1 if $M_{highest} \leq 4$
b) N = $M_{highest}$ DIV 4 +1 if $M_{highest}$ MOD 4 ≠ 0
c) N = $M_{highest}$ DIV 4 if $M_{highest}$ MOD 4 = 0

where

N is the number of octets Modul_Status_Octet or the number of array elements,

$M_{highest}$ is the highest number of configured modules within a device (maximum 244).

The last Modul_Status_Octet (octet N) may not contain all 4 module status entries. If $M_{highest}$ MOD 4 ≠0 the octet N is not fully filled and the remaining bits shall be set to zero referred to as Padding Bits.

The status of the device's modules shall be structured in ascending order without gaps. The status of module number one is always placed in Modul_Status_Entry_1 of the Modul_Status_Octet number one. The coding of a Modul_Status_Octet shall be according to 3.4 and the individual bits shall have the following meaning:

NOTE The term "configured modules" addresses physical or virtual modules of a device that have had a related format field in the Chk_Cfg-REQ-PDU. Modules that are not part of the configuration are purely local and therefore not related to the Modul_Status_Array field.

Bit 0 to 1: Modul_Status_Entry_1
It shall contain the module status of a configured module with the number that meets m MOD 4 = 1. Padding bits shall not be used here.

Bit 2 to 3: Modul_Status_Entry_2
It shall contain the module status of a configured module with the number that meets m MOD 4 = 2. Otherwise padding bits (00 binary) shall be used.

Bit 4 to 5: Modul_Status_Entry_3
It shall contain the module status of a configured module with the number that meets m MOD 4 = 3. Otherwise padding bits (00 binary) shall be used.

Bit 6 to 7: Modul_Status_Entry_4
These 2 bits shall contain the module status of a configured module with the number that meets m MOD 4 = 0. Otherwise padding bits (00 binary) shall be used.

where

m is the number of the current module with 1 ≤ m ≤ N.

Figure 2 illustrates the arrangement of Modul_Status_Entry_(1 to 4) fields as an example for a device with 6 configured modules.

	Bit Number							
	7	6	5	4	3	2	1	0
Modul_ Status_ Octet 1	status module number 4		status module number 3		status module number 2		status module number 1	
Modul_ Status_ Octet 2	Padding Bits		Padding Bits		status module number 6		status module number 5	
	Modul_Status_Entry_4		Modul_Status_Entry_3		Modul_Status_Entry_2		Modul_Status_Entry_1	

Figure 2 – Example Modul_Status_Array

Each entry Modul_Status_Entry_1 to Modul_Status_Entry_4 shall contain the module status according to Table 16.

Table 16 – Range of Modul_Status_Entry (1-4)

Value (decimal)	Meaning
0	data valid
1	data invalid: the data of the corresponding module are not valid due to an error (e.g. short circuit)
2	data invalid/wrong module: the data of the corresponding module are not valid, due to a wrong module in place
3	data invalid/no module: the data of the corresponding module are not valid, because there is no module in place

5.3.14 Coding of the field Identifier_Diagnosis_Data_Array

The field Identifier_Diagnosis_Data_Array is an array of octets that shall contain at least one and at most 31 octets referred to as Identifier_Diagnosis_Data_Octet. A Identifier_Diagnosis_Data_Octet shall consist of at least one and at most 8 diagnosis entries referred to as Identifier_Diagnosis_Entry_1 to Identifier_Diagnosis_Entry_8. Therefore, the number of Identifier_Diagnosis_Data_Octet corresponds to the number of configured modules within a device and shall be calculated as follows:

a) $N = 1$ if $M_{highest} \leq 8$

b) $N = M_{highest}$ DIV 8 +1 if $M_{highest}$ MOD 8 $\neq 0$

c) $N = M_{highest}$ DIV 8 if $M_{highest}$ MOD 8 = 0

where

N is the number of octets Identifier_Diagnosis_Data_Octet or the number of array elements,

$M_{highest}$ is the highest number of configured modules within a device (maximum 244).

The last Identifier_Diagnosis_Data_Octet (octet N) may not contain all 8 diagnosis entries. If $M_{highest}$ MOD 8 $\neq 0$ the octet N is not fully filled and the remaining bits shall be set to zero referred to as Padding Bits.

NOTE The term DIV stands for division without rest. The term MOD stands for the rest of the division.

The identifier diagnosis of the device's modules shall be structured in ascending order without gaps. The identifier diagnosis of module number one is always placed in Identifier_Diagnosis_Entry_1 of the Identifier_Diagnosis_Data_Octet number one. The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: Identifier_Diagnosis_Entry_1
This bit shall be set to one if the module with the number that meets m MOD 8 = 1 indicates diagnosis otherwise it shall be set to zero. A padding bit shall not be used here.

Bit 1: Identifier_Diagnosis_Entry_2
This bit shall be set to one if the module with the number that meets m MOD 8 = 2 indicates diagnosis otherwise it shall be set to zero. A padding bit shall be used if there is no module configured.

Bit 2: Identifier_Diagnosis_Entry_3
This bit shall be set to one if the module with the number that meets m MOD 8 = 3 indicates diagnosis otherwise it shall be set to zero. A padding bit shall be used if there is no module configured.

Bit 3: Identifier_Diagnosis_Entry_4
This bit shall be set to one if the module with the number that meets m MOD 8 = 4 indicates diagnosis otherwise it shall be set to zero. A padding bit shall be used if there is no module configured.

Bit 4: Identifier_Diagnosis_Entry _5
This bit shall be set to one if the module with the number that meets m MOD 8 = 5 indicates diagnosis otherwise it shall be set to zero. A padding bit shall be used if there is no module configured.

Bit 5: Identifier_Diagnosis_Entry _6
This bit shall be set to one if the module with the number that meets m MOD 8 = 6 indicates diagnosis otherwise it shall be set to zero. A padding bit shall be used if there is no module configured.

Bit 6: Identifier_Diagnosis_Entry _7
This bit shall be set to one if the module with the number that meets m MOD 8 = 7 indicates diagnosis otherwise it shall be set to zero. A padding bit shall be used if there is no module configured.

Bit 7: Identifier_Diagnosis_Entry_8
This bit shall be set to one if the module with the number that meets m MOD 8 = 0 indicates diagnosis otherwise it shall be set to zero. A padding bit shall be used if there is no module configured.

where m is the number of the current module with $1 \leq m \leq N$.

5.3.15 Coding of the field Identifier_Number

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 5: Identifier_Number
The values 0 to 63 (decimal) are valid.

Bit 6 to 7: Selection
The Selection shall contain the value Channel Related Diagnosis according to Table 10.

5.3.16 Coding of the field Channel_Number

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 5: Channel_Number
The values 0 to 63 (decimal) are valid.

Bit 6 to 7: Input_Output_Selection
The Input_Output_Selection shall be set as shown in Table 17.

Table 17 – Input_Output_Selection

Value (decimal)	Meaning
0	Reserved
1	Input.
2	Output.
3	Input and output

5.3.17 Coding of the field Type_of_Diagnosis

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 4: Error_Type
The Error_Type shall be set as shown in Table 18.

Table 18 – Error type

Value (decimal)	Meaning
0 to 31	Coding of the field ChannelErrorType within IEC 61158-6-10 applies.

Bit 5 to 7: Channel_Type
The Channel_Type shall be set as shown in Table 19.

Table 19 – Channel_Type

Value (decimal)	Meaning
0	Unspecific, may be used for any type
1	1 bit
2	2 bit.
3	4 bit
4	Octet
5	Word
6	2 words
7	Reserved

5.3.18 Coding of the field Revision_Number

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 5: Revision_Number
The values 1 to 63 (decimal) are valid.

Bit 6 to 7: Selection
The Selection shall contain the value Revision_Number according to Table 10.

5.3.19 Coding of the field Publisher_Address

This field shall be coded as data type Unsigned8. The range of values shall be 0 to 125. This parameter shall contain the address of the DP-slave acting as Publisher.

5.3.20 Coding of the field Publisher_Status

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 7: Link_Status
FALSE (0): error or not active during the last monitoring period

TRUE (1) means no error and active

Bit 6: Link_Error
FALSE (0): means no error

TRUE (1): means length error

Bit 0 to 5: reserved

5.3.21 Coding of the field RedSpecifier

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 1: Status_Specifier
Decimal (0): means no further differentiation

Decimal (1, 2 and 3): reserved

Bit 3 to 7: Seq Number
Contains Sequence Number of the last PrmCmd processed

5.3.22 Coding of the field Function

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: reserved

Bit 1: Primary
FALSE (0) means no Primary request was executed last

TRUE (1) means a Primary request was executed last

Bit 2: Start MS1
FALSE (0) means no Start MS1 request was executed last

TRUE (1) means a Start MS1 request was executed last

Bit 3: Stop MS1
FALSE (0) means no Stop MS1 request was executed last

TRUE (1) means a Stop MS1 request was executed last

Bit 4: Check_Properties
FALSE (0) means no Check_Properties request was executed last

TRUE (1) means a Check_Properties request was executed last

Bit 5: reserved

Bit 6: MasterStateClear
FALSE (0) means no MasterStateClear request was executed last

TRUE (1) means a MasterStateClear request was executed last

Bit 7: reserved

5.3.23 Coding of the field Red_Status1

This field represents the Status of the Component which sent the status. The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: Backup
FALSE (0) means Component is not in Backup State

TRUE (1) means Component is in Backup State

Bit 1: Primary
FALSE (0) means Component is not in Primary State

TRUE (1) means Component is in Primary State

Bit 2: HW-Defect
FALSE (0) means has no Hardware Defect

TRUE (1) means Component has a Hardware Defect

Bit 3: DataExchange
FALSE (0) means Component is not in DataExchange State

TRUE (1) means Component is in DataExchange State

Bit 4: Fail Safe
FALSE (0) means Component is not in Fail Safe State

TRUE (1) means Component is in Fail Safe State

Bit 5: DatarateSet
FALSE (0) means has no Datarate set

TRUE (1) means Component has a Datarate set

Bit 6: TohStarted
FALSE (0) means has not started Toh

TRUE (1) means Component has a started Toh

Bit 7: reserved

5.3.24 Coding of the field Red_Status2

This field represents the Status of a Partner Component of the Component which sent the status. The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: Backup
FALSE (0) means Component is not in Backup State

TRUE (1) means Component is in Backup State

Bit 1: Primary
FALSE (0) means Component is not in Primary State

TRUE (1) means Component is in Primary State

Bit 2: HW-Defect
FALSE (0) means has no Hardware Defect

TRUE (1) means Component has a Hardware Defect

Bit 3: DataExchange
FALSE (0) means Component is not in DataExchange State

TRUE (1) means Component is in DataExchange State

Bit 4: Fail Safe
FALSE (0) means Component is not in Fail Safe State

TRUE (1) means Component is in Fail Safe State

Bit 5: DatarateSet
FALSE (0) means has no Datarate set

TRUE (1) means Component has a Datarate set

Bit 6: TohStarted
FALSE (0) means has not started Toh

TRUE (1) means Component has a started Toh

Bit 7: reserved

5.3.25 Coding of the field Red_Status3

This field shall be coded as data type Unsigned8. The values are user specific.

5.4 Coding section related to parameterization PDU

5.4.1 Coding of the field Station_status

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: reserved

Bit 1: reserved

Bit 2: reserved

Bit 3: WD_On (Watchdog on)
FALSE (0)

TRUE (1)

Bit 4: Freeze_Req
FALSE (0): Freeze mode not requested

TRUE (1): Freeze mode requested

Bit 5: Sync_Req
FALSE (0): Sync mode not requested

TRUE (1): Sync mode requested

Bit 6: Unlock_Req
Shall be used as described in Table 20.

Bit 7: Lock_Req
Shall be used as described in Table 20.

Table 20 – Specification of the bits Lock_Req and Unlock_Req

Bit 7	Bit 6	Meaning
0	0	The parameter min T_{SDR} can be changed. All other parameters remain unchanged.
0	1	The DP-slave will be unlocked for other Masters.
1	0	The DP-slave is locked for other Masters, all parameters are accepted (exception: min T_{SDR} = 0)
1	1	The DP-slave is unlocked for other Masters.

5.4.2 Coding of the field WD_Fact_1

This field shall be coded as data type Unsigned8 with the following value range:

Decimal 1 to 255

5.4.3 Coding of the field WD_Fact_2

This field shall be coded as data type Unsigned8 with the following value range:

Decimal 1 to 255

The Watch Dog Time (T_{WD}) shall be calculated according to Formula (1).

$$T_{WD} = WD\_Fact\_1 \times WD\_Fact\_2 \times WD\_Base \quad (1)$$

where

T_{WD}	is the Watch Dog Time;
WD_Fact_1	is the first factor to be multiplied with the time base;
WD_Fact_2	is the second factor to be multiplied with the time base;
WD_Base	is the time base (1 ms or 10 ms) derived from DPV1_Status_1.WD_Base_1ms. If no DPV1_Status_1.WD_Base_1ms exist, then 10 ms is used as time base.

5.4.4 Coding of the field min_T_{SDR}

This field shall be coded as data type Unsigned8 with the following value range:

Decimal (0 to max T_{SDR} (see Part 4)) if DP operation

Decimal (0): means current value remains unchanged

5.4.5 Coding of the field Group_Ident

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: Group_1

Bit 1: Group_2

Bit 2: Group_3

Bit 3: Group_4

Bit 4: Group_5

Bit 5: Group_6

Bit 6: Group_7

Bit 7: Group_8

Each of the bits above shall set to one if the device belongs to the indicated group number. Otherwise it shall be set to zero.

NOTE A device may belong to no groups (Group_Ident=decimal(0)) or all groups (Group_Ident=decimal(255)).

5.4.6 Coding of the field User_Prm_Data_Element

One of the following data types shall be used for each User_Prm_Data_Element:

Boolean, Integer, Unsigned, Floating Point, Visible String, Octet String, Date, TimeOfDay, Time-Difference.

5.4.7 Coding of the field DPV1_Status_1

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: reserved

Bit 1: reserved

Bit 2: WD_Base_1ms
Value (0): Watchdog time base equals 10 milliseconds

Value (1): Watchdog time base equals 1 millisecond

Bit 3 to 4: reserved

Bit 5: Publisher_Enable
FALSE (0): Publisher disabled

TRUE (1): Publisher enabled

Bit 6: Fail_Safe
FALSE (0): Fail_Safe disabled

TRUE (1): Fail_Safe enabled

Bit 7: DPV1_Enable
FALSE (0): DPV1 disabled

TRUE (1): DPV1 enabled

5.4.8 Coding of the field DPV1_Status_2

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: Check_Cfg_Mode
FALSE (0): normal check (restrictive)

TRUE (1): user specific check (more flexible)

Bit 1: reserved

Bit 2: Enable_Update_Alarm
FALSE (0): Update_Alarm disabled

TRUE (1): Update_Alarm enabled

Bit 3: Enable_Status_Alarm
FALSE (0): Status_Alarm disabled

TRUE (1): Status_Alarm enabled

Bit 4: Enable_Manufacturer_Specific_Alarm
FALSE (0): Manufacturer_Specific_Alarm disabled

TRUE (1): Manufacturer_Specific_Alarm enabled

Bit 5: Enable_Diagnostic_Alarm
FALSE (0): Diagnostic_Alarm disabled

TRUE (1): Diagnostic_Alarm enabled

Bit 6: Enable_Process_Alarm
FALSE (0): Process_Alarm disabled

TRUE (1): Process_Alarm enabled

Bit 7: Enable_Pull_Plug_Alarm
FALSE (0): Pull_Plug_Alarm disabled

TRUE (1): Pull_Plug_Alarm enabled

5.4.9 Coding of the field DPV1_Status_3

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 2: Alarm_Mode
Decimal (0): 1 alarm of each type

Decimal (1): 2 alarms in total

Decimal (2): 4 alarms in total

Decimal (3): 8 alarms in total

Decimal (4): 12 alarms in total

Decimal (5): 16 alarms in total

Decimal (6): 24 alarms in total

Decimal (7): 32 alarms in total

Bit 3: Prm_Structure
FALSE (0): Prm_Structure disabled

TRUE (1): Prm_Structure enabled

Bit 4: IsoM_Req
FALSE (0): IsoM_Req disabled

TRUE (1): IsoM_Req enabled

Bit 5 to 6: reserved

Bit 7: PrmCmd
FALSE (0): PrmCmd disabled

TRUE (1): PrmCmd enabled

5.4.10 Coding of the field Structure_Length

This field shall be coded as data type Unsigned8 with the following value range:

Decimal 5 to 244, 255

The field Structure_Length contains the number of octets including itself. A value of 255 means that all elements in the Prm_User_Data following this element belongs to this structure.

5.4.11 Coding of the field Structure_Type

This field shall be coded as data type Unsigned8 with the following value range:

Decimal 0 to 1: reserved

Decimal 2: PrmCmd

Decimal 3: DXB Linktable

Decimal 4: IsoM Parameter

Decimal 5: F-Parameter

Decimal 6: reserved

Decimal 7: DXB Subscribertable

Decimal 8: Time AR Parameter

Decimal 9 to 31: reserved

Decimal 32 to 128: Manufacturer Specific

Decimal 129: User Prm Data

Decimal 130 to 255: reserved

5.4.12 Coding of the field Version

This field shall be coded as data type Unsigned8 with the following value range:

Decimal 0: reserved

Decimal 1: Version 1

Decimal 2 to 255: reserved

5.4.13 Coding of the field Publisher_Addr

This field shall be coded as data type Unsigned8 with the following value range:

Decimal 0 to 125, 128

Decimal 126, 127, 129 to 255: reserved

In case of the DXB-Subscribertable the value 128 indicates an entry for the DP-master (Class 1).

5.4.14 Coding of the field Publisher_Length

This field shall be coded as data type Unsigned8 with the following value range:

Decimal 1 to 244

Decimal 0, 245 to 255: reserved

In case of a Master Data entry in the DXB-Subscribertable the value in this field is not relevant and shall be set to 0.

5.4.15 Coding of the field Sample_Offset

This field shall be coded as data type Unsigned8 with the following value range:

Decimal 0 to 243

Decimal 244 to 255: reserved

5.4.16 Coding of the field Sample_Length

This field shall be coded as data type Unsigned8 with the following value range:

Decimal 1 to 244

Decimal 0, 245 to 255: reserved

5.4.17 Coding of the Dest_Slot_Number

This field shall be coded as data type Unsigned8 with the following value range:

Decimal 1 to 244

Decimal 0, 245 to 255: reserved

5.4.18 Coding of the Offset_Data_Area

This field shall be coded as data type Unsigned8 with the following value range:

Decimal 0 to 244

Decimal 245 to 255: reserved

5.4.19 Coding of the field T_{BASE_DP}

This field shall be coded as data type Unsigned32 with the allowed values of decimal 375, 750, 1 500 (default), 3 000, 6 000, 12 000. All other values are reserved.

5.4.20 Coding of the field T_{DP}

This field shall be coded as data type Unsigned16 with the value range from 154 to 2^{16}-1.

5.4.21 Coding of the field T_{MAPC}

This field shall be coded as data type Unsigned8.

5.4.22 Coding of the field T_{BASE_IO}

This field shall be coded as data type Unsigned32 with the allowed values of decimal 375, 750, 1 500 (default), 3 000, 6 000, 12 000. All other values are reserved.

5.4.23 Coding of the field T_I

This field shall be coded as data type Unsigned16.

5.4.24 Coding of the field T_O

This field shall be coded as data type Unsigned16.

5.4.25 Coding of the field T_{DX}

This field shall be coded as data type Unsigned32.

5.4.26 Coding of the field T_{PLL_W}

This field shall be coded as data type Unsigned16 with the value range from 1 to 2^{16}-1.

5.4.27 Coding of the field T_{PLL_D}

This field shall be coded as data type Unsigned16 with the value range from 0 to 2^{16}-1.

5.4.28 Coding of the field Specifier

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 1: Specifier
Decimal (0): means no further differentiation

Bit 3 to 7: Seq Number
Contains Sequence Number of the PrmCmd

5.4.29 Coding of the field Function

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: reserved

Bit 1: Primary
FALSE (0): means no Primary request is requested

TRUE (1): means a Primary request is requested

Bit 2: Start MS1
FALSE (0): means no Start MS1 request is requested

TRUE (1): means a Start MS1 request is requested

Bit 3: Stop MS1
FALSE (0): means no Stop MS1 request is requested

TRUE (1): means a Stop MS1 request is requested

Bit 4: Check_Properties
FALSE (0): means no Check_Properties request is requested

TRUE (1): means a Check_Properties request is requested

Bit 5: reserved

Bit 6: MasterStateClear
FALSE (0): means the MasterState is Operate

TRUE (1): means the MasterState is Clear

Bit 7: reserved

5.4.30 Coding of the field Properties

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: Primary
FALSE (0): means no Primary request is used

TRUE (1): means a Primary request is used

Bit 1: Start Stop MS1
FALSE (0): means no Start Stop MS1 request is used

TRUE (1): means a Start Stop MS1 request is used

Bit 2: AddressChange
FALSE (0): means no Address Change is required with Primary/Backup role

TRUE (1): means Address Change is required with Primary/Backup role

Bit 3: AddressOffset
FALSE (0): means Address Offset is 0

TRUE (1): means Address Offset is 64

Bit 4 to 7: reserved

5.4.31 Coding of the field Output Hold Time

This field shall be coded as data type Unsigned16 with the value range from 0 to 2^{16}-1.

5.4.32 Coding of the field Clock Sync Interval

This field shall be coded as data type Unsigned16 with the value range from 0 to 2^{16}-1.

5.4.33 Coding of the field CS Delay Time

This field shall be coded as data type Network Time Difference.

5.5 Coding section related to configuration PDUs

5.5.1 Coding of the field Cfg_Identifier

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 3: Data_Length
Decimal (0): means 1 octet or 1 word according to the Length_Format

Decimal (1): means 2 octets or 2 words according to the Length_Format

Decimal (2): means 3 octets or 3 words according to the Length_Format

Decimal (3): means 4 octets or 4 words according to the Length_Format

Decimal (4): means 5 octets or 5 words according to the Length_Format

…

Decimal (15): means 16 octets or 16 words according to the Length_Format

Bit 4 to 5: Input_Output_Selection
Decimal (0): shall not be used here, means special identifier format

Decimal (1): means input

Decimal (2): means output

Decimal (3): means input and output

Bit 6: Length_Format
Value (0): means octet structure used

Value (1): means word structure used (high octet transferred first, big-endian)

Bit 7: Consistency
Value (0): means octet or word consistency

Value (1): means consistency over the whole length

5.5.2 Coding of the field Special_Cfg_Identifier

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 3: Length_of_Manufacturer_Specific_Data
The length in octet information of the manufacturer specific data shall be in dependency of the use in the Chk_Cfg-REQ-PDU or Get_Cfg-RES-PDU as shown in Table 21 and Table 22. Data type fields shall always be counted if present.

Table 21 – Range of Length_of_Manufacturer_Specific_Data if used in Chk_Cfg-REQ-PDU

Value (decimal)	Meaning
0	No manufacturer specific data follow; no data in Real_Cfg_Data.
1 to 14	Manufacturer specific data of specified length follow; these shall be identical with the data in Real_Cfg_Data.
15	No manufacturer specific data follow; the verification can be omitted.

Table 22 – Range of Length_of_Manufacturer_Specific_Data if used in Get_Cfg-RES-PDU

Value (decimal)	Meaning
0	No manufacturer specific data follow.
1 to 14	Manufacturer specific data with specified length follow.
15	Not allowed.

Bit 4 to 5: shall be set to zero (decimal)

Bit 6 to 7: Input_Output_Selection
This field contains the information about the structure following the Special_Cfg_Identifier as shown in Table 23.

Table 23 – Input_Output_Selection

Value (decimal)	Meaning
0	means empty slot, no input or output data for this module
1	means one length octet for input data follows
2	means one length octet for output data follows
3	means one length octet for output data followed by one length octet for input data follows

5.5.3 Coding of the fields Length_Octet

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 5: Data_Length
Decimal (0): means 1 octet or 1 word according to the Length_Format

Decimal (1): means 2 octets or 2 words according to the Length_Format

...

Decimal (63): means 64 octets or 64 words according to the Length_Format

Bit 6: Length_Format
Value (0): means octet structure used

Value (1): means word structure used (high octet transferred first, big-endian)

Bit 7: Consistency
Value (0): means octet or word consistency

Value (1): means consistency over the whole length indicated in Data_Length

5.5.4 Coding of the field Manufacturer_Specific_Data

Each Manufacturer_Specific_Data field shall be coded with the data types stated in 5.5.6.

5.5.5 Coding of the field Extended_Length_Octet

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 5: Data_Length
Decimal (0): means 1 octet

Decimal (1): means 2 octets

Decimal (2): means 3 octets

Decimal (3): means 4 octets

Decimal (4): means 5 octets

...

Decimal (63): means 64 octets

Bit 6: Format
This bit shall be set to zero that means octet structure is used.

Bit 7: Consistency
This bit shall be set to one that means consistency over the whole length.

5.5.6 Coding of the field Data_Type

This field shall be coded as data type Unsigned8 with values according to Table 24.

Table 24 – Data types

Data type name	Code / DataLength	Remark
See IEC 61158-5-10, Clause 5		

Sequences of data types shall be coded as a list of data types. The sum of the lengths of the data types shall correspond to the input- or output data length.

Data types with user specific coding (DataTypeNumber >= 128) shall not be used in any combination with type specific coding (DataTypeNumber < 128) in one Chk_Cfg-REQ-PDU or Get_Cfg-REQ-PDU.

5.6 Coding section related to global control PDUs

5.6.1 Coding of the field Control_Command

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: reserved

Bit 1: Clear_Data
Value (0): no command

Value (1): clear output (set output data in safe state)

Bit 2: UnFreeze
Value (0): no command

Value (1): cancels Freeze command

Bit 3: Freeze
Value (0): no command

Value (1): freeze input values

Table 25 illustrates the meaning and priority of the Bits 2 to 3.

Table 25 – Specification of the bits for Un-/Freeze

Bit 2	Bit 3	Meaning
0	0	no function
0	1	function is activated
1	0	function is deactivated
1	1	function is deactivated

Bit 4: Unsync
Value (0): no command

Value (1): cancels Sync command

Bit 5: Sync
Value (0): no command

Value (1): synchronize output values

Table 26 illustrates the meaning and priority of the Bits 4 to 5.

Table 26 – Specification of the bits for Un-/Sync

Bit 4	Bit 5	Meaning
0	0	no function
0	1	function is activated
1	0	function is deactivated
1	1	function is deactivated

Bit 6: reserved

Bit 7: reserved

5.6.2 Coding of the field Group_Select

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: Group_1

Bit 1: Group_2

Bit 2: Group_3

Bit 3: Group_4

Bit 4: Group_5

Bit 5: Group_6

Bit 6: Group_7

Bit 7: Group_8

Each of the bits above shall set to one if the control command belongs to the indicated group number. Otherwise it shall be set to zero. Zero means that all assigned Slaves shall execute the Control Command.

NOTE A control command may belong to no groups (Group_Ident=decimal(0)) or all groups (Group_Ident=decimal(255)).

5.7 Coding section related to clock-value-PDUs

5.7.1 Coding of the field Clock_value_time_event

The coding of this field shall be as data type Network Time with the fixed length of 8.

5.7.2 Clock_value_previous_TE

The coding of this field shall be as data type Network Time with the fixed length of 8.

5.7.3 Coding of the field Clock_value_status1

The coding of this field shall be according to 3.4, with C and CV mapped to the parameter local time diff in the set time service. The individual bits shall have the following meaning:

Bit 0: reserved

Bit 1: reserved

Bit 2 to 6: CV (Correction Value)
Value (0): 0 min

Value (1): 30 min

Value (2): 60 min

Value (3): 90 min

...

Value (31): 930 min

Bit 7: C (Sign of CV)
Value (0): CV to be added from Time

Value (1): CV to be subtracted from Time

5.7.4 Coding of the field Clock_value_status2

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: SYF (Synchronisation fault)
Value (0): Clock_value_time_event is synchronized

Value (1): Clock_value_time_event is not synchronized (with other clocks in the system)

Bit 1: reserved

Bit 2: reserved

Bit 3 to 4: CR (Accuracy)
Value (0): 1 ms

Value (1): 10 ms

Value (2): 100 ms

Value (3): 1 s

Bit 5: reserved

Bit 6: SWT (Summer/Winter-Time)
Value (0): Winter Time

Value (1): Summer Time

Bit 7: ANH (Announcement Hour)
Value (0): No Change planned within the next hour

Value (1): A Change of SWT will occur within the next hour

5.8 Coding section related to function identification and errors

5.8.1 Coding of the field Function_Num

The coding of this field shall be according to 3.4 and the individual bits shall have the following meanings. The meaning of the complete field is defined in Table 27.

Bit 0 to 4: Function_Code / Error_Code
These bits contain the Function_Code or the Error_Code corresponding to the DLPDU_Selector. The Function_Code shall be set according to Table 27.

Bit 5 to 6: PDU_Identifier
These two bits shall be set to 2 (decimal) to indicate a DP-PDU.

Bit 7: DLPDU_Selector
Value (0): means request DLPDU or positive response DLPDU

Value (1): means error or negative response DLPDU

The field Function_Num is transferred in the request and response DLPDU. If the service is processed correctly, bit 7 of Function_Num in the response APDU shall be set to zero. In case of an error an error DLPDU is transferred. In this APDU Function_Num contains an error code and the bit 7 shall be set to one.

Table 27 – Coding of the Function_Code/ Function_Num

DLPDU_ Selector (decimal)	PDU_ Identifier (decimal)	Function_Code (decimal)	Meaning	Complete field: Function_Num (hexadecimal)
0	2	0	reserved	0x40
0	2	1	Get_Master_Diag	0x41
0	2	2	Start_Seq	0x42
0	2	3	Download	0x43
0	2	4	Upload	0x44
0	2	5	End_Seq	0x45
0	2	6	Act_Para_Brct	0x46
0	2	7	Act_Param	0x47
0	2	8	Idle	0x48
0	2	9 to 16	reserved	0x49 to 0x50

DLPDU_ Selector (decimal)	PDU_ Identifier (decimal)	Function_Code (decimal)	Meaning	Complete field: Function_Num (hexadecimal)
0	2	17	Data_Transport	0x51
0	2	18 to 21	reserved	0x52 to 0x55
0	2	22	RM	0x56
0	2	23	Initiate	0x57
0	2	24	Abort	0x58
0	2	25	reserved	0x59
0	2	26	reserved	0x5A
0	2	27	reserved	0x5B
0	2	28	Alarm_Ack	0x5C
0	2	29	reserved	0x5D
0	2	30	Read	0x5E
0	2	31	Write	0x5F

The Error_Code shall be set according to Table 28.

Table 28 – Coding of the Error_Code / Function_Num

DLPDU_ Selector (decimal)	PDU_ Identifier (decimal)	Error_Code (decimal)	Meaning	Complete field: Function_Num (hexadecimal)
1	2	0	reserved	0xC0
1	2	1	FE	0xC1
1	2	2	NI	0xC2
1	2	3	AD	0xC3
1	2	4	EA	0xC4
1	2	5	LE	0xC5
1	2	6	RE	0xC6
1	2	7	IP	0xC7
1	2	8	SC	0xC8
1	2	9	SE	0xC9
1	2	10	NE	0xCA
1	2	11	DI	0xCB
1	2	12	NC	0xCC
1	2	13	TO	0xCD
1	2	14	CA	0xCE
1	2	15 to 16	reserved	0xCF to 0xD0
1	2	17	Error Data_Transport	0xD1
1	2	18 to 22	reserved	0xD2 to 0xD6
1	2	23	Error Initiate	0xD7
1	2	24	reserved	0xD8
1	2	25	reserved	0xD9
1	2	26	reserved	0xDA
1	2	27	reserved	0xDB

DLPDU_ Selector (decimal)	PDU_ Identifier (decimal)	Error_Code (decimal)	Meaning	Complete field: Function_Num (hexadecimal)
1	2	28	Error Alarm_Ack	0xDC
1	2	29	reserved	0xDD
1	2	30	Error Read	0xDE
1	2	31	Error Write	0xDF

5.8.2 Coding of the field Error_Decode

The coding of this field shall be as data type Unsigned8 and the values shall be set according to Table 29.

Table 29 – Values of Error_Decode

Value (decimal)	Meaning
0 to 127	reserved
128	DPV1
129 to 253	reserved
254 to 255	PROFILE_SPECIFIC

The field Error_Decode defines the meaning of the following Octets Error_Code_x. The Error_Decode values PROFILE_SPECIFIC indicate that the parameters Error_Code_x shall be set as defined in the corresponding protocols:

PROFILE_SPECIFIC

Error_Code_1= taken from profile specification

Error_Code_2= taken from profile specification

5.8.3 Coding of the field Error_Code_1

If field Error_Decode = DPV1

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 3: Error_Code
The values shall be set according to the Error_Code column in Table 30.

Bit 4 to 7: Error_Class
The values shall be set according to the Error_Class column in Table 30.

Table 30 – Coding of Error_Code_1 at DPV1

Error_Class (decimal)	Meaning	Error_Code (decimal)
0 to 9	not specified	not specified [a]
10	application	0 = read error 1 = write error 2 = module failure 3 to 6 = not specified [a] 7 = busy 8 = version conflict 9 = feature not supported 10 to 15 = User specific
11	access	0 = invalid index 1 = write length error 2 = invalid slot 3 = type conflict 4 = invalid area 5 = state conflict 6 = access denied 7 = invalid range 8 = invalid parameter 9 = invalid type 10= backup 11 to 15 = User specific
12	resource	0 = read constrain conflict 1 = write constrain conflict 2 = resource busy 3 = resource unavailable 4 to 7 = not specified [a] 8 to 15 = User specific
13 to 15	User specific	

[a] Not specified values are used to serve legacy codes and are intended to be passed unchanged to the application.

If field Error_Decode = PROFILE_SPECIFIC

Error_Code_1= from profile specification, needs to be specified in the profile definition

5.8.4 Coding of the field Error_Code_2

If field Error_Decode = DPV1

The coding of this field shall be as data type Unsigned8. The values are user specific.

If field Error_Decode = PROFILE_SPECIFIC

Error_Code_2 = from profile specification, needs to be specified in the profile definition.

5.9 Coding section related to master diagnosis PDU

5.9.1 Coding of the field MDiag_Identifier

The coding of this field shall be as data type Unsigned8 and the values shall be set according Table 31:

Table 31 – Values of MDiag_Identifier

Value (decimal)	Meaning
0 to 125	Diag_Data of the DP-slave
126	System_Diagnosis
127	Master_Status
128	Data_Transfer_List
129 to 255	reserved

5.9.2 Coding of the field System_Diagnosis

The coding of this field shall be as data type Octet String with the fixed length of 16. The 16 octets of System_Diagnosis shall be coded as follows:

Octet 1
Bit 0 shall be set to one if station number 0 has reported diagnosis otherwise it shall be set to zero.

Bit 1 shall be set to one if station number 1 has reported diagnosis otherwise it shall be set to zero.

Bit 2 shall be set to one if station number 2 has reported diagnosis otherwise it shall be set to zero.

Bit 3 shall be set to one if station number 3 has reported diagnosis otherwise it shall be set to zero.

Bit 4 shall be set to one if station number 4 has reported diagnosis otherwise it shall be set to zero.

Bit 5 shall be set to one if station number 5 has reported diagnosis otherwise it shall be set to zero.

Bit 6 shall be set to one if station number 6 has reported diagnosis otherwise it shall be set to zero.

Bit 7 shall be set to one if station number 7 has reported diagnosis otherwise it shall be set to zero.

Octet 2
Bit 0 shall be set to one if station number 8 has reported diagnosis otherwise it shall be set to zero.

Bit 1 shall be set to one if station number 9 has reported diagnosis otherwise it shall be set to zero.

etc.

Octet 16
Bit 5 shall be set to one if station number 125 has reported diagnosis otherwise it shall be set to zero.

Bit 6 to 7 shall be set to zero (not used).

5.9.3 Coding of the field USIF_State

This field shall be coded as data type Unsigned8. The following values are defined:

Value (0x40): shall be set if operation mode of DP-master (Class 1) is STOP

Value (0x80): shall be set if operation mode of DP-master (Class 1) is CLEAR

Value (0xC0): shall be set if operation mode of DP-master (Class 1) is OPERATE

5.9.4 Coding of the field Hardware_Release_DP

This field shall be coded as data type Unsigned8.

5.9.5 Coding of the field Firmware Release_DP

This field shall be coded as data type Unsigned8.

5.9.6 Coding of the field Hardware_Release_User

This field shall be coded as data type Unsigned8.

5.9.7 Coding of the field Firmware Release_User

This field shall be coded as data type Unsigned8.

5.9.8 Coding of the field Data_Transfer_List

The coding of this field shall be as data type Octet String with the fixed length of 16. The 16 octets of Data_Transfer_List shall be coded as follows:

Octet 1
Bit 0 shall be set to one if data exchange executed with station number 0 otherwise it shall be set to zero.

Bit 1 shall be set to one if data exchange executed with station number 1 otherwise it shall be set to zero.

Bit 2 shall be set to one if data exchange executed with station number 2 otherwise it shall be set to zero.

Bit 3 shall be set to one if data exchange executed with station number 3 otherwise it shall be set to zero.

Bit 4 shall be set to one if data exchange executed with station number 4 otherwise it shall be set to zero.

Bit 5 shall be set to one if data exchange executed with station number 5 otherwise it shall be set to zero.

Bit 6 shall be set to one if data exchange executed with station number 6 otherwise it shall be set to zero.

Bit 7 shall be set to one if data exchange executed with station number 7 otherwise it shall be set to zero.

Octet 2
Bit 0 shall be set to one if data exchange executed with station number 8 otherwise it shall be set to zero.

Bit 1 shall be set to one if data exchange executed with station number 9 otherwise it shall be set to zero.

etc.

Octet 16
.

.

Bit 5 shall be set to one if data exchange executed with station number 125 otherwise it shall be set to zero.

Bit 6 to 7 shall be set to zero (not used)

5.10 Coding section related to upload/download/act para PDUs

5.10.1 Coding of the field Area_Code_UpDownload

The coding of this field shall be as data type Unsigned8 and the values shall be set according to Table 32.

Table 32 – Values for Area_Code_UpDownload

Value (decimal)	Meaning
0 to 125	DP-slave parameter set (see field DP_Slave_Parameter_Set)
126	reserved
127	Bus parameter set (see field Bus_Parameter_Set)
128	reserved
129	statistic counters (see field Statistic_Counter)
130 to 135	reserved
136 to 139	Master parameter set (see field Master_Parameter_Set)
140 to 254	reserved
255	No local access protection in Start_Seq-REQ-PDU or Start_Seq-RES-PDU

5.10.2 Coding of the field Timeout

The coding of this field shall be as data type Unsigned16 and the time base for the timer shall be 1 ms.

5.10.3 Coding of the field Max_Len_Data_Unit

The coding of this field shall be as data type Unsigned8 and the value for the Max_Len_Data_Unit shall be in the range from 1 to 240.

NOTE The minimum value for the Max_Len_Data_Unit is 68 in case of Initiate-RES_PDU.

5.10.4 Coding of the field Add_Offset

The coding of this field shall be as data type Unsigned16.

5.10.5 Coding of the field Data

One of the following data types shall be used for each Data field:

Boolean, Integer, Unsigned, Floating Point, Visible String, Octet String, Date, TimeOfDay, Time-Difference.

5.10.6 Coding of the field Data_Len

The coding of this field shall be as data type Unsigned8 and the value for the Data_Len shall be in the range from 1 to 240.

5.10.7 Coding of the field Area_CodeActBrct

The coding of this field shall be as data type Unsigned8 and the values shall be set according to Table 33.

Table 33 – Values for Area_CodeActBrct

Value (decimal)	Meaning
0 to 126	reserved
127	Bus_parameter set
128 to 129	reserved
130 to 135	reserved
136 to 139	Master parameter set
140 to 254	reserved
255	reserved

5.10.8 Coding of the field Area_CodeAct

The coding of this field shall be as data type Unsigned8 and the values shall be set according Table 34.

Table 34 – Values for Area_CodeAct

Value (decimal)	Meaning
0 to 125	DP-slave parameter set The Active Flag in the DP-slave parameter set of the DP-master (Class 1) is influenced accordingly. The DP-slave concerned takes part in the cyclic data exchange mode or is removed from this mode and no longer addressed.
126	reserved
127	Bus parameter set
128	operation mode
129	reserved
130 to 135	reserved
136 to 139	Master parameter set
140 to 255	reserved

5.10.9 Coding of the field Activate

This field shall be coded as data type Unsigned8 according to Table 35.

Table 35 – Values for Activate

Value (hexadecimal)	Meaning
0x80	means activate the corresponding DP-slave parameter set if the field Area_CodeAct equals 0..125
0x00	means deactivate if the field Area_CodeAct equals 0..125
0xFF	means activate the bus parameter set if the field Area_CodeAct equals 127
0x40	means set operation mode to STOP if the field Area_CodeAct equals 128
0x80	means set operation mode to CLEAR if the field Area_CodeAct equals 128
0xC0	means set operation mode to OPERATE if the field Area_CodeAct equals 128

5.11 Coding section related to the bus parameter set

5.11.1 Coding of the field Bus_Para_Len

This field shall contain the length of Bus_Para inclusive the field Bus_Para_Len itself. This field shall be coded as data type Unsigned16. The range of values shall be from 66 to 2^{16}-1.

5.11.2 Coding of the field DL_Add

This field shall be coded as data type Unsigned8. The range of values shall be 0 to 125. This parameter shall contain the own address of the DP-master.

5.11.3 Coding of the field Data_rate

This field shall be coded as data type Unsigned8. The values shall be set as shown at Table 36:

Table 36 – Values for Data_rate

Value (decimal)	Meaning
0	9,6 kbit/s
1	19,2 kbit/s
2	93,75 kbit/s
3	187,5 kbit/s
4	500 kbit/s
5	reserved
6	1 500 kbit/s
7	3 000 kbit/s
8	6 000 kbit/s
9	12 000 kbit/s
10	31,25 kbit/s
11	45,45 kbit/s
12 to 255	reserved

5.11.4 Coding of the fields T_{SL}, min T_{SDR}, max T_{SDR}

These parameters are described in IEC 61158-3-3. The coding of these fields shall be as data type Unsigned16.

5.11.5 Coding of the fields T_{QUI}, T_{SET}, G, HSA, max_retry_limit

These parameters are described in IEC 61158-3-3. The coding of these fields shall be as data type Unsigned8.

5.11.6 Coding of the field T_{TR} (Target Token Rotation time)

This field is described in IEC 61158-3-3. The coding of this field shall be as data type Unsigned32.

5.11.7 Coding of the field Bp_Flag (Busparameter flag)

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 2: reserved

Bit 3 to 4: Isochronous Mode
Decimal (0): Not Synchronized

Decimal (1): Buffered Synchronized

Decimal (2): Enhanced Synchronized

Decimal (3): reserved

Bit 5: IsoM Sync
This bit shall be set to one if the DP master (Class 1) shall send a Global_Control-REQ-PDU with Control Command Sync in case of isochronous operation. Otherwise this bit shall be set to zero.

Bit 6: IsoM Freeze
This bit shall be set to one if the DP master (Class 1) shall send a Global_Control-REQ-PDU with Control Command Freeze in case of isochronous operation. Otherwise this bit shall be set to zero.

Bit 7: Error_Action_Flag
This bit shall be set to one if the DP master (Class 1) shall change the operation mode in case of an error. Otherwise this bit shall be set to zero.

5.11.8 Coding of the field Min_Slave_Interval

The coding of this field shall be as data type Unsigned16 with the value range from 1 to 2^{16}-1 and the time base for the timer shall be 100 µs.

5.11.9 Coding of the field Poll_Timeout

The coding of this field shall be as data type Unsigned16 with the value range from 1 to 2^{16}-1 and the time base for the timer shall be 1 ms.

5.11.10 Coding of the field Data_Control_Time

The coding of this field shall be as data type Unsigned16 with the value range from 1 to 2^{16}-1 and the time base for the timer shall be 10 ms.

5.11.11 Coding of the field Alarm_Max

The coding of this field shall be as data type Unsigned8 with the value range from 7 to 32.

5.11.12 Coding of the field Max_User_Global_Control

The coding of this field shall be as data type Unsigned8 with the value range from 1 to 255. The value Zero is reserved.

NOTE This parameter defines the maximum number of Global_Control requests, which may be started by the User at the same time. The parameter describes the ability of the DP-master. A practicable value for Max_User_Global_Control is 16. For each of the 8 Slave groups one SYNC and one FREEZE command may be started at the same time.

5.11.13 Coding of the field Master_User_Data_Len

This field shall contain the length of Master_User_Data inclusive the field Master_User_Data_Len itself. This field shall be coded as data type Unsigned16. The range of values shall be from 2 to (2^{16}–1).

5.11.14 Coding of the field Master_Class2_Name

The coding of this field shall be as data type Visible String with the fixed length of 32.

5.11.15 Coding of the field Master_User_Data

One of the following data types shall be used for each Master_User_Data field:

Boolean, Integer, Unsigned, Floating Point, Visible String, Octet String, Date, TimeOfDay, Time-Difference.

5.11.16 Coding of the field T_{CT}

The coding of this field shall be as data type Unsigned32 with the value range from 1 to 2^{24}–1.

5.11.17 Coding of the field $maxT_{SH}$

The coding of this field shall be as data type Unsigned8 with the value range from 1 to 2^{8}–1.

5.12 Coding section related to the slave parameter set

5.12.1 Coding of the field Slave_Para_Len

The coding of this field shall be as data type Unsigned16 and the value for the Slave_Para_Len shall be in the range from 0 to 2^{16}–1.This parameter shall contain the length of Slave_Para inclusive the length parameter itself. A DP-slave parameter set can be deleted by setting the Slave_Para_Len to zero.

5.12.2 Coding of the field SI_Flag (slave flag)

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: reserved

Bit 1: Extra_Alarm_SAP
This bit shall be set if the DP-master (Class 1) shall acknowledge alarms via SAP 50. Otherwise, if the DP-master (Class 1) acknowledges alarms via SAP 51 this bit shall not be set.

Bit 2: DPV1_Data_Types
This bit shall be set according to Table 37.

Table 37 – DPV1_Data_Types

Value (decimal)	Meaning
0	Identifier_Format or Special_Identifier_Format Octet or word data types
1	Extended_Special_Identifier_Format Extended data types are used in configuration

Bit 3: DPV1_Supported
This bit shall be set to enable the DPV1 extensions. Otherwise the extensions shall be disabled.

Bit 4: Publisher_Enable
This bit shall be set to enable the publisher function. Otherwise the publisher function shall be disabled.

Bit 5: Fail_Safe
This bit shall be set to force the DP-master (Class 1) to convey a NULL-PDU during operation mode Clear. Otherwise data with the value zero will be sent during Clear.

Bit 6: New_Prm
This bit shall be set to force the DP-master (Class 1) to convey a new Set_Prm-REQ-PDU during the data transfer phase. Otherwise this bit shall be set to zero.

Bit 7: Active
This bit shall be set to force the DP-master (Class 1) to activate the corresponding DP-slave. Otherwise this bit shall be set to zero.

5.12.3 Coding of the field Slave_Type

This field shall be coded as data type Unsigned8 with values according to Table 38:

Table 38 – Values for Slave_Type

Value (decimal)	Meaning
0	DP-slave
1 to 15	reserved
16 to 255	manufacturer specific

5.12.4 Coding of the field Max_Diag_Data_Len

This field shall be coded as data type Unsigned8 and with values from the range 6 to 244.

5.12.5 Coding of the field Max_Alarm_Len

This field shall be coded as data type Unsigned8 and with values from the range 4 to Min(Max_Diag_Data_Len-6, 64).

5.12.6 Coding of the field Max_Channel_Data_Length

This field shall be coded as data type Unsigned8 and with values from the range 4 to 244.

NOTE This parameter defines the maximum APDU size for the corresponding DP-slaves on the MS1-AR.

5.12.7 Coding of the field Diag_Upd_Delay

This field shall be coded as data type Unsigned8 and with values from the range 0 to 15 (extendible up to 255).

NOTE The parameter is used to count the number of Slave_Diag.cnf in the state DIAG2 while Diag_Data.Prm_Req is still set (for Slaves with reduced performance).

5.12.8 Coding of the field Alarm_Mode

This parameter specifies the maximum number of possible active alarms.

This field shall be coded as data type Unsigned8 with values according to Table 39:

Table 39 – Values for Alarm_Mode

Value (decimal)	Meaning
0	1 alarm of each type
1	2 alarms in total
2	4 alarms in total
3	8 alarms in total
4	12 alarms in total
5	16 alarms in total
6	24 alarms in total
7	32 alarms in total

5.12.9 Coding of the field Add_Sl_Flag

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: NA_To_Abort
This bit shall be set if the DP-master (Class 1) shall proceed with data exchange in case the corresponding DP-slave is not responding. Otherwise this bit shall not be set.

Bit 1: Ignore_ACIr
This bit shall be set if the DP-master (Class 1) shall exclude the corresponding DP-slave from the auto-clear behaviour. Otherwise this bit shall not be set.

Bit 2 to 7: reserved

5.12.10 Coding of the field MS1_Timeout

This field shall be coded as data type Unsigned16 with values from the range decimal 1 to $2^{16}-1$. The value 0 is reserved but may be used for back-up values.

5.12.11 Coding of the field Prm_Data_Len

This parameter contains the length of Prm_Data inclusive the length parameter.

This field shall be coded as data type Unsigned16 and with values from the range 9 to 246.

5.12.12 Coding of the field Prm_Data

This field shall be coded as a Set_Prm-REQ-PDU.

5.12.13 Coding of the field Cfg_Data_Len

This parameter contains the length of Cfg_Data inclusive the length parameter.

This field shall be coded as data type Unsigned16 and with values from the range 3 to 246.

5.12.14 Coding of the field Cfg_Data

This field shall be coded as a Chk_Cfg-REQ-PDU.

5.12.15 Coding of the field Add_Tab_Len

This parameter contains the length of Add_Tab inclusive the length parameter.

This field shall be coded as data type Unsigned16 and with values from the range 2 to $2^{16}-31$.

5.12.16 Coding of the field Number_of_Entries

This field contains the number of entries in the address table.

This field shall be coded as data type Unsigned16 and with values from the range 1 to 488.

NOTE This parameter contains the number of entries in the address assignment table in the host to the addressed DP-slave. At maximum 488 entries per DP-slave are allowed (maximal 244 configuration strings doubled for inputs and outputs are possible).

5.12.17 Coding of the field Add_Tab_Entry_Header

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: IO_Selection
This bit shall be set if the following addresses belong to the output part. Otherwise this bit shall not be set and the following addresses belong to the input part.

Bit 1: Address_Format_Selection
This bit shall be set to one if the following addresses are coded as Unsigned32. This bit shall be set to zero if the following addresses are coded as Unsigned16.

Bit 2 to 7: reserved

5.12.18 Coding of the field I/O_Data_Length

This field shall be coded as data type Unsigned8 with the values from 1 to 244.

5.12.19 Coding of the field I/O_Config_Address

This field represents the local address of the related configuration identifier.

This field shall be coded as data type Unsigned16 if the bit Address_Format_Selection in the field Add_Tab_Entry_Header is set to zero.

This field shall be coded as data type Unsigned32 if the bit Address_Format_Selection in the field Add_Tab_Entry_Header is set to one.

5.12.20 Coding of the field Host_Address

This field represents the local address of the related input or output data.

This field shall be coded as data type Unsigned16 if the bit Address_Format_Selection in the field Add_Tab_Entry_Header is set to zero.

This field shall be coded as data type Unsigned32 if the bit Address_Format_Selection in the field Add_Tab_Entry_Header is set to one.

5.12.21 Coding of the field Slave_User_Data_Len

This parameter contains the length of Slave_User_Data inclusive the length parameter.

This field shall be coded as data type Unsigned16 and with values from the range 2 to $2^{16}-31$.

5.12.22 Coding of the field Slave_User_Data

One of the following data types shall be used for each Slave_User_Data field:

Boolean, Integer, Unsigned, Floating Point, Visible String, Octet String, Date, TimeOfDay, Time-Difference.

5.12.23 Coding of the field Ext_Prm_Data_Len

This parameter contains the length of Ext_Prm_Data inclusive the length parameter (2 octets).

This field shall be coded as data type Unsigned16 and with values from the range 2 (no Ext_Prm_Data follow) and 7 to 246.

5.12.24 Coding of the field Ext_Prm_Data

This field shall be coded as a Set_Ext_Prm-REQ-PDU.

5.13 Coding section related to statistic counters

5.13.1 Coding of the field DLPDU_sent_count and SD_count

The coding of this field shall be as data type Unsigned32.

5.13.2 Coding of the field Error_count and SD_error_count

The coding of this field shall be as data type Unsigned16.

5.14 Coding section related to set slave address PDU

5.14.1 Coding of the field New_Slave_Add

This field shall be coded as data type Unsigned8 with values from the range 0 to 125.

5.14.2 Coding of the field No_Add_Change

This field shall be coded as data type Boolean. The value TRUE shall be used to indicate that the change of the address is only once possible. The value FALSE shall be used to indicate that the change of the address is more than once possible.

5.14.3 Coding of the field Rem_Slave_Data

One of the following data types shall be used for each Rem_Slave _Data field:

Boolean, Integer, Unsigned, Floating Point, Visible String, Octet String, Date, TimeOfDay, Time-Difference.

5.15 Coding section related to initiate/abort PDUs

5.15.1 Coding of the field Features_Supported_1

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: Read_Write
This bit shall be set to indicate that Read and Write services on MS2-AR are supported.

NOTE Read and Write are mandatory on a MS2-AR.

Bit 1 to 7: reserved

5.15.2 Coding of the field Features_Supported_2

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 7: reserved

5.15.3 Coding of the field Profile_Features_Supported_1

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 7: defined by profile

This field shall be set to zero (decimal) if no profile is used. Otherwise the value shall be taken from the appropriate profile specification.

5.15.4 Coding of the field Profile_Features_Supported_2

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 7: defined by profile

This field shall be set to zero (decimal) if no profile is used. Otherwise the value shall be taken from the appropriate profile specification.

5.15.5 Coding of the field Profile_Ident_Number

The coding of this field shall be as data type Unsigned16. The value zero shall be set if no profile is used. Otherwise the value shall be taken from the appropriate profile specification.

5.15.6 Coding of the field S_Type (source type)

The coding of this field shall be as data type Unsigned8. The value zero shall be set to indicate the use of the short address format. The value one shall be set to indicate the use of the long address format. Other values are reserved.

NOTE The terms source and destination always belong to the direction of the used PDU.

5.15.7 Coding of the field D_Type (destination type)

The coding of this field shall be as data type Unsigned8. The value zero shall be set to indicate the use of the short address format. The value one shall be set to indicate the use of the long address format. Other values are reserved.

5.15.8 Coding of the field S_Len (source length)

This field shall be coded as data type Unsigned8.

5.15.9 Coding of the field D_Len (destination length)

This field shall be coded as data type Unsigned8.

5.15.10 Coding of the field S_API (source application identifier)

This field shall be coded as data type Unsigned8.

5.15.11 Coding of the field D_API (destination application identifier)

This field shall be coded as data type Unsigned8.

5.15.12 Coding of the field S_SCL (source security level)

This field shall be coded as data type Unsigned8. The value zero shall be set to indicate that no access level is used.

5.15.13 Coding of the field D_SCL (destination security level)

This field shall be coded as data type Unsigned8. The value zero shall be set to indicate that no access level is used.

5.15.14 Coding of the field S_Network_Address

The field shall only be present if the field S_Type is set to one.

The coding of this field shall be as data type Octet String with the fixed length of 6.

5.15.15 Coding of the field D_Network_Address

The field shall only be present if the field D_Type is set to one.

The coding of this field shall be as data type Octet String with the fixed length of 6.

5.15.16 Coding of the field S_MAC_Address

The field shall only be present if the field S_Type is set to one.

The coding of this field shall be as data type Octet String.

5.15.17 Coding of the field D_MAC_Address

The field shall only be present if the field D_Type is set to one.

The coding of this field shall be as data type Octet String.

5.15.18 Coding of the field Send_Timeout

This field shall be coded as data type Unsigned16. The time base for the timer shall be 10 ms. The values shall be set from the range 1 to $2^{16}-1$.

5.15.19 Coding of the field Server_SAP

This field shall be coded as data type Unsigned8. The values shall be set from the range 0 to 48.

5.15.20 Coding of the field Subnet

The coding of this field shall be as data type Unsigned8 and the values shall be set according to Table 40.

Table 40 – Values for Subnet

Value (decimal)	Meaning
0	means no subnet
1	means subnet local
2	means subnet remote
3 to 255	reserved

5.15.21 Coding of the field Instance_Reason_Code

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0 to 3: Reason Code
The values shall be set according to Table 41 if Instance is set to DLL.

Table 41 – Values of reason code if instance is DLL

Value (decimal)	Meaning
1	UE (meaning see IEC 61158-3-3)
2	RR (meaning see IEC 61158-3-3)
3	RS (meaning see IEC 61158-3-3)
9	NR (meaning see IEC 61158-3-3)
10	DH (meaning see IEC 61158-3-3)
11	LR (meaning see IEC 61158-3-3)
12	RDL (meaning see IEC 61158-3-3)
13	RDH (meaning see IEC 61158-3-3)
14	DS (meaning see IEC 61158-3-3)
15	NA (meaning see IEC 61158-3-3)

The values shall be set according to Table 42 if Instance is set to MS2.

Table 42 – Values of reason code if instance is MS2

Value (decimal)	Meaning
1	ABT_SE means sequence error
2	ABT_FE means invalid request APDU received
3	ABT_TO means connection timed out
4	ABT_RE means invalid response APDU received
5	ABT_IV means invalid service from the User
6	ABT_STO means requested value of Send_Timeout was too short
7	ABT_IA means invalid additional address information
8	ABT_OC means S-Timer expired, response APDU has not been sent yet

Bit 4 to 5: Instance
The values shall be set as follows:

Decimal (0): DLL

Decimal (1): MS2

Decimal (2): USER

Decimal (3): reserved

Bit 6 to 7: reserved

5.16 Coding section related to read/write/data transport PDUs

5.16.1 Coding of the field Index

This field shall be coded as data type Unsigned8 with values from the range 0 to 255.

5.16.2 Coding of the field Length

This field contains the number of octets of the Data field.

This field shall be coded as data type Unsigned8 with values from the range 0 to 240.

5.17 Coding section related to load region and function invocation PDUs

5.17.1 Coding of the field Extended_Function_Num

In addition to the field Function_Num, as described in 5.8.1, this field specifies the function of the Load Region or Function Invocation service.

The coding of this field shall be as data type Unsigned8 and the values shall be set according to Table 43:

Table 43 – Values of Extended_Function_Num

Value (decimal)	Meaning
0	Initiate_Load
1	Push
2	Pull
3	Terminate
4	Start
5	Stop
6	Resume
7	Reset
8	Call
9	Get_FI_State
10 to 255	reserved

5.17.2 Coding of the field Options

The coding of this field shall be according to 3.4 and the individual bits shall have the following meaning:

Bit 0: More_follows
This bit is only valid for the Push APDU and Pull APDU. This bit shall be set if the Load Region sequence is not finished and additional Push APDUs or Pull APDUS follow subsequently. This bit shall be reset, if the sequence is finished and no additional Push APDUs or Pull APDUS follow.

For the Initiate_Load APDU this bit shall be reset.

Bit 1 to 6: reserved

Bit 7: Pull / Push
This bit is only valid in the Initiate_Load APDU. This bit shall be set, if Pull LR sequence is initialized. This bit shall be reset, if a Push LR sequence is initialized.

For the Push APDU and Pull APDU this bit shall be reset.

5.17.3 Coding of the field Sequence_Number

This field shall be coded as data type Unsigned32.

5.17.4 Coding of the field LR_Data

This field shall be coded as data type Octet String.

5.17.5 Coding of the field Max_Segment_Length

This field shall be coded as data type Unsigned8.

5.17.6 Coding of the field LR_Index

This field shall be coded as data type Unsigned16.

5.17.7 Coding of the field LR_Length

This field shall be coded as data type Unsigned32.

5.17.8 Coding of the field Max_Response_Delay

This field shall be coded as data type Unsigned16.

5.17.9 Coding of the field Intersegment_Request_Timeout

This field shall be coded as data type Unsigned16.

5.17.10 Coding of the field User_Specific

This field contains user specific information, e.g. the format of the LR_Data.

This field shall be coded as data type Octet String.

5.17.11 Coding of the field FI_Index

This field shall be coded as data type Unsigned16 with values according to Table 44.

Table 44 – Values of FI_Index

Value (decimal)	Meaning
0 to 32 767	Indices reserved for manufacturer specific use
32 768 to 64 999	Indices reserved for Profibus system services
65 000 to 65 199	Indices reserved for Identification and Maintenance (I&M) services
65 200 to 65 535	Reserved

5.17.12 Coding of the field Entity Number

This field shall be coded as data type Unsigned8.

The meaning of this parameter is profile specific. If not used it shall be set to 0.

5.17.13 Coding of the field Execution_Argument

This field shall be coded as data type Octet String.

5.17.14 Coding of the field Result_Argument

This field shall be coded as data type Octet String.

5.17.15 Coding of the field FI_State

This field contains the state of the Function Invocation entity. The coding of this field shall be as data type Unsigned8 and the values shall be set according to Table 45:

Table 45 – Values of FI_State

Value (decimal)	Meaning
0	IDLE
1	STARTING
2	RUNNING
3	STOPPING
4	STOPPED
5	RESUMING
6	UNRUNNABLE
7	RESETTING
8	WAIT-SET-IN-USE
9	WAIT-SET-IN-USE-2
10	WAIT-DEL-IN-USE
11	WAIT-DEL-IN-USE-2
12	EXEC-ERR
13	EXEC-TERM
14 to 255	Reserved

5.17.16 Coding of the field IMData_Execution_Argument

5.17.16.1 General

This field contains the IMData_Execution_Argument of the CALL-REQ-PDU which are coded according to Table 46.

Table 46 – IMData_Execution_Argument

IMData_Execution_Argument	Reference
IM_Header	See 5.17.16.2
IM_Profile_Specific_Data	See 5.17.16.3
I&M_Manufacturer_Specific_x	See 5.17.16.4
IM_Date	See field IM_Date within IEC 61158-6-10
IM_Descriptor	See field IM_Descriptor within IEC 61158-6-10
IM_Signature	See field IM_Signature within IEC 61158-6-10
IM_Tag_Function	See field IM_Tag_Function within IEC 61158-6-10
IM_Tag_Location	See field IM_Tag_Location within IEC 61158-6-10
IM_Manufacturer_ID	See field VendorID within IEC 61158-6-10

5.17.16.2 Coding of the field IM_Header

This field shall be coded as data type OctetString with 10 octets. The value shall be set manufacturer specific.Any unused octet shall be filled with 0x00.

5.17.16.3 Coding of the field IM_Profile_Specific_Data

This field shall be coded as data type OctetString with 54 octets. The value shall be set profile specific and shall be filled with 0x00 if it is shorter than 54.

5.17.16.4 Coding of the field I&M_Manufacturer_Specific_x

This field shall be coded as data type OctetString with 64 octets. The value shall be set manufacturer specific and shall be filled with 0x00 if it is shorter than 64.

5.17.17 Coding of the field IMData_Result_Argument

This field contains the IMData_Result_Argument of the CALL-RES-PDU which are coded according to Table 47.

Table 47 – IMData_Result_Argument

IMData_Result_Argument	Reference
IM_Header	See 5.17.16.2
IM_Profile_Specific_Data	See 5.17.16.3
I&M_Manufacturer_Specific_x	See 5.17.16.4
IM_Date	See field IM_Date within IEC 61158-6-10
IM_Descriptor	See field IM_Descriptor within IEC 61158-6-10
IM_Signature	See field IM_Signature within IEC 61158-6-10
IM_Tag_Function	See field IM_Tag_Function within IEC 61158-6-10
IM_Tag_Location	See field IM_Tag_Location within IEC 61158-6-10
IM_Manufacturer_ID	See field VendorID within IEC 61158-6-10
IM_Hardware_Revision	See field IM_Hardware_Revision within IEC 61158-6-10

IMData_Result_Argument	Reference
IM_Profile_ID	See field IM_Profile_ID within IEC 61158-6-10
IM_Profile_Specific_Type	See field IM_Profile_Specific_Type within IEC 61158-6-10
IM_Revision_Counter	See field IM_Revision_Counter within IEC 61158-6-10
IM_Serial_Number	See field IM_Serial_Number within IEC 61158-6-10
IM_Supported	See field IM_Supported within IEC 61158-6-10
IM_SWRevision_Bug_Fix	See field IM_SWRevision_Bug_Fix within IEC 61158-6-10
IM_SWRevision_Functional_Enhancement	See field IM_SWRevision_Functional_Enhancement within IEC 61158-6-10
IM_SWRevision_Internal_Change	See field IM_SWRevision_Internal_Change within IEC 61158-6-10
IM_Version_Major	See field IM_Version_Major within IEC 61158-6-10
IM_Version_Minor	See field IM_Version_Minor within IEC 61158-6-10
OrderID	See field OrderID within IEC 61158-6-10
SWRevisionPrefix	See field SWRevisionPrefix within IEC 61158-6-10

5.18 Examples of Diagnosis-RES-PDUs

Figure 3 illustrates an example for a diagnosis APDU with a device related status message, a device related process alarm and an identifier related diagnostic.

bits	7	6	5	4	3	2	1	0	
octets									
1	0	0	0	0	0	1	1	1	Headeroctet: device related diagnostic
2	1	0	0	0	0	0	0	1	Status_Type: Status_Message
3	0	0	0	0	0	0	1	0	Slot_Number: 2 (sensor A)
4	0	0	0	0	0	0	0	0	Specifier: no further differentiation
5	0	0	0	0	0	1	0	1	Diagnostic_User_Data: 5 (average temperature)
6									Diagnostic_User_Data: temperature
7									value (Unsigned16)
8	0	0	0	0	1	0	0	1	Headeroctet: device related diagnostic
9	0	0	0	0	0	0	1	0	Alarm_Type: Process_Alarm
10	0	0	0	0	0	0	1	1	Slot_Number: 3 (valve B)
11	0	0	0	0	0	0	0	1	Specifier: alarm appears
12	0	1	0	1	0	0	0	0	Diagnostic_User_Data: 0x50 (upper limit exceeded pressure)
13									Diagnostic_User_Data: time stamp
14									(Time_of_day = 4 octets)
15									
16									
17	0	1	0	0	0	0	1	0	Headeroctet: identifier related diagnostic
18	0	0	0	0	0	1	0	1	1st identifier Number with diagnosis
	msb								

Figure 3 – Example of Ext_Diag_Data in case of DPV1 diagnosis format with alarm and status PDU

Corresponding textual part
Text assignments for sensor A and valve B

Unit_Diag_Area = 24-27

Value (1) = “Minimum temperature”

Value (2) = “Maximum temperature”

Value (5) = “Average temperature”

Unit_Diag_Area_End

Unit_Diag_Area = 28-31

Value (1) = “lower limit exceeded pressure”

Value (5) = “upper limit exceeded pressure”

Unit_Diag_Area_End

Unit_Diag_Area = 8-15

Value (2) = “sensor A”

Value (3) = “valve B”

Unit_Diag_Area_End

Unit_Diag_Area = 16-17

Value (1) = “alarm/status appearing”

Value (2) = “alarm/status disappearing”

Unit_Diag_Area_End

Since these definitions are used for both alarms and status messages their values should be different. That means different values for alarms and status messages should be used at the same position within the diagnostic field.

Figure 4 illustrates an example for a diagnosis APDU with a device related user specific diagnostic, an identifier related diagnostic and a channel related diagnostic.

bits	7	6	5	4	3	2	1	0	
octets 1	0	0	0	0	0	1	0	0	device related diagnostic:
2	device specific								meaning of the bits is defined
3	diagnostic field								manufacturer specific
4	of length 3								
5	0	1	0	0	0	1	0	0	identifier related diagnostic:
6	0	0	0	0	0	0	0	1	identifier number 0 has diagnostic
7	0	0	0	1	0	0	0	0	identifier number 12 has diagnostic
8	0	0	0	0	0	1	0	0	identifier number 18 has diagnostic
9	1	0	0	0	0	0	0	0	channel related diagnostic: identifier number 0
10	0	0	0	0	0	0	1	0	channel 2
11	0	0	1	0	0	1	0	0	overload, channel bit organized
12	1	0	0	0	1	1	0	0	identifier number 12
13	0	0	0	0	0	1	1	0	channel 6
14	1	0	1	0	0	1	1	1	upper limit value exceeded, channel word organized
	msb								

Figure 4 – Example of Ext_Diag_Data in case of the basic diagnosis format

5.19 Example of Chk_Cfg-REQ-PDU

Figure 5 illustrates an example for a Chk_Cfg APDU with a special identifier format.

bits	7	6	5	4	3	2	1	0	
octets 1	1	1	0	0	0	0	1	1	input/output, 3 octets manufacturer specific data
2	1	1	0	0	1	1	1	1	consistency, output, 16 words
3	1	1	0	0	0	1	1	1	consistency, input, 8 words
4	manufacturer								
5	specific								
6	data								
	msb								

Figure 5 – Example of a special identifier format

5.20 Examples of Chk_Cfg-REQ-PDUs with DPV1 data types

The data types are coded into the manufacturer-specific part.

Simple data types are coded in a octet which contains the code of the data type.

Sequences of data types are coded as a list of simple data types. The sum of the lengths of the data types has to correspond to the input- or output data length. Data types with variable length (visible string, octet string, time of day, time difference) cannot be handled in a sequence, but as single element. Comments may be added at the end of the list of the data types.

Figure 6 shows the special identifier format for an analogue input block containing a value (Floating Point), Status (Unsigned8) and Comment (3 Octets).

bits	7	6	5	4	3	2	1	0	
octets 1	0	1	0	0	0	1	0	1	Manufacturer-specific format
2	1	0	0	0	0	1	0	0	Consistency, 5 Octets (Input)
3	0	0	0	0	1	0	0	0	Data type Floating Point
4	0	0	0	0	0	1	0	1	Data type Unsigned8
5	Manufacturer								Comment
6	Specific								Comment
7	Data								Comment
	msb								

Figure 6 – Example of a special identifier format with data types

Figure 7 shows the special identifier format for an analogue output block where the output value can be read back.

The representation of this analogue output block is structured as follows:

a) Output Value (Floating Point)
b) Output Status (Unsigned8)
c) read back Value (Floating Point)
d) read back Status (Unsigned8)
e) comment (1 octet)

bits	7	6	5	4	3	2	1	0	
octets 1	1	1	0	0	0	1	0	1	Manufacturer-specific format
2	1	0	0	0	0	1	0	0	Consistency, 5 Octets (Output)
3	1	0	0	0	0	1	0	0	Consistency, 5 Octets (Input)
4	0	0	0	0	1	0	0	0	Data Type Floating Point
5	0	0	0	0	0	1	0	1	Data Type Unsigned8
6	0	0	0	0	1	0	0	0	Data Type Floating Point
7	0	0	0	0	0	1	0	1	Data Type Unsigned8
8	1	1	1	1	0	0	0	0	Comment
	msb								

Figure 7 – Example of a special identifier format with data types

Figure 8 shows the special identifier format for an empty slot with 3 octets comment.

The representation of this empty slot is structured as follows:

a) empty slot
b) comment (3 octets)

bits	7	6	5	4	3	2	1	0	
octets 1	0	0	0	0	0	0	1	1	Manufacturer-specific format
2	1	1	1	1	0	0	0	0	Comment
3	1	1	1	1	0	1	0	0	Comment
4	1	1	1	1	0	0	1	0	Comment
	msb								

Figure 8 – Example of an empty slot with data types

Figure 9 shows the use of the special identifier format for a multi-variable device which utilizes the user specific coding instead of data types in the data type field.

The representation of this multi variable device is structured as follows:

a) Analog Input (AI)

b) Discrete Output (DO) with setpoint (SP_D) and readback (READBACK_D)

bits	7	6	5	4	3	2	1	0	Multi variable device, AI and DO
octets 1	0	1	0	0	0	0	1	0	**Analog Input** Special Config Identifier for the example AI (0x42)
2	1	0	0	0	0	1	0	0	Extended Length Octet for input (0x84)
3	1	0	0	0	0	0	0	1	User specific instead of data type >128 (e.g block code for AI 0x81)
4	1	1	1	1	0	0	0	1	User specific (e.g. Identification of cyclic parameter for AI 0x81)
5	1	1	0	0	0	0	1	0	**Discrete Output** Special Config Identifier for the example DO (0xC2)
6	1	0	0	0	0	0	0	1	Extended Length Octet for output (0x81)
7	1	0	0	0	0	0	0	1	Extended Length Octet for input (0x81)
8	1	0	0	0	0	1	0	0	User specific instead of data type >128 (e.g. block code for DO 0x84)
9	1	0	0	0	0	0	1	1	User specific instead of data type >128 (e.g. Identification of cyclic parameter for DO with SP_D and READBACK_D 0x83)
	msb								

Figure 9 – Example for multi-variable device with AI and DO function blocks

5.21 Example structure of the Data_Unit for Data_Exchange

The structure of the Data_Unit for the service Data_Exchange is defined by the identifiers which are passed to the DP-slave with the service Chk_Cfg at configuration.

The following example illustrates how the data in the Data_Unit will be transferred, corresponding to the given identifiers. Figure 10 shows the data which shall be sent by Data_Exchange and their order as defined by the service Chk_Cfg.

1. ID	2. ID	3. ID	4. ID	5. ID	6. ID	7. ID	
1 B_I	1 W_I	2 W_O	2 B_IO	1 B_O	2 W_I	3 W_A	

Figure 10 – Identifiers (ID)

Figure 11 shows the type and length of the data corresponding to their Identifiers and the direction of their transfer, meaning whether they are Input or Output.

1 B_I: 1 Octet input;
1 B_O: 1 Octet output;
2 B_IO: 2 Octet out- and input
1 W_I: 1 Word input
1 W_O: 1 Word output
etc.

Figure 11 – Identifier list

The sequence of data in the Data_Unit is shown in Figure 12.

Data_Exchange.req

Bit	msb 7	6	5	4	3	2	1	0
octets								
1	Word_high				(3. ID)			
2	Word_low				(3. ID)			
3	Data				(4. ID)			
4	Data				(4. ID)			
5	Data				(5. ID)			
6	1.Word_high				(7. ID)			
7	1.Word_low				(7. ID)			
8	2.Word_high				(7. ID)			
9	2.Word_low				(7. ID)			
10	3.Word_high				(7. ID)			
11	3.Word_low				(7. ID)			
12	etc.							
…								
n								

Data_Exchange.res

Bit	msb 7	6	5	4	3	2	1	0
octets								
1	Data				(1. ID)			
2	Word_high				(2. ID)			
3	Word_low				(2. ID)			
4	Data				(4. ID)			
5	Data				(4. ID)			
6	1.Word_high				(6. ID)			
7	1.Word_low				(6. ID)			
8	2.Word_high				(6. ID)			
9	2.Word_low				(6. ID)			
10	etc.							
11								
12								
…								
n								

Figure 12 – Structure of the Data_Unit for the request- and response-DLPDU

For the configuration identifier “empty slot” no data will be transferred in the Data_Unit.

6 FAL protocol state machines

6.1 Overall structure

6.1.1 Fieldbus Service Protocol Machines (FSPM)

The FSPM State Machines co-ordinate the underlying state machines used for processing of the various services and application relations.

There exists one state machine for each device type (FSPMS for DP-slave, FSPMM1 for DP-master (Class 1) and FSPMM2 for DP-master (Class 2).

6.1.2 Master to Slave cyclic (MS0)

The MSCY1 State Machines are responsible for the cyclic data transfer between the DP-master (Class 1) and one DP-slave (MSCY1M/MSCY1S). This comprises the parameterisation, configuration, diagnostic and the data exchange as well as the Global Control commands. There exist one MSCY1S for a DP-slave and up to 125 MSCY1M for a DP-master (Class 1). To subscribe Input Data objects a DP-slave use up to 125 SSCY1S.

6.1.3 Master (class 1) to Slave acyclic (MS1)

The MSAC1 State Machines (MSAC1M, MSAC1S) are responsible for the acyclic data transfer between the DP-master (Class 1) and one DP-slave. The MS1 communication relationship is coupled to the cyclic communication relationship and processes the services Read, Write and Alarm Ack as well as the Load Region and Function Invocation services.

Additionally the MSAL1M state machine is used for alarm handling at the DP-master.

There exist one MSAC1S for a DP-slave and up to 125 MSAC1M/MSAL1M for a DP-master (Class 1).

6.1.4 Master (class 2) to Slave acyclic (MS2)

The MS2 State Machines (MSAC2M, MSRM2S, MSAC2S) are responsible for the acyclic data transfer between the DP-master (Class 2) and one DP-slave. The MS2 communication relationship processes the services Read, Write and Data_Transport as well as the Load Region and Function Invocation services.

The Master Slave Resource Manager (MSRM2S) is responsible for the assignment of a DLSAP and the related resources to each established MSAC2 communication relationship in the DP-slave. The Master Slave Resource Manager additionally starts the appropriate MSAC2S State Machine.

At the DP-master (Class 2) an arbitrary number of MSAC2M state machines exist. The maximum number of MSAC2M state machines is given by the maximum number of DP-slaves (=125) where each DP-slave provides the maximum number of MS2 CRs (=49).

6.1.5 Master to Slave clock synchronisation (MS3)

By means of the MSCS1 State Machines the clock synchronisation can be accomplished between the DP-master (Class 1) and the DP-slaves.

There exist at most one state machine at DP-master (Class 1) and one at DP-slave.

6.1.6 Master Master acyclic (MM1/MM2)

By means of the MMAC State Machines the Master_Diag, the Master Parameter Set, Slave Parameter Set, Bus Parameter Set can be transferred between the DP-master (Class 1) and the DP-master (Class 2).

There exist at most one state machine at DP-master (Class 1) and one at DP-master (Class 2).

6.1.7 DLL Mapping Protocol Machines (DMPM)

The DL Mapping Protocol Machines (DMPM) connects the other State machines and Layer 2. DMPM provides the coordination of all state machines concerning the configuration and error handling of the Data Link Layer Usage. The functions are mapped by the DMPM to the DLL and DLM services of Layer 2. The DMPM generates the necessary Layer 2 parameters of the service, receives the confirmations and indications from Layer 2 and passes them to the appropriate DMPM-User.

There exist one state machine for each device type (FSPMS for DP-slave, FSPMM1 for DP-master (Class 1) and FSPMM2 for DP-master (Class 2).

Interface between DMPM and Layer 2

The DMPM is directly put upon the DLL User – DLL interface and on the DLMS User – DLM interface as described in Data Link Layer Service Definitions (Refer to DL Service Definition in IEC 61158-3-3).

The DMPM uses for the mapping of the DMPM functions on the Layer 2 the following services:

a) DLL services
   - Send and Request Data with Reply (SRD)
   - Send Data with No Acknowledge (SDN)
   - Send and Request Data with Multicast Reply (MSRD)
   - Clock Synchronization (CS)

b) DLM services
   - Reset
   - Set Value
   - Get Value
   - Event
   - SAP Activate
   - SAP Activate Responder
   - SAP Deactivate

6.2 Assignment of state machines to devices

Table 48 shows the assignment of State Machines to devices. Some State Machines are always required and thus marked with ‘M’ (mandatory) others are marked with ‘O’ (optional). The Remarks show options in mandatory machines.

Table 48 – Assignment of state machines

Device type	Machine	Mandatory/ optional	Remarks
DP-slave	FSPMS	M	Alarm handling is optional
	MSCY1S	M	
	SSCY1S	O	
	MSAC1S	O	
	MSAC2S	O	
	MSRM2S	O	
	DMPMS	M	
DP-master (Class 1)	FSPMM1	M	
	MSCY1M	M	
	MSAL1M	O	
	MSAC1M	O	
	MMAC1	O	
	DMPMM1	M	
DP-master (Class 2)	FSPMM2	M	
	MSAC2M	O	
	MMAC2	O	
	DMPMM2	M	

6.3 Overview DP-slave

Figure 13 illustrates the general structure of the AL of a DP-slave by showing its state machines and the services used by them.

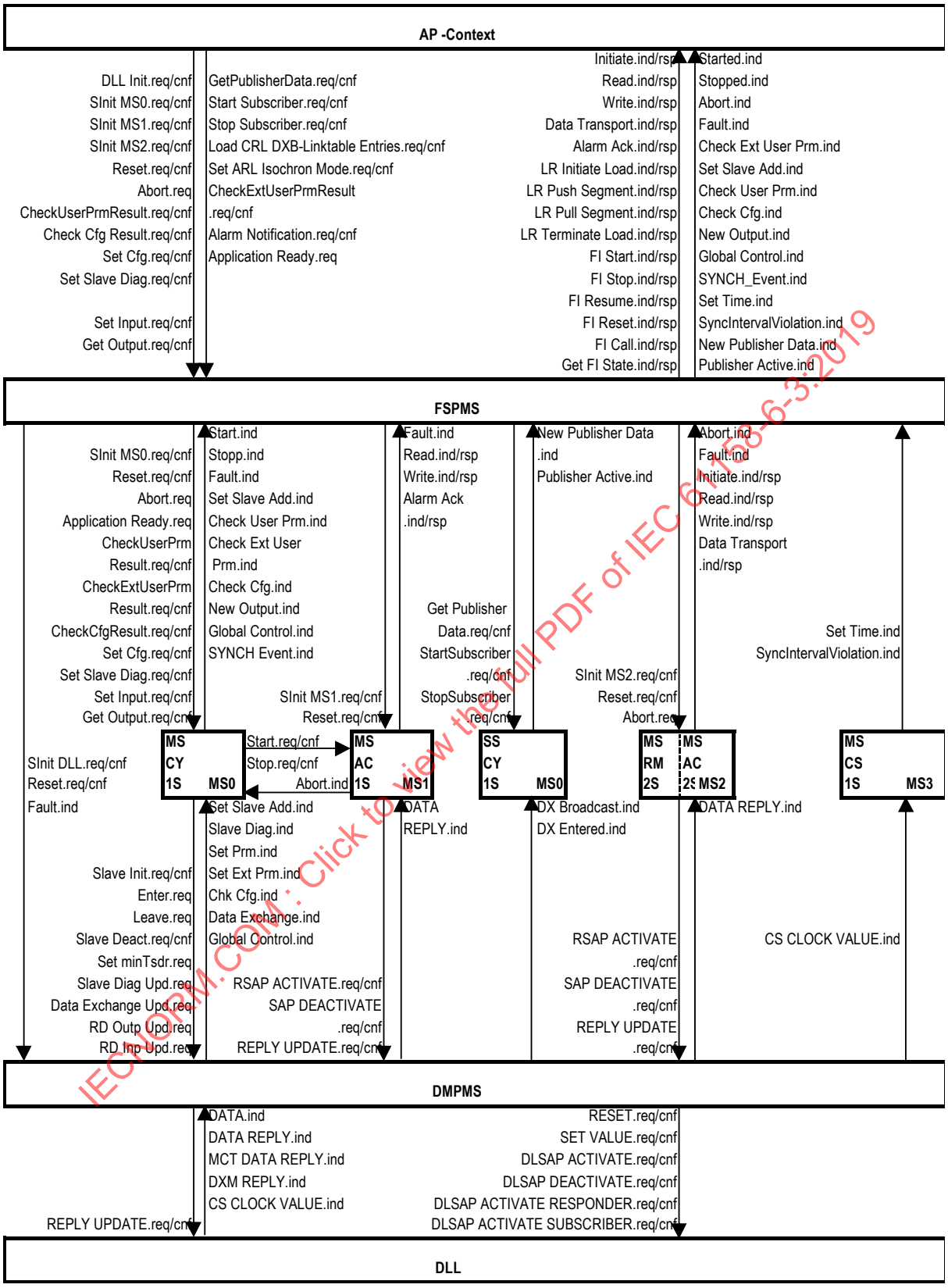


Figure 13 – Structuring of the protocol machines and adjacent layers in a DP-slave

6.4 Overview DP-master (class 1)

Figure 14 illustrates the general structure of the AL of a DP-master (Class 1) by showing its state machines and the services used by them.

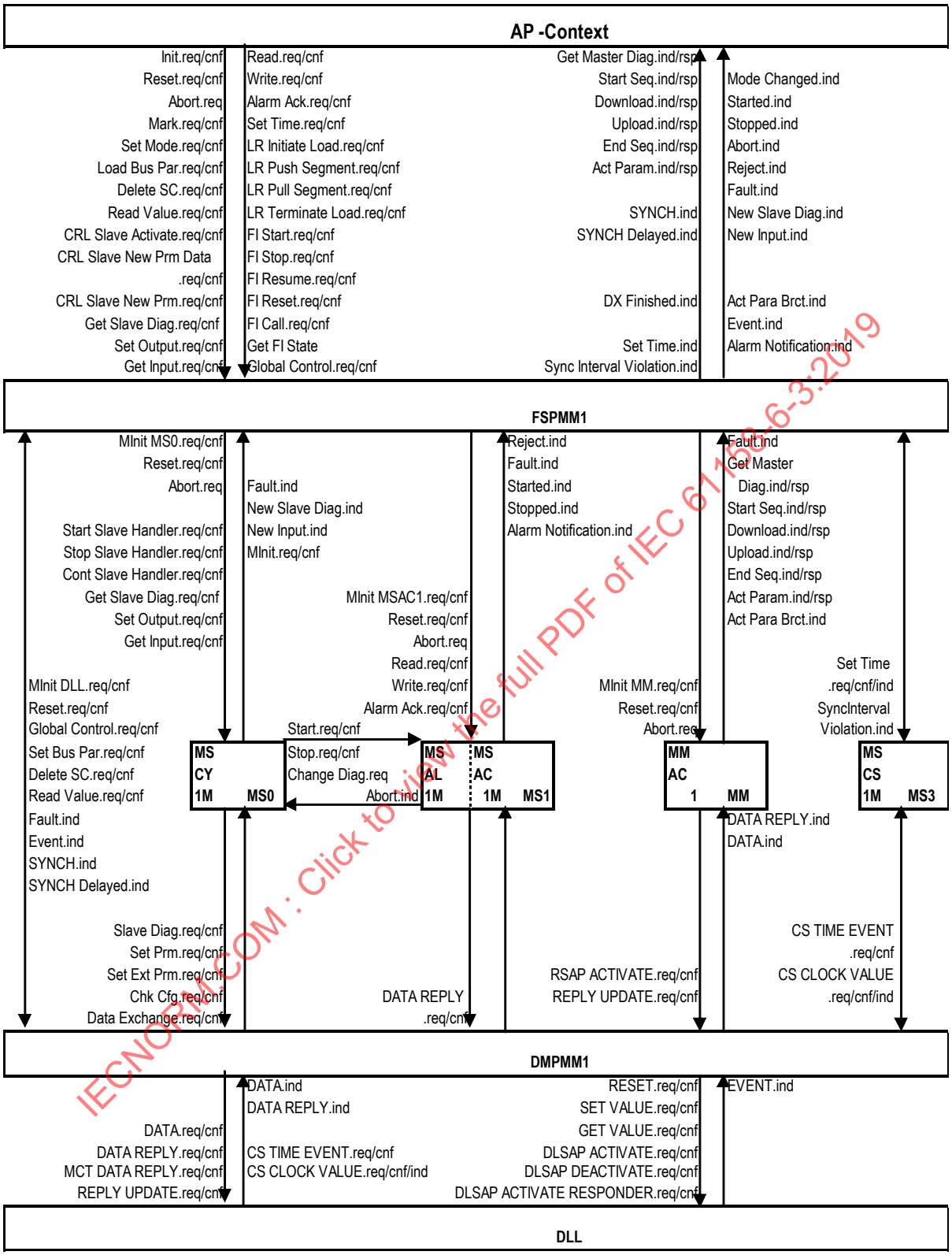


Figure 14 – Structuring of the protocol machines and adjacent layers in a DP-master (class 1)

6.5 Overview DP-master (class 2)

Figure 15 illustrates the general structure of the AL of a DP-master (Class 2) showing its state machines and the services used.

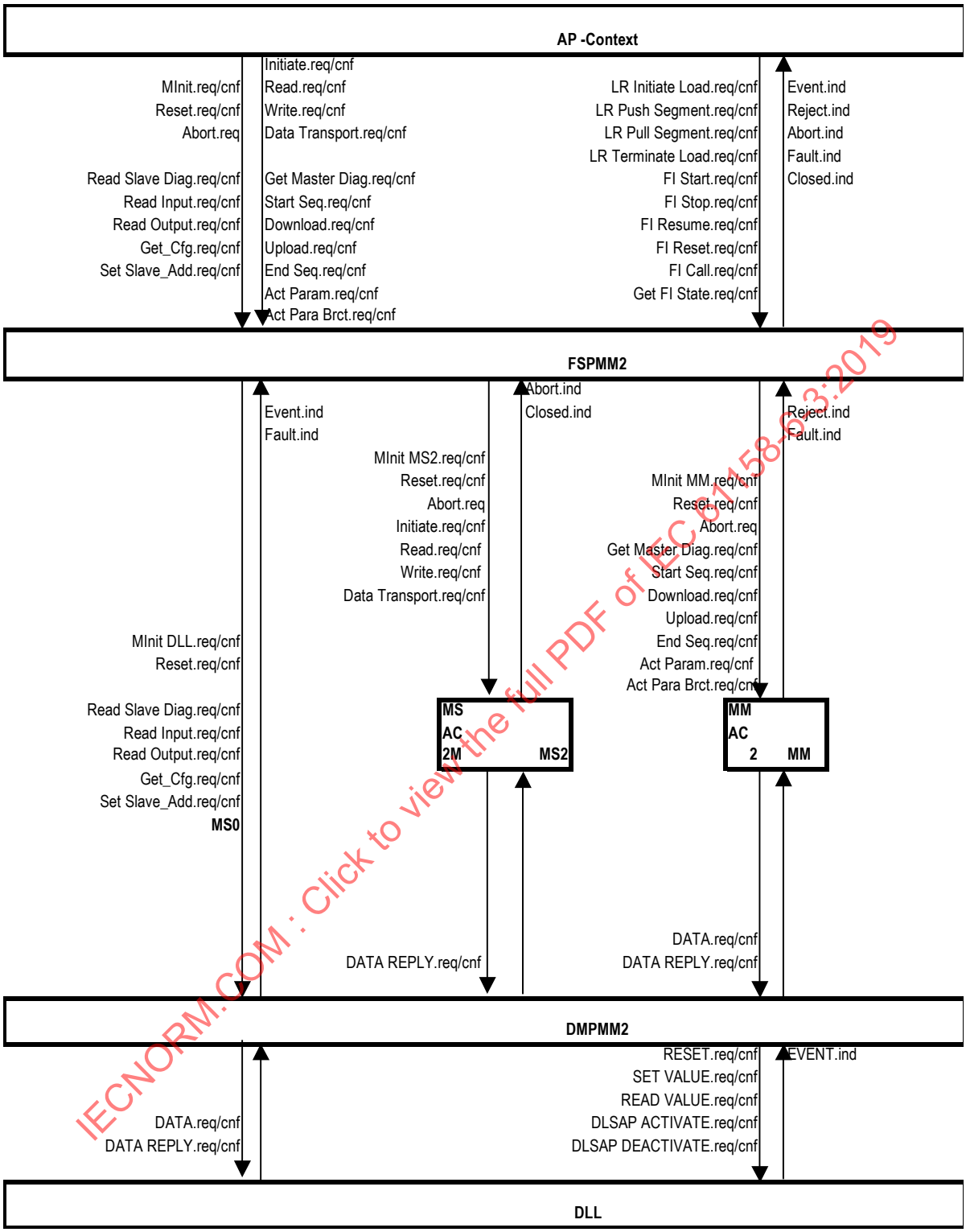


Figure 15 – Structuring of the protocol machines and adjacent layers in a DP-master (class 2)

6.6 Cyclic communication between DP-master (class 1) and DP-slave

The communication between DP-master and DP-slave is established with the sequence as shown in Figure 16:

If the DP-master wants to communicate with a DP-slave, the DP-master requests the Diag_Data of the DP-slave, in order to check the readiness for operation of the DP-slave.

This Diagnosis request will be repeated until the DP-slave responds with the requested data.

If the DP-slave responds with the requested diagnostic information, the DP-master checks whether another DP-master occupies this DP-slave. If this is not the case, the DP-slave will be parameterized and configured (Set_Prm / Chk_Cfg).

Subsequently the DP-master requests again the diagnostic data from the DP-slave. The following messages can occur:

- Parameterization or configuration error
- The DP-slave is occupied by another DP-master
- Static User diagnostics, no normal data transfer possible (for Example: start-up procedure for DP correctly executed, but some external parameterization outstanding)
- DP-slave is not ready for operation

In the cases a) and b) the DP-master returns to the starting state and checks again the readiness for operation of the DP-slave.

In the cases c) and d) the DP-master requests the diagnostic information from the DP-slave until the mentioned messages disappear.

If no error messages occur the DP-master starts the data exchange to the DP-slave. The input and output data will be transferred to and from the DP-slave. The DP-master (Class 1) indicates his own operation mode to the DP-slaves for the first time.

DP-master (Class 1)					DP-slave		
	MSCY1M	MSInt	DL		DL	WD	MSCY1S	
	/*Startup of Master*/						/*Slave offline*/	
Start_SH-> (SH=Slave Handler)		-						
	(Slave Diag)->	|					<-SET_VALUE (Initialize DLL)	
		|		SRD.req ->				
	(NA)<-	|						<-MS0_Init
<-		|					<-RSAP_ACTIVATE	
		|						
Cont_SH->		|					<- (Slave Diag-RES-PDU)	
		-					Master_Add = invalid	
	(Slave Diag)->	|					->	
		|		SRD.req ->				
		|		<-(Data)				
	(SlaveDiag-RES-PDU)<-	|						
<-		|						
Cont_SH->		|						
		-						
	(Set_Prm-REQ-PDU)->	|						
		|		SRD.req ->				
		|		<-(OK)				
	(OK)<-	|				-	check User Prm ->	
<-		|				|	<- (Slave Diag-RES-PDU)	
		|				|	Master_Add = M_Add	
		|				|	Prm_Req = False	
		|				|	Station_Not_Ready = True	

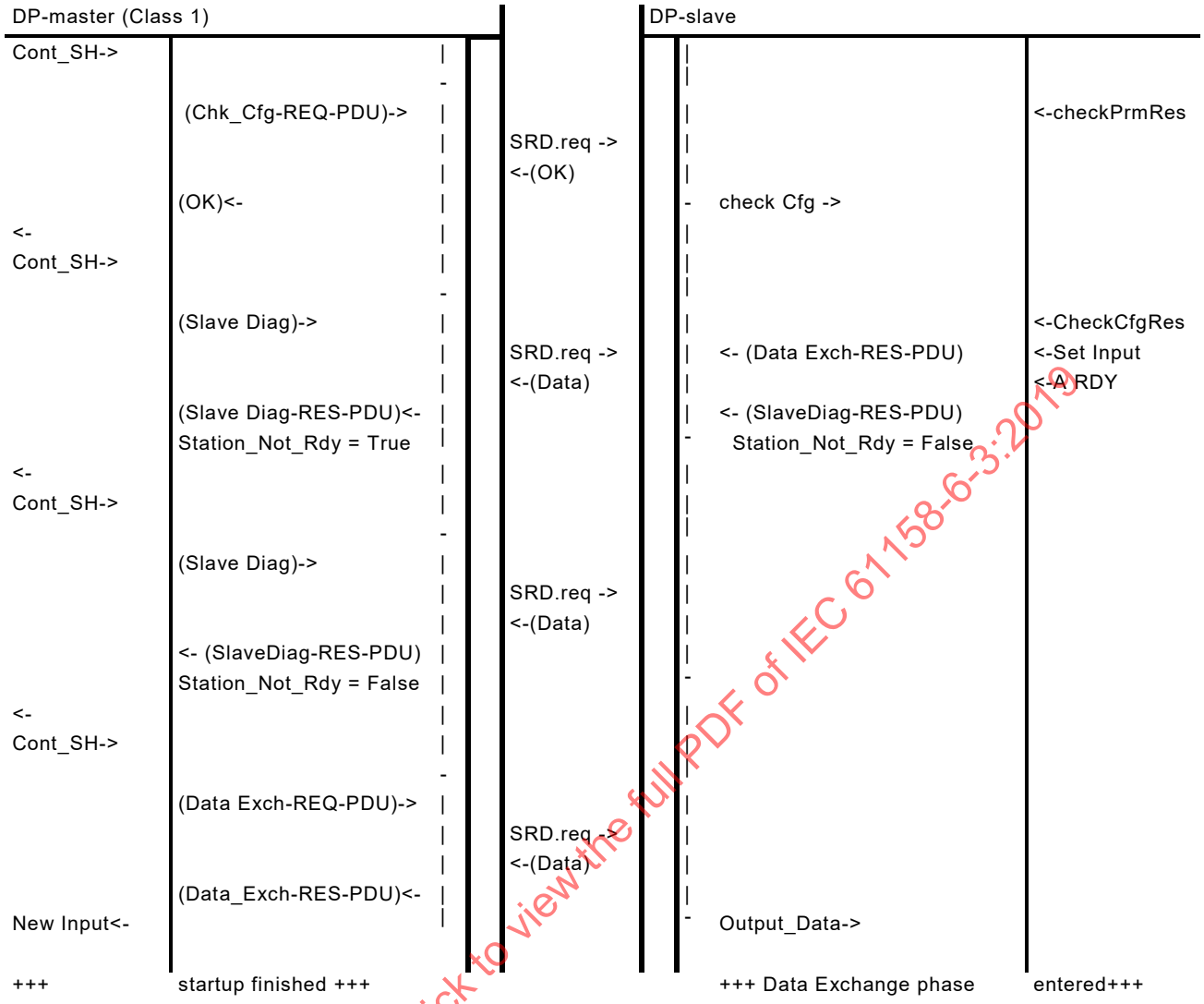


Figure 16 – Sequence of the communication between DP-master and DP-slave

6.7 Acyclic communication between DP-master (class 2) and DP-master (class 1)

The communication between DP-master (Class 2) and DP-master (Class 1) is handled in a fixed way (Request-Poll-Response), see Figure 17, with one exception.

This exception is the function Act_Para_Brct, which is transferred as a multicast from the DP-master (Class 2) to the corresponding DP-master (Class 1).

Communication is always initiated by the DP-master (Class 2). The DP-master (Class 2) sends a request to the DP-master (Class 1) and waits for the response. The service is processed by the MMAC2 of the DP-master (Class 2). After the receipt of the response the DP-master (Class 2) can make a new request.

At one point of time the DP-master (Class 1) can only communicate with one DP-master (Class 2).

The DP-master communication is initiated by a xxx.req from the DP-master (Class 2). The MMAC1 passes this request to Layer 2 and checks the status after receipt of the DMPM-response. If the response status is NR (no reply data) it will continue with a new SRD with no L_sdu to get response data. This procedure will be repeated by the DP-master (Class 2) until

an erroneous response occurs,

or a SRD_RES_PDU with response data is received,

or the Timer T1 expires.

The result will then be passed to the User of the DP-master (Class 2).

The timer T1 will be started after receipt of the xxx.req in the MMAC2 of the DP-master (Class 2) and will be stopped after release of the xxx.cnf. The value loaded in the timer T1 depends on the DLL parameter and the performance of the DP-master (Class 1).

The responder (DP-master (Class 1)) waits after a reset for a request. If the MMAC1 of the DP-master (Class 1) receives a valid request, this request is passed to the User by the corresponding xxx.ind. At this point of time an access protection was set to the local service access point (SAP) for the communication with the DP-master (Class 2). This is necessary to make sure that the response data will be sent to the right DP-master (Class 2).

After receipt of the response from the User (xxx.rsp) the response data will be passed with a REPLY_UPDATE.req to DMPM. Additionally, a timer will be started which controls the request for the response. If this timer expires and the response data are still not fetched by the DP-master (Class 2), the MMAC1 deletes the response data because the MMAC1 presumes that the DP-master (Class 2) is no longer available. In this case the access protection for the DP-master (Class 2) will be cancelled. This will also be done in the case of a single transaction if a DATA_REPLY.ind indicates that the response data are fetched by the DP-master (Class 2). The value loaded in the timer of the DP-master (Class 1) depends on the DL parameter and the performance of the DP-master (Class 2).

In case of Master-Master communication a sequence of requests can put into parentheses. Therefore the access protection will be set by the “Start_Seq” and released by the “End_Seq”. To avoid errors the time between two requests will be supervised. The value for the time supervision is added to the Start_Seq request. With the Start_Seq.req the User can additionally send an areacode. By means of this areacode at the beginning of a sequence an area can be reserved at the User in the DP-master (Class 1).

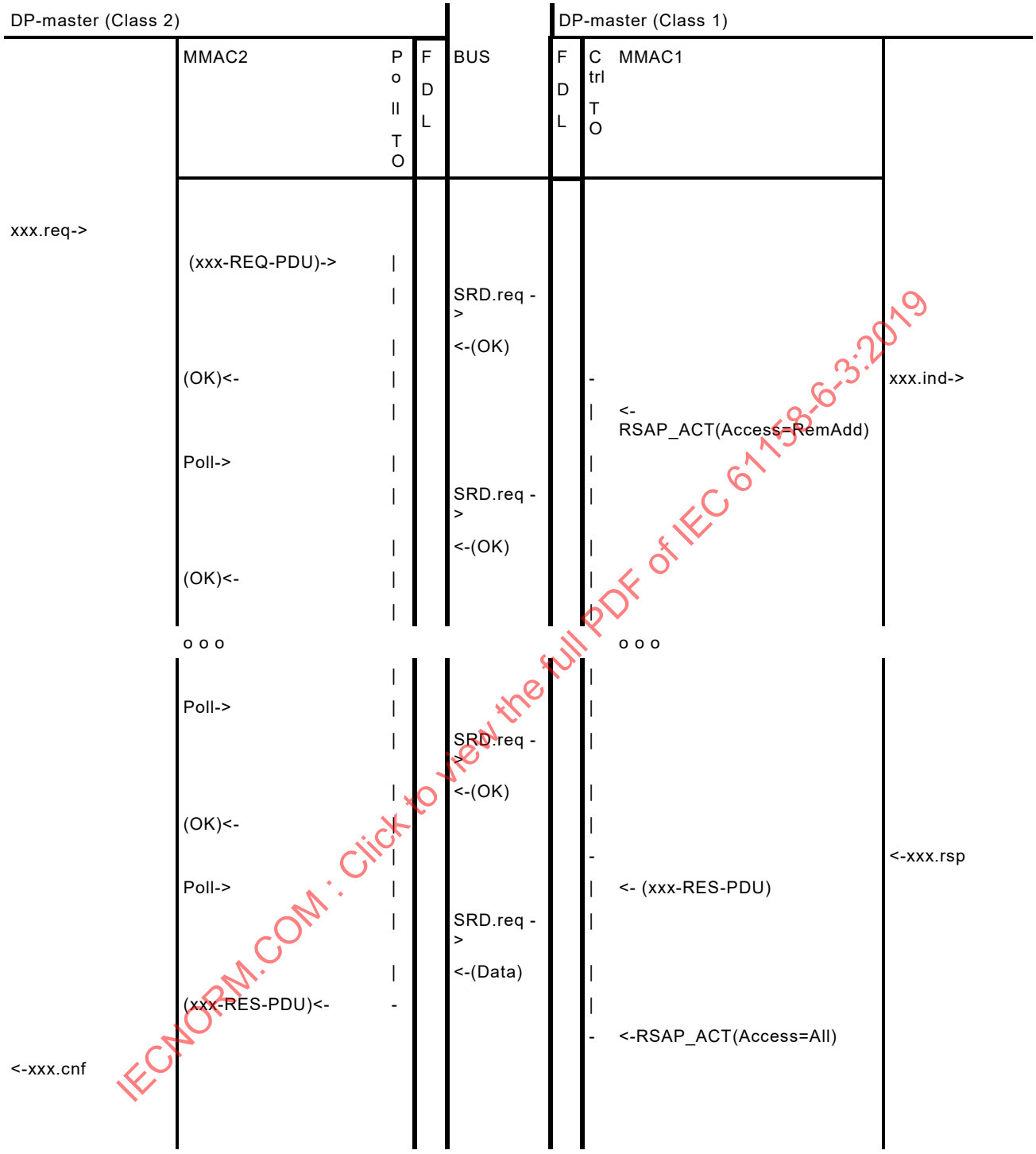


Figure 17 – Sequence of communication between DP-master (class 2) and DP-master (class 1)

6.8 Acyclic communication between DP-master (class 1) and DP-slave

The acyclic communication between DP-master (Class 1) and DP-slave is handled in a fixed way (Request-Poll-Response) on the MS1 application relationship, see Figure 18.

The communication is always initiated by the DP-master (Class 1). The connection is established if the cyclic communication MS0 AR has entered the data exchange mode. The DP-master (Class 1) issues a request and polls for the response.

The communication is initiated by a xxx.req from the DP-master (Class 1). The MSAC1M passes this request to Layer 2 and checks the status after receipt of the DMPMM1-response. If the response status is NR (no reply data) it will continue with a new SRD with a NULL-PDU to get response data. This procedure will be repeated by the DP-master (Class 1) until

– an erroneous response occurs,

– or a SRD_RES_PDU with response data is received.

The result will then be passed to the User of the DP-master (Class 1) by means of a corresponding xxx.cnf.

The responder (DP-slave) waits for a request. If the MSAC1S of the DP-slave receives a valid request, this request is passed to the User by the corresponding xxx.ind.

After receipt of the response from the User (xxx.rsp) the response data will be passed with a REPLY_UPDATE.req to DMPMS.

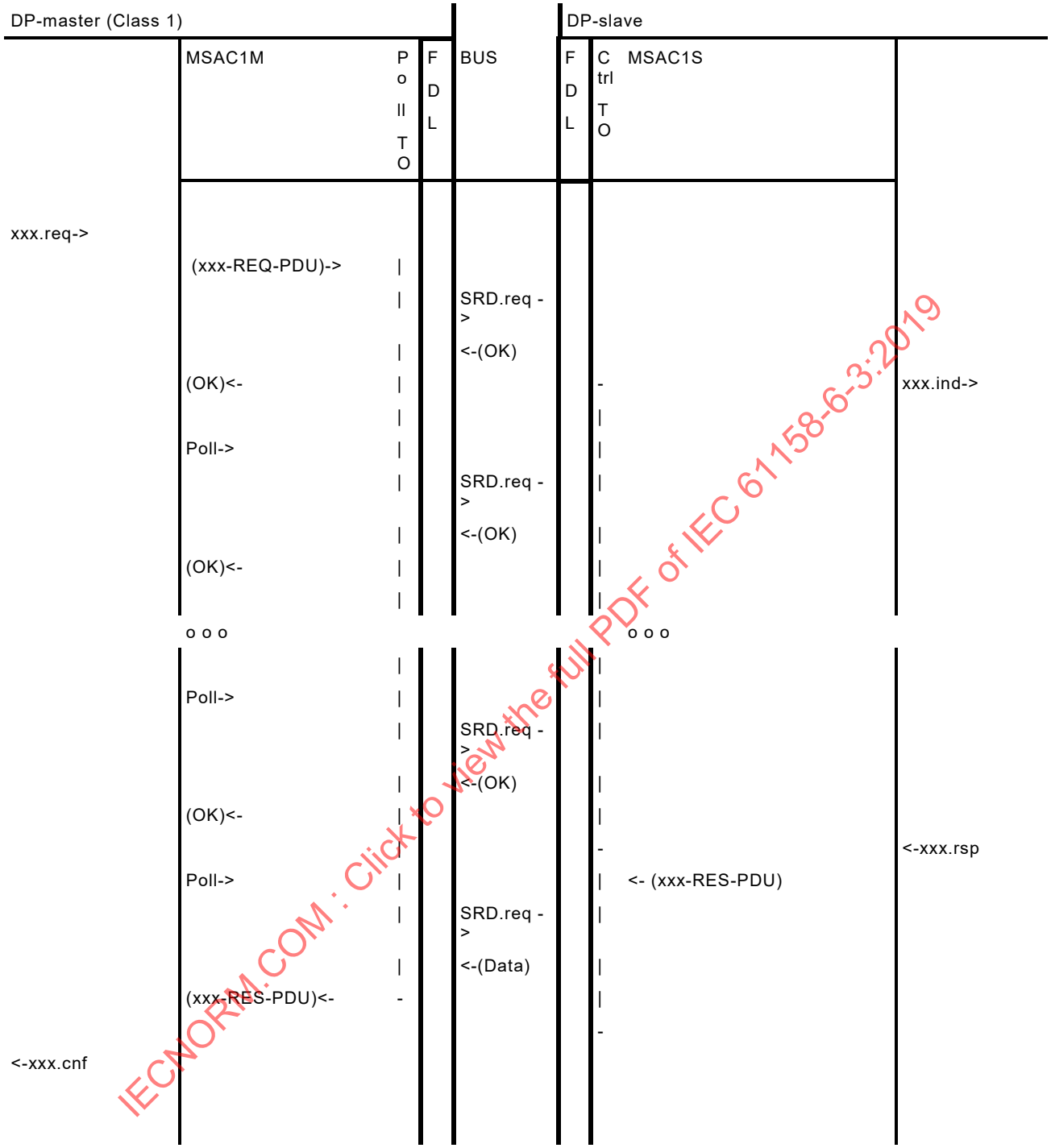


Figure 18 – Sequence of acyclic communication between DP-master (class 1) and DP-slave

6.9 Application relationship monitoring

6.9.1 Monitoring of the MS0 – AR

6.9.1.1 General

In industrial control systems there is a need for checking the correct functioning of each individual part. In such a system a medium failure or a defect in a bus master shall not cause errors in the peripheral devices. In the above mentioned cases, a peripheral device shall

switch to a safe state. Additionally, the User of a bus master (Class 1) has to be informed about transmission breakdowns after a certain amount of time.

6.9.1.2 Control interval at the DP-slave

Watchdog control

This Timer, restarted by received Requests on the bus master side, will set the outputs of a bus slave to the safe state after the expiration of the timer.

Rules for issuing Check User Prm Result.req and Check Cfg Result.req

The order of issuing these primitives shall be the same as the order of invocation of Check User Prm.ind and Check Cfg Result.ind primitives by MSCY1S.

6.9.1.3 Control intervals at the DP-master (class 1)

Min_Slave_Interval

This monitoring time specifies the smallest allowed period of time between two Slave poll cycles. This ensures that the sequence of function requests from the bus master can be handled by the bus slave. This period of time will be complied by the bus master (Class 1) for every master-slave function with exception of the function Global_Control. For the function Global_Control the User is responsible for compliance with Min_Slave_Interval.

Data_Control_Time

The Data_Control_Time defines the time in which a DP-master (Class 1) checks the data transfer to its associated DP-slaves. Furthermore, in half of this time the DP-master (Class 1) indicates his operation mode by means of a Global Control Service to the dedicated DP-slaves .

The Master parameter set contains the Data_Control_Time.

Rule

$$T_{WD} > T_{TR} \quad (2)$$

$$T_{WD} > \text{Min\_Slave\_Interval} \quad (3)$$

$$\text{Data\_Control\_Time} \geq 6 \cdot T_{WD} \quad (4)$$

6.9.2 Monitoring of the MS2 – AR

The objective of the connection monitoring is to detect the failure of the communication partner. Idle PDUs are exchanged to retrigger the monitoring timer of the communication partner.

The Idle service cycle initiated by a bus master is called Master-Idle. The Idle service cycle initiated by the Slave is called Slave-Idle.

All monitoring and idle sequences are controlled with a single timebase “Send Timeout”.

Send_Timeout

Each MSAC2 communication relationship is supervised by a Timer (Send_Timeout).

If this Timer expires the bus master (Class 2) aborts the MSAC2-communication relationship. Additionally the User of a bus master (Class 2) has to be informed.

This Timer has to be set up according to

a) the network load

b) the number of parallel connections per DP-master

Rule

$$T_{Send\_TO(TS)} \geq \left(\sum ATCyc\right) \times Num\_of\_max(TS) \quad (5)$$

where

ATCyc means delay of all activities of a token cycle,

Num_of_max means number of maximal acyclic connections.

NOTE The Send_Timeout parameter in the Initiate service is coded as an unsigned integer value, each unit represents 10ms. Due to Timer errors and the round-up of integer, the next possible integer plus one has to be used as Send_Timeout.

It is assumed that one acyclic interaction is executed within a Token cycle at each DP-master. If more than one can be executed, the Send_TO may be divided by the number of acyclic interactions per token cycle.

Figure 19 shows an example for the establishment of an MS2 connection between a Master (Class 2) and a Slave. The connection monitoring starts immediately after this.

Master				BUS	Slave			
	MSAC2M	Timer	FDL		FDL	Timer	MSRM2S/MSAC2S	
	/*Connection Establishment*/							<<<RM_SAP used
Initiate.req->	SRD(Initiate-REQ-PDU) ->	S						
		S		SRD.req ->				
	(NA) <-	S					MSAC2S_Init(all C2-SM started)	<- RM_Init.req
	<- Abort						<- SAP_Activate(RM_SAP)	
Initiate.req->							<- RPL_UPD (RM-REQ-PDU)	
	SRD(Initiate-REQ-PDU)->	S						
		S		SRD.req ->				
		S		<- (Data)			-> Initiate-REQ-PDU	
		S					<- RPL_UPD(RM-REQ-PDU)	
	<- RM-REQ-PDU	S					<- SAP_Activate(Server_SAP)	
								<<<Server_SAP used
	Poll.req ->	R				U	Initiate.ind ->	
		R		SRD.req->		U		
		R		<-(no Data)		U		
		R				U		
		R		SRD.req->		U		
		R		<-(no Data)		U		<- Initiate.res(+)
		R				F	<- RPL_UPD(Initiate-RES-PDU)	
		R		SRD.req ->		F		
		R		<- (Data)		F		
	Initiate-RES-PDU<-	R				I	-> Initiate send	
	<- Initiate.cnf	S				I		
		S				I	+++Connection open+++	
		S				I		

Figure 19 – Example for connection establishment on MS2

Phase a) Connection monitoring during pending MS2 service response or running Master-Idle:

The DP-master starts the R-Timer (receive timer) when the MS2 service request is transferred to the local DLL. The R-Timer monitors the remote MSAC2S and the local DLL.

The R-Timer is stopped if a response is received or restarted if a Slave-Idle request is received. The connection monitoring aborts the connection if a negative DL confirmation appears or the R-Timer expires.

The slave starts the U-Timer (user response timer) when a confirmed service indication is passed to the User. The U-Timer is stopped when the User provides the service response. If the U-Timer expires a Slave-Idle request is sent to retrigger the monitoring R-Timer of the Master.

The DP-slave starts the F-Timer (fetch response timer) when a service response or Slave-Idle request is passed to the local DLL. The F-Timer is stopped when the master has fetched the PDU. If the F-Timer expires the connection monitoring aborts the connection.

The connection monitoring of phase a) is shown in Figure 20.

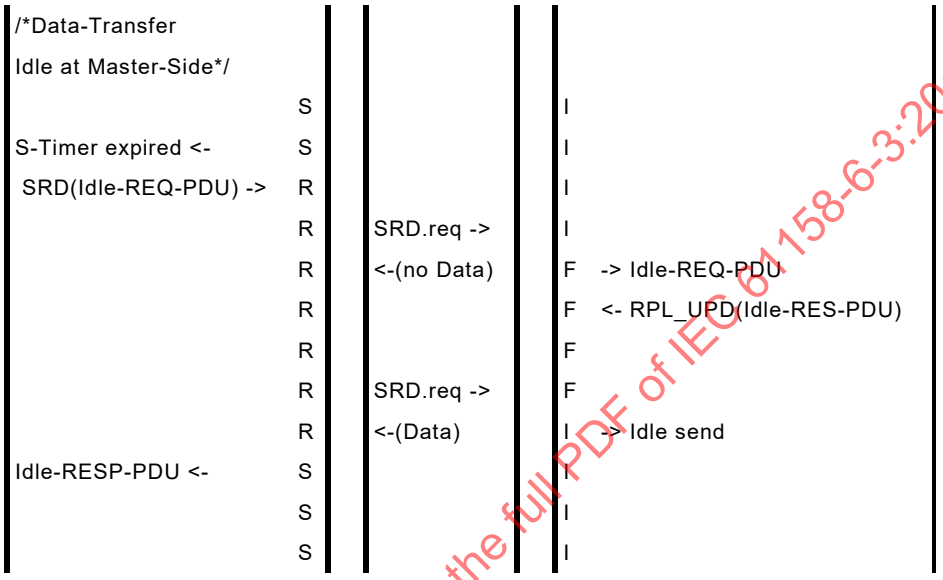


Figure 20 – Idle at master-side on MS2

Phase b) Connection monitoring without pending service response:

The bus master starts the S-Timer (send timer) if a response PDU or an Idle-PDU is received. The S-Timer is stopped if a service request is provided by the User. If the S-Timer expires a Master-Idle request is sent to stop the monitoring I-Timer of the Slave.

During pending Master-Idle responses the connection monitoring is handled as described in Phase a).

The Slave starts the I-Timer (indication timer) when the reply update buffer has been fetched by the Master. The I-Timer is stopped if a Master-Idle request or a service indication is received. If the I-Timer expires the connection monitoring aborts the connection.

The Master monitors the transmission of the Abort-REQ-PDU. The Master starts the S-Timer when the Abort.req is transferred to the local DLL. The S-Timer is stopped when the Abort.req is sent. If the S-Timer expires the connection is closed and the User is informed with a MSAC2M_Closed.ind.

The Slave monitors the fetching of the Abort-REQ-PDU by the Master. The Slave starts the F-Timer when the Abort.req is transferred to the local DL. The F-Timer is stopped if the Master has fetched the Abort-REQ-PDU. If the F-Timer expires the Server_SAP is deactivated and the connection is closed.

The connection monitoring of phase b) is shown in Figure 21.

Master					Slave			
	MSAC2M	Timer	FDL	BUS	FDL	Timer	MSRM2S/MSAC2S	
	/*Data-Transfer							
	Idle at Slave-Side*/							
Read.req->		S				I		
	SRD(Read-REQ-PDU)->	R				I		
		R		SRD.req ->		I		
		R		<-(no Data)			-> Read-REQ-PDU	
	Poll.req ->	R				U	Read.ind ->	
		R		SRD.req ->		U		
		R		<-(no Data)		U		
		R				U	-> U-Timer expired	
		R				F	<- RPL_UPD(Idle-REQ-PDU)	
		R				F		
		R		SRD.req ->		F		
		R		<-(Data)		I	-> Idle send	
	Idle-REQ-PDU<-	R				I		
	SRD(Idle-RES-PDU) ->	R				I		
		R		SRD.req ->		I		
		R		<-(no Data)		U	-> Idle-RES-PDU	
		R				U		
		R		SRD.req ->		U		<- Read.res
		R		<-(no Data)		F	<- RPL_UPD(Read-RES-PDU)	
		R				F		
		R		SRD.req ->		F		
		R		<-(Data)		I	-> Response send	
	Read-RES-PDU <-	R				I		
	<- Read.cnf	S				I		
		S				I		

Figure 21 – Idle at slave-side on MS2

7 AP-context state machine

There is no AP-Context State Machine defined for this Protocol.

8 FAL service protocol machines (FSPMs)

8.1 FSPMS

8.1.1 Primitive definitions

8.1.1.1 Primitives exchanged between FSPMS and AP-Context

Table 49 shows the service primitives including their associated parameters issued by the AP-Context and received by the FSPMS.

Table 49 – Primitives issued by AP-Context to FSPMS

Primitive name	Source	Associated parameters	Functions
DLL Init DP slave.req	AP-Context	Bus Para	Refer to FAL Service Definition Type 3 Fieldbus, IEC 61158-5-3
SInit MS0.req	AP-Context	AREP	
SInit MS1.req	AP-Context	AREP	
SInit MS2.req	AP-Context	AREP	
Reset DP slave.req	AP-Context	(none)	
Abort.req	AP-Context	AREP, Subnet, Instance, Reason Code	
DP slave Application Ready.req	AP-Context	AREP	
Check User Prm Result.req	AP-Context	AREP, Prm_OK	
Check Ext User Prm Result	AP-Context	AREP, Ext_Prm_OK	
Check Cfg Result.req	AP-Context	AREP, Cfg_OK, Input Data Len, Output Data Len	
Set Cfg.req	AP-Context	AREP, Cfg Data	
Set Slave Diag.req	AP-Context	AREP, Ext Diag Overflow, Ext Diag Flag, Ext Diag Data	
Set Input.req	AP-Context	AREP, Input Data	
Get Output.req	AP-Context	AREP	
Alarm Notification.req	AP-Context	AREP, Slot Number, Alarm Type, Seq Nr, Add Ack, Alarm Specifier, Alarm Data	
Initiate.rsp(+)	AP-Context	AREP, Max Len Data Unit, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param	
Initiate.rsp(-)	AP-Context	AREP, Error Decode, Error Code 1 Error Code 2	

Primitive name	Source	Associated parameters	Functions
Read.rsp(+)	AP-Context	AREP, Length, Data	
Read.rsp(-)	AP-Context	AREP, Error Decode, Error Code 1 Error Code 2	
Write.rsp(+)	AP-Context	AREP, Length	
Write.rsp(-)	AP-Context	AREP, Error Decode, Error Code 1 Error Code 2	
Data Transport.rsp(+)	AP-Context	AREP, Length, Data	
Data Transport.rsp(-)	AP-Context	AREP, Error Decode, Error Code 1 Error Code 2	
Alarm Ack.rsp(+)	AP-Context	AREP, Slot Number, Alarm Type, Seq Nr	
Alarm Ack.rsp(-)	AP-Context	AREP, Slot Number, Alarm Type, Seq Nr	
Load ARL DP slave.req	AP-Context	Min Send Timeout, List of Resource Manager Entries, List of S_SAP_indices, List of ARL Entries	
Get ARL DP slave.req	AP-Context	(none)	
Load CRL DP slave.req	AP-Context	List of CRL Entries	
Get CRL DP slave.req	AP-Context	(none)	
Set ARL Isochronous Mode.req	AP-Context	Isochronous Mode	
Get Publisher Data.req	AP-Context	AREP, CREP	
Start Subscriber.req	AP-Context	AREP, CREP	
Stop Subscriber.req	AP-Context	AREP, CREP	
Load CRL DXB-Linktable Entries.req	AP-Context	AREP, DXB-Linktable	
Initiate Load.rsp(+)	AP-Context	AREP, Actual LR Size, Max Response Delay, Max Segment Length User Specific	
Initiate Load.rsp(-)	AP-Context	AREP, Error Code	
Pull Segment.rsp(+)	AP-Context	AREP, Segment Length, Segment Number, More Follows, Data	
Pull Segment.rsp(-)	AP-Context	AREP, Error Code	
Push Segment.rsp(+)	AP-Context	AREP	

Primitive name	Source	Associated parameters	Functions
Push Segment.rsp(-)	AP-Context	AREP, Error Code	
Terminate Load.rsp(+)	AP-Context	AREP	
Terminate Load.rsp(-)	AP-Context	AREP, Error Code	
Start.rsp(+)	AP-Context	AREP	
Start.rsp(-)	AP-Context	AREP Error Code	
Stop.rsp(+)	AP-Context	AREP	
Stop.rsp(-)	AP-Context	AREP Error Code	
Resume.rsp(+)	AP-Context	AREP	
Resume.rsp(-)	AP-Context	AREP Error Code	
Reset.rsp(+)	AP-Context	AREP	
Reset.rsp(-)	AP-Context	AREP Error Code	
Call.rsp(+)	AP-Context	AREP Result Argument	
Call.rsp(-)	AP-Context	AREP Error Code	
Get FI State.rsp(+)	AP-Context	AREP FI State	
Get FI State.rsp(-)	AP-Context	AREP Error Code	

Table 50 shows the service primitives including their associated parameters issued by the FSPMS and received by the AP-Context.

Table 50 – Primitives issued by FSPMS to AP-Context

Primitive name	Source	Associated parameters	Functions
Sinit DLL.cnf	FSPMS	(none)	Refer to FAL Service Definition Type 3 Fieldbus, IEC 61158-5-3
Sinit MS0.cnf	FSPMS	AREP	
Sinit MS1.cnf	FSPMS	AREP	
Sinit MS2.cnf	FSPMS	AREP	
Reset.cnf	FSPMS	(none)	
Check User Prm Result.cnf(+)	FSPMS	AREP	
Check User Prm Result.cnf(-)	FSPMS	AREP, Status	
Check Ext User Prm Result.cnf(+)	FSPMS	AREP	
Check Ext User Prm Result.cnf(-)	FSPMS	AREP, Status	
Check Cfg Result.cnf(+)	FSPMS	AREP	
Check Cfg Result.cnf(-)	FSPMS	AREP, Status	
Set Cfg.cnf(+)	FSPMS	AREP	
Set Cfg.cnf(-)	FSPMS	AREP	
Set Slave Diag.cnf	FSPMS	AREP	

Primitive name	Source	Associated parameters	Functions
Set Input.cnf(+)	FSPMS	AREP	
Set Input.cnf(-)	FSPMS	AREP	
Get Output.cnf(+)	FSPMS	AREP, Output Data, Clear Flag, New Flag	
Get Output.cnf(-)	FSPMS	AREP	
Started.ind	FSPMS	AREP, Actual Enabled Alarms, Alarm Sequence, Alarm Limit	
Stopped.ind	FSPMS	AREP	
Abort.ind	FSPMS	AREP, Locally Generated, Subnet, Instance, Reason Code, Additional Detail	
Fault.ind	FSPMS	AREP	
Alarm Reject.ind	FSPMS	AREP, Slot Number, Alarm Type, Seq Nr	
Set Slave Add.ind	FSPMS	AREP, New Slave Add, Ident Number, No Add Chg, Rem Slave Data	
Check User Prm.ind	FSPMS	AREP, User Prm Data	
Check Ext User Prm.ind	FSPMS	AREP, Ext User Prm Data	
Check Cfg.ind	FSPMS	AREP, Cfg Data	
New Output.ind	FSPMS	AREP, Output Data, Clear Flag	
Global Control.ind	FSPMS	AREP, Clear Command, Sync Command, Freeze Command, Group Select	
Initiate.ind	FSPMS	AREP, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param	
Read.ind	FSPMS	AREP, Slot Number, Index, Length	
Write.ind	FSPMS	AREP, Slot Number, Index, Length, Data	
Data Transport.ind	FSPMS	AREP, Slot Number, Index, Length, Data	

Primitive name	Source	Associated parameters	Functions
Alarm Ack.ind	FSPMS	AREP, Slot Number, Alarm Type, Seq Nr	
Load ARL DP slave.cnf(+)	FSPMS	(none)	
Load ARL DP slave.cnf(-)	FSPMS	Status	
Get ARL DP slave.cnf(+)	FSPMS	Min Send Timeout, List of Resource Manager Entries, List of S_SAP_indices, List of ARL Entries	
Get ARL DP slave.cnf(-)	FSPMS	Status	
Load CRL DP slave.cnf(+)	FSPMS	(none)	
Load CRL DP slave.cnf(-)	FSPMS	Status	
Get CRL DP slave.cnf(+)	FSPMS	List of CRL Entries	
Get CRL DP slave.cnf(-)	FSPMS	Status	
Set ARL Isochronous Mode.cnf(+)	FSPMS		
Set ARL Isochronous Mode.cnf(-)	FSPMS	Status	
SYNCH Event.ind	FSPMS	AREP, Status	
Publisher Active.ind	FSPMS	AREP, CREP, Status	
New Publisher Data.ind	FSPMS	AREP, CREP	
Get Publisher Data.cnf(+)	FSPMS	AREP, CREP, Data, New Flag	
Get Publisher Data.cnf(-)	FSPMS	AREP, CREP	
Start Subscriber.cnf(+)	FSPMS	AREP, CREP	
Start Subscriber.cnf(-)	FSPMS	AREP, CREP	
Stop Subscriber.cnf(+)	FSPMS	AREP, CREP	
Stop Subscriber.cnf(-)	FSPMS	AREP, CREP	
Load CRL DXB-Linktable Entries.cnf(+)	FSPMS	AREP	
Load CRL DXB-Linktable Entries.cnf(-)	FSPMS	AREP	
Set Time.ind	FSPMS	AREP, Time Value, Local Time Diff, Summertime, Accuracy, Synchronisation Active, Announcement Hour	
Sync Interval Violation.ind	FSPMS	AREP	

Primitive name	Source	Associated parameters	Functions
Initiate Load.ind	FSPMS	AREP, Slot Number, LR Index, Load Type, Load Image Size, Intersegment Request Timeout User Specific	
Pull Segment.ind	FSPMS	AREP, Slot Number, LR Index, Segment Length	
Push Segment.ind	FSPMS	AREP, Slot Number LR Index, Segment Length Segment Number, More Follows, Data	
Terminate Load.ind	FSPMS	AREP Slot Number LR Index	
Start.ind	FSPMS	AREP Slot Number FI Index Execution Argument	
Stop.ind	FSPMS	AREP Slot Number FI Index	
Resume.ind	FSPMS	AREP Slot Number FI Index Execution Argument	
Reset.ind	FSPMS	AREP Slot Number FI Index	
Call.ind	FSPMS	AREP Slot Number Entity Number FI Index Execution Argument	
Get FI State	FSPMS	AREP Slot Number FI Index	

8.1.1.2 Parameters of FSPMS primitives

The parameters used with the primitives exchanged between the FSPMS and the AP-Context are described in FAL Service Definition Type 3 Fieldbus, see IEC 61158-5-3.

8.1.2 State machine description

This Machine is used to co-ordinate the Interactions between AP-Context and DMPMS, MSCY1S, MSAC1S, MSRM2S and MSAC2S. As Alarms requires the support of several State Machines the alarms are handled by this machine.

With the SInit_MS1.req, the local variables of the State Machines are set to the start-up values and MSAC1 State Machines is initialized.

When the MSCY1S enters the Data-Exchange-State (This is signalled by a start.ind) alarm messages can be generated by the User.

The parallel processing of alarm messages is limited by:

a) the number of Alarm_Classes (Alarm_Sequence = False)
b) the number of Alarms accepted by the Master (Alarm_Sequence = True)
c) the number of Alarms handled by this Slave (Alarm_Sequence=True).

Alarms may be processed only if the appropriate Alarm_Class is enabled by the DP-master. An Alarm issued by an Alarm_Notification will be transmitted via a diagnosis (MSCY1S) to a DP-master (Class 1). The Acknowledgement will be received by an Alarm Ack service processed by MSAC1S.

The State Machine needs local variables which are described below. Furthermore the machine uses functions and macros which are listed in 8.1.4.

Local variables

Alarm_Sequence
(Boolean)
This variable indicates whether

a) only one alarm of a specific Alarm_Class can be active at one time (Alarm_Sequence = False), or

b) several alarms (2 to 32) of any type can be active at one time (Alarm_Sequence = True).

Initial_Alarm_Sequence
(Boolean)

This variable stores the basic ability of the Slave.

Outstanding_Alarm_TABLE
(Array 0…7, 0…31 of Alarm_Status, Alarm_Type)

Alarm_Status: pending, not_pending

Alarm_Type: 1 to 6, 32 to 126

The Alarm_State_Table is a two dimensional array with 7 * 32 elements. Each element within the array stores information about the actual state of any alarm. The array is addressed with the Alarm_Class calculated from the Alarm_Type and the Seq_Nr of the alarm.

Alarm_Max
(Unsigned8)

This variable indicates the maximum number of parallel alarms of this Slave.

Range: 1 to 32

Alarm_Limit
(Unsigned8)

This variable contains the maximum number of Alarms allowed by the actual Master-Slave-connection.

Range: 1 to 32

Alarm_Decode
(Array 0 to 7 of Unsigned8)

Decoding the number of parallel alarms:

Alarm_Decode[0] = 1

Alarm_Decode[1] = 2

Alarm_Decode[2] = 4

Alarm_Decode[3] = 8

Alarm_Decode[4] = 12

Alarm_Decode[5] = 16

Alarm_Decode[6] = 24

Alarm_Decode[7] = 32

Alarm_Count
(Unsigned8)

This counter contains the number of alarms (0 to 32) which have actually been sent by the DP-slave. The counter is only used in the sequence mode.

Range: 0 to Alarm_Limit

Req_Alarm_FIFO
The Req_Alarm_FIFO is used to store the Alarms which are still to be sent. Each element consists of an Alarm-PDU. The FIFO shall be able to hold up to 32 respectively Alarm_Max entries.

Actual_Enabled_Alarms

(Array of Bits)

This variable indicates the type of alarms which are actually supported by the Master.

Bit 0 = reserved
Bit 1 = reserved
Bit 2 = Update_Alarm
Bit 3 = Status_Alarm
Bit 4 = Manufacturer_Specific_Alarm
Bit 5 = Diagnostic_Alarm
Bit 6 = Process_Alarm
Bit 7 = Pull_Plug_Alarm

Alarms_Supported
(Array of Bits)

This variable indicates the type of alarms which are supported by the Slave.

Bit 0 = reserved
Bit 1 = reserved
Bit 2 = Update_Alarm
Bit 3 = Status_Alarm
Bit 4 = Manufacturer_Specific_Alarm
Bit 5 = Diagnostic_Alarm
Bit 6 = Process_Alarm
Bit 7 = Pull_Plug_Alarm

Local_Diag_Buffer
(Octet-String)

This local variable is used to store the Diag-Data except the first 6 Octets and the Alarm- and Status-PDUs.

L_Ext_Diag_Flag
(Bit)

This local variable is used to store the Ext_Diag_Flag.

Stored_Diag
(Boolean)

Flag which indicates, that new Diag_Data are stored in the Diag_Buffer while waiting for transmission of a previous Diagnosis.

Diag_Buffer

(Octet-String)

Buffer for storing a Slave-Diagnostic(Identification-Related-Diagnosis, Channel-Related-Diagnosis, Revision_Number)

Index
(Structure of Alarm_Class, Seq_Nr)

This index is used for addressing outstanding Alarm_Table.

LR_State
(Unsigned8)

This variable stores the state of the Load Region sequence.

CurrentBuffer
(Array of structure Empty, WriteLength and Data)

This buffer stores APDUs for Load Region for each AREP.

CurrentLoadType
(Unsigned8)

This variable stores whether a Pull or a Push sequence is active.

CurrentLRIndex
(Unsigned16)

This variable stores the LR type.

8.1.3 FSPMS state table

Table 51 contains the complete description of the FSPMS state machine.

Processing constraints for isochronous mode:
The time from Issuing the SYNCH.ind until the Service Requests from the MSCY1S to the DL shall be small enough to allow the MAC to send out all requests from MSCY1S within a cycle.

The following definition for a state set is used in Table 51.

t_state = W-START or W-DIA-UPD or W-FETCHED

Table 51 – FSPMS state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	Load ARL DP Slave.req (Min Send Timeout, List of Resource Manager Entries, List of S_SAP_indices, List of ARL Entries) /valid parameters => Load ARL DP Slave.cnf(+)	POWER-ON
2	POWER-ON	Load ARL DP Slave.req (Min Send Timeout, List of Resource Manager Entries, List of S_SAP_indices, List of ARL Entries) /invalid parameters => Status:= NO Load ARL DP Slave.cnf(-) (Status)	POWER-ON
3	POWER-ON	Get ARL DP Slave.req /ARL loaded => Get ARL DP Slave.cnf(+) (Min Send Timeout, List of Resource Manager Entries, List of S_SAP_indices, List of ARL Entries)	POWER-ON
4	POWER-ON	Get ARL DP Slave.req /ARL not loaded => Status:= NO Get ARL DP Slave.cnf(-) (Status)	POWER-ON
5	POWER-ON	Load CRL DP Slave.req (List of CRL Entries) /valid parameters => Load CRL DP Slave.cnf(+)	POWER-ON
6	POWER-ON	Load CRL DP Slave.req (List of CRL Entries) /invalid parameters => Status:= NO Load CRL DP Slave.cnf(-) (Status)	POWER-ON
7	POWER-ON	Get CRL DP Slave.req /CRL loaded => Get CRL DP Slave.cnf(+) (List of CRL Entries)	POWER-ON
8	POWER-ON	Get CRL DP Slave.req /CRL not loaded => Status:= NO Get CRL DP Slave.cnf(-) (Status)	POWER-ON
9	POWER-ON	FSPMS_SInit_MS1.req (AREP) /Alarm_Mode_Slave <> 0 => Alarm_Sequence:= True Initial_Alarm_Sequence:= Alarm_Sequence Alarm_Max:= Alarm_Decode[Alarm_Mode_Slave] Alarm_Max:= max(Alarm_Max, 7) Local_Diag_Buffer:= Default_Ext_Diag_Data L_Ext_Diag_Flag:= Default_Ext_Diag FILL (Outstanding_Alarm_TABLE.Alarm_Status, not_pending) Stored_Diag:=False Reset_Req_Alarm_FIFO() Act_Ref:= 0 MSAC1_SInit_MS1.req	W-START-C1

#	Current State	Event /Condition =>Action	Next State
10	POWER-ON	FSPMS_SInit_MS1.req (AREP) /Alarm_Mode_Slave = 0 => Alarm_Sequence:= False Initial_Alarm_Sequence:= Alarm_Sequence Alarm_Max:= 7 Local_Diag_Buffer:= Default_Ext_Diag_Data L_Ext_Diag_Flag:= Default_Ext_Diag FILL(Outstanding_Alarm_TABLE.Alarm_Status, not_pending) Stored_Diag:=False Reset_Req_Alarm_FIFO() FSPMS_User_Reset:=False Act_Ref:= 0 CurrentBuffer[AREP].Empty:=TRUE LR_State[AREP]:=W-LR-IND MSAC1_SInit_MS1.req	W-START-C1
11	W-START-C1	MSAC1S_SInit_MS1.cnf => FSPMS_SInit_MS1.cnf(AREP)	W-START
12	W-START	FSPMS_Abort.req(AREP) /AREP.AR_Type=MS0 => CurrentBuffer[AREP].Empty:=TRUE LR_State[AREP]:=W-LR-IND MSCY1S_Abort.req	W-START
13	W-START	FSPMS_Alarm_Notification.req(AREP, Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data) => FSPMS_Alarm_Notification.cnf(-)(AREP, Alarm_Type, Slot_Number, Seq_Nr, Status=No_Start)	W-START
14	W-START	FSPMS_Set_Slave_Diag.req(AREP,Ext_Diag_Flag, Ext_Diag) => L_Ext_Diag_Flag:= Ext_Diag_Flag L_Ext_Diag_Overflow:= Ext_Diag_Overflow Local_Diag_Buffer:= Ext_Diag Act_Ref:= Act_Ref + 1 MSCY1S_Set_Slave_Diag.req(Ext_Diag_Flag, Ext_Diag_Overflow, Ext_Diag, Reference:= Act_Ref) FSPMS_Set_Slave_Diag.cnf(AREP)	W-START
15	W-START	MSAC1S_Alarm_Ack.ind(Slot_Number, Alarm_Type, Seq_Nr) => MSCY1S_Abort.req FSPMS_Abort.ind(AREP)	W-START
16	W-START	MSCY1S_Set_Slave_Diag.cnf(-) (Reference) => ignore	W-START
17	W-START	MSCY1S_Set_Slave_Diag.cnf(+) (Reference) => ignore	W-START
18	W-START	MSCY1S_Start.ind(Enabled_Alarms, Alarm_Mode_Master, Diag_Flag) /(Enabled Alarms & Alarms_Supported) = 0 => FSPMS DP Slave Started.ind(AREP,Actual_Enabled_Alarms, Alarm_Sequence, Alarm_Limit)	W-START

#	Current State	Event /Condition =>Action	Next State
19	W-START	MSCY1S_Start.ind(Enabled_Alarms, Alarm_Mode_Master, Diag_Flag) / ((Enabled Alarms & Alarms_Supported) <> 0) && Alarm_Mode_Master <> 0 && Initial_Alarm_Sequence = TRUE && Diag_Flag=FALSE => Alarm_Sequence:= True Actual_Enabled_Alarms:= Enabled_Alarms Alarm_Limit:= min(Alarm_Max, Alarm_Decode[Alarm_Mode_Master]) Alarm_Count:= 0 FSPMS DP Slave Started.ind(AREP,Actual_Enabled_Alarms, Alarm_Sequence, Alarm_Limit)	W-DIA-UPD
20	W-START	MSCY1S_Start.ind(Enabled_Alarms, Alarm_Mode_Master, Diag_Flag) / ((Enabled Alarms & Alarms_Supported) <> 0) && Alarm_Mode_Master <> 0 && Initial_Alarm_Sequence =FALSE => MSCY1S_Abort.req	W-START
21	W-START	MSCY1S_Start.ind(Enabled_Alarms, Alarm_Mode_Master, Diag_Flag) / ((Enabled Alarms & Alarms_Supported) <> 0) && Alarm_Mode_Master = 0 && Diag_Flag=TRUE => Alarm_Sequence:= False Actual_Enabled_Alarms:= Enabled_Alarms FSPMS DP Slave Started.ind(AREP,Actual_Enabled_Alarms, Alarm_Sequence, Alarm_Limit)	W-FETCHED
22	W-START	MSCY1S_Start.ind(Enabled_Alarms, Alarm_Mode_Master, Diag_Flag) / ((Enabled Alarms & Alarms_Supported) <> 0) && Alarm_Mode_Master = 0 && Diag_Flag=FALSE => Alarm_Sequence:= False Actual_Enabled_Alarms:= Enabled_Alarms FSPMS DP Slave Started.ind(AREP,Actual_Enabled_Alarms, Alarm_Sequence, Alarm_Limit)	W-DIA-UPD
23	W-START	MSCY1S_Stop.ind => FSPMS_Stopped.ind(AREP)	W-START
24	W-DIA-UPD	FSPMS_Abort.req(AREP) /AREP.AR_Type=MS0 => Index:= Search_In_Outstanding_Alarm_TABLE(Alarm_Status:=pending) CurrentBuffer[AREP].Empty:=TRUE LR_State[AREP]:=W-LR-IND MSCY1S_Abort.req	RESET-P-LEAVE
25	W-DIA-UPD	FSPMS_Alarm_Notification.req(AREP, Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data) /ALARM_ENABLED=FALSE => FSPMS_Alarm_Notification.cnf(-)(AREP,Alarm_Type, Slot_Number, Seq_Nr, Status=Not_Enabled)	W-DIA-UPD
26	W-DIA-UPD	FSPMS_Alarm_Notification.req(AREP, Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data) /ALARM_ENABLED=TRUE && Alarm_Sequence=TRUE && Alarm_Count=Alarm_Limit => FSPMS_Alarm_Notification.cnf(-)(AREP, Alarm_Type, Slot_Number, Seq_Nr, Status=Limit_Expired)	W-DIA-UPD

#	Current State	Event /Condition =>Action	Next State
27	W-DIA-UPD	FSPMS_Alarm_Notification.req(AREP, Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data) /ALARM_ENABLED=TRUE && Alarm_Sequence=TRUE && Alarm_Count<Alarm_Limit && Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]=not_pending => Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]:=pending Outstanding_Alarm_TABLE.Alarm_Type[Acls(Alarm_Type),Seq_Nr]:=Alarm_Type Alarm_Count:=Alarm_Count+1 Ext_Diag_Flag:=L_Ext_Diag_Flag Ext_Diag_Overflow:=L_Ext_Diag_Overflow Ext_Diag:=(Local_Diag_Buffer, Alarm(Slot_Number,Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data)) Act_Ref:= Act_Ref + 1 MSCY1S_Set_Slave_Diag.req(Ext_Diag_Flag, Ext_Diag_Overflow, Ext_Diag, Reference:= Act_Ref) FSPMS_Alarm_Notification.cnf(+)(AREP,Alarm_Type, Slot_Number, Seq_Nr)	W-FETCHED
28	W-DIA-UPD	FSPMS_Alarm_Notification.req(AREP, Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data) /ALARM_ENABLED=TRUE && Alarm_Sequence=TRUE && Alarm_Count<Alarm_Limit && Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]= pending => FSPMS_Alarm_Notification.cnf(-)(AREP,Alarm_Type, Slot_Number, Seq_Nr, Status=Sequence_Nr_Pending)	W-DIA-UPD
29	W-DIA-UPD	FSPMS_Alarm_Notification.req(AREP, Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data) /ALARM_ENABLED=TRUE && Alarm_Sequence=FALSE && Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]= pending => FSPMS_Alarm_Notification.cnf(-)(AREP,Alarm_Type, Slot_Number, Seq_Nr, Status=Sequence_Nr_Pending)	W-DIA-UPD
30	W-DIA-UPD	FSPMS_Alarm_Notification.req(AREP, Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data) /ALARM_ENABLED=TRUE && Alarm_Sequence=FALSE && Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]= not_pending => Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]:=pending Outstanding_Alarm_TABLE.Alarm_Type[Acls(Alarm_Type),Seq_Nr]:=Alarm_Type Ext_Diag_Flag:=L_Ext_Diag_Flag Ext_Diag_Overflow:=L_Ext_Diag_Overflow Ext_Diag:=(Local_Diag_Buffer, Alarm(Slot_Number,Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data)) Act_Ref:= Act_Ref + 1 MSCY1S_Set_Slave_Diag.req(Ext_Diag_Flag, Ext_Diag_Overflow, Ext_Diag, Reference:= Act_Ref) FSPMS_Alarm_Notification.cnf(+)(AREP,Alarm_Type, Slot_Number, Seq_Nr)	W-FETCHED
31	W-DIA-UPD	FSPMS_Set_Slave_Diag.req(AREP,Ext_Diag_Flag, Ext_Diag) => if (PrmCmdAck in Ext_Diag or Stored_PrmCmdAck != NULL) Stored_PrmCmdAck:= PrmCmdAck, Ext_Diag.Red_Status:= PrmCmdAck endif L_Ext_Diag_Flag:= Ext_Diag_Flag L_Ext_Diag_Overflow:= Ext_Diag_Overflow Local_Diag_Buffer:= Ext_Diag Act_Ref:= Act_Ref + 1 MSCY1S_Set_Slave_Diag.req(Ext_Diag_Flag, Ext_Diag_Overflow, Ext_Diag, Reference:= Act_Ref) FSPMS_Set_Slave_Diag.cnf(AREP)	W-DIA-UPD
32	W-DIA-UPD	MSAC1S_Alarm_Ack.ind(Slot_Number, Alarm_Type, Seq_Nr) /Alarm_Sequence=TRUE && Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]=pending => Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]:= not_pending Alarm_Count:=Alarm_Count-1 FSPMS_Alarm_Ack.ind(Slot_Number, Alarm_Type, Seq_Nr) MSAC1S_Alarm Ack.rsp(+) (Slot Number, Alarm Type, Seq Nr)	W-DIA-UPD

#	Current State	Event /Condition =>Action	Next State
33	W-DIA-UPD	MSAC1S_Alarm_Ack.ind(Slot_Number, Alarm_Type, Seq_Nr) /Alarm_Sequence=FALSE && Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr] =pending => Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]:= not_pending FSPMS_Alarm_Ack.ind(Slot_Number, Alarm_Type, Seq_Nr) MSAC1S_Alarm Ack.rsp(+) (Slot Number, Alarm Type, Seq Nr)	W-DIA-UPD
34	W-DIA-UPD	MSAC1S_Alarm_Ack.ind(Slot_Number, Alarm_Type, Seq_Nr) /Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr] =not_pending => Index:= Search_In_Outstanding_Alarm_TABLE(Alarm_Status:=pending) MSCY1S_Abort.req FSPMS_Abort.ind(AREP)	RESET-P-LEAVE
35	W-DIA-UPD	MSCY1S_Set_Slave_Diag.cnf(-) (Reference) => R_Cnt:=0 FSPMS DP Slave Fault.ind DMPMS_Reset.req MSCY1S_Reset.req MSAC1S_Reset.req MSRM2S_Reset.req	W-RESET
36	W-DIA-UPD	MSCY1S_Set_Slave_Diag.cnf(+) (Reference) /Reference < Act_Ref => ignore	W-DIA-UPD
37	—	—	—
38	W-DIA-UPD	MSCY1S_Start.ind(Enabled_Alarms, Alarm_Mode_Master, Diag_Flag) => MSCY1S_Abort.req	W-START
39	W-DIA-UPD	MSCY1S_Stop.ind => Alarm_Count:=0 Index:= Search_In_Outstanding_Alarm_TABLE(Alarm_Status:=pending)	RESET-PENDING
40	RESET-PENDING	/Index.Seq_Nr <> 255 => Outstanding_Alarm_TABLE.Alarm_Status[Index]:= not_pending Alarm_Type:= Outstanding_Alarm_TABLE.Alarm_Type[Acls(Alarm_Type),Seq_Nr] Index:= Search_In_Outstanding_Alarm_TABLE(Alarm_Status:=pending) FSPMS_Alarm_Notification.cnf(-)(AREP,Alarm_Type, Slot_Number=0, Index.Seq_Nr, Status=MSAL1S_Stopped)	RESET-PENDING
41	RESET-PENDING	/Index.Seq_Nr = 255 => Reset_Req_Alarm_FIFO() FSPMS DP Slave Stopped.ind(AREP)	W-START
42	RESET-P-LEAVE	/Index.Seq_Nr <> 255 => Outstanding Alarm_TABLE.Alarm_Status[Index]:= not_pending Alarm_Type:= Outstanding_Alarm_TABLE.Alarm_Type[Acls(Alarm_Type),Seq_Nr] Index:= Search_In_Outstanding_Alarm_TABLE(Alarm_Status:=pending) FSPMS_Alarm_Notification.cnf(-)(AREP,Alarm_Type, Slot_Number=0, Index.Seq_Nr, Status=Stopped)	RESET-P-LEAVE
43	RESET-P-LEAVE	/Index.Seq_Nr = 255 => Reset_Req_Alarm_FIFO()	W-START

#	Current State	Event /Condition =>Action	Next State
44	W-FETCHED	FSPMS_Abort.req(AREP) /AREP.AR_Type=MS0 && Alarm_Sequence=TRUE && Alarm_Count=0 => Stored_Diag:=False Ext_Diag_Flag:=L_Ext_Diag_Flag Ext_Diag_Overflow:=L_Ext_Diag_Overflow Ext_Diag:=(Local_Diag_Buffer)Act_Ref:= Act_Ref + 1 CurrentBuffer[AREP].Empty:=TRUE LR_State[AREP]:=W-LR-IND MSCY1S_Set_Slave_Diag.req(Ext_Diag_Flag, Ext_Diag_Overflow, Ext_Diag, Reference:= Act_Ref) MSCY1S_Abort.req	W-START
45	W-FETCHED	FSPMS_Abort.req(AREP) /AREP.AR_Type=MS0 && Alarm_Sequence=FALSE || Alarm_Count<>0 => Stored_Diag:=False Ext_Diag_Flag:=L_Ext_Diag_Flag Ext_Diag_Overflow:=L_Ext_Diag_Overflow Ext_Diag:=(Local_Diag_Buffer) Index:= Search_In_Outstanding_Alarm_TABLE(Alarm_Status:=pending) Act_Ref:= Act_Ref + 1 CurrentBuffer[AREP].Empty:=TRUE LR_State[AREP]:=W-LR-IND MSCY1S_Set_Slave_Diag.req(Ext_Diag_Flag, Ext_Diag_Overflow, Ext_Diag, Reference:= Act_Ref) MSCY1S_Abort.req	RESET-P-LEAVE
46	W-FETCHED	FSPMS_Alarm_Notification.req(AREP, Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data) /ALARM_ENABLED=FALSE => FSPMS_Alarm_Notification.cnf(-)(AREP,Alarm_Type, Slot_Number, Seq_Nr, Status=Not_Enabled)	W-FETCHED
47	W-FETCHED	FSPMS_Alarm_Notification.req(AREP, Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data) /ALARM_ENABLED=TRUE && Alarm_Sequence=TRUE && Alarm_Count=Alarm_Limit => FSPMS_Alarm_Notification.cnf(-)(AREP,Alarm_Type, Slot_Number, Seq_Nr, Status=Limit_Expired)	W-FETCHED
48	W-FETCHED	FSPMS_Alarm_Notification.req(AREP, Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data) /ALARM_ENABLED=TRUE && Alarm_Sequence=TRUE && Alarm_Count<Alarm_Limit && Outstanding_Alarm_TABLE.Alarm_Status[AclsAlarm_Type),Seq_Nr]=not_pending => Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]:=pending Alarm_Count:=Alarm_Count+1 Store_to_Req_Alarm_FIFO(Alarm(Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data)) FSPMS_Alarm_Notification.cnf(+)(AREP,Alarm_Type, Slot_Number, Seq_Nr)	W-FETCHED
49	W-FETCHED	FSPMS_Alarm_Notification.req(AREP, Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data) /ALARM_ENABLED=TRUE && Alarm_Sequence=TRUE && Alarm_Count<Alarm_Limit&& Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]= pending => FSPMS_Alarm_Notification.cnf(-)(AREP,Alarm_Type, Slot_Number, Seq_Nr, Status=Sequence_Nr_Pending)	W-FETCHED
50	W-FETCHED	FSPMS_Alarm_Notification.req(AREP, Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data) /ALARM_ENABLED=TRUE && Alarm_Sequence=FALSE && Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]= pending => FSPMS_Alarm_Notification.cnf(-)(AREP, Alarm_Type, Slot_Number, Seq_Nr, Status=Sequence_Nr_Pending)	W-FETCHED

#	Current State	Event /Condition =>Action	Next State
51	W-FETCHED	FSPMS_Alarm_Notification.req(AREP, Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data) /ALARM_ENABLED=TRUE && Alarm_Sequence=FALSE && Outstanding_Alarm_TABLE.Alarm_Status[AclsL(Alarm_Type),Seq_Nr]= not_pending => Outstanding_Alarm_TABLE.Alarm_Status[AclsL(Alarm_Type),Seq_Nr]:= pending Outstanding_Alarm_TABLE.Alarm_Type[Acls(Alarm_Type),Seq_Nr]:= Alarm_Type Store_to_Req_Alarm_FIFO(Alarm(Slot_Number, Alarm_Type, Seq_Nr, Add_Ack, Alarm_Specifier, Alarm_Data)) FSPMS_Alarm_Notification.cnf(+)(AREP,Alarm_Type, Slot_Number, Seq_Nr)	W-FETCHED
52	W-FETCHED	FSPMS_Set_Slave_Diag.req(AREP,Ext_Diag_Flag, Ext_Diag) => if (PrmCmdAck in Ext_Diag or Stored_PrmCmdAck != NULL) Stored_PrmCmdAck:= PrmCmdAck, Ext_Diag.Red_Status:= PrmCmdAck endif Stored_Diag:= True L_Ext_Diag_Flag:= Ext_Diag_Flag L_Ext_Diag_Overflow:= Ext_Diag_Overflow Local_Diag_Buffer:= Ext_Diag FSPMS_Set_Slave_Diag.cnf(AREP)	W-FETCHED
53	W-FETCHED	MSAC1S_Alarm_Ack.ind(Slot_Number, Alarm_Type, Seq_Nr) /Alarm_Sequence=TRUE && Outstanding_Alarm_TABLE.Alarm_Status[Acls (Alarm_Type),Seq_Nr]=pending => Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]:=not_pending Alarm_Count:=Alarm_Count-1 FSPMS_Alarm_Ack.ind(Slot_Number, Alarm_Type, Seq_Nr) MSAC1S_Alarm Ack.rsp(+) (Slot Number, Alarm Type, Seq Nr)	W-FETCHED
54	W-FETCHED	MSAC1S_Alarm_Ack.ind(Slot_Number, Alarm_Type, Seq_Nr) /Alarm_Sequence=FALSE && Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]=pending => Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]:=not_pending FSPMS_Alarm_Ack.ind(Slot_Number, Alarm_Type, Seq_Nr) MSAC1S_Alarm Ack.rsp(+) (Slot Number, Alarm Type, Seq Nr)	W-FETCHED
55	W-FETCHED	MSAC1S_Alarm_Ack.ind(Slot_Number, Alarm_Type, Seq_Nr) /Outstanding_Alarm_TABLE.Alarm_Status[Acls(Alarm_Type),Seq_Nr]=not_pending => Index:= Search_In_Outstanding_Alarm_TABLE(Alarm_Status:=pending) MSCY1S_Abort.req FSPMS_Abort.ind(AREP)	RESET-P-LEAVE
56	W-FETCHED	MSCY1S_Set_Slave_Diag.cnf(-) (Reference) => R_Cnt:=0 FSPMS DP Slave Fault.ind DMPMS_Reset.req MSCY1S_Reset.req MSAC1S_Reset.req MSRM2S_Reset.req	W-RESET
57	W-FETCHED	MSCY1S_Set_Slave_Diag.cnf(+) (Reference) /Reference < Act_Ref => ignore	W-FETCHED
58	W-FETCHED	MSCY1S_Set_Slave_Diag.cnf(+) (Reference) /Reference >= Act_Ref => Stored_PrmCmdAck:= NULL	CHK-TRANSMIT
59	W-FETCHED	MSCY1S_Start.ind(Enabled_Alarms, Alarm_Mode_Master, Diag_Flag) => MSCY1S_Abort.req	W-START

#	Current State	Event /Condition =>Action	Next State
60	W-FETCHED	MSCY1S_Stop.ind /Alarm_Sequence=TRUE && Alarm_Count=0 => Stored_Diag:=False Ext_Diag_Flag:=L_Ext_Diag_Flag Ext_Diag_Overflow:=L_Ext_Diag_Overflow Ext_Diag:=(Local_Diag_Buffer) Act_Ref:= Act_Ref + 1 CurrentBuffer[AREP].Empty:=TRUE LR_State[AREP]:=W-LR-IND MSCY1S_Set_Slave_Diag.req(Ext_Diag_Flag, Ext_Diag_Overflow, Ext_Diag, Reference:= Act_Ref) FSPMS_Stopped.ind(AREP)	W-START
61	W-FETCHED	MSCY1S_Stop.ind /Alarm_Sequence=FALSE || Alarm_Count<>0 => Stored_Diag:=False Alarm_Count:=0 Index:= Search_In_Outstanding_Alarm_TABLE(Alarm_Status:=pending) Ext_Diag_Flag:=L_Ext_Diag_Flag Ext_Diag_Overflow:=L_Ext_Diag_Overflow Ext_Diag:=(Local_Diag_Buffer) Act_Ref:= Act_Ref + 1 CurrentBuffer[AREP].Empty:=TRUE LR_State[AREP]:=W-LR-IND MSCY1S_Set_Slave_Diag.req(Ext_Diag_Flag, Ext_Diag_Overflow, Ext_Diag, Reference:= Act_Ref)	RESET-PENDING
62	CHK-TRANSMIT	/Req_Alarm_FIFO_state=empty &&Stored_Diag = FALSE	W-DIA-UPD
63	CHK-TRANSMIT	/Req_Alarm_FIFO_state=empty && Stored_Diag = TRUE=>Stored_Diag:=FalseExt_Diag_Flag:=L_Ext_Diag_FlagExt_Diag_Overflow:=L_Ext_Diag_OverflowExt_Diag:=(Local_Diag_Buffer)Act_Ref:= Act_Ref + 1MSCY1S_Set_Slave_Diag.req(Ext_Diag_Flag, Ext_Diag_Overflow, Ext_Diag, Reference:= Act_Ref)	W-FETCHED
64	CHK-TRANSMIT	/Req_Alarm_FIFO_state<>empty => Alarm:=Load_from_Req_Alarm_FIFO() Stored_Diag:=False Ext_Diag_Flag:=L_Ext_Diag_Flag Ext_Diag_Overflow:=L_Ext_Diag_Overflow Ext_Diag:=(Local_Diag_Buffer, Alarm) Act_Ref:= Act_Ref + 1 MSCY1S_Set_Slave_Diag.req(Ext_Diag_Flag, Ext_Diag_Overflow, Ext_Diag, Reference:= Act_Ref)	W-FETCHED
65	W-RESET	DMPMS_Reset.cnf /R_Cnt<3 => R_Cnt:=R_Cnt+1	W-RESET
66	W-RESET	MSCY1S_Reset.cnf /R_Cnt<3 => R_Cnt:=R_Cnt+1	W-RESET
67	W-RESET	MSAC1S_Reset.cnf /R_Cnt<3 => R_Cnt:=R_Cnt+1	W-RESET
68	W-RESET	MSRM2S_Reset.cnf /R_Cnt<3 => R_Cnt:=R_Cnt+1	W-RESET

#	Current State	Event /Condition =>Action	Next State

69	W-RESET	DMPMS_Reset.cnf
		/R_Cnt>=3
		=>
		if (FSPMS_User_Reset=True)
		FSPMS_Reset DP Slave.cnf
		endif
			POWER-ON

70	W-RESET	MSCY1S_Reset.cnf
		/R_Cnt>=3
		=>
		if (FSPMS_User_Reset=True)
		FSPMS_Reset DP Slave.cnf
		endif
			POWER-ON

71	W-RESET	MSAC1S_Reset.cnf
		/R_Cnt>=3
		=>
		if (FSPMS_User_Reset=True)
		FSPMS_Reset DP Slave.cnf
		endif
			POWER-ON

72	W-RESET	MSRM2S_Reset.cnf
		/R_Cnt>=3
		=>
		if (FSPMS_User_Reset=True)
		FSPMS_Reset DP Slave.cnf
		endif
			POWER-ON

73	any state	DMPMS_SYCL_Clock_Value.ind (Clock Value Time Event, Clock Value previous TE, Clock Value Status, Receive Delay Time, Src add Clk msg) => FSPMS_SYCL_Clock_Value.ind (AREP, Clock Value Time Event, Clock Value previous TE, Clock Value Status, Receive Delay Time, Src add Clk msg)	SAME
74	any state	DMPMS_Fault.ind => R_Cnt:=0 FSPMS DP Slave Fault.ind DMPMS_Reset.req MSCY1S_Reset.req MSAC1S_Reset.req MSRM2S_Reset.req	W-RESET
75	any state	MSAC1S_Fault.ind => R_Cnt:=0 FSPMS DP Slave Fault.ind DMPMS_Reset.req MSCY1S_Reset.req MSAC1S_Reset.req MSRM2S_Reset.req	W-RESET
76	any state	MSCY1S_Fault.ind => R_Cnt:=0 FSPMS DP Slave Fault.ind DMPMS_Reset.req MSCY1S_Reset.req MSAC1S_Reset.req MSRM2S_Reset.req	W-RESET
77	any state	MSRM2S_Fault.ind => R_Cnt:=0 FSPMS DP Slave Fault.ind DMPMS_Reset.req MSCY1S_Reset.req MSAC1S_Reset.req MSRM2S_Reset.req	W-RESET

#	Current State	Event /Condition =>Action	Next State
78	any state	FSPMS_Reset DP Slave .req => R_Cnt:=0 FSPMS_User_Reset:=True DMPMS_Reset.req MSCY1S_Reset.req MSAC1S_Reset.req MSRM2S_Reset.req	W-RESET
79	W-FETCHED	Set_ARL_Isochron_Mode.req(Isochron_Mode) => Set_ARL_Isochron_Mode.cnf(+)	W-FETCHED
80	W-DIA-UPD	Set_ARL_Isochron_Mode.req(Isochron_Mode) => Set_ARL_Isochron_Mode.cnf(+)	W-DIA-UPD
81	W-START	Set_ARL_Isochron_Mode.req(Isochron_Mode) => Status:= NO Set_ARL_Isochron_Mode.cnf(-)(Status)	W-START
82	t-state	Load CRL DXB Linktable Entries.req(AREP, DXB-Linktable) /valid Parameters => Load CRL DXB Linktable Entries.cnf(+)	
83	t-state	Load CRL DXB Linktable Entries.req(AREP, DXB-Linktable) /invalid Parameters => Status:= NO Load CRL DXB Linktable Entries.cnf(-) (Status)	
84	t-state	FSPMS DLL Init DP Slave.req (Bus Para) => DMPMS_Sinit_DLL.req (Bus_Para)	SAME
85	t-state	FSPMS_SInit MS0.req (AREP) => MSCY1S_SInit MS0.req ( )	SAME
86	t-state	FSPMS_SInit MS2.req ( ) => MSRM2S_SInit MS2.req ( )	SAME
87	t-state	FSPMS_Abort.req (AREP, Subnet, Instance, Reason Code) /AREP.AR_Type=MS2 => CurrentBuffer[AREP].Empty:=TRUE LR_State[AREP]:=W-LR-IND MSAC2S_Abort.req (Subnet, Instance, Reason Code)	SAME
88	t-state	FSPMS_DP Slave Application Ready.req (AREP) /AREP.AR_Type=MS0 => MSCY1S_Application Ready.req ( )	SAME
89	t-state	FSPMS_Check User Prm Result.req (AREP, Prm_OK) /AREP.AR_Type=MS0 => MSCY1S_Check User Prm Result.req (Prm_OK)	SAME
90	t-state	FSPMS_Check Cfg Result.req (AREP, Cfg_OK, Input Data Len, Output Data Len) /AREP.AR_Type=MS0 => MSCY1S_Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len)	SAME
91	t-state	FSPMS_Set Cfg.req (AREP, Cfg Data) /AREP.AR_Type=MS0 => MSCY1S_Set Cfg.req (Cfg Data)	SAME
92	t-state	FSPMS_Set Input.req (AREP, Input Data) /AREP.AR_Type=MS0 => MSCY1S_Set Input.req (Input Data)	SAME

#	Current State	Event /Condition =>Action	Next State
93	t-state	FSPMS_Get Output.req (AREP) /AREP.AR_Type=MS0 => MSCY1S_Get Output.req ( )	SAME
94	t-state	FSPMS_Initiate.rsp(+) (AREP, Max Len Data Unit, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param) /AREP.AR_Type=MS1 => MSAC2S_Initiate.rsp(+) (Res SAP, Max Len Data Unit, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param)	SAME
95	t-state	FSPMS_Initiate.rsp(-) (AREP, Error Decode, Error Code 1 Error Code 2) /AREP.AR_Type=MS2 => MSAC2S_Initiate.rsp(-) (Res SAP, Error Decode, Error Code 1 Error Code 2)	SAME
96	t-state	FSPMS_Read.rsp(+) (AREP, Length, Data) /AREP.AR_Type=MS1 => MSAC1S_Read.rsp(+) (Length, Data)	SAME
97	t-state	FSPMS_Read.rsp(+) (AREP, Length, Data) /AREP.AR_Type=MS2 => MSAC2S_Read.rsp(+) (Length, Data)	SAME
98	t-state	FSPMS_Read.rsp(-) (AREP, Error Decode, Error Code 1 Error Code 2) /AREP.AR_Type=MS1 => MSAC1S_Read.rsp(-) (Error Decode, Error Code 1 Error Code 2)	SAME
99	t-state	FSPMS_Read.rsp(-) (AREP, Error Decode, Error Code 1 Error Code 2) /AREP.AR_Type=MS2 => MSAC2S_Read.rsp(-) (Error Decode, Error Code 1 Error Code 2)	SAME
100	t-state	FSPMS_Write.rsp(+) (AREP, Length) /AREP.AR_Type=MS1 => MSAC1S_Write.rsp(+) (Length)	SAME
101	t-state	FSPMS_Write.rsp(+) (AREP, Length) /AREP.AR_Type=MS2 => MSAC2S_Write.rsp(+) (Length)	SAME
102	t-state	FSPMS_Write.rsp(-) (AREP, Error Decode, Error Code 1 Error Code 2) /AREP.AR_Type=MS1 => MSAC1S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2)	SAME
103	t-state	FSPMS_Write.rsp(-) (AREP, Error Decode, Error Code 1 Error Code 2) /AREP.AR_Type=MS2 => MSAC2S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2)	SAME
104	t-state	FSPMS_Data Transport.rsp(+) (AREP, Length, Data) /AREP.AR_Type=MS2 => MSAC2S_Data Transport.rsp(+) (Res SAP, Length, Data)	SAME
105	t-state	FSPMS_Data Transport.rsp(-) (AREP, Error Decode, Error Code 1 Error Code 2) /AREP.AR_Type=MS2 => MSAC2S_Data Transport.rsp(-) (Res SAP, Error Decode, Error Code 1 Error Code 2)	SAME
106	t-state	FSPMS_Alarm Ack.rsp(+) (AREP, Slot Number, Alarm Type, Seq Nr) /AREP.AR_Type=MS1 =>	SAME
107	t-state	FSPMS_Alarm Ack.rsp(-) (AREP, Slot Number, Alarm Type, Seq Nr) /AREP.AR_Type=MS1 =>	SAME

#	Current State	Event /Condition =>Action	Next State
108	t-state	DMPMS_SInit DLL.cnf ( ) => FSPMS DLL Init DP Slave.cnf ( )	SAME
109	t-state	MSCY1S_SInit MS0.cnf ( ) => AREP=MS0-AR FSPMS_SInit MS0.cnf (AREP)	SAME
110	t-state	MSRM2S_SInit MS2.cnf ( ) => FSPMS_SInit MS2.cnf ( )	SAME
111	t-state	MSCY1S_Check User Prm Result.cnf(+) ( ) => AREP=MS0-AR FSPMS_Check User Prm Result.cnf(+) (AREP)	SAME
112	t-state	MSCY1S_Check User Prm Result.cnf(-) (Status) => AREP=MS0-AR FSPMS_Check User Prm Result.cnf(-) (AREP, Status)	SAME
113	t-state	MSCY1S_Check Cfg Result.cnf(+) ( ) => AREP=MS0-AR FSPMS_Check Cfg Result.cnf(+) (AREP)	SAME
114	t-state	MSCY1S_Check Cfg Result.cnf(-) (Status) => AREP=MS0-AR FSPMS_Check Cfg Result.cnf(-) (AREP, Status)	SAME
115	t-state	MSCY1S_Set Cfg.cnf(+) ( ) => AREP=MS0-AR FSPMS_Set Cfg.cnf(+) (AREP)	SAME
116	t-state	MSCY1S_Set Cfg.cnf(-) ( ) => AREP=MS0-AR FSPMS_Set Cfg.cnf(-) (AREP)	SAME
117	t-state	MSCY1S_Set Input.cnf(+) ( ) => AREP=MS0-AR FSPMS_Set Input.cnf(+) (AREP)	SAME
118	t-state	MSCY1S_Set Input.cnf(-) ( ) => AREP=MS0-AR FSPMS_Set Input.cnf(-) (AREP)	SAME
119	t-state	MSCY1S_Get Output.cnf(+) (Output Data, Clear Flag, New Flag) => AREP=MS0-AR FSPMS_Get Output.cnf(+) (AREP, Output Data, Clear Flag, New Flag)	SAME
120	t-state	MSCY1S_Get Output.cnf(-) ( ) => AREP=MS0-AR FSPMS_Get Output.cnf(-) (AREP)	SAME
121	t-state	MSCY1S_Set Slave Add.ind (New Slave Add, Ident Number, No Add Chg, Rem Slave Data) => AREP=MS0-AR FSPMS_Set Slave Add.ind (AREP, New Slave Add, Ident Number, No Add Chg, Rem Slave Data)	SAME
122	t-state	MSCY1S_Check User Prm.ind (User Prm Data) => AREP=MS0-AR FSPMS_Check User Prm.ind (AREP, User Prm Data)	SAME

#	Current State	Event /Condition =>Action	Next State
123	t-state	MSCY1S_Check Cfg.ind (Check Cfg Mode, Cfg Data) => AREP=MS0-AR FSPMS_Check Cfg.ind (AREP, Check Cfg Mode, Cfg Data)	SAME
124	t-state	MSCY1S_New Output.ind (Clear Flag) => AREP=MS0-AR FSPMS_New Output.ind (AREP, Clear Flag)	SAME
125	t-state	MSCY1S_Global Control.ind (Clear Command, Sync Command, Freeze Command, Group Select) => AREP=MS0-AR FSPMS_Global Control.ind (AREP, Clear Command, Sync Command, Freeze Command, Group Select)	SAME
126	t-state	MSAC2S_Initiate.ind (Res SAP, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param) /SetContext(Res_SAP, Service)=TRUE => FSPMS_Initiate.ind (AREP, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param)	SAME
127	t-state	MSAC2S_Abort.ind (Res SAP, Locally Generated, Subnet, Instance, Reason Code, Additional Detail) /SetContext(Res_SAP, Service)=TRUE => CurrentBuffer[AREP].Empty:=TRUE LR_State[AREP]:=W-LR-IND FSPMS_Abort.ind (AREP, Locally Generated, Subnet, Instance, Reason Code, Additional Detail)	SAME
128	t-state	MSAC2S_Read.ind (Res SAP, Slot Number, Index, Length) /SetContext(Res_SAP, Service)=TRUE && Index<>255 => FSPMS_Read.ind (AREP, Slot Number, Index, Length)	SAME
129	t-state	MSAC1S_Read.ind (Res SAP, Slot Number, Index, Length) /SetContext(Res_SAP, Service)=TRUE && Index<>255 => FSPMS_Read.ind (AREP, Slot Number, Index, Length)	SAME
130	t-state	MSAC2S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /SetContext(Res_SAP, Service)=TRUE && Index<>255 => FSPMS_Write.ind (AREP, Slot Number, Index, Length, Data)	SAME
131	t-state	MSAC1S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /SetContext(Res_SAP, Service)=TRUE &&Index<>255 => FSPMS_Write.ind (AREP, Slot Number, Index, Length, Data)	SAME
132	t-state	MSAC2S_Data Transport.ind (Res SAP, Slot Number, Index, Length, Data) /SetContext(Res_SAP, Service)=TRUE => FSPMS_Data Transport.ind (AREP, Slot Number, Index, Length, Data)	SAME
133	t-state	FSPMS_Get_Publisher_Data.req (AREP, CREP) /AREP.AR_Type=MS0 => Rem_Add=CREP.Rem_Add SSCY1S_Get_Publisher_Data.req (Rem_Add)	SAME
134	t-state	FSPMS_Start_Subscriber.req (AREP, CREP) /AREP.AR_Type=MS0 => Rem_Add=CREP.Rem_Add SSCY1S_Start_Subscriber.req (Rem_Add)	SAME

#	Current State	Event /Condition =>Action	Next State
135	t-state	FSPMS_Stop_Subscriber.req (AREP, CREP) /AREP.AR_Type=MS0 => Rem_Add=CREP.Rem_Add SSCY1S_Stop_Subscriber.req (Rem_Add)	SAME
136	t-state	FSPMS_Check_Ext_User_Prm_Result.req (AREP, Ext_Prm_OK) /AREP.AR_Type=MS0 => MSCY1S_Check_Ext_User_Prm_Result.req (Ext_Prm_OK)	SAME
137	t-state	SSCY1S_Get_Publisher_Data.cnf(+) (Rem_Add, Data, New_Flag) /SetContext(Rem_Add, Service)=TRUE => FSPMS_Get_Publisher_Data.cnf(+) (AREP, CREP, Data, New_Flag)	SAME
138	t-state	SSCY1S_Get_Publisher_Data.cnf(-) (Rem_Add) /SetContext(Rem_Add, Service)=TRUE => FSPMS_Get_Publisher_Data.cnf(-) (AREP, CREP)	SAME
139	t-state	SSCY1S_Start_Subscriber.cnf(+) (Rem_Add) /SetContext(Rem_Add, Service)=TRUE => FSPMS_Start_Subscriber.cnf(+) (AREP, CREP)	SAME
140	t-state	SSCY1S_Start_Subscriber.cnf(-) (Rem_Add) /SetContext(Rem_Add, Service)=TRUE => FSPMS_Start_Subscriber.cnf(-) (AREP, CREP)	SAME
141	t-state	SSCY1S_Stop_Subscriber.cnf(+) (Rem_Add) /SetContext(Rem_Add, Service)=TRUE => FSPMS_Stop_Subscriber.cnf(+) (AREP, CREP)	SAME
142	t-state	SSCY1S_Stop_Subscriber.cnf(-) (Rem_Add) /SetContext(Rem_Add, Service)=TRUE => FSPMS_Stop_Subscriber.cnf(-) (AREP, CREP)	SAME
143	t-state	SSCY1S_New_Publisher_Data.ind (Rem_Add) /SetContext(Rem_Add, Service)=TRUE => FSPMS_New_Publisher_Data.ind(AREP, CREP)	SAME
144	t-state	SSCY1S_Publisher_Active.ind (Rem_Add, Status) /SetContext(Rem_Add, Service)=TRUE => FSPMS_Publisher_Active.ind(AREP, CREP, Status)	SAME
145	t-state	MSAC1S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.w.IL.Extended_Function_Num=Initiate_Load) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length Current Load Type[AREP]:=Data.w.IL.Load_Type LR Index:=Data.w.IL.LR_Index Load Type:=Data.w.IL.Load_Type Load Image Size:=Data.w.IL.Load_Image_Size User Specific:=Data.w.IL.User_Specific Intersegment Request Timeout:=Data.w.IL.Intersegment_Request_Timeout FSPMS_Initiate_Load.ind (AREP, Slot Number, LR Index, Load Type, Load Image Size, Intersegment Request Timeout, User Specific) LR_State[AREP]:=W-LR-RES	SAME

#	Current State	Event /Condition =>Action	Next State
146	t-state	MSAC1S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.w.IL.Extended_Function_Num=Initiate_Load) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length Current Load Type[AREP]:=Data.w.IL.Load_Type LR Index:=Data.w.IL.LR_Index Load Type:=Data.w.IL.Load_Type Load Image Size:=Data.w.IL.Load_Image_Size User Specific:=Data.w.IL.User_Specific Intersegment Request Timeout:=Data.w.IL.Intersegment_Request_Timeout FSPMS_Initiate_Load.ind (AREP, Slot Number, LR Index, Load Type, Load Image Size, Intersegment Request Timeout, User Specific) LR_State[AREP]:=W-LR-RES	SAME
147	t-state	MSAC2S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.w.IL.Extended_Function_Num=Initiate_Load) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length CurrentLoadType[AREP]:=Data.w.IL.Load_Type CurrentLRIndex[AREP]:=Data.w.IL.LR_Index LR Index:=Data.w.IL.LR_Index Load Type:=Data.w.IL.Load_Type Load Image Size:=Data.w.IL.Load_Image_Size User Specific:=Data.w.IL.User_Specific Intersegment Request Timeout:=Data.w.IL.Intersegment_Request_Timeout FSPMS_Initiate_Load.ind (AREP, Slot Number, LR Index, Load Type, Load Image Size, Intersegment Request Timeout, User Specific) LR_State[AREP]:=W-LR-RES	SAME
148	t-state	FSPMS_Initiate_Load.rsp(+) (AREP, Actual LR Size, Max Response Delay, Max Segment Length, User Specific) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => CurrentBuffer[AREP].LR:=TRUE CurrentBuffer[AREP].Empty:=FALSE CurrentBuffer[AREP].data.r.IL.Actual_LR_Size:=Actual LR Size CurrentBuffer[AREP].data.r.IL.Max_Response_Delay:=Max Response Delay CurrentBuffer[AREP].data.r.IL.Max_Segment_Length:=Max Segment Length CurrentBuffer[AREP].data.r.IL.User_Specific:=User Specific MSAC1S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME
149	t-state	FSPMS_Initiate_Load.rsp(+) (AREP, Actual LR Size, Max Response Delay, Max Segment Length, User Specific) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].LR:=TRUE CurrentBuffer[AREP].Empty:=FALSE CurrentBuffer[AREP].data.r.IL.Actual_LR_Size:=Actual LR Size CurrentBuffer[AREP].data.r.IL.Max_Response_Delay:=Max Response Delay CurrentBuffer[AREP].data.r.IL.Max_Segment_Length:=Max Segment Length CurrentBuffer[AREP].data.r.IL.User_Specific:=User Specific MSAC2S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME

#	Current State	Event /Condition =>Action	Next State
150	t-state	MSAC1S_Read.ind (Res SAP, Slot Number, Index, Length) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && CurrentBuffer[AREP].Empty=FALSE && CurrentBuffer[AREP].LR=FALSE) => Data:=CurrentBuffer[AREP].data Length:=sizeof(CurrentBuffer[AREP].data) MSAC1S_Read.rsp(+) (Length, Data) LR_State[AREP]:=W-LR-IND	SAME
151	t-state	MSAC1S_Read.ind (Res SAP, Slot Number, Index, Length) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && CurrentBuffer[AREP].Empty=FALSE && CurrentBuffer[AREP].LR=TRUE) => CurrentBuffer[AREP].Empty:=TRUE Data:=CurrentBuffer[AREP].data Length:=sizeof(CurrentBuffer[AREP].data) MSAC1S_Read.rsp(+) (Length, Data) LR_State[AREP]:=W-LR-IND	SAME
152	t-state	MSAC2S_Read.ind (Res SAP, Slot Number, Index, Length) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && CurrentBuffer[AREP].Empty=FALSE && CurrentBuffer[AREP].LR=FALSE) => Data:=CurrentBuffer[AREP].data Length:=sizeof(CurrentBuffer[AREP].data) MSAC2S_Read.rsp(+) (Length, Data) LR_State[AREP]:=W-LR-IND	SAME
153	t-state	MSAC2S_Read.ind (Res SAP, Slot Number, Index, Length) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && CurrentBuffer[AREP].Empty=FALSE && CurrentBuffer[AREP].LR=TRUE) => CurrentBuffer[AREP].Empty:=TRUE Data:=CurrentBuffer[AREP].data Length:=sizeof(CurrentBuffer[AREP].data) MSAC2S_Read.rsp(+) (Length, Data) LR_State[AREP]:=W-LR-IND	SAME
154	t-state	FSPMS_Initiate_Load.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC1S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
155	t-state	FSPMS_Initiate_Load.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC2S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME

#	Current State	Event /Condition =>Action	Next State
156	t-state	MSAC1S_Read.ind (Res SAP, Slot Number, Index, Length) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE &&(Index=255 && CurrentBuffer[AREP].Empty=TRUE && CurrentBuffer[AREP].LR=TRUE) => LR Index:=CurrentLRIndex[AREP] Segment Length:=Length FSPMS_Pull_Segment.ind (AREP, Slot Number, LR Index, Segment Length) LR_State[AREP]:=W-LR-RES	SAME
157	t-state	MSAC1S_Read.ind (Res SAP, Slot Number, Index, Length) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && CurrentBuffer[AREP].Empty=TRUE && CurrentBuffer[AREP].LR=FALSE) => Error Decode:=DPV1 Error Code 1:=state conflict Error Code 2:=0 MSAC1S_Read.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-RES	SAME
158	t-state	MSAC2S_Read.ind (Res SAP, Slot Number, Index, Length) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && CurrentBuffer[AREP].Empty=TRUE && CurrentBuffer[AREP].LR=TRUE) => LR Index:=CurrentLRIndex[AREP] Segment Length:=Length FSPMS_Pull_Segment.ind (AREP, Slot Number, LR Index, Segment Length) LR_State[AREP]:=W-LR-RES	SAME
159	t-state	MSAC2S_Read.ind (Res SAP, Slot Number, Index, Length) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && CurrentBuffer[AREP].Empty=TRUE && CurrentBuffer[AREP].LR=FALSE) => Error Decode:=DPV1 Error Code 1:=state conflict Error Code 2:=0 MSAC2S_Read.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-RES	SAME
160	t-state	FSPMS_Pull_Segment.rsp(+) (AREP, Segment Length, Segment Number, More Follows, Data) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => Data.PULL.Extended_Function_Num=Pull Data.PULL.Sequence Number:=Segment Number Data.PULL.Options.More Follows:=More Follows Data.PULL.Region Data:= Data Length:=6+Segment Length MSAC1S_Read.rsp(+) (Length, Data) LR_State[AREP]:=W-LR-IND	SAME
161	t-state	FSPMS_Pull_Segment.rsp(+) (AREP, Segment Length, Segment Number, More Follows, Data) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => Data.PULL.Extended_Function_Num=Pull Data.PULL.Sequence Number:=Segment Number Data.PULL.Options.More Follows:=More Follows Data.PULL.Region Data:= Data Length:=6+Segment Length MSAC2S_Read.rsp(+) (Length, Data) LR_State[AREP]:=W-LR-IND	SAME

#	Current State	Event /Condition =>Action	Next State
162	t-state	FSPMS_Pull Segment.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC1S_Read.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
163	t-state	FSPMS_Pull Segment.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC2S_Read.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
164	t-state	MSAC1S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.w.PUSH.Extended_Function_Num=Push) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length LR Index:=Data.PUSH.LR_Index Segment Length:=Length Segment Number:=Data.PUSH.Segment_Number More Follows:=Data.PUSH.More_Follows Data:=Data.PUSH.data FSPMS_Push_Segment.ind (AREP, Slot Number, LR Index, Segment Length, Segment Number, More Follows, Data) LR_State[AREP]:=W-LR-RES	SAME
165	t-state	MSAC2S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.w.PUSH.Extended_Function_Num=Push) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length LR Index:=Data.PUSH.LR_Index Segment Length:=Length Segment Number:=Data.PUSH.Segment_Number More Follows:=Data.PUSH.More_Follows Data:=Data.PUSH.data FSPMS_Push_Segment.ind (AREP, Slot Number, LR Index, Segment Length, Segment Number, More Follows, Data) LR_State[AREP]:=W-LR-RES	SAME
166	t-state	FSPMS_Push_Segment.rsp(+) (AREP) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => Length:=CurrentBuffer[AREP].WriteLength MSAC1S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME
167	t-state	FSPMS_Push_Segment.rsp(+) (AREP) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => Length:=CurrentBuffer[AREP].WriteLength MSAC2S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME

#	Current State	Event /Condition =>Action	Next State
168	t-state	FSPMS_Push_Segment.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC1S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
169	t-state	FSPMS_Push_Segment.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC2S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
170	t-state	MSAC1S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.w.TL.Extended_Function_Num=Terminate_Load) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length LR Index:=Data.w.TL.LR_Index FSPMS_Terminate_Load.ind (AREP, Slot Number, LR Index) LR_State[AREP]:=W-LR-RES	SAME
171	t-state	MSAC2S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.w.TL.Extended_Function_Num=Terminate_Load) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length LR Index:=Data.w.TL.LR_Index FSPMS_Terminate_Load.ind (AREP, Slot Number, LR Index) LR_State[AREP]:=W-LR-RES	SAME
172	t-state	FSPMS_Terminate_Load.rsp(+) (AREP) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => CurrentBuffer[AREP].LR:=FALSE CurrentBuffer[AREP].Empty:=FALSE CurrentBuffer[AREP].data.r:=NIL Length:=CurrentBuffer[AREP].WriteLength MSAC1S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME
173	t-state	FSPMS_Terminate_Load.rsp(+) (AREP) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].LR:=FALSE CurrentBuffer[AREP].Empty:=FALSE CurrentBuffer[AREP].data.r:=NIL Length:=CurrentBuffer[AREP].WriteLength MSAC2S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME

#	Current State	Event /Condition =>Action	Next State
174	t-state	FSPMS_Terminate_Load.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC1S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
175	t-state	FSPMS_Terminate_Load.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC2S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
176	t-state	MSAC1S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &CurrentBuffer[AREP].LR:=FALSE &&SetContext(Res_SAP, Service)=TRUE &&(Index=255 && Length >= 4) &&(Data.w.CALL.Extended_Function_Num=CALL) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length FI Index:=Data.w.CALL.FI_Index Entity_Number:=Data.w.CALL. Entity_Number Execution Argument:=Data.w.CALL.Execution_Argument FSPMS_Call.ind (AREP, Slot Number, Entity_Number, FI Index, Execution Argument) LR_State[AREP]:=W-LR-RES	SAME
177	t-state	MSAC1S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &CurrentBuffer[AREP].LR:=True && SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.w.CALL.Extended_Function_Num=CALL) => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=state conflict Error Code 2:=0 MSAC1S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
178	t-state	MSAC2S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &CurrentBuffer[AREP].LR:=FALSE && SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.w.CALL.Extended_Function_Num=CALL) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length Entity_Number:=Data.w.CALL. Entity_Number FI Index:=Data.w.CALL.FI_Index Execution Argument:=Data.w.CALL.Execution_Argument FSPMS_Call.ind (AREP, Slot Number, Entity_Number, FI Index, Execution Argument) LR_State[AREP]:=W-LR-RES	SAME

#	Current State	Event /Condition =>Action	Next State
179	t-state	MSAC2S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &CurrentBuffer[AREP].LR:=True && SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.w.CALL.Extended_Function_Num=CALL) => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=state conflict Error Code 2:=0 MSAC2S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
180	t-state	FSPMS_Call.rsp(+) (AREP, Result Argument) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => CurrentBuffer[AREP].Empty:=FALSE CurrentBuffer[AREP].data.r.CALL.ResultArgument:=Result Argument Length:=CurrentBuffer[AREP].WriteLength MSAC1S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME
181	t-state	FSPMS_Call.rsp(+) (AREP, Result Argument) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].Empty:=FALSE CurrentBuffer[AREP].data.r.CALL.ResultArgument:=Result Argument Length:=CurrentBuffer[AREP].WriteLength MSAC2S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME
182	t-state	FSPMS_Call.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC1S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
183	t-state	FSPMS_Call.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC1S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
184	t-state	MSAC1S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.FIS.Extended_Function_Num=Start) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length FI Index:=Data.FIS.FI_Index Execution Argument:=Data.FIS.Execution_Argument FSPMS_Start.ind (AREP, Slot Number, FI Index, Execution Argument) LR_State[AREP]:=W-LR-RES	SAME

#	Current State	Event /Condition =>Action	Next State
185	t-state	MSAC2S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.FIS.Extended_Function_Num=Start) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length FI Index:=Data.FIS.FI_Index Execution Argument:=Data.FIS.Execution_Argument FSPMS_Start.ind (AREP, Slot Number, FI Index, Execution Argument) LR_State[AREP]:=W-LR-RES	SAME
186	t-state	FSPMS_Start.rsp(+) (AREP) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => Length:=CurrentBuffer[AREP].WriteLength MSAC1S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME
187	t-state	FSPMS_Start.rsp(+) (AREP)/LR_State[AREP]==W-LR-RES&AREP.AR_Type=MS2=>Length:=CurrentBuffer[AREP].WriteLengthMSAC2S_Write.rsp(+) (Length)LR_State[AREP]:=W-LR-IND	SAME
188	t-state	FSPMS_Start.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC1S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
189	t-state	FSPMS_Start.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC2S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
190	t-state	MSAC1S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.FIS.Extended_Function_Num=Stop) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length FI Index:=Data.FIS.FI_Index FSPMS_Stop.ind (AREP, Slot Number, FI Index) LR_State[AREP]:=W-LR-RES	SAME
191	t-state	MSAC2S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.FIS.Extended_Function_Num=Stop) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length FI Index:=Data.FIS.FI_Index FSPMS_Stop.ind (AREP, Slot Number, FI Index) LR_State[AREP]:=W-LR-RES	SAME

#	Current State	Event /Condition =>Action	Next State
192	t-state	FSPMS_Stop.rsp(+) (AREP) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => Length:=CurrentBuffer[AREP].WriteLength MSAC1S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME
193	t-state	FSPMS_Stop.rsp(+) (AREP) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => Length:=CurrentBuffer[AREP].WriteLength MSAC2S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME
194	t-state	FSPMS_Stop.rsp(-) (AREP, Error Code)/LR_State[AREP]==W-LR-RES&AREP.AR_Type=MS1=>CurrentBuffer[AREP].Empty:=TRUEError Decode:=DPV1Error Code 1:=Error CodeError Code 2:=0MSAC1S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2)LR_State[AREP]:=W-LR-IND	SAME
195	t-state	FSPMS_Stop.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC2S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
196	t-state	MSAC1S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.FIS.Extended_Function_Num=Resume) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length FI Index:=Data.FIS.FI_Index Execution Argument:=Data.FIS.Execution_Argument FSPMS_Resume.ind (AREP, Slot Number, FI Index, Execution Argument) LR_State[AREP]:=W-LR-RES	SAME
197	t-state	MSAC2S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.FIS.Extended_Function_Num=Resume) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length FI Index:=Data.FIS.FI_Index Execution Argument:=Data.FIS.Execution_Argument FSPMS_Resume.ind (AREP, Slot Number, FI Index, Execution Argument) LR_State[AREP]:=W-LR-RES	SAME
198	t-state	FSPMS_Resume.rsp(+) (AREP) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => Length:=CurrentBuffer[AREP].WriteLength MSAC1S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME
199	t-state	FSPMS_Resume.rsp(+) (AREP) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => Length:=CurrentBuffer[AREP].WriteLength MSAC2S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME

#	Current State	Event /Condition =>Action	Next State
200	t-state	FSPMS_Resume.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC1S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
201	t-state	FSPMS_Resume.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC2S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
202	t-state	MSAC1S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.FIS.Extended_Function_Num=Reset) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length FI Index:=Data.FIS.FI_Index FSPMS_Reset.ind (AREP, Slot Number, FI Index) LR_State[AREP]:=W-LR-RES	SAME
203	t-state	MSAC2S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.FIS.Extended_Function_Num=Reset) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length FI Index:=Data.FIS.FI_Index FSPMS_Reset.ind (AREP, Slot Number, FI Index) LR_State[AREP]:=W-LR-RES	SAME
204	t-state	FSPMS_Reset.rsp (+) (AREP) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => Length:=CurrentBuffer[AREP].WriteLength MSAC1S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME
205	t-state	FSPMS_Reset.rsp (+) (AREP) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => Length:=CurrentBuffer[AREP].WriteLength MSAC2S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME
206	t-state	FSPMS_Reset.rsp (-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC1S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME

#	Current State	Event /Condition =>Action	Next State
207	t-state	FSPMS_Reset.rsp (-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC2S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME
208	t-state	MSAC1S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.w.STATE.Extended_Function_Num=Get_State) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length FI Index:=Data.w.STATE.FI_Index FSPMS_Get_FI_State.ind (AREP, Slot Number, FI Index) LR_State[AREP]:=W-LR-RES	SAME
209	t-state	MSAC2S_Write.ind (Res SAP, Slot Number, Index, Length, Data) /LR_State[AREP]==W-LR-IND &SetContext(Res_SAP, Service)=TRUE && (Index=255 && Length >= 4) && (Data.w.STATE.Extended_Function_Num=Get_State) => CurrentBuffer[AREP].Empty:=TRUE CurrentBuffer[AREP].WriteLength:=Length FI Index:=Data.w.STATE.FI_Index FSPMS_Get_FI_State.ind (AREP, Slot Number, FI Index) LR_State[AREP]:=W-LR-RES	SAME
210	t-state	FSPMS_Get_FI_State.rsp (+) (AREP, FI State) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => CurrentBuffer[AREP].Empty:=FALSE CurrentBuffer[AREP].data.r.STATE.FI_State:=FI State Length:=CurrentBuffer[AREP].WriteLength MSAC1S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME
211	t-state	FSPMS_Get_FI_State.rsp (+) (AREP, FI State) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].Empty:=FALSE CurrentBuffer[AREP].data.r.STATE.FI_State:=FI State Length:=CurrentBuffer[AREP].WriteLength MSAC2S_Write.rsp(+) (Length) LR_State[AREP]:=W-LR-IND	SAME
212	t-state	FSPMS_Get_FI_State.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS1 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC1S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME

#	Current State	Event /Condition =>Action	Next State
213	t-state	FSPMS_Get_FI_State.rsp(-) (AREP, Error Code) /LR_State[AREP]==W-LR-RES &AREP.AR_Type=MS2 => CurrentBuffer[AREP].Empty:=TRUE Error Decode:=DPV1 Error Code 1:=Error Code Error Code 2:=0 MSAC1S_Write.rsp(-) (Error Decode, Error Code 1 Error Code 2) LR_State[AREP]:=W-LR-IND	SAME

8.1.4 Functions

Table 52 contains the functions used by the FSPMS, their arguments and their descriptions.

Table 52 – Functions used by the FSPMS

Function name	Description
SetContext(Rem_Add, Service)	This function checks if there exist an entry for the inputs Rem_Add resp. Res_SAP and invoked Service in the ARL and related CRL of the DP-slave. A) If there exist an entry then it returns TRUE and sets the local context according to the current CREP and AREP. B) Otherwise it returns FALSE.
FILL(Outstanding_Alarm_TABLE.Alarm_Status, State)	This function sets each entry of the two dimensional Outstanding_Alarm_TABLE to the given initial value of State. Valid parameters for State are pending/not_pending. The function has no return value.
Search_In_Outstanding_Alarm_TABLE(Alarm_Status)	This function scans each entry of the two dimensional Outstanding_Alarm_TABLE for a specific Alarm_Status. The return value of this function is an index that refers to an Alarm_Table entry that fulfills the Alarm_Status criteria pending/not_pending.
SReset_Req_Alarm_FIFO()	This function resets the FIFO for alarms not yet sent. During this function all entries are cleared. The function has no return value.
Store_To_Req_Alarm_FIFO(Alarm)	By means of this function the Alarm is stored into the Req_Alarm_FIFO. The function has no return value.
Load_From_Req_Alarm_FIFO ()	By means of this function one Alarm is read from the Req_Alarm_FIFO. The return value is an Alarm.
Req_Alarm_FIFO_state ()	This function checks the FIFO for alarms not yet sent. The function has the return values empty/not empty.

AcIs(Alarm_Type)	This function calculates the Alarm_Type related index as return value as follows:
	if (Alarm_Type = 1) return 0
	if (Alarm_Type = 2) return 1
	if (Alarm_Type = 3) return 2
	if (Alarm_Type = 4) return 3
	if (Alarm_Type = 5) return 4
	if (Alarm_Type = 6) return 5
	if (Alarm_Type >= 32) && (Alarm_Type <= 126) return 6

IsDynamicAttributeLoadable(Attribute)	This function checks the ARL and CRL if the given attribute is loadable in the current state. For the attribute DXB-Linktable the function returns TRUE: if all elements in the DXB-Linktable have a corresponding attribute in the CRL and none of the corresponding SSCY1S state machines is in the state RUN.

Function name	Description
SetDynamicAtrribute(Attribute)	This function stores the given attribute in the ARL or CRL. For the attribute DXB-Linktable each entry is stored at the corresponding CRL attribute.

Macros

ALARM_ENABLED

(
((Alarm_Type=3 OR Alarm_Type=4) &&
Actual_Enabled_Alarms[7]=TRUE)
OR ((Alarm_Type=2) && Actual_Enabled_Alarms[6]=TRUE)
OR ((Alarm_Type=1) && Actual_Enabled_Alarms[5]=TRUE)
OR ((Alarm_Type>=32 && Alarm_Type<=126)
&& Actual_Enabled_Alarms[4]=TRUE)
OR ((Alarm_Type=5) && Actual_Enabled_Alarms[3]=TRUE)
OR ((Alarm_Type=6) && Actual_Enabled_Alarms[2]=TRUE)
)

8.2 FSPMM1

8.2.1 Primitive definitions

8.2.1.1 Primitives exchanged between AP-Context and FSPMM1

Table 53 shows the service primitives including their associated parameters issued by the AP-Context and received by the FSPMM1.

Table 53 – Primitives issued by AP-Context to FSPMM1

Primitive name	Source	Associated parameters	Functions
Init DP master Cl1.req	AP-Context	Bus Para	Refer to FAL Service Definition in IEC 61158-5-3 and in IEC 61158-3-3
Reset DP master Cl1.req	AP-Context	(none)	Refer to FAL Service Definition in IEC 61158-5-3
Load ARL DP master Cl1.req	AP-Context	Alarm Max, List of ARL Entries	
Get ARL DP master Cl1.req	AP-Context	(none)	
Load CRL DP master Cl1.req	AP-Context	Update, List of CRL Entries	
Get CRL DP master Cl1.req	AP-Context	(none)	
CRL Slave Activate.req	AP-Context	Activate	
CRL New Prm.req	AP-Context	New Prm	
CRL New Prm Data.req	AP-Context	Prm Data	
Abort DP master Cl1.req	AP-Context	AREP	
Mark DP master Cl1.req	AP-Context	AREP	
Set Mode DP master Cl1.req	AP-Context	AREP, USIF State	
Load Bus Par DP master Cl1.req	AP-Context	Bus Para	
Delete SC DP master Cl1.req	AP-Context	Address	

Primitive name	Source	Associated parameters	Functions
Read Value DP master Cl1.req	AP-Context	Variable	
Get Slave Diag.req	AP-Context	AREP, CREP	
Set Output.req	AP-Context	AREP, CREP, Slot Number, Output Data, Final	
Get Input.req	AP-Context	AREP, CREP, Slot Number	
Global Control.req	AP-Context	AREP, Sync Command, Freeze Command, Group Select	
Read.req	AP-Context	AREP, Slot Number, Index, Length	
Write.req	AP-Context	AREP, Slot Number, Index, Length, Data	
Alarm Ack.req(+)	AP-Context	AREP, Slot Number, Alarm Type, Seq Nr	
Get Master Diag.rsp(+)	AP-Context	AREP, Diagnosis Data	
Get Master Diag.rsp(-)	AP-Context	AREP, Status	
Start Seq.rsp(+)	AP-Context	AREP, Max Len Data Unit	
Start Seq.rsp(-)	AP-Context	AREP, Status	
Download.rsp(+)	AP-Context	AREP	
Download.rsp(-)	AP-Context	AREP, Status	
Upload.rsp(+)	AP-Context	AREP, Data	
Upload.rsp(-)	AP-Context	AREP, Status	
End Seq.rsp(+)	AP-Context	AREP	
End Seq.rsp(-)	AP-Context	AREP, Status	
Act Param.rsp(+)	AP-Context	AREP	
Act Param.rsp(-)	AP-Context	AREP, Status	
Set Time.req	AP-Context	AREP, Time Value Local Time Diff Summertime Accuracy Synchronisation Active Announcement Hour	

Primitive name	Source	Associated parameters	Functions
Initiate Load.req	AP-Context	AREP, Slot Number, LR Index, Load Type, Load Image Size, Intersegment Request, Timeout User_Specific	
Pull Segment.req	AP-Context	AREP, Slot Number, LR Index, Segment Length	
Push Segment.req	AP-Context	AREP, Slot Number LR Index, Segment Length Segment Number, More Follows, Data	
Terminate Load.req	AP-Context	AREP Slot Number LR Index	
Start.req	AP-Context	AREP Slot Number FI Index Execution Argument	
Stop.req	AP-Context	AREP Slot Number FI Index	
Resume.req	AP-Context	AREP Slot Number FI Index Execution Argument	
Reset.req	AP-Context	AREP Slot Number FI Index	
Call.req	FSPMS	AREP Slot Number Entity Number FI Index Execution Argument	
Get_FI_State.req	FSPMS	AREP Slot Number FI Index	

Table 54 shows the service primitives including their associated parameters issued by the FSPMM1 and received by the AP-Context.

Table 54 – Primitives issued by FSPMM1 to AP-Context

Primitive name	Source	Associated parameters	Functions
Init DP master Cl1.cnf	FSPMM1	(none)	Refer to FAL Service Definition in IEC 61158-5-3
Reset DP master Cl1.cnf	FSPMM1	(none)	
Load ARL DP master Cl1.cnf(+)	FSPMM1	(none)	
Load ARL DP master Cl1.cnf(-)	FSPMM1	Status	
Get ARL DP master Cl1.cnf(+)	FSPMM1	Alarm Max, List of ARL Entries	
Get ARL DP master Cl1.cnf(-)	FSPMM1	Status	
Load CRL DP master Cl1.cnf(+)	FSPMM1	(none)	
Load CRL DP master Cl1.cnf(-)	FSPMM1	Status	
Get CRL DP master Cl1.cnf(+)	FSPMM1	List of CRL Entries	
Get CRL DP master Cl1.cnf(-)	FSPMM1	Status	
CRL Slave Activate.cnf(+)	FSPMM1		
CRL Slave Activate.cnf(-)	FSPMM1	Status	
CRL New Prm.cnf(+)	FSPMM1		
CRL New Prm.cnf(-)	FSPMM1	Status	
CRL New Prm Data.cnf(+)	FSPMM1		
CRL New Prm Data.cnf(-)	FSPMM1	Status	
Mark DP master Cl1.cnf(+)	FSPMM1	AREP, Dia	
Mark DP master Cl1.cnf(-)	FSPMM1	AREP, Status	
Set Mode DP master Cl1.cnf(+)	FSPMM1	AREP, Bus Accessible	
Set Mode DP master Cl1.cnf(-)	FSPMM1	AREP, Bus Accessible	
Load Bus Par DP master Cl1.cnf	FSPMM1	Status	
Delete SC DP master Cl1.cnf	FSPMM1	Status	
Read Value DP master Cl1.cnf	FSPMM1	Value, Status	
Get Slave Diag.cnf(+)	FSPMM1	AREP, CREP, Diag Data	
Get Slave Diag.cnf(-)	FSPMM1	AREP, CREP	
Set Output.cnf(+)	FSPMM1	AREP, CREP, Slot Number	
Set Output.cnf(-)	FSPMM1	AREP, CREP, Slot Number, Status	
Get Input.cnf(+)	FSPMM1	AREP, CREP, Slot Number, Input Data	
Get Input.cnf(-)	FSPMM1	AREP, CREP, Slot Number, Status	

Primitive name	Source	Associated parameters	Functions
Global Control.cnf(+)	FSPMM1	AREP	
Global Control.cnf(-)	FSPMM1	AREP, Status	
Read.cnf(+)	FSPMM1	AREP, Length, Data	
Read.cnf(-)	FSPMM1	AREP, Error Decode, Error Code 1 Error Code 2	
Write.cnf(+)	FSPMM1	AREP, Length	
Write.cnf(-)	FSPMM1	AREP, Error Decode, Error Code 1 Error Code 2	
Alarm Ack.cnf(+)	FSPMM1	AREP, Slot Number, Alarm Type, Seq Nr	
Alarm Ack.cnf(-)	FSPMM1	AREP Slot Number, Alarm Type, Seq Nr	
Get Master Diag.ind	FSPMM1	AREP, Mdiag Identifier	
Start Seq.ind	FSPMM1	AREP, Area Code, Timeout	
Download.ind	FSPMM1	AREP, Area Code, Add Offset, Data	
Upload.ind	FSPMM1	AREP, Area Code, Add Offset, Data Len	
End Seq.ind	FSPMM1	AREP	
Act Param.ind	FSPMM1	AREP, Area Code, Activate	
Act Para Brct.ind	FSPMM1	AREP, Area Code	
New Slave Diag.ind	FSPMM1	AREP, CREP	
New Input.ind	FSPMM1	AREP	
Alarm Notification.ind	FSPMM1	AREP, Slot Number, Alarm Type, Seq Nr, Add Ack, Alarm Specifier, Alarm Data	
DP master Cl1 Mode Changed.ind	FSPMM1	AREP, USIF State	
DP master Cl1 Event.ind		Event, Add Info	
DP master Cl1 Started.ind	FSPMM1	AREP	
DP master Cl1 Stopped.ind	FSPMM1	AREP	

Primitive name	Source	Associated parameters	Functions
Abort.ind	FSPMM1	AREP	
DP master Cl1 Reject.ind	FSPMM1	AREP, Reason	
DP master Cl1 Fault.ind	FSPMM1	(none)	
SYNCH.ind	FSPMM1	AREP	
SYNCH Delayed.ind	FSPMM1	AREP, T_{SH}	
DX Finished.ind	FSPMM1	AREP	
Set Time.ind	FSPMM1	AREP, Time Value, Local Time Diff, Summertime, Accuracy, Synchronisation Active, Announcement Hour	
Set Time.cnf	FSPMM1	AREP, Status	
Sync Interval Violation.ind	FSPMM1	AREP	
Initiate Load.cnf(+)	FSPMM1	AREP, Actual LR Size, Max Response Delay, Max Segment Length User Specific	
Initiate Load.cnf(-)	FSPMM1	AREP, Error Code	
Pull Segment.cnf(+)	FSPMM1	AREP, Segment Length, Segment Number, More Follows, Data	
Pull Segment.cnf(-)	FSPMM1	AREP, Error Code	
Push Segment.cnf(+)	FSPMM1	AREP	
Push Segment.cnf(-)	FSPMM1	AREP, Error Code	
Terminate Load.cnf(+)	FSPMM1	AREP	
Terminate Load.cnf(-)	FSPMM1	AREP, Error Code	
Start.cnf(+)	FSPMM1	AREP	
Start.cnf(-)	FSPMM1	AREP Error Code	
Stop.cnf(+)	FSPMM1	AREP	
Stop.cnf(-)	FSPMM1	AREP Error Code	
Resume.cnf(+)	FSPMM1	AREP	
Resume.cnf(-)	FSPMM1	AREP Error Code	
Reset.cnf(+)	FSPMM1	AREP	
Reset.cnf(-)	FSPMM1	AREP Error Code	
Call.cnf(+)	FSPMM1	AREP Result Argument	
Call.cnf(-)	FSPMM1	AREP Error Code	
Get FI State.cnf(+)	FSPMM1	AREP FI State	

Primitive name	Source	Associated parameters	Functions
Get FI State.cnf(-)	FSPMM1	AREP Error Code	

8.2.1.2 Parameters of FSPMM1 primitives

The parameters used with the primitives exchanged between the FSPMM1 and the AP-Context are described in FAL Service Definition in IEC 61158-5-3.

8.2.2 State machine description

This Machine is used to co-ordinate the Interactions between AP-Context and DMPMM1, MSCY1M, MSAC1M, MSAL1M and MMAC1. As the support of several State Machines for the communication with individual slave as well as timing constraints of the operation of slave require a consistent scheduling, this task is accomplished by FSPMM1 a s well.

For each DP-slave an individual set of state machines (MSCY1M, MSAL1M, MSAC1M) will be established which will be controlled by the Scheduler in the DP-master (Class 1). The Scheduler will be controlled by the AP-Context with special services. The Slave-Handlers(MSCY1M) are triggered sequentially from the FSPMS-Scheduler to perform the necessary actions. The FSPMM1 notifies the operation modes CLEAR and OPERATE in fixed time intervals (Dx_Control_Timer) to the DP-slaves. This occurs also at state transitions of FSPMM1.

The Scheduler provides four operation modes for the User:

OFFLINE

STOP

CLEAR

OPERATE

Local Variables

Go_AClr
(Boolean)

State variable which indicates by the value True that after finishing the current Slave cycle the FSPMM1 has to force all DP-slaves into a safe state (Scheduler-State = CLEAR).

Internal_Go_AClr
(Boolean)

Only after finishing the current Slave cycle the FSPMS has to go to state CLEAR if Go_AClr is True. During each cycle the state variable Internal_Go_AClr stores the actual value of Go_AClr. At the end of each cycle the value of Internal_Go_AClr is copied into Go_AClr.

Mark_Active
(Unsigned8)

Indicates the state of the local function Mark.

Range:

Possible values are:

0 => Mark is not active

1 => Mark is active, but not yet in execution

2 => Mark is active and in execution

Diag_Active
(Boolean)

Used to store the information if at least one of the DP-slaves is not in the data exchange mode (True). The information is issued to the User with Mark.cnf.

UGC_Count
(Unsigned8)

Global_Control requests of the User are possible at any time during the FSPMM1 operation modes CLEAR and OPERATE. Confirmations of these requests will be passed to the User. The FSPMM1 uses the service Global_Control to send cyclic Global_Control requests with its actual operation mode to all assigned DP-slaves. Confirmations of cyclic Global_Control requests have to be utilized by the FSPMM1. The FSPMM1 has to know which Global_Control.cnf belongs to which previously called Global_Control.req It is not possible to resolve this relationship only by using DL services. Therefore the FSPMM1 shall get this relationship information by using the sequential order of the received Global_Control confirmations. The counters UGC_Count and CGC_Count are used to store information about the order of the requests and also information about the order of the confirmations.

With the UGC_Count counter all User Global_Control requests are counted incrementally. The initial value is 0. It is decremented with each Global_Control.cnf. In addition the counter is related with the parameter Bus_Para.Max_User_Global_Control.

Range: 0 .. Bus_Para.Max_User_Global_Control

CGC_Count
(Unsigned8)

With this counter all cyclic Global_Control requests are counted which were started by the Scheduler. The initial value is 0. With each new cyclic Global_Control.req UGC_Count is incremented and its value is copied to CGC_Count. UGC_Count also is decremented with each new Global_Control.cnf. If this counter reaches the value 1, the actual Global_Control.cnf is a confirmation for a cyclic Global_Control.req

Range: 0 .. Bus_Para.Max_User_Global_Control

Bus_Accessible
(Boolean)

Indicates the actual state of the bus system. If services can be sent, its value is True.

Act_Rem_Add
(Unsigned8)

This variable stores the actually used station address of a DP-slave.

Range: 0 .. 125

Act_msac1m_Max_L_sdu_len
(Unsigned8)

This variable is used to store the actual value of the parameter msac1m_Max_L_sdu_len.

Range: 0 .. 240

User_Sched_Reset
(Boolean)

This variable indicates whether the actual reset was started by the User (AP-Context) or by any other State Machine.

Act_New_Bus_Para
(Bus_Para)

This structure is used to save an actual bus parameter set passed from the AP-Context to FSPMM1 temporarily.

DX_Lock
(Boolean)

This variable marks an active data exchange cycle to lock the data buffer and block user access. The value TRUE is set when the first Continue Slave Handler.req is issued until DX Finished.ind. The value remains FALSE until the next SYNCH.ind starts again the data exchange cycle.

DX_Synch
(Boolean)

This variable marks the interval between a SYNCH.ind and the next data exchange cycle. The value TRUE is set when the SYNCH.ind is received until the first Continue Slave Handler.req is issued. The value remains FALSE until the next SYNCH.ind.

Chg Buffer
(Boolean)

The value TRUE indicates that the buffers for input and output data below shall be changed with the next DX Finished.ind. It remains FALSE until the next Set Output.req (Final=TRUE) is received.

Bfr_Set_Output
(Octet String)

This variable is a local buffer for output data that belongs to the application. It allows asynchronous access from the application process to the output data in Buffered Synchronized Mode. The contents of the buffer is refreshed by the FSPMM1 when the application process has issued a Set Output.req (Final = TRUE) and the DX Finished.ind has been received. The buffer may accessed again after the next SYNCH.ind.

Bfr_Get_Input
(Octet String)

This variable is a local buffer for input data that belongs to the application. It allows asynchronous access from the application process to the input data in Buffered Synchronized Mode. The refreshment is done together with the Bfr_Set_Output.

Set_Output
(Octet String)

This variable is a local buffer for output data that belongs to the network. It allows asynchronous access from the application process to the output data in Buffered Synchronized Mode. The contents of the buffer is refreshed by the FSPMM1 when the application process has issued a Set Output.req (Final = TRUE) and the DX Finished.ind has been received. The buffer may be accessed again after the next SYNCH.ind.

Get_Input
(Octet String)

This variable is a local buffer for input data that belongs to the network. It allows asynchronous access from the application process to the input data in Buffered Synchronized Mode. The refreshment is done together with the Bfr_Set_Output

LR_State
(Array of Unsigned8)

This variable stores the state of the Load Region sequence of the corresponding CREP.

CurrentExtendedFunctionNum
(Array of Unsigned 8)

This variable stores the extended function number of the currently processed LR service of the corresponding CREP.

CurrentSlotNumber
(Array of Unsigned8)

This variable stores the slot number of the currently processed LR service of the corresponding CREP.

Timers

Min_SI_Interval_Timer
This timer supervises the minimal access control time of the DP-slaves. It is evaluated only within the FSPMM1 states CCLEAR and COPERATE.

Dx_Control_Interval_Timer
If this timer expires the broadcast service Global_Control is executed in order to inform the DP-slave whether the DP-master (Class 1) is in the state CLEAR or OPERATE. If the timer expires and the confirmation of the last Global_Control has not yet been received a physical bus access problem has occurred.

Macros

INDICATE_BUS_ACCESSIBILITY
If (Accessible = True) Bus_Accessible = True

8.2.3 FSPMM1 state table

Table 55 contains the complete description of the FSPMM1 state machine.

The following definition for a state set is used in Table 55.

t_state = STOP or CLEAR or OPERATE

Table 55 – FSPMM1 state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	Load ARL DP Master CI1.req (Alarm Max, List of ARL Entries) /valid parameters => Load ARL DP Master CI1.cnf(+)	POWER-ON
2	POWER-ON	Load ARL DP Master CI1.req (Alarm Max, List of ARL Entries) /invalid parameters => Status:= NO Load ARL DP Master CI1.cnf(-) (Status)	POWER-ON
3	POWER-ON	Get ARL DP Master CI1.req /ARL loaded => Get ARL DP Master CI1.cnf(+) (Alarm Max, List of ARL Entries)	POWER-ON
4	POWER-ON	Get ARL DP Master CI1.req /ARL not loaded => Status:= NO Get ARL DP Master CI1.cnf(-) (Status)	POWER-ON
5	POWER-ON	Load CRL DP Master CI1.req (List of CRL Entries) /valid parameters => Load CRL DP Master CI1.cnf(+)	POWER-ON

#	Current State	Event /Condition =>Action	Next State
6	POWER-ON	Load CRL DP Master CI1.req (List of CRL Entries) /invalid parameters => Status:= NO Load CRL DP Master CI1.cnf(-) (Status)	POWER-ON
7	POWER-ON	Get CRL DP Master CI1.req /CRL loaded => Get CRL DP Master CI1.cnf(+) (List of CRL Entries)	POWER-ON
8	POWER-ON	Get CRL DP Master CI1.req /CRL not loaded => Status:= NO Get CRL DP Master CI1.cnf(-) (Status)	POWER-ON
9	POWER-ON	FSPMM1_Init DP Master CI1.req(Bus_Para) => User_Init:=TRUE USER_ChangedCritical_Para:= FALSE DX_Lock:= FALSE Chg_Buffer:= FALSE Current Extended Function Num[CREP]:=No_Service LR_State[CREP]:=W-LR-REQ DMPMM1_Minit_DLL.req(Bus_Para)	INIT-DMPM
10	INIT-DMPM	DMPMM1_Minit_DLL.cnf => Act_Rem_Add:= 0 MSAC1M_MInit_MSAC1.req(Rem_Add:=Act_Rem_Add)	INIT-AC1
11	—	—	—
12	INIT-AC1	MSAC1M_MInit_MSAC1.cnf(Rem_Add) /Act_Rem_Add<125 => Act_Rem_Add:= Act_Rem_Add + 1 MSAC1M_MInit_MSAC1.req(Rem_Add:=Act_Rem_Add)	INIT-AC1
13	INIT-AC1	MSAC1M_MInit_MSAC1.cnf(Rem_Add) /Act_Rem_Add=125 => Act_Rem_Add:= 0 MSAL1M_MInit.req(Rem_Add:=Act_Rem_Add)	INIT-AL1
14	INIT-AL1	MSAL1M_MInit.cnf(Rem_Add) /Act_Rem_Add<125 => Act_Rem_Add:= Act_Rem_Add + 1 MSAL1M_MInit.req(Rem_Add:=Act_Rem_Add)	INIT-AL1
15	INIT-AL1	MSAL1M_MInit.cnf(Rem_Add) /Act_Rem_Add = 125 => Act_Rem_Add:= 0 MSCY1M_Minit_MS0.req(Rem_Add:=Act_Rem_Add)	INIT-CY1
16	INIT-CY1	MSCY1M_Minit_MS0.cnf(Rem_Add) /Act_Rem_Add<125 => Act_Rem_Add:= Act_Rem_Add + 1 MSCY1M_Minit_MS0.req(Rem_Add:=Act_Rem_Add)	INIT-CY1
17	INIT-CY1	MSCY1M_Minit_MS0.cnf(Rem_Add) /Act_Rem_Add=125 && IsARExistent(MM)=TRUE => MMAC1_Minit_MM.req	INIT-MM1

#	Current State	Event /Condition =>Action	Next State

18	INIT-CY1	MSCY1M_Minit_MS0.cnf(Rem_Add)
		/Act_Rem_Add=125 && IsARExistent(MM)=FALSE &&
		USER_ChangedCritical_Para = FALSE
		=>
		Bus_Accessible:= FALSE
		if(USER_SetMode_Offline=FALSE)
		FSPMM1_Init DP Master CI1.cnf
		else
		FSPMM1_Set_Mode DP Master CI1.cnf(+)(AREP, Bus_Accessible)
		USER_SetMode_Offline:=FALSE
		endif
			OFFLINE

19	INIT-CY1	MSCY1M_Minit_MS0.cnf(Rem_Add) /Act_Rem_Add=125 && IsARExistent(MM)=FALSE && USER_ChangedCritical_Para = TRUE => DMPMM1_Set_Bus_Par.req(Bus_Para:=Act_New_Bus_Para)	LDBP
20	INIT-MM1	MMAC1_Minit_MM.cnf /USER_ChangedCritical_Para = TRUE => DMPMM1_Set_Bus_Par.req(Bus_Para:=Act_New_Bus_Para)	LDBP

21	INIT-MM1	MMAC1_Minit_MM.cnf
		/USER_ChangedCritical_Para = FALSE
		=>
		Bus_Accessible:= FALSE
		if(USER_SetMode_Offline=FALSE)
		FSPMM1_Init DP Master CI1.cnf
		else
		FSPMM1_Set_Mode DP Master CI1.cnf(+)(AREP, Bus_Accessible)
		USER_SetMode_Offline:=FALSE
		endif
			OFFLINE

22	OFFLINE	FSPMM1_Set_Mode.req(USIF_State) /USIF_State <> Stop && USIF_State <> Offline => FSPMM1_Set_Mode DP Master CI1.cnf(-)(AREP, Bus_Accessible)	OFFLINE
23	OFFLINE	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Offline => FSPMM1_Set_Mode DP Master CI1.cnf(+)(AREP, Bus_Accessible)	OFFLINE
24	OFFLINE	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Stop && Bus_Para invalid => FSPMM1_Set_Mode DP Master CI1.cnf(-)(AREP, Bus_Accessible)	OFFLINE
25	OFFLINE	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Stop && Bus_Para valid => Mark_Active:= 0 Diag_Active:= False DMPMM1_Set_Bus_Par.req(Bus_Para)	LDBP
26	OFFLINE	FSPMM1_Mark.req(AREP) => FSPMM1_Mark.cnf(-)(AREP, Status:=NO)	OFFLINE
27	OFFLINE	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) => FSPMM1_Global_Control.cnf(-)(AREP, Status:=NO)	OFFLINE
28	OFFLINE	FSPMM1_Load_Bus_Par DP Master CI1.req(Bus_Para) => FSPMM1_Load_Bus_Par.cnf(-)(Status:=NO)	OFFLINE
29	OFFLINE	FSPMM1_Delete_SC DP Master CI1.req(Address) => DMPMM1_Delete_SC.req(Address)	DELSC-OFF
30	OFFLINE	FSPMM1_Read_Value DP Master CI1.req(Variable) => DMPMM1_Read_Value.req(Variable)	RDVAL-OFF

#	Current State	Event /Condition =>Action	Next State
31	LDBP	DMPMM1_Set_Bus_Par.cnf /USER_ChangeCritical_Para = FALSE => FSPMM1_Set_Mode DP Master CI1.cnf(+)(AREP, Bus_Accessible)	STOP
32	LDBP	DMPMM1_Set_Bus_Par.cnf /USER_ChangeCritical_Para = TRUE => FSPMM1_Load_Bus_Para.cnf(+)	STOP
33	DELSC-OFF	DMPMM1_Delete_SC.cnf => FSPMM1_Delete_SC DP Master CI1.cnf	OFFLINE
34	RDVAL-OFF	DMPMM1_Read_Value.cnf(Status,Value) => FSPMM1_Read_Value DP Master CI1.cnf(Status, Value)	OFFLINE
35	STOP	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Offline => USER_SetMode_Offline:=TRUE	RESET
36	STOP	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Stop => FSPMM1_Set_Mode DP Master CI1.cnf(+)(AREP, Bus_Accessible)	STOP
37	STOP	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Operate => FSPMM1_Set_Mode DP Master CI1.cnf(-)(AREP, Bus_Accessible)	STOP
38	STOP	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Clear => UGC_Count:= 0 CGC_Count:= 0 Bus_Acessible:= True Go_AClr:= Bus_Para.Error_Action_Flag Internal_Go_AClr:= False Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2) MSCY1M_Start_Slave_Handler.req(0 .. 125)	CLEAR
39	STOP	FSPMM1_Mark.req(AREP) => FSPMM1_Mark.cnf(-)(AREP, Status:=NO)	STOP
40	STOP	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) => FSPMM1_Global_Control.cnf(-)(AREP, Status:=NO)	STOP
41	STOP	FSPMM1_Load_Bus_Par DP Master CI1.req(Bus_Para) /Bus_Para invalid => FSPMM1_Load_Bus_Par.cnf(-)(Status:=IV)	STOP
42	STOP	FSPMM1_Load_Bus_Par DP Master CI1.req(Bus_Para) /(Bus_Para valid) && (critical parameters unchanged) => DMPMM1_Set_Bus_Par.req(Bus_Para)	LDBP-ST-CPU
43	STOP	FSPMM1_Load_Bus_Par DP Master CI1.req(Bus_Para) /(Bus_Para valid) && (critical parameters changed) => Act_New_Bus_Para:= Bus_Para User_ChangedCritical_Para:= TRUE	RESET
44	STOP	FSPMM1_Delete_SC DP Master CI1.req(Address) => DMPMM1_Delete_SC.req(Address)	DELSC-ST
45	STOP	FSPMM1_Read_Value DP Master CI1.req(Variable) => DMPMM1_Read_Value.req(Variable)	RDVAL-ST

#	Current State	Event /Condition =>Action	Next State
46	LDBP-ST-CPU	DMPMM1_Set_Bus_Par.cnf => FSPMM1_Load_Bus_Para.cnf(+)	STOP
47	DELSC-ST	DMPMM1_Delete_SC.cnf => FSPMM1_Delete_SC DP Master CI1.cnf	STOP
48	RDVAL-ST	DMPMM1_Read_Value.cnf(Status,Value) => FSPMM1_Read_Value DP Master CI1.cnf(Status, Value)	STOP
49	CLEAR	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Offline => FSPMM1_Set_Mode DP Master CI1.cnf(-)(AREP, Bus_Accessible)	CLEAR
50	CLEAR	FSPMM1_Mark DP Master CI1.req(AREP) /Mark_Active = 0 => Mark_Active:= 1	CLEAR
51	CLEAR	FSPMM1_Mark DP Master CI1.req(AREP) /Mark_Active <> 0 => FSPMM1_Mark.cnf(-)(AREP, Status:=NO)	CLEAR
52	CLEAR	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Stop && Mark_Active = 0 => Stop Dx_Control_Interval_Timer MSCY1M_Stop_Slave_Handler.req(0..125)	SLHSTP
53	CLEAR	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Stop && Mark_Active <> 0 => Mark_Active:= 0 Stop Dx_Control_Interval_Timer FSPMM1_Mark.cnf(-)(AREP; Status:=NO) MSCY1M_Stop_Slave_Handler.req(0..125)	SLHSTP
54	CLEAR	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Clear => FSPMM1_Set_Mode DP Master CI1.cnf(+)(AREP, Bus_Accessible)	CLEAR
55	CLEAR	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Operate && Go_AClr = False && Internal_Go_AClr = False && CGC_Count > 0 => Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2	SGC-CO-WGC
56	CLEAR	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Operate && Go_AClr = False && Internal_Go_AClr = False && CGC_Count = 0 => UGC_Count:= UGC_Count+1 CGC_Count:= UGC_Count Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2 DMPMM1_Global_Control.req(Rem_Add:=127, Control_Command:=0, Group_Select:=0)	SGC-CO
57	CLEAR	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Operate && (Go_AClr = True || Internal_Go_AClr = True) => FSPMM1_Set_Mode DP Master CI1.cnf(-)(AREP, Bus_Accessible)	CLEAR
58	CLEAR	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /NOT received all MSCY1M_Start_Slave_Handler.cnf => FSPMM1_Global_Control.cnf(-)(AREP, Status:=NO)	CLEAR

#	Current State	Event /Condition =>Action	Next State
59	CLEAR	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /received all MSCY1M_Start_Slave_Handler.cnf && CGC_Count = 0 && UGC_Count >= Bus_Para.Max_User_Global_Control => FSPMM1_Global_Control.cnf(-)(AREP, Status:=NO)	CLEAR
60	CLEAR	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /received all MSCY1M_Start_Slave_Handler.cnf && CGC_Count = 0 && UGC_Count < Bus_Para.Max_User_Global_Control && (Isochronous Mode = Not Synchronized || Group_Select < 0x80) => UGC_Count:= UGC_Count+1 DMPMM1_Global_Control.req(Rem_Add:=CREP.Rem_Add, Control_Command.1:=1, Group_Select)	CLEAR
61	CLEAR	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /received all MSCY1M_Start_Slave_Handler.cnf && CGC_Count > 0 && UGC_Count >= Bus_Para.Max_User_Global_Control+1 => FSPMM1_Global_Control.cnf(-)(AREP, Status:=NO)	CLEAR
62	CLEAR	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /received all MSCY1M_Start_Slave_Handler.cnf && CGC_Count > 0 && UGC_Count < Bus_Para.Max_User_Global_Control+1 && (Isochronous Mode = Not Synchronized || Group_Select < 0x80) => UGC_Count:= UGC_Count+1 DMPMM1_Global_Control.req(Rem_Add:=CREP.Rem_Add, Control_Command.1:=1, Group_Select)	CLEAR
63	CLEAR	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count = 0 => INDICATE_BUS_ACCESSIBILITY UGC_Count:= UGC_Count-1 FSPMM1_Global_Control.cnf(+)(AREP)	CLEAR
64	CLEAR	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count = 1 => INDICATE_BUS_ACCESSIBILITY UGC_Count:= UGC_Count-1 CGC_Count:= 0	CLEAR
65	CLEAR	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /received all MSCY1M_Start_Slave_Handler.cnf && CGC_Count = 0 && UGC_Count < Bus_Para.Max_User_Global_Control && Isochronous Mode = (Buffered Synchronized || Enhanced Synchronized) && Group_Select >= 0x80 => FSPMM1_Global_Control.cnf(-)(AREP, Status:=GE)	CLEAR
66	CLEAR	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /received all MSCY1M_Start_Slave_Handler.cnf && CGC_Count > 0 && UGC_Count < Bus_Para.Max_User_Global_Control+1 && Isochronous Mode = (Buffered Synchronized || Enhanced Synchronized) && Group_Select >= 0x80 => FSPMM1_Global_Control.cnf(-)(AREP, Status:=GE)	CLEAR

#	Current State	Event /Condition =>Action	Next State
67	CLEAR	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count > 1 => INDICATE_BUS_ACCESSIBILITY UGC_Count:= UGC_Count-1 CGC_Count:= CGC_Count-1 FSPMM1_Global_Control.cnf(+)(AREP)	CLEAR
68	CLEAR	MSCY1M_Start_Slave_Handler.cnf(Rem_Add) /NOT received all MSCY1M_Start_Slave_Handler.cnf => ignore	CLEAR
69	CLEAR	MSCY1M_Start_Slave_Handler.cnf(Rem_Add) /received all MSCY1M_Start_Slave_Handler.cnf && Isochronous Mode = Not Synchronized => Internal_Go_AClr:= False Start Min_SI_Interval_Timer(Bus_Para.Min_Slave_Interval) Set_Mode.cnf(Status:=OK, Bus_Accessible) MSCY1M_Cont_Slave_Handler.req(0..125, Output_Clear:=True)	CLEAR
70	CLEAR	MSCY1M_Start_Slave_Handler.cnf(Rem_Add) /received all MSCY1M_Start_Slave_Handler.cnf && Isochronous Mode <> Not Synchronized => Internal_Go_AClr:= False Set_Mode.cnf(Status:=OK, Bus_Accessible) MSCY1M_Cont_Slave_Handler.req(0..125, Output_Clear:=True)	CLEAR
71	CLEAR	DMPMM1_SYNCH.ind => FSPMM1_DP Master Cl1 Fault.ind	RESET
72	CLEAR	MSCY1M_Cont_Slave_Handler.cnf(Rem_Add, Diag, No_AClr) /NOT received all MSCY1M_Cont_Slave_Handler.cnf && Mark_Active = 2 => Diag_Active:= Diag_Active OR Diag Internal_Go_AClr:= Internal_Go_AClr OR (Bus_Para.Error_Action_Flag AND NOT No_AClr)	CLEAR
73	CLEAR	MSCY1M_Cont_Slave_Handler.cnf(Rem_Add, Diag, No_AClr) /NOT received all MSCY1M_Cont_Slave_Handler.cnf && Mark_Active <> 2 => Internal_Go_AClr:= Internal_Go_AClr OR (Bus_Para.Error_Action_Flag AND NOT No_AClr)	CLEAR
74	CLEAR	MSCY1M_Cont_Slave_Handler.cnf(Rem_Add, Diag, No_AClr) /received all MSCY1M_Cont_Slave_Handler.cnf && Mark_Active = 0 => Internal_Go_AClr:= Internal_Go_AClr OR (Bus_Para.Error_Action_Flag AND NOT No_AClr) DX_Lock:= FALSE	CCLEAR
75	CLEAR	MSCY1M_Cont_Slave_Handler.cnf(Rem_Add, Diag, No_AClr) /received all MSCY1M_Cont_Slave_Handler.cnf && Mark_Active = 1 => Mark_Active:=2 Diag_Active:= False Internal_Go_AClr:= Internal_Go_AClr OR (Bus_Para.Error_Action_Flag AND NOT No_AClr) DX_Lock:= FALSE	CCLEAR

#	Current State	Event /Condition =>Action	Next State
76	CLEAR	MSCY1M_Cont_Slave_Handler.cnf(Rem_Add, Diag, No_AClr) /received all MSCY1M_Cont_Slave_Handler.cnf && Mark_Active = 2 => Diag_Active:= Diag_Active OR Diag Internal_Go_AClr:= Internal_Go_AClr OR (Bus_Para.Error_Action_Flag AND NOT No_AClr) Mark_Active:=0 DX_Lock:= FALSE FSPMM1_Mark.cnf(+)(AREP, Dia:=Diag_Active)	CCLEAR
77	CLEAR	Dx_Control_Interval_Timer expired /CGC_Count = 0 => Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2) UGC_Count:= UGC_Count+1 CGC_Count:= UGC_Count DMPMM1_Global_Control.req(Rem_Add:=127, Control_Command:=2, Group_Select:=0)	CLEAR
78	CLEAR	Dx_Control_Interval_Timer expired /CGC_Count > 0 => Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2) Bus_Accessible:= False	CLEAR
79	CLEAR	FSPMM1_Load_Bus_Par DP Master CI1.req(Bus_Para) /Bus_Para invalid => FSPMM1_Load_Bus_Par.cnf(-)(Status:=IV)	CLEAR
80	CLEAR	FSPMM1_Load_Bus_Par DP Master CI1.req(Bus_Para) /(Bus_Para valid) && (critical parameters changed) => FSPMM1_Load_Bus_Par.cnf(-)(Status:=NO)	CLEAR
81	CLEAR	FSPMM1_Load_Bus_Par DP Master CI1.req(Bus_Para) /(Bus_Para valid) && (critical parameters unchanged) => DMPMM1_Set_Bus_Par.req(Bus_Para)	LDBP-CL-CPU
82	CLEAR	FSPMM1_Delete_SC DP Master CI1.req(Address) => DMPMM1_Delete_SC.req(Address)	DELSC-CL
83	CLEAR	FSPMM1_Read_Value DP Master CI1.req(Variable) => DMPMM1_Read_Value.req(Variable)	RDVAL-CL
84	SLHSTP	MSCY1M_Stop_Slave_Handler.cnf(Rem_Add) /NOT received all MSCY1M_Stop_Slave_Handler.cnf => ignore	SLHSTP
85	SLHSTP	MSCY1M_Stop_Slave_Handler.cnf(Rem_Add) /received all MSCY1M_Stop_Slave_Handler.cnf => Set_Mode.cnf(Status:=OK, Bus_Accessible)	STOP
86	SGC-CO-WGC	Dx_Control_Interval_Timer expired => Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2) Bus_Accessible:= False FSPMM1_Set_Mode DP Master CI1.cnf(-)(AREP, Bus_Accessible)	CLEAR
87	SGC-CO-WGC	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count > 1 => INDICATE_BUS_ACCESSIBILITY UGC_Count:= UGC_Count-1 CGC_Count:= CGC_Count-1 FSPMM1_Global_Control.cnf(+)(AREP)	SGC-CO-WGC

#	Current State	Event /Condition =>Action	Next State
88	SGC-CO-WGC	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count = 1 => INDICATE_BUS_ACCESSIBILITY CGC_Count:= UGC_Count DMPMM1_Global_Control.req(Rem_Add:=127, Control_Command:=0, Group_Select:=0)	SGC-CO
89	SGC-CO	Dx_Control_Interval_Timer expired => Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2) Bus_Accessible:= False Set_Mode.cnf(Status:=OK, Bus_Accessible)	OPERATE
90	SGC-CO	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count > 1 => INDICATE_BUS_ACCESSIBILITY UGC_Count:= UGC_Count-1 CGC_Count:= CGC_Count-1 FSPMM1_Global_Control.cnf(+)(AREP)	SGC-CO
91	SGC-CO	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count = 1 => INDICATE_BUS_ACCESSIBILITY UGC_Count:= UGC_Count-1 CGC_Count:= 0 FSPMM1_Set_Mode DP Master Cl1.cnf(+)(AREP, Bus_Accessible)	OPERATE
92	CCLEAR	Min_SI_Interval_Timer expired => Go_AClr:= Internal_Go_AClr Internal_Go_AClr:= False Start Min_SI_Interval_Timer(Bus_Para.Min_Slave_Interval) MSCY1M_Cont_Slave_Handler.req(0..125, Output_Clear:=True)	CLEAR
93	CCLEAR	DMPMM1_SYNCH.ind => DX_Lock:= TRUE Output_Data = Set_Output MSCY1M_Cont_Slave_Handler.req(0..125, Output_Clear:=False) MSCY1M_Set_Output.req(Rem_Add:=0..125, Slot_Number:=ALL, Output_Data) MSCY1M_Get_Input.req(Rem_Add:=0..125, Slot_Number:=ALL) FSPMM1_SYNCH.ind (AREP)	CLEAR
94	LDBP-CL-CPU	DMPMM1_Set_Bus_Par.cnf => FSPMM1_Load_Bus_Para.cnf(+)	CLEAR
95	DELSC-CL	DMPMM1_Delete_SC.cnf => FSPMM1_Delete_SC DP Master Cl1.cnf	CLEAR
96	RDVAL-CL	DMPMM1_Read_Value.cnf(Status,Value) => FSPMM1_Read_Value DP Master Cl1.cnf(Status, Value)	CLEAR
97	OPERATE	FSPMM1_Set_Mode.req(USIF_State) /USIF_State <> Clear && USIF_State <> Operate => FSPMM1_Set_Mode DP Master Cl1.cnf(-)(AREP, Bus_Accessible)	OPERATE
98	OPERATE	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Operate => FSPMM1_Set_Mode DP Master Cl1.cnf(+)(AREP, Bus_Accessible)	OPERATE
99	OPERATE	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Clear && CGC_Count > 0 => Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2)	SGC-OC-WGC

#	Current State	Event /Condition =>Action	Next State
100	OPERATE	FSPMM1_Set_Mode.req(USIF_State) /USIF_State = Clear && CGC_Count = 0 => Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2 UGC_Count:= UGC_Count+1 CGC_Count:= UGC_Count DMPMM1_Global_Control.req(Rem_Add:=127, Control_Command:=2, Group_Select:=0)	SGC-OC
101	OPERATE	FSPMM1_Mark DP Master CI1.req(AREP) /Mark_Active = 0 => Mark_Active:= 1	OPERATE
102	OPERATE	FSPMM1_Mark DP Master CI1.req(AREP) /Mark_Active <> 0 => FSPMM1_Mark.cnf(-)(AREP, Status:=NO)	OPERATE
103	OPERATE	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /Control_Command.1 = 1 => FSPMM1_Global_Control.cnf(-)(AREP, Status:=NO)	OPERATE
104	OPERATE	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /Control_Command.1 = 0 && CGC_Count = 0 && UGC_Count >= Bus_Para.Max_User_Global_Control => FSPMM1_Global_Control.cnf(-)(AREP, Status:=NO)	OPERATE
105	OPERATE	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /Control_Command.1 = 0 && CGC_Count = 0 && UGC_Count < Bus_Para.Max_User_Global_Control && (Isochronous Mode = Not Synchronized || Group_Select < 0x80) => UGC_Count:= UGC_Count+1 DMPMM1_Global_Control.req(Rem_Add:=CREP.Rem_Add, Control_Command.1:=0, Group_Select)	OPERATE
106	OPERATE	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /Control_Command.1 = 0 && CGC_Count > 0 && UGC_Count >= Bus_Para.Max_User_Global_Control+1 => FSPMM1_Global_Control.cnf(-)(AREP, Status:=NO)	OPERATE
107	OPERATE	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /Control_Command.1 = 0 && CGC_Count > 0 && UGC_Count < Bus_Para.Max_User_Global_Control+1 && (Isochronous Mode = Not Synchronized || Group_Select < 0x80) => UGC_Count:= UGC_Count+1 DMPMM1_Global_Control.req(Rem_Add:=CREP.Rem_Add, Control_Command.1:=0, Group_Select)	OPERATE
108	OPERATE	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count = 0 => INDICATE_BUS_ACCESSIBILITY UGC_Count:= UGC_Count-1 FSPMM1_Global_Control.cnf(+)(AREP)	OPERATE

#	Current State	Event /Condition =>Action	Next State
109	OPERATE	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count = 1 => INDICATE_BUS_ACCESSIBILITY UGC_Count:= UGC_Count-1 CGC_Count:= 0	OPERATE
110	OPERATE	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count > 1 => INDICATE_BUS_ACCESSIBILITY UGC_Count:= UGC_Count-1 CGC_Count:= CGC_Count-1 FSPMM1_Global_Control.cnf(+)(AREP)	OPERATE
111	OPERATE	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /Control_Command.1 = 0 && CGC_Count = 0 && UGC_Count < Bus_Para.Max_User_Global_Control && Isochronous Mode = (Buffered Synchronized || Enhanced Synchronized) && Group_Select >= 0x80 => FSPMM1_Global_Control.cnf(-)(AREP, Status:=GE)	OPERATE
112	OPERATE	FSPMM1_Global_Control.req(AREP, Sync_Command, Freeze_Command, Group_Select) /Control_Command.1 = 0 && CGC_Count > 0 && UGC_Count < Bus_Para.Max_User_Global_Control+1 && Isochronous Mode = (Buffered Synchronized || Enhanced Synchronized) && Group_Select >= 0x80 => FSPMM1_Global_Control.cnf(-)(AREP, Status:=GE)	OPERATE
113	OPERATE	DMPMM1_SYNCH.ind => FSPMM1_DP Master Cl1 Fault.ind	RESET
114	OPERATE	MSCY1M_Cont_Slave_Handler.cnf(Rem_Add, Diag, No_AClr) /NOT received all MSCY1M_Cont_Slave_Handler.cnf && Mark_Active = 2 => Diag_Active:=Diag_Active OR Diag Go_AClr:= Go_AClr OR (Bus_Para.Error_Action_Flag AND NOT No_AClr)	OPERATE
115	OPERATE	MSCY1M_Cont_Slave_Handler.cnf(Rem_Add, Diag, No_AClr) /NOT received all MSCY1M_Cont_Slave_Handler.cnf && Mark_Active <> 2 => Go_AClr:= Go_AClr OR (Bus_Para.Error_Action_Flag AND NOT No_AClr)	OPERATE
116	OPERATE	MSCY1M_Cont_Slave_Handler.cnf(Rem_Add, Diag, No_AClr) /received all MSCY1M_Cont_Slave_Handler.cnf && Mark_Active = 0 && (Isochronous Mode = Not Synchronized || Isochronous Mode = Enhanced Synchronized) => Go_AClr:= Go_AClr OR (Bus_Para.Error_Action_Flag AND NOT No_AClr) DX_Lock:= FALSE FSPMM1_DX_Finished.ind (AREP)	COPERATE

#	Current State	Event /Condition =>Action	Next State
117	OPERATE	MSCY1M_Cont_Slave_Handler.cnf(Rem_Add, Diag, No_AClr) /received all MSCY1M_Cont_Slave_Handler.cnf && Mark_Active = 1 && (Isochronous Mode = Not Synchronized || Isochronous Mode = Enhanced Synchronized) => Mark_Active:=2 Diag_Active:= False Go_AClr:= Go_AClr OR (Bus_Para.Error_Action_Flag AND NOT No_AClr) DX_Lock:= FALSE FSPMM1_DX_Finished.ind (AREP)	COPERATE
118	OPERATE	MSCY1M_Cont_Slave_Handler.cnf(Rem_Add, Diag, No_AClr) /received all MSCY1M_Cont_Slave_Handler.cnf && Mark_Active = 2 && (Isochronous Mode = Not Synchronized || Isochronous Mode = Enhanced Synchronized) => Diag_Active:=Diag_Active OR Diag Go_AClr:= Go_AClr OR (Bus_Para.Error_Action_Flag AND NOT No_AClr) Mark_Active:=0 DX_Lock:= FALSE FSPMM1_Mark.cnf(+)(AREP, Dia:=Diag_Active) DX_Finished.ind (AREP)	COPERATE
119	OPERATE	MSCY1M_Cont_Slave_Handler.cnf(Rem_Add, Diag, No_AClr) /received all MSCY1M_Cont_Slave_Handler.cnf && Mark_Active = 0 && Isochronous Mode = Buffered Synchronized => Go_AClr:= Go_AClr OR (Bus_Para.Error_Action_Flag AND NOT No_AClr) DX_Lock:= FALSE Chg_Bfr (Bfr_Get_Input, Get_Input) FSPMM1_DX_Finished.ind (AREP)	COPERATE
120	OPERATE	MSCY1M_Cont_Slave_Handler.cnf(Rem_Add, Diag, No_AClr) /received all MSCY1M_Cont_Slave_Handler.cnf && Mark_Active = 1 && Isochronous Mode = Buffered Synchronized => Mark_Active:=2 Diag_Active:= False Go_AClr:= Go_AClr OR (Bus_Para.Error_Action_Flag AND NOT No_AClr) DX_Lock:= FALSE Chg_Bfr (Bfr_Get_Input, Get_Input) FSPMM1_DX_Finished.ind (AREP)	COPERATE
121	OPERATE	MSCY1M_Cont_Slave_Handler.cnf(Rem_Add, Diag, No_AClr) /received all MSCY1M_Cont_Slave_Handler.cnf && Mark_Active = 2 && Isochronous Mode = Buffered Synchronized => Diag_Active:=Diag_Active OR Diag Go_AClr:= Go_AClr OR (Bus_Para.Error_Action_Flag AND NOT No_AClr) Mark_Active:=0 DX_Lock:= FALSE FSPMM1_Mark.cnf(+)(AREP, Dia:=Diag_Active) DX_Finished.ind (AREP)	COPERATE
122	OPERATE	Dx_Control_Interval_Timer expired /CGC_Count = 0 => Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2) UGC_Count:= UGC_Count+1 CGC_Count:= UGC_Count DMPMM1_Global_Control.req(Rem_Add:=127, Control_Command:=0, Group_Select:=0)	OPERATE

#	Current State	Event /Condition =>Action	Next State
123	OPERATE	Dx_Control_Interval_Timer expired /CGC_Count > 0 => Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2) Bus_Accessible:= False	OPERATE
124	OPERATE	FSPMM1_Load_Bus_Par DP Master CI1.req(Bus_Para) /Bus_Para invalid => FSPMM1_Load_Bus_Par.cnf(-)(Status:=IV)	OPERATE
125	OPERATE	FSPMM1_Load_Bus_Par DP Master CI1.req(Bus_Para) /(Bus_Para valid) && (critical parameters changed) => FSPMM1_Load_Bus_Par.cnf(-)(Status:=NO)	OPERATE
126	OPERATE	FSPMM1_Load_Bus_Par DP Master CI1.req(Bus_Para) /(Bus_Para valid) && (critical parameters unchanged) => DMPMM1_Set_Bus_Par.req(Bus_Para)	LDBP-OP-CPU
127	OPERATE	FSPMM1_Delete_SC DP Master CI1.req(Address) => DMPMM1_Delete_SC.req(Address)	DELSC-OP
128	OPERATE	FSPMM1_Read_Value DP Master CI1.req(Variable) => DMPMM1_Read_Value.req(Variable)	RDVAL-OP
129	SGC-OC-WGC	Dx_Control_Interval_Timer expired /Go_AClr = False => Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2) Bus_Accessible:= False FSPMM1_Set_Mode DP Master CI1.cnf(-)(AREP, Bus_Accessible)	OPERATE
130	SGC-OC-WGC	Dx_Control_Interval_Timer expired /Go_AClr = True => Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2) Bus_Accessible:= False	SGC-OC-WGC
131	SGC-OC-WGC	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count > 1 => INDICATE_BUS_ACCESSIBILITY UGC_Count:= UGC_Count-1 CGC_Count:= CGC_Count-1 FSPMM1_Global_Control.cnf(+)(AREP)	SGC-OC-WGC
132	SGC-OC-WGC	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count = 1 => INDICATE_BUS_ACCESSIBILITY CGC_Count:= UGC_Count DMPMM1_Global_Control.req(Rem_Add:=127, Control_Command:=2, Group_Select:=0)	SGC-OC
133	SGC-OC	Dx_Control_Interval_Timer expired /Go_AClr = False => Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2) Bus_Accessible:= False Set_Mode.cnf(Status:=OK, Bus_Accessible)	CLEAR
134	SGC-OC	Dx_Control_Interval_Timer expired /Go_AClr = True => Start Dx_Control_Interval_Timer(Bus_Para.Data_Control_Time/2) Bus_Accessible:= False FSPMM1_DP Master CI1 Mode_Changed.ind(AREP, USIF_State:=Clear) MSCY1M_Cont_Slave_Handler.req(0..125, Output_Clear:=True)	CLEAR

#	Current State	Event /Condition =>Action	Next State
135	SGC-OC	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count > 1 => INDICATE_BUS_ACCESSIBILITY UGC_Count:= UGC_Count-1 CGC_Count:= CGC_Count-1 FSPMM1_Global_Control.cnf(+)(AREP)	SGC-OC
136	SGC-OC	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count = 1 && Go_AClr = False => INDICATE_BUS_ACCESSIBILITY UGC_Count:= UGC_Count-1 CGC_Count:= 0 FSPMM1_Set_Mode DP Master CI1.cnf(+)(AREP, Bus_Accessible)	CLEAR
137	SGC-OC	DMPMM1_Global_Control.cnf(Rem_Add, Status) /CGC_Count = 1 && Go_AClr = True => INDICATE_BUS_ACCESSIBILITY UGC_Count:= UGC_Count-1 CGC_Count:= 0 FSPMM1_DP Master CI1 Mode_Changed.ind(AREP, USIF_State:=Clear) MSCY1M_Cont_Slave_Handler.req(0..125, Output_Clear:=True)	CLEAR
138	COPERATE	Min_SI_Interval_Timer expired /Go_AClr = False => Start Min_SI_Interval_Timer (Bus_Para.Min_Slave_Interval) MSCY1M_Cont_Slave_Handler.req(0..125, Output_Clear:=False)	OPERATE
139	COPERATE	Min_SI_Interval_Timer expired /Go_AClr = True && CGC_Count > 0 => Start Dx_Control_Interval_Timer (Bus_Para.Data_Control_Time/2) Start Min_SI_Interval_Timer (Bus_Para.Min_Slave_Interval)	SGC-OC-WGC
140	COPERATE	Min_SI_Interval_Timer expired /Go_AClr = True && CGC_Count = 0 => Start Dx_Control_Interval_Timer (Bus_Para.Data_Control_Time/2) UGC_Count:= UGC_Count+1 CGC_Count:= UGC_Count Start Min_SI_Interval_Timer (Bus_Para.Min_Slave_Interval) DMPMM1_Global_Control.req(Rem_Add:=127, Control_Command:=2, Group_Select:=0)	SGC-OC
141	COPERATE	DMPMM1_SYNCH.ind /Go_AClr = False => DX_Lock:= TRUE Output_Data = Set_Output MSCY1M_Cont_Slave_Handler.req(0..125, Output_Clear:=False) MSCY1M_Set_Output.req(Rem_Add:=0..125, Slot_Number:=ALL, Output_Data) MSCY1M_Get_Input.req(Rem_Add:=0..125, Slot_Number:=ALL) FSPMM1_SYNCH.ind (AREP)	COPERATE
142	COPERATE	DMPMM1_SYNCH.ind /Go_AClr = True && CGC_Count > 0 => Start Dx_Control_Interval_Timer (Bus_Para.Data_Control_Time/2) DX_Lock:= TRUE Output_Data = Set_Output MSCY1M_Set_Output.req(Rem_Add:=0..125, Slot_Number:=ALL, Output_Data) MSCY1M_Get_Input.req(Rem_Add:=0..125, Slot_Number:=ALL) FSPMM1_SYNCH.ind (AREP)	COPERATE

#	Current State	Event /Condition =>Action	Next State
143	COPERATE	DMPMM1_SYNCH.ind /Go_AClr = True && CGC_Count = 0 => Start Dx_Control_Interval_Timer (Bus_Para.Data_Control_Time/2) UGC_Count:= UGC_Count+1 CGC_Count:= UGC_Count DX_Lock:= TRUE Output_Data = Set_Output DMPMM1_Global_Control.req(Rem_Add:=127, Control_Command:=2, Group_Select:=0) MSCY1M_Set_Output.req(Rem_Add:=0..125, Slot_Number:=ALL, Output_Data) MSCY1M_Get_Input.req(Rem_Add:=0..125, Slot_Number:=ALL) FSPMM1_SYNCH.ind (AREP)	COPERATE
144	LDBP-OP-CPU	DMPMM1_Set_Bus_Par.cnf => FSPMM1_Load_Bus_Para.cnf(+)	OPERATE
145	DELSC-OP	DMPMM1_Delete_SC.cnf => FSPMM1_Delete_SC DP Master CI1.cnf	OPERATE
146	RDVAL-OP	DMPMM1_Read_Value.cnf(Status,Value) => FSPMM1_Read_Value DP Master CI1.cnf(Status, Value)	OPERATE
147	RESET	/spontanous => Act_Rem_Add:= 0 MSCY1M_Reset.req(Act_Rem_Add)	RESET-MS0
148	RESET-MS0	MSCY1M_Reset.cnf(Act_Rem_Add) /Act_Rem_Add<125 => Act_Rem_Add:= Act_Rem_Add+1 MSCY1M_Reset.req(Act_Rem_Add)	RESET-MS0
149	RESET-MS0	MSCY1M_Reset.cnf(Act_Rem_Add) /Act_Rem_Add=125 && IsARExistent(MS1)=TRUE => Act_Rem_Add:= 0 MSAC1M_Reset.req(Act_Rem_Add)	RESET-MS1
150	RESET-MS0	MSCY1M_Reset.cnf(Act_Rem_Add) /Act_Rem_Add=125 && IsARExistent(MS1)=FALSE && && IsARExistent(MM1)=TRUE => MMAC1_Reset.req	RESET-MM1
151	RESET-MS0	MSCY1M_Reset.cnf(Act_Rem_Add) /Act_Rem_Add=125 && IsARExistent(MS1)=FALSE && && IsARExistent(MM1)=FALSE => DMPMM1_Reset.req	RESET-DMPM
152	RESET-MS1	MSAC1M_Reset.cnf(Act_Rem_Add) /Act_Rem_Add=125 => Act_Rem_Add:= 0 MSAL1M_Reset.req(Act_Rem_Add)	RESET-MS1
153	RESET-MS1	MSAL1M_Reset.cnf(Act_Rem_Add) /Act_Rem_Add<125 => Act_Rem_Add:= Act_Rem_Add + 1 MSAL1M_Reset.req(Act_Rem_Add)	RESET-MS1
154	RESET-MS1	MSAL1M_Reset.cnf(Act_Rem_Add) /Act_Rem_Add = 125 && IsARExistent(MM1)=TRUE => MMAC1_Reset.req	RESET-MM1

#	Current State	Event /Condition =>Action	Next State
155	RESET-MS1	MSAL1M_Reset.cnf(Act_Rem_Add) /Act_Rem_Add = 125 && IsARExistent(MM1)=FALSE => DMPMM1_Reset.req	RESET-DMPM
156	RESET-MM1	MMAC1_Reset.cnf => DMPMM1_Reset.req	RESET-DMPM
157	RESET-DMPM	DMPMM1_Reset.cnf /USER_SetMode_Offline = TRUE || USER_ChangedCritical_Para = TRUE	INIT-DMPM

158	RESET-DMPM	DMPMM1_Reset.cnf
		/USER_SetMode_Offline = FALSE && USER_ChangedCritical_Para = FALSE
		=>
		if(USER_sched_Reset = TRUE)
		FSPMM1_Reset DP Master Cl1.cnf
		endif
			POWER-ON

159	t state	FSPMM1_Abort DP Master Cl1.req(AREP) /AREP.AR_Type=MS0 => Current Extended Function Num[CREP]:=No_Service LR_State[CREP]:=W-LR-REQ MSCY1M_Abort.req(Rem_Add:=CREP.Rem_Add)	SAME
160	t state	FSPMM1_Abort DP Master Cl1.req(AREP) /AREP.AR_Type=MS1 => Current Extended Function Num[CREP]:=No_Service LR_State[CREP]:=W-LR-REQ MSAC1M_Abort.req(Rem_Add:=CREP.Rem_Add)	SAME
161	t state	FSPMM1_Abort DP Master Cl1.req(AREP) /AREP.AR_Type=MM1 => MMAC1_Abort.req(Rem_Add:=CREP.Rem_Add)	SAME
162	t state	FSPMM1_Get_Slave_Diag.req(AREP, CREP) /AREP.AR_Type=MS0 => MSCY1M_Get_Slave_Diag.req(CREP.Rem_Add)	SAME
163	t state	MSCY1M_Get_Slave_Diag.cnf(+)(Rem_Add, Diag_Data) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Get_Slave_Diag.cnf(+)(AREP, CREP, Diag_Data)	SAME
164	t state	MSCY1M_Get_Slave_Diag.cnf(-)(Rem_Add) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Get_Slave_Diag.cnf(-)(AREP, CREP)	SAME
165	t state	MSCY1M_Set_Output.cnf(+)(Rem_Add, Slot_Number) /SetContext(Rem_Add, Service)=TRUE && Isochronous Mode = (Not Synchronized || Enhanced Synchronized) => FSPMM1_Set_Output.cnf(+)(AREP, CREP, Slot_Number)	SAME
166	t state	MSCY1M_Set_Output.cnf(-)(Rem_Add, Slot_Number) /SetContext(Rem_Add, Service)=TRUE && Isochronous Mode = (Not Synchronized || Enhanced Synchronized) => Status:= SC FSPMM1_Set_Output.cnf(-)(AREP, CREP, Slot_Number, Status)	SAME
167	t state	MSCY1M_Get_Input.cnf(+)(Rem_Add, Slot_Number, Input_Data) /SetContext(Rem_Add, Service)=TRUE && Isochronous Mode = (Not Synchronized || Enhanced Synchronized) => FSPMM1_Get_Input.cnf(+)(AREP, CREP, Slot_Number, Input_Data)	SAME

#	Current State	Event /Condition =>Action	Next State
168	t state	MSCY1M_Get_Input.cnf(-)(Rem_Add, Slot_Number) /SetContext(Rem_Add, Service)=TRUE && Isochronous Mode = (Not Synchronized || Enhanced Synchronized) => Status:= SC FSPMM1_Get_Input.cnf(-)(AREP, CREP, Slot_Number, Status)	SAME
169	t state	FSPMM1_Read.req(AREP, Slot_Number, Index, Length) /AREP.AR_Type=MS1 => MSAC1M_Read.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length)	SAME
170	t state	MSAC1M_Read.cnf(+)(Rem_Add, Length, Data) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Read.cnf(+)(AREP, Length, Data)	SAME
171	t state	MSAC1M_Read.cnf(-)(Rem_Add, Error_Decode, Error_Code_1, Error_Code_2) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Read.cnf(-)(AREP, Rem_Add, Error_Decode, Error_Code_1, Error_Code_2)	SAME
172	t state	FSPMM1_Write.req(AREP, Slot_Number, Index, Length, Data) /AREP.AR_Type=MS1 => MSAC1M_Write.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length, Data)	SAME
173	t state	MSAC1M_Write.cnf(+)(Rem_Add, Length) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Write.cnf(+)(AREP, Length)	SAME
174	t state	MSAC1M_Write.cnf(-)(Rem_Add, Error_Decode, Error_Code_1, Error_Code_2) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Write.cnf(-)(AREP, Rem_Add, Error_Decode, Error_Code_1, Error_Code_2)	SAME
175	t state	FSPMM1_Alarm_Ack.req(AREP, Slot_Number, Alarm_Type, Seq_Nr) /AREP.AR_Type=MS1 => MSAL1M_Alarm_Ack.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Alarm_Type, Seq_Nr)	SAME
176	t state	MSAL1M_Alarm_Ack.cnf(+)(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Alarm_Ack.cnf(+)(AREP, Slot_Number, Alarm_Type,Seq_Nr)	SAME
177	t state	MSAL1M_Alarm_Ack.cnf(-)(Rem_Add) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Alarm_Ack.cnf(-)(AREP, Slot Number, Alarm Type,Seq Nr)	SAME
178	t state	FSPMM1_Get_Master_Diag.rsp(+)(AREP, Diagnosis_Data) /AREP.AR_Type=MM1 => MMAC1_Get_Master_Diag.rsp(Status:=OK, Diagnosis_Data)	SAME
179	t state	FSPMM1_Get_Master_Diag.rsp(-)(AREP, Status) /AREP.AR_Type=MM1 => MMAC1_Get_Master_Diag.rsp(Status)	SAME
180	t state	MMAC1_Get_Master_Diag.ind(Identifier) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Get_Master_Diag.ind(AREP, Mdiag_Identifier:=Identifier)	SAME
181	t state	FSPMM1_Start_Seq.rsp(+)(AREP, Max_Len_Data_Unit) /AREP.AR_Type=MM1 => MMAC1_Start_Seq.rsp(Status:=OK, Max_Length_Data_Unit)	SAME

#	Current State	Event /Condition =>Action	Next State
182	t state	FSPMM1_Start_Seq.rsp(-)(AREP, Status) /AREP.AR_Type=MM1 => MMAC1_Start_Seq.rsp(Status)	SAME
183	t state	MMAC1_Start_Seq.ind(Req_Add, Area_Code, Timeout) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Start_Seq.ind(AREP, Area_Code, Timeout)	SAME
184	t state	FSPMM1_Download.rsp(+)(AREP) /AREP.AR_Type=MM1 => MMAC1_Download.rsp(Status:=OK)	SAME
185	t state	FSPMM1_Download.rsp(-)(AREP, Status) /AREP.AR_Type=MM1 => MMAC1_Download.rsp(Status)	SAME
186	t state	MMAC1_Download.ind(Req_Add, Area_Code, Address_Offset, Data) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Download.ind(AREP, Area_Code, Add_Offset, Data)	SAME
187	t state	FSPMM1_Upload.rsp(+)(AREP, Data) /AREP.AR_Type=MM1 => MMAC1_Upload.rsp(Status:=OK, Data)	SAME
188	t state	FSPMM1_Upload.rsp(-)(AREP, Status) /AREP.AR_Type=MM1 => MMAC1_Upload.rsp(Status)	SAME
189	t state	MMAC1_Upload.ind(Req_Add, Area_Code, Address_Offset, Data_Length) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Upload.ind(AREP, Area_Code, Add_Offset, Data_Len:=Data_Length)	SAME
190	t state	FSPMM1_End_Seq.rsp(+)(AREP) /AREP.AR_Type=MM1 => MMAC1_End_Seq.rsp(Status:=OK)	SAME
191	t state	FSPMM1_End_Seq.rsp(-)(AREP, Status) /AREP.AR_Type=MM1 => MMAC1_End_Seq.rsp(Status)	SAME
192	t state	MMAC1_End_Seq.ind(Req_Add) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_End_Seq.ind(AREP)	SAME
193	t state	FSPMM1_Act_Param.rsp(+)(AREP) /AREP.AR_Type=MM1 => MMAC1_Act_Param.rsp(Status:=OK)	SAME
194	t state	FSPMM1_Act_Param.rsp(-)(AREP, Status) /AREP.AR_Type=MM1 => MMAC1_Act_Param.rsp(Status)	SAME
195	t state	MMAC1_Act_Param.ind(Area_Code, Activate) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Act_Param.ind(AREP, Area_Code, Activate)	SAME

#	Current State	Event /Condition =>Action	Next State
196	t state	MMAC1_Act_Para_Brct.ind(Area_Code) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Act_Para_Brct.ind(AREP, Area_Code)	SAME
197	t state	MSAC1M_Reject.ind(Rem_Add, Status) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_DP Master Cl1 Reject.ind(AREP, Reason:=Status)	SAME
198	t state	MSAL1M_Started.ind(Rem_Add, Alarm_Limit) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_DP Master Cl1 Started.ind(AREP, Alarm_Limit)	SAME
199	t state	MSAL1M_Stopped.ind(Rem_Add) /SetContext(Rem_Add, Service)=TRUE => Current Extended Function Num[CREP]:=No_Service LR_State[CREP]:=W-LR-REQ FSPMM1_DP Master Cl1 Stopped.ind(AREP)	SAME
200	t state	MSAL1M_Alarm_Notification.ind(Rem_Add, Alarm_Type, Slot_Number, Seq_Nr, Alarm_Specifier, Add_Ack, Alarm_Data) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_Alarm_Notification.ind(AREP, Alarm_Type, Slot_Number, Seq_Nr, Alarm_Specifier, Add_Ack, Alarm_Data)	SAME
201	t state	MSCY1M_New_Slave_Diag.ind(Rem_Add) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_New_Slave_Diag.ind(AREP, CREP)	SAME
202	t state	MSCY1M_New_Input.ind(Rem_Add) /SetContext(Rem_Add, Service)=TRUE => FSPMM1_New_Input.ind(AREP, CREP)	SAME
203	t state	FSPMM1_Set_Output.req(AREP, CREP, Slot_Number, Output_Data, Final) /AREP.AR_Type=MS0 && (DX_Lock || Chg_Buffer) => Status:= SE FSPMM1_Set_Output.cnf(-)(AREP, CREP, Slot_Number, Status)	SAME
204	t state	FSPMM1_Get_Input.req(AREP, CREP, Slot_Number) /AREP.AR_Type=MS0 && (DX_Lock || Chg_Buffer) => Status:= SE FSPMM1_Get_Input.cnf(-)(AREP, CREP, Slot_Number, Status)	SAME
205	t state	FSPMM1_Set_Output.req(AREP, CREP, Slot_Number, Output_Data, Final) /AREP.AR_Type=MS0 && !DX_Lock && !Chg_Buffer && Isochronous Mode <> Buffered Synchronized => MSCY1M_Set_Output.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Output_Data)	SAME
206	t state	FSPMM1_Get_Input.req(AREP, CREP, Slot_Number) /AREP.AR_Type=MS0 && !DX_Lock && !Chg_Buffer && Isochronous Mode <> Buffered Synchronized => MSCY1M_Get_Input.req(Rem_Add:=CREP.Rem_Add, Slot_Number)	SAME
207	t state	FSPMM1_Set_Output.req(AREP, CREP, Slot_Number, Output_Data, Final) /AREP.AR_Type=MS0 && !DX_Lock && !Chg_Buffer && Isochronous Mode = Buffered Synchronized => Bfr_Set_Output:= Output_Data Chg_Buffer:= Final FSPMM1_Set_Output.cnf(+)(AREP, CREP, Slot_Number)	SAME

#	Current State	Event /Condition =>Action	Next State
208	t state	FSPMM1_Get_Input.req(AREP, CREP, Slot_Number) /AREP.AR_Type=MS0 && !DX_Lock && !Chg_Buffer && Isochronous Mode = Buffered Synchronized => Input_Data:= Bfr_Get_Input FSPMM1_Get_Input.cnf(+)(AREP, CREP, Slot_Number, Input_Data)	SAME
209	t state	MSCY1M_Set_Output.cnf(+)(Rem_Add, Slot_Number) /SetContext(Rem_Add, Service)=TRUE && Isochronous Mode = Buffered Synchronized =>	SAME
210	t state	MSCY1M_Set_Output.cnf(-)(Rem_Add, Slot_Number) /SetContext(Rem_Add, Service)=TRUE && Isochronous Mode = Buffered Synchronized => Set_Output = Nil	SAME
211	t state	MSCY1M_Get_Input.cnf(+)(Rem_Add, Slot_Number, Input_Data) /SetContext(Rem_Add, Service)=TRUE && Isochronous Mode = Buffered Synchronized && NOT received all MSCY1M_Get_Input.cnf => Get_Input = Input_Data	SAME
212	t state	MSCY1M_Get_Input.cnf(+)(Rem_Add, Slot_Number, Input_Data) /SetContext(Rem_Add, Service)=TRUE && Isochronous Mode = Buffered Synchronized&& NOT received all MSCY1M_Get_Input.cnf => Chg_Buffer:= FALSE Get_Input = Input_Data	SAME
213	t state	MSCY1M_Get_Input.cnf(-)(Rem_Add, Slot_Number) /SetContext(Rem_Add, Service)=TRUE && Isochronous Mode = Buffered Synchronized => Get_Input = Nil	SAME
214	ANY-STATE	DMPMM1_Event.ind(Event, Add_Info) => FSPMM1_DP Master Cl1 Event.ind(Event, Add_Info)	SAME
215	ANY-STATE	DMPMM1_SYNCH_Delayed.ind (TSH) => FSPMM1_SYNCH_Delayed.ind (AREP, TSH)	SAME
216	ANY-STATE	DMPMM1_Fault.ind => USER_sched_Reset:= False FSPMM1_DP Master Cl1 Fault.ind	RESET
217	ANY-STATE	MSCY1M_Fault.ind => USER_sched_Reset:= False FSPMM1_DP Master Cl1 Fault.ind	RESET
218	ANY-STATE	MSAL1M_Fault.ind => USER_sched_Reset:= False FSPMM1_DP Master Cl1 Fault.ind	RESET
219	ANY-STATE	MSAC1M_Fault.ind => USER_sched_Reset:= False FSPMM1_DP Master Cl1 Fault.ind	RESET
220	ANY-STATE	MMAC1_Fault.ind => USER_sched_Reset:= False FSPMM1_DP Master Cl1 Fault.ind	RESET
221	ANY-STATE	FSPMM1_Reset DP Master Cl1.req => USER_sched_Reset:= True	RESET

#	Current State	Event /Condition =>Action	Next State
222	ANY-STATE	DMPMM1_SYCL_Time_Event.cnf (Send Delay Time, Status) => FSPMM1_SYCL_Time_Event.cnf (AREP, Send Delay Time, Status)	SAME
223	ANY-STATE	FSPMM1_SYCL_Time_Event.req (AREP) => DMPMM1_SYCL_Time_Event.req	SAME
224	ANY-STATE	FSPMM1_SYCL_Clock_Value.req (AREP, Clock Value Time Event, Clock Value previous TE, Clock Value Status) => DMPMM1_SYCL_Clock_Value.req (Clock Value Time Event, Clock Value previous TE, Clock Value Status)	SAME
225	ANY-STATE	DMPMM1_SYCL_Clock_Value.cnf (Status) => FSPMM1_SYCL_Clock_Value.cnf (AREP, Status)	SAME
226	ANY-STATE	DMPMM1_SYCL_Clock_Value.ind (Clock Value Time Event, Clock Value previous TE, Clock Value Status, Receive Delay Time, Src add Clk msg) => FSPMM1_SYCL_Clock_Value.ind (AREP, Clock Value Time Event, Clock Value previous TE, Clock Value Status, Receive Delay Time, Src add Clk msg)	SAME
227	t state	FSPMM1_Initiate_Load.req (AREP, Slot Number, LR Index, Load Type, Load Image Size, Intersegment Request Timeout, User Specific) /LR_State[CREP]==W-LR-REQ &AREP.AR_Type=MS1 => Current Extended Function Num[CREP]:=Initiate_Load Current Slot Number[CREP]:=Slot Number Index:=255 Length:=10+sizeof(User Specific) Data.w.IL.Extended_Function_Num:=Initiate_Load Data.w.IL.LR_Index:=LR Index Data.w.IL.Load_Type:=Load Type Data.w.IL.Load_Image_Size:=Load Image Size Data.w.IL.User_Specific:=User Specific Data.w.IL.Intersegment_Request_Timeout:=Intersegment Request Timeout MSAC1M_Write.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length, Data) LR_State[CREP]:=W-LR-CNF	SAME
228	t state	MSAC1M_Write.cnf(+)(Rem_Add, Length) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Initiate_Load => Slot_Number:=Current Slot Number[CREP] Index:=255 Length:=10 MSAC1M_Read.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length) LR_State[CREP]:=W-LR-CNF	SAME
229	t state	MSAC1M_Read.cnf(+)(Rem_Add, Length, Data) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Initiate_Load => Current Extended Function Num[CREP]=No_Service Actual LR Size:=Data.r.IL.Actual_LR_Size Max Response Delay:=Data.r.IL.Max_Response_Delay Max Segment Length:=Data.r.IL.Max_Segment_Length User Specific:=Data.r.IL.User_Specific FSPMM1_Initiate_Load.cnf(+) (AREP, Actual LR Size, Max Response Delay, Max Segment Length, User Specific) LR_State[CREP]:=W-LR-REQ	SAME

#	Current State	Event /Condition =>Action	Next State
230	t state	MSAC1M_Write.cnf(-)(Rem_Add, Error_Decode, Error_Code_1, Error_Code_2) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Initiate_Load => Current Extended Function Num[CREP]=No_Service Error Code:=Error_Code_1 FSPMM1_Initiate_Load.cnf(-) (AREP, Error Code) LR_State[CREP]:=W-LR-REQ	SAME
231	t state	FSPMM1_Pull_Segment.req (AREP, Slot Number, LR Index, Segment Length) /LR_State[CREP]==W-LR-REQ &AREP.AR_Type=MS1 => Current Extended Function Num[CREP]:=Pull Current LR Index[CREP]:=LR Index Index:=255 Length:=Segment Length MSAC1M_Read.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length) LR_State[CREP]:=W-LR-CNF	SAME
232	t state	MSAC1M_Read.cnf(+)(Rem_Add, Length, Data) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Pull => Current Extended Function Num[CREP]=No_Service Segment Number:=Data.PULL.Sequence Number More Follows:=Data.PULL.Options.More Follows Data:=Data.PULL.Region Data Segment Length:=Length-6 FSPMM1_Pull_Segment.cnf(+) (AREP, Segment Length, Segment Number, More Follows, Data) LR_State[CREP]:=W-LR-REQ	SAME
233	t state	MSAC1M_Read.cnf(-)(Rem_Add, Error_Decode, Error_Code_1, Error_Code_2) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Pull => Current Extended Function Num[CREP]=No_Service Error Code:=Error_Code_1 FSPMM1_Pull_Segment.cnf(-) (AREP, Error Code) LR_State[CREP]:=W-LR-REQ	SAME
234	t state	FSPMM1_Push_Segment.req (AREP, Slot Number, LR Index, Segment Length, Segment Number, More Follows, Data) /LR_State[CREP]==W-LR-REQ &AREP.AR_Type=MS1 => Current Extended Function Num[CREP]:=Push Current Slot Number[CREP]:=Slot Number Index:=255 Length:=6+Segment Length Data.PUSH.Extended_Function_Num:=Push Data.PUSH.LR_Index:=LR Index Data.PUSH.Segment_Number:=Segment Number Data.PUSH.Option.More_Follows:=More Follows Data.PUSH.Data:=Data MSAC1M_Write.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length, Data) LR_State[CREP]:=W-LR-CNF	SAME

#	Current State	Event /Condition =>Action	Next State
235	t state	MSAC1M_Write.cnf(+)(Rem_Add, Length) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Push => Current Extended Function Num[CREP]=No_Service Segment Number:=Data.PUSH.Sequence Number More Follows:=Data.PUSH.Options.More Follows Data:=Data.PUSH.Region Data Segment Length:=Length-6 FSPMM1_Push_Segment.cnf(+) (AREP) LR_State[CREP]:=W-LR-REQ	SAME
236	t state	MSAC1M_Write.cnf(-)(Rem_Add, Error_Decode, Error_Code_1, Error_Code_2) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Push => Current Extended Function Num[CREP]=No_Service Error Code:=Error_Code_1 FSPMM1_Push_Segment.cnf(-) (AREP, Error Code) LR_State[CREP]:=W-LR-REQ	SAME
237	t state	FSPMM1_Terminate_Load.req (AREP, Slot Number, LR Index) /LR_State[CREP]==W-LR-REQ &AREP.AR_Type=MS1 => Current Extended Function Num[CREP]:=Terminate_Load Current Slot Number[CREP]:=Slot Number Index:=255 Length:=6 Data.TL.Extended_Function_Num:=Terminate_Load Data.TL.LR_Index:=LR Index MSAC1M_Write.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length, Data) LR_State[CREP]:=W-LR-CNF	SAME
238	t state	MSAC1M_Write.cnf(+)(Rem_Add, Length) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Terminate_Load => Current Extended Function Num[CREP]=No_Service FSPMM1_Terminate_Load.cnf(+) (AREP) LR_State[CREP]:=W-LR-REQ	SAME
239	t state	MSAC1M_Write.cnf(-)(Rem_Add, Error_Decode, Error_Code_1, Error_Code_2) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Terminate_Load => Current Extended Function Num[CREP]=No_Service Error Code:=Error_Code_1 FSPMM1_Terminate_Load.cnf(-) (AREP, Error Code) LR_State[CREP]:=W-LR-REQ	SAME
240	t state	FSPMM1_Call.req (AREP, Slot Number, Entity Number, FI Index, Execution Argument) /LR_State[CREP]==W-LR-REQ &AREP.AR_Type=MS1 => Current Extended Function Num[CREP]:=Call Current Slot Number[CREP]:=Slot Number Index:=255 Length:=4+sizeof(Execution Argument) Data.w.CALL.Extended_Function_Num:=Call Data.w.CALL. Entity_Number:= Entity_Number Data.w.CALL.FI_Index:=FI Index Data.w.CALL.Execution_Argument:=Execution Argument MSAC1M_Write.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length, Data) LR_State[CREP]:=W-LR-CNF	SAME

#	Current State	Event /Condition =>Action	Next State
241	t state	MSAC1M_Write.cnf(+)(Rem_Add, Length) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Call => Slot_Number:=Current Slot Number[CREP] Index:=255 Length:=CREP.Max_PDU_Length MSAC1M_Read.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length) LR_State[CREP]:=W-LR-CNF	SAME
242	t state	MSAC1M_Read.cnf(+)(Rem_Add, Length, Data) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Call => Current Extended Function Num[CREP]=No_Service Result Argument:=Data.r.CALL.Result_Argument FSPMM1_Call.cnf(+) (AREP, Result Argument) LR_State[CREP]:=W-LR-REQ	SAME
243	t state	MSAC1M_Write.cnf(-)(Rem_Add, Error_Decode, Error_Code_1, Error_Code_2) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Call => Current Extended Function Num[CREP]=No_Service Error Code:=Error_Code_1 FSPMM1_Call.cnf(-) (AREP, Error Code) LR_State[CREP]:=W-LR-REQ	SAME
244	t state	FSPMM1_Start.req (AREP, Slot Number, FI Index, Execution Argument) /LR_State[CREP]==W-LR-REQ &AREP.AR_Type=MS1 => Current Extended Function Num[CREP]:=Start Current Slot Number[CREP]:=Slot Number Index:=255 Length:=4+sizeof(Execution Argument) Data.FIS.Extended_Function_Num:=Start Data.FIS.FI_Index:=FI Index Data.FIS.Execution_Argument:=Execution Argument MSAC1M_Write.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length, Data) LR_State[CREP]:=W-LR-CNF	SAME
245	t state	MSAC1M_Write.cnf(+)(Rem_Add, Length) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Start => Current Extended Function Num[CREP]=No_Service FSPMM1_Start.cnf(+) (AREP) LR_State[CREP]:=W-LR-REQ	SAME
246	t state	MSAC1M_Write.cnf(-)(Rem_Add, Error_Decode, Error_Code_1, Error_Code_2) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Start => Current Extended Function Num[CREP]=No_Service Error Code:=Error_Code_1 FSPMM1_Start.cnf(-) (AREP, Error Code) LR_State[CREP]:=W-LR-REQ	SAME

#	Current State	Event /Condition =>Action	Next State
247	t state	FSPMM1_Stop.req (AREP, Slot Number, FI Index) /LR_State[CREP]==W-LR-REQ &AREP.AR_Type=MS1 => Current Extended Function Num[CREP]:=Stop Current Slot Number[CREP]:=Slot Number Index:=255 Length:=4 Data.FIS.Extended_Function_Num:=Stop Data.FIS.FI_Index:=FI Index MSAC1M_Write.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length, Data) LR_State[CREP]:=W-LR-CNF	SAME
248	t state	MSAC1M_Write.cnf(+)(Rem_Add, Length) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Stop => Current Extended Function Num[CREP]=No_Service FSPMM1_Stop.cnf(+) (AREP) LR_State[CREP]:=W-LR-REQ	SAME
249	t state	MSAC1M_Write.cnf(-)(Rem_Add, Error_Decode, Error_Code_1, Error_Code_2) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Stop => Current Extended Function Num[CREP]=No_Service Error Code:=Error_Code_1 FSPMM1_Stop.cnf(-) (AREP, Error Code) LR_State[CREP]:=W-LR-REQ	SAME
250	t state	FSPMM1_Resume.req (AREP, Slot Number, FI Index, Execution Argument) /LR_State[CREP]==W-LR-REQ &AREP.AR_Type=MS1 => Current Extended Function Num[CREP]:=Resume Current Slot Number[CREP]:=Slot Number Index:=255 Length:=4+sizeof(Execution Argument) Data.FIS.Extended_Function_Num:=Resume Data.FIS.FI_Index:=FI Index Data.FIS.Execution_Argument:=Execution Argument MSAC1M_Write.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length, Data) LR_State[CREP]:=W-LR-CNF	SAME
251	t state	MSAC1M_Write.cnf(+)(Rem_Add, Length) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Resume => Current Extended Function Num[CREP]=No_Service FSPMM1_Resume.cnf(+) (AREP) LR_State[CREP]:=W-LR-REQ	SAME
252	t state	MSAC1M_Write.cnf(-)(Rem_Add, Error_Decode, Error_Code_1, Error_Code_2) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Resume => Current Extended Function Num[CREP]=No_Service Error Code:=Error_Code_1 FSPMM1_Resume.cnf(-) (AREP, Error Code) LR_State[CREP]:=W-LR-REQ	SAME

#	Current State	Event /Condition =>Action	Next State
253	t state	FSPMM1_Reset.req (AREP, Slot Number, FI Index) /LR_State[CREP]==W-LR-REQ &AREP.AR_Type=MS1 => Current Extended Function Num[CREP]:=Reset Current Slot Number[CREP]:=Slot Number Index:=255 Length:=4 Data.FIS.Extended_Function_Num:=Reset Data.FIS.FI_Index:=FI Index MSAC1M_Write.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length, Data) LR_State[CREP]:=W-LR-CNF	SAME
254	t state	MSAC1M_Write.cnf(+)(Rem_Add, Length) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Reset => Current Extended Function Num[CREP]=No_Service FSPMM1_Reset.cnf(+) (AREP) LR_State[CREP]:=W-LR-REQ	SAME
255	t state	MSAC1M_Write.cnf(-)(Rem_Add, Error_Decode, Error_Code_1, Error_Code_2) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Reset => Current Extended Function Num[CREP]=No_Service Error Code:=Error_Code_1 FSPMM1_Reset.cnf(-) (AREP, Error Code) LR_State[CREP]:=W-LR-REQ	SAME
256	t state	FSPMM1_Get_FI_State.req (AREP, Slot Number, FI Index) /LR_State[CREP]==W-LR-REQ &AREP.AR_Type=MS1 => Current Extended Function Num[CREP]:=Get_State Current Slot Number[CREP]:=Slot Number Index:=255 Length:=4 Data.w.STATE.Extended_Function_Num:=Get_State Data.w.STATE.FI_Index:=FI Index MSAC1M_Write.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length, Data) LR_State[CREP]:=W-LR-CNF	SAME
257	t state	MSAC1M_Write.cnf(+)(Rem_Add, Length) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Get_State => Slot_Number:=Current Slot Number[CREP] Index:=255 Length:=5 MSAC1M_Read.req(Rem_Add:=CREP.Rem_Add, Slot_Number, Index, Length) LR_State[CREP]:=W-LR-CNF	SAME
258	t state	MSAC1M_Read.cnf(+)(Rem_Add, Length, Data) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Get_State => Current Extended Function Num[CREP]=No_Service FI State:=Data.r.STATE.State FSPMM1_Get_FI_State.cnf (+) (AREP, FI State) LR_State[CREP]:=W-LR-REQ	SAME

#	Current State	Event /Condition =>Action	Next State
259	t state	MSAC1M_Write.cnf(-)(Rem_Add, Error_Decode, Error_Code_1, Error_Code_2) /LR_State[CREP]==W-LR-CNF &SetContext(Rem_Add, Service)=TRUE && Current Extended Function Num[CREP]=Get_State => Current Extended Function Num[CREP]=No_Service Error Code:=Error_Code_1 FSPMM1_Get_State.cnf(-) (AREP, Error Code) LR_State[CREP]:=W-LR-REQ	SAME

8.2.4 Functions

Table 56 contains the functions used by the FSPMM1, their arguments and their descriptions.

Table 56 – Functions used by the FSPMM1

Function name	Description
SetContext (Rem_Add, Service-Id)	This function checks if there exist an entry for the inputs Rem_Add resp. Res_SAP and Service in the ARL and related CRL of the DP-master (Class 1). A) If there exists an entry then it returns TRUE and sets the local context according to the current CREP and AREP. B) Otherwise it returns FALSE.
IsARexistent (AR)	This function checks if there exist an entry with ARtype=AR.
Chg_Bfr(Buffer1, Buffer 2)	This function exchanges Buffer 1 and Buffer 2.

8.3 FSPMM2

8.3.1 Primitive definitions

8.3.1.1 Primitives exchanged between AP-Context and FSPMM2

Table 57 shows the service primitives including their associated parameters issued by the AP-Context and received by the FSPMM2.

Table 57 – Primitives issued by AP-Context to FSPMM2

Primitive name	Source	Associated parameters	Functions
Init DP master CI2.req	AP-Context	Bus Para	Refer to FAL Service Definition in IEC 61158-5-3 and in IEC 61158-3-3
Reset DP master CI2.req	AP-Context	(none)	Refer to FAL Service Definition in IEC 61158-5-3
Abort.req	AP-Context	AREP Subnet, Instance, Reason Code	
Read Slave Diag.req	AP-Context	AREP	
Read Output.req	AP-Context	AREP	
Read Input.req	AP-Context	AREP	
Get Cfg.req	AP-Context	AREP	

Primitive name	Source	Associated parameters	Functions
Set Slave Add.req	AP-Context	AREP, New Slave Add, Ident Number, No Add Chg, Rem Slave Data	
Initiate.req	AP-Context	AREP, Send Timeout, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param	
Read.req	AP-Context	AREP, Slot Number, Index, Length	
Write.req	AP-Context	AREP, Slot Number, Index, Length, Data	
Data Transport.req	AP-Context	AREP, Slot Number, Index, Length, Data	
Get Master Diag.req	AP-Context	AREP, MDiag Identifier	
Start Seq.req	AP-Context	AREP, Area Code, Timeout	
Download.req	AP-Context	AREP, Area Code, Add Offset, Data	
Upload.req	AP-Context	AREP, Area Code, Add Offset, Data Len	
End Seq.req	AP-Context	AREP	
Act Param.req	AP-Context	AREP, Area Code, Activate	
Act Para Brct.req	AP-Context	AREP, Area Code	
Load ARL DP master Cl2.req	AP-Context	List of ARL Entries	
Get ARL DP master Cl2.req	AP-Context	(none)	
Load CRL DP master Cl2.req	AP-Context	List of CRL Entries	
Get CRL DP master Cl2.req	AP-Context	(none)	
Initiate Load.req	AP-Context	AREP, Slot Number, LR Index, Load Type, Load Image Size, Intersegment Request, Timeout	
Pull Segment.req	AP-Context	AREP, Slot Number, LR Index, Segment Length	

Primitive name	Source	Associated parameters	Functions
Push Segment.req	AP-Context	AREP, Slot Number LR Index, Segment Length Segment Number, More Follows, Data	
Terminate Load.req	AP-Context	AREP Slot Number LR Index	
Start.req	AP-Context	AREP Slot Number FI Index Execution Argument	
Stop.req	AP-Context	AREP Slot Number FI Index	
Resume.req	AP-Context	AREP Slot Number FI Index Execution Argument	
Reset.req	AP-Context	AREP Slot Number FI Index	
Call.req	FSPMS	AREP Slot Number Entity Number FI Index Execution Argument	

Table 58 shows the service primitives including their associated parameters issued by the AP-Context and received by the FSPMS.

Table 58 – Primitives issued by FSPMM2 to AP-Context

Primitive name	Source	Associated parameters	Functions
Init DP master CI2.cnf	FSPMM2	(none)	Refer to FAL Service Definition in IEC 61158-5-3
Reset.cnf	FSPMM2	(none)	
Read Slave Diag.cnf(+)	FSPMM2	AREP, Diag Data	
Read Slave Diag.cnf(-)	FSPMM2	AREP, Status	
Get Cfg.cnf(+)	FSPMM2	AREP, Cfg Data	
Get Cfg.cnf(-)	FSPMM2	AREP, Status	
Read Output.cnf(+)	FSPMM2	AREP, Output Data	
Read Output.cnf(-)	FSPMM2	AREP, Status	
Read Input.cnf(+)	FSPMM2	AREP, Input Data	
Read Input.cnf(-)	FSPMM2	AREP, Status	
Set Slave Add.cnf(+)	FSPMM2	AREP	
Set Slave Add.cnf(-)	FSPMM2	AREP, Status	

Primitive name	Source	Associated parameters	Functions
Initiate.cnf(+)	FSPMM2	AREP, Max Len Data Unit, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param	
Initiate.cnf(-)	FSPMM2	AREP, Error Decode, Error Code 1 Error Code 2	
Read.cnf(+)	FSPMM2	AREP, Length, Data	
Read.cnf(-)	FSPMM2	AREP, Error Decode, Error Code 1 Error Code 2	
Write.cnf(+)	FSPMM2	AREP, Length	
Write.cnf(-)	FSPMM2	AREP, Error Decode, Error Code 1 Error Code 2	
Data Transport.cnf(+)	FSPMM2	AREP, Length, Data	
Data Transport.cnf(-)	FSPMM2	AREP, Error Decode, Error Code 1 Error Code 2	
Get Master Diag.cnf(+)	FSPMM2	AREP, Diagnosis Data	
Get Master Diag.cnf(-)	FSPMM2	AREP, Status	
Start Seq.cnf(+)	FSPMM2	AREP, Max Len Data Unit	
Start Seq.cnf(-)	FSPMM2	AREP, Status	
Download.cnf(+)	FSPMM2	AREP	
Download.cnf(-)	FSPMM2	AREP, Status	
Upload.cnf(+)	FSPMM2	AREP, Data	
Upload.cnf(-)	FSPMM2	AREP, Status	
End Seq.cnf(+)	FSPMM2	AREP	
End Seq.cnf(-)	FSPMM2	AREP, Status	
Act Param.cnf(+)	FSPMM2	AREP	
Act Param.cnf(-)	FSPMM2	AREP, Status	
Act Para Brct.cnf(+)	FSPMM2	AREP	
Act Para Brct.cnf(-)	FSPMM2	AREP, Status	
Event.ind	FSPMM2	Event, Add Info	

Primitive name	Source	Associated parameters	Functions
DP master CI2 Reject.ind	FSPMM2	AREP, Reason	
Abort.ind	FSPMM2	AREP, Locally Generated, Subnet, Instance, Reason_Code, Additional_Detail	
DP master CI2 Fault.ind	FSPMM2	(none)	
Load ARL DP master CI2.cnf(+)	FSPMM2	(none)	
Load ARL DP master CI2.cnf(-)	FSPMM2	Status	
Get ARL DP master CI2.cnf(+)	FSPMM2	List of ARL Entries	
Get ARL DP master CI2.cnf(-)	FSPMM2	Status	
Load CRL DP master CI2.cnf(+)	FSPMM2	(none)	
Load CRL DP master CI2.cnf(-)	FSPMM2	Status	
Get CRL DP master CI2.cnf(+)	FSPMM2	List of CRL Entries	
Get CRL DP master CI2.cnf(-)	FSPMM2	Status	
Initiate Load.cnf(+)	FSPMM2	AREP, Actual LR Size, Max Response Delay, Max Segment Length	
Initiate Load.cnf(-)	FSPMM2	AREP, Error Code	
Pull Segment.cnf(+)	FSPMM2	AREP, Segment Length, Segment Number, More Follows, Data	
Pull Segment.cnf(-)	FSPMM2	AREP, Error Code	
Push Segment.cnf(+)	FSPMM2	AREP	
Push Segment.cnf(-)	FSPMM2	AREP, Error Code	
Terminate Load.cnf(+)	FSPMM2	AREP	
Terminate Load.cnf(-)	FSPMM2	AREP, Error Code	
Start.cnf(+)	FSPMM2	AREP	
Start.cnf(-)	FSPMM2	AREP Error Code Function Invocation State	
Stop.cnf(+)	FSPMM2	AREP	
Stop.cnf(-)	FSPMM2	AREP Error Code Function Invocation State	
Resume.cnf(+)	FSPMM2	AREP	
Resume.cnf(-)	FSPMM2	AREP Error Code Function Invocation State	
Reset.cnf(+)	FSPMM2	AREP	
Reset.cnf(-)	FSPMM2	AREP Error Code	
Call.cnf(+)	FSPMM2	AREP Result Argument	

Primitive name	Source	Associated parameters	Functions
Call.cnf(-)	FSPMM2	AREP Result Argument Error Code	

8.3.1.2 Parameters of FSPMM2 primitives

The parameters used with the primitives exchanged between the FSPMM2 and the AP-Context are described in FAL Service Definition in IEC 61158-5-3.

8.3.2 State machine description

This Machine is used to co-ordinate the Interactions between AP-Context and DMPMM1, MSCY1M, MSAC1M, MSAL1M and MMAC1. As the support of several State Machines for the communication with individual slave as well as timing constraints of the operation of slave require a consistent scheduling, this task is accomplished by FSPMM1 as well.

Local Variables

LR_State
(Array of Unsigned8)

This variable stores the state of the Load Region sequence of the corresponding AREP.

CurrentExtendedFunctionNum
(Array of Unsigned 8)

This variable stores the extended function number of the currently processed LR service of the corresponding AREP.

CurrentSlotNumber
(Array of Unsigned8)

This variable stores the slot number of the currently processed LR service of the corresponding AREP.

8.3.3 FSPMM2 state table

Table 59 contains the complete description of the FSPMM2 state machine.

Table 59 – FSPMM2 state table

#	Current state	Event / condition => action	Next state
1	POWER-ON	Load ARL DP master CI2.req (List of ARL Entries) /valid parameters => Load ARL DP master CI2.cnf(+)	POWER-ON
2	POWER-ON	Load ARL DP master CI2.req (List of ARL Entries) /invalid parameters => Status:= NO Load ARL DP master CI2.cnf(-) (Status)	POWER-ON
3	POWER-ON	Get ARL DP master CI2.req /ARL loaded => Get ARL DP master CI2.cnf(+) (List of ARL Entries)	POWER-ON

#	Current state	Event / condition => action	Next state
4	POWER-ON	Get ARL DP master CI2.req /ARL not loaded => Status:= NO Get ARL DP master CI2.cnf(-) (Status)	POWER-ON
5	POWER-ON	Load CRL DP master CI2.req (List of CRL Entries) /valid parameters => Load CRL DP master CI2.cnf(+)	POWER-ON
6	POWER-ON	Load CRL DP master CI2.req (List of CRL Entries) /invalid parameters => Status:= NO Load CRL DP master CI2.cnf(-) (Status)	POWER-ON
7	POWER-ON	Get CRL DP master CI2.req /CRL loaded => Get CRL DP master CI2.cnf(+) (List of CRL Entries)	POWER-ON
8	POWER-ON	Get CRL DP master CI2.req /CRL not loaded => Status:= NO Get CRL DP master CI2.cnf(-) (Status)	POWER-ON
9	PON	FSPMM2_Init DP master CI2 .req(Bus_Para) => StoreNumberIssuedMInit() Current Extended Function Num[AREP]=No_Service LR_State[AREP]:=W-LR-REQ DMPMM2_Minit_DLL.req(Bus_Para) MMAC_MInit_MM.req MSAC2M_MInit_MS2.req(First_MS2_AREP ..Last_MS2_AREP)	INIT-DLM
10	INIT-DLM	DMPMM2_Minit_DLL.cnf => ignore	INIT-DLM
11	INIT-DLM	MMAC2_MInit_MM.cnf => ignore	INIT-DLM
12	INIT-DLM	MSAC2M_MInit_MS2.cnf (CREP) /IsLastWaitingMInit()=FALSE => ignore	INIT-DLM
13	INIT-DLM	MSAC2M_MInit_MS2.cnf (CREP) /IsLastWaitingMInit()=TRUE => FSPMM2_Init DP master CI2 .cnf	RUN
14	RUN	Any FSPMM2.req /wrong AREP.AR_Type => ignore	RUN
15	RUN	Any MMAC1.cnf /SetContext(Rem_add, Service) = FALSE => ignore	RUN
16	RUN	Any MSAC2M.req /SetContext(CREP, Service) = FALSE => ignore	RUN
17	RUN	Any DMPMM2.req /SetContext(Rem_add, Service) = FALSE => ignore	RUN

#	Current state	Event / condition => action	Next state
18	RUN	FSPMM2_Abort.req(AREP, Subnet, Instance, Reason_Code) /AREP.AR_Type = MS0 => DMPMM2_Abort.req	RUN
19	RUN	FSPMM2_Abort.req(AREP, Subnet, Instance, Reason_Code) /AREP.AR_Type = MM1 => MMAC2_Abort.req	RUN
20	RUN	FSPMM2_Abort.req(AREP, Subnet, Instance, Reason_Code) /AREP.AR_Type = MS2 => Current Extended Function Num[AREP]=No_Service LR_State[AREP]:=W-LR-REQ MSAC2M_Abort.req(CREP, Subnet, Instance, Reason_Code)	RUN
21	RUN	MSAC2M_Abort.ind(CREP, Locally_Generated, Subnet, Instance, Reason_Code, Abort_Detail) /SetContext(CREP, Service) = TRUE => Current Extended Function Num[AREP]=No_Service LR_State[AREP]:=W-LR-REQ FSPMM2_Abort.ind(AREP, Locally Generated, Subnet, Instance, Reason Code, Additional Detail)	RUN
22	RUN	DMPMM2_Event.ind(Event, Add_info) => FSPMM2_Event.ind(Event=Event/Fault, Add_Info=Add_info)	RUN
23	RUN	FSPMM2_Read_Slave_Diag.req(AREP) /AREP.AR_Type = MS0 => DMPMM2_Read_Slave_Diag.req(CREP.Rem_Add)	RUN
24	RUN	DMPMM2_Read_Slave_Diag.cnf(+)(Rem_Add, Diag_Data) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Read_Slave_Diag.cnf(+)(AREP, Diag_Data)	RUN
25	RUN	DMPMM2_Read_Slave_Diag.cnf(-)(Rem_Add, Status) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Read_Slave_Diag.cnf(-)(AREP, Status=Status)	RUN
26	RUN	FSPMM2_Get_Cfg.req(AREP) /AREP.AR_Type = MS0 => DMPMM2_Get_Cfg.req(CREP.Rem_Add)	RUN
27	RUN	DMPMM2_Get_Cfg.cnf(+)(Rem_Add, Cfg_Data) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Get_Cfg.cnf(+)(AREP, Cfg_Data)	RUN
28	RUN	DMPMM2_Get_Cfg.cnf(-)(Rem_Add, Status) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Get_Cfg.cnf(-)(AREP, Status)	RUN
29	RUN	FSPMM2_Read_Input.req(AREP) /AREP.AR_Type = MS0 => DMPMM2_Read_Input.req(CREP.Rem_Add)	RUN
30	RUN	DMPMM2_Read_Input.cnf(+)(Rem_Add, Inp_Data) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Read_Input.cnf(+)(AREP, Inp_Data)	RUN
31	RUN	DMPMM2_Read_Input.cnf(-)(Rem_Add, Status) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Read_Input.cnf(-)(AREP, Status)	RUN

#	Current state	Event / condition => action	Next state
32	RUN	FSPMM2_Read_Output.req(AREP) /AREP.AR_Type = MS0 => DMPMM2_Read_Output.req(CREP.Rem_Add)	RUN
33	RUN	DMPMM2_Read_Output.cnf(+)(Rem_Add, Outp_Data) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Read_Output.cnf(+)(AREP, Outp_Data)	RUN
34	RUN	DMPMM2_Read_Output.cnf(-)(Rem_Add, Status) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Read_Output.cnf(-)(AREP, Status)	RUN
35	RUN	FSPMM2_Set_Slave_Add.req(AREP, New_Slave_Add, Ident_Number, No_Add_Chg, Rem_Slave_Data) /AREP.AR_Type = MS0 => DMPMM2_Set_Slave_Add.req(CREP.Rem_Add, New_Slave_Add, Ident_Number, No_Add_Chg, Rem_Slave_Data)	RUN
36	RUN	DMPMM2_Set_Slave_Add.cnf(+)(Rem_Add) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Set_Slave_Add.cnf(+)(AREP)	RUN
37	RUN	DMPMM2_Set_Slave_Add.cnf(-)(Rem_Add, Status) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Set_Slave_Add.cnf(-)(AREP, Status=Status)	RUN
38	RUN	Any DMPMM2.req /SetContext(Rem_add, Service) = FALSE => ignore	RUN
39	RUN	FSPMM2_Initiate.req(AREP, Send Timeout, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param) /AREP.AR_Type = MS2 => MSAC2M_Initiate.req(CREP, Send Timeout, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param)	RUN
40	RUN	MSAC2M_Initiate.cnf(+)(CREP, Max Len Data Unit, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param) /SetContext(CREP, Service) = TRUE => FSPMM2_Initiate.cnf(+)(AREP, Max Len Data Unit, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param)	RUN
41	RUN	MSAC2M_Initiate.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /SetContext(CREP, Service) = TRUE => FSPMM2_Initiate.cnf(-)(AREP, Error Decode, Error Code 1, Error Code 2)	RUN
42	RUN	FSPMM2_Read.req(AREP, Slot Number, Index, Length) /AREP.AR_Type = MS2 => MSAC2M_Read.req(CREP, Slot Number, Index, Length)	RUN
43	RUN	MSAC2M_Read.cnf(+)(CREP, Length, Data) /SetContext(CREP, Service) = TRUE => FSPMM2_Read.cnf(+)(AREP, Length, Data)	RUN
44	RUN	MSAC2M_Read.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /SetContext(CREP, Service) = TRUE => FSPMM2_Read.cnf(-)(AREP, Error Decode, Error Code 1, Error Code 2)	RUN

#	Current state	Event / condition => action	Next state
45	RUN	FSPMM2_Write.req(AREP, Slot Number, Index, Length, Data) /AREP.AR_Type = MS2 => MSAC2M_Write.req(CREP, Slot Number, Index, Length, Data)	RUN
46	RUN	MSAC2M_Write.cnf(+)(CREP, Length) /SetContext(CREP, Service) = TRUE => FSPMM2_Write.cnf(+)(AREP, Length)	RUN
47	RUN	MSAC2M_Write.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /SetContext(CREP, Service) = TRUE => FSPMM2_Write.cnf(-)(AREP, Error Decode, Error Code 1, Error Code 2)	RUN
48	RUN	FSPMM2_Data_Transport.req(AREP, Slot Number, Index, Length, Data) /AREP.AR_Type = MS2 => MSAC2M_Data_Transport.req(CREP, Slot Number, Index, Length, Data)	RUN
49	RUN	MSAC2M_Data_Transport.cnf(+)(CREP, Length, Data) /SetContext(CREP, Service) = TRUE => FSPMM2_Data_Transport.cnf(+)(AREP, Length, Data)	RUN
50	RUN	MSAC2M_Data_Transport.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /SetContext(CREP, Service) = TRUE => FSPMM2_Data_Transport.cnf(-)(AREP, Error Decode, Error Code 1, Error Code 2)	RUN
51	RUN	MSAC2M_Closed.ind(CREP) /SetContext(CREP, Service) = TRUE => FSPMM2 DP master Cl2 _Closed.ind(AREP)	RUN
52	RUN	MMAC2_Reject.ind(Rem_Add, Reason_Code) /SetContext(Rem_add, Service) = TRUE => FSPMM2 DP master Cl2 _Reject.ind(AREP, Reason)	RUN
53	RUN	FSPMM2_Get_Master_Diag.req(AREP, MDiag Identifier) /AREP.AR_Type = MM1 => MMAC2_Get_Master_Diag.req(AREP, MDiag Identifier)	RUN
54	RUN	MMAC2_Get_Master_Diag.cnf(+)(Rem_Add, Diagnosis Data) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Get_Master_Diag.cnf(+)(AREP, Diagnosis Data)	RUN
55	RUN	MMAC2_Get_Master_Diag.cnf(-)(Rem_Add, Status) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Get_Master_Diag.cnf(-)(AREP, Status)	RUN
56	RUN	FSPMM2_Start_Seq.req(AREP, Area Code, Timeout) /AREP.AR_Type = MM1 => MMAC2_Start_Seq.req(AREP, Area Code, Timeout)	RUN
57	RUN	MMAC2_Start_Seq.cnf(+)(Rem_Add, Max Len Data Unit) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Start_Seq.cnf(+)(AREP, Max Len Data Unit)	RUN
58	RUN	MMAC2_Start_Seq.cnf(-)(Rem_Add, Status) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Start_Seq.cnf(-)(AREP, Status)	RUN
59	RUN	FSPMM2_End_Seq.req(AREP) /AREP.AR_Type = MM1 => MMAC2_End_Seq.req(AREP)	RUN

#	Current state	Event / condition => action	Next state
60	RUN	MMAC2_End_Seq.cnf(+)(Rem_Add) /SetContext(Rem_add, Service) = TRUE => FSPMM2_End_Seq.cnf(+)(AREP)	RUN
61	RUN	MMAC2_End_Seq.cnf(-)(Rem_Add, Status) /SetContext(Rem_add, Service) = TRUE => FSPMM2_End_Seq.cnf(-)(AREP, Status)	RUN
62	RUN	FSPMM2_Download.req(AREP, Area Code, Add Offset, Data) /AREP.AR_Type = MM1 => MMAC2_Download.req(AREP, Area Code, Add Offset, Data)	RUN
63	RUN	MMAC2_Download.cnf(+)(Rem_Add) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Download.cnf(+)(AREP)	RUN
64	RUN	MMAC2_Download.cnf(-)(Rem_Add, Status) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Download.cnf(-)(AREP, Status)	RUN
65	RUN	FSPMM2_Upload.req(AREP, Area Code, Add Offset, Data Len) /AREP.AR_Type = MM1 => MMAC2_Upload.req(AREP, Area Code, Add Offset, Data Len)	RUN
66	RUN	MMAC2_Upload.cnf(+)(Rem_Add, Data) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Upload.cnf(+)(AREP, Data)	RUN
67	RUN	MMAC2_Upload.cnf(-)(Rem_Add, Status) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Upload.cnf(-)(AREP, Status)	RUN
68	RUN	FSPMM2_Act_Param.req(AREP, Area Code, Activate) /AREP.AR_Type = MM1 => MMAC2_Act_Param.req(AREP, Area Code, Activate)	RUN
69	RUN	MMAC2_Act_Param.cnf(+)(Rem_Add) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Act_Param.cnf(+)(AREP)	RUN
70	RUN	MMAC2_Act_Param.cnf(-)(Rem_Add, Status) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Act_Param.cnf(-)(AREP, Status)	RUN
71	RUN	FSPMM2_Act_Param_Brct.req(AREP, Area Code) /AREP.AR_Type = MM2 => MMAC2_Act_Param_Brct.req(AREP, Area Code, Activate)	RUN
72	RUN	MMAC2_Act_Param_Brct.cnf(+)(Rem_Add) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Act_Param_Brct.cnf(+)(AREP)	RUN
73	RUN	MMAC2_Act_Param_Brct.cnf(-)(Rem_Add, Status) /SetContext(Rem_add, Service) = TRUE => FSPMM2_Act_Param_Brct.cnf(-)(AREP, Status)	RUN
74	any state	DMPMM2_SYCL_Time_Event.cnf (Send Delay Time, Status) => FSPMM2_SYCL_Time_Event.cnf (AREP, Send Delay Time, Status)	SAME
75	any state	FSPMM2_SYCL_Time_Event.req (AREP) => DMPMM2_SYCL_Time_Event.req	SAME

#	Current state	Event / condition => action	Next state
76	any state	FSPMM2_SYCL_Clock_Value.req (AREP, Clock Value Time Event, Clock Value previous TE, Clock Value Status) => DMPMM2_SYCL_Clock_Value.req (Clock Value Time Event, Clock Value previous TE, Clock Value Status)	SAME
77	any state	DMPMM2_SYCL_Clock_Value.cnf (Status) => FSPMM2_SYCL_Clock_Value.cnf (AREP, Status)	SAME
78	any state	DMPMM2_SYCL_Clock_Value.ind (Clock Value Time Event, Clock Value previous TE, Clock Value Status, Receive Delay Time, Src add Clk msg) => FSPMM2_SYCL_Clock_Value.ind (AREP, Clock Value Time Event, Clock Value previous TE, Clock Value Status, Receive Delay Time, Src add Clk msg)	SAME
79	any state	FSPMM2_Reset DP master Cl2 .req => StoreNumberIssuedResets() FSPMM2_UserReset = TRUE DMPMM2_Reset.req MMAC2_Reset.req MSAC2M_Reset.req(First_MS2_AREP ..Last_MS2_AREP)	WAIT-RESET
80	any state	DMPMM2_Fault.ind => StoreNumberIssuedResets() FSPMM2_UserReset = FALSE DMPMM2_Reset.req MMAC2_Reset.req MSAC2M_Reset.req(First_MS2_AREP ..Last_MS2_AREP) FSPMM2 DP master Cl2 Fault.ind	WAIT-RESET
81	any state	MMAC2_Fault.ind => StoreNumberIssuedResets() FSPMM2_UserReset = FALSE DMPMM2_Reset.req MMAC2_Reset.req MSAC2M_Reset.req(First_MS2_AREP ..Last_MS2_AREP) FSPMM2 DP master Cl2 Fault.ind	WAIT-RESET
82	WAIT-RESET	DMPMM2_Reset.cnf => no action	WAIT-RESET
83	WAIT-RESET	MMAC2_Reset.cnf => no action	WAIT-RESET
84	WAIT-RESET	MSAC2M_Reset.cnf (CREP) /IsLastWaitingReset()=FALSE => no action	WAIT-RESET
85	WAIT-RESET	MSAC2M_Reset.cnf (CREP) /IsLastWaitingReset()=TRUE AND FSPMM2_UserReset = TRUE => FSPMM2_Reset DP master Cl2 .cnf	PON
86	WAIT-RESET	MSAC2M_Reset.cnf (CREP) /IsLastWaitingReset()=TRUE AND FSPMM2_UserReset = FALSE => no action	PON

#	Current state	Event / condition => action	Next state
87	RUN	FSPMM2_Initiate_Load.req (AREP, Slot Number, LR Index, Load Type, Load Image Size, Intersegment Request Timeout, User Specific) /LR_State[AREP]==W-LR-REQ &AREP.AR_Type=MS2 => Current Extended Function Num[AREP]:=Initiate_Load Current Slot Number[AREP]:=Slot Number Index:=255 Length:=10+sizeof(User Specific) Data.w.IL.Extended_Function_Num:=Initiate_Load Data.w.IL.LR_Index:=LR Index Data.w.IL.Load_Type:=Load Type Data.w.IL.Load_Image_Size:=Load Image Size Data.w.IL.User_Specific:=User Specific Data.w.IL.Intersegment_Request_Timeout:=Intersegment Request Timeout MSAC2M_Write.req(CREP, Slot Number, Index, Length, Data) LR_State[AREP]:=W-LR-CNF	RUN
88	RUN	MSAC2M_Write.cnf(+)(CREP, Length) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Initiate_Load => Slot_Number:=Current Slot Number[AREP] Index:=255 Length:=10 MSAC2M_Read.req(CREP, Slot Number, Index, Length) LR_State[AREP]:=W-LR-CNF	RUN
89	RUN	MSAC2M_Read.cnf(+)(CREP, Length, Data) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Initiate_Load => Current Extended Function Num[AREP]=No_Service Actual LR Size:=Data.r.IL.Actual_LR_Size Max Response Delay:=Data.r.IL.Max_Response_Delay Max Segment Length:=Data.r.IL.Max_Segment_Length User Specific:=Data.r.IL.User_Specific FSPMM2_Initiate_Load.cnf(+) (AREP, Actual LR Size, Max Response Delay, Max Segment Length, User Specific) LR_State[AREP]:=W-LR-REQ	RUN
90	RUN	MSAC2M_Write.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Initiate_Load => Current Extended Function Num[AREP]=No_Service Error Code:=Error_Code_1 FSPMM2_Initiate_Load.cnf(-) (AREP, Error Code) LR_State[AREP]:=W-LR-REQ	RUN
91	RUN	FSPMM2_Pull_Segment.req (AREP, Slot Number, LR Index, Segment Length) /LR_State[AREP]==W-LR-REQ &AREP.AR_Type=MS2 => Current Extended Function Num[AREP]:=Pull Current LR Index[AREP]:=LR Index Index:=255 Length:=Segment Length MSAC2M_Read.req(CREP, Slot Number, Index, Length) LR_State[AREP]:=W-LR-CNF	RUN

#	Current state	Event / condition => action	Next state
92	RUN	MSAC2M_Read.cnf(+)(CREP, Length, Data) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Pull => Current Extended Function Num[AREP]=No_Service Segment Number:=Data.PULL.Sequence Number More Follows:=Data.PULL.Options.More Follows Data:=Data.PULL.Region Data Segment Length:=Length-6 FSPMM2_Pull_Segment.cnf(+) (AREP, Segment Length, Segment Number, More Follows, Data) LR_State[AREP]:=W-LR-REQ	RUN
93	RUN	MSAC2M_Read.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /LR_State[AREP]==W-LR-CNF &SetContext(Service, Service)=TRUE && Current Extended Function Num[AREP]=Pull => Current Extended Function Num[AREP]=No_Service Error Code:=Error_Code_1 FSPMM2_Pull_Segment.cnf(-) (AREP, Error Code) LR_State[AREP]:=W-LR-REQ	RUN
94	RUN	FSPMM2_Push_Segment.req (AREP, Slot Number, LR Index, Segment Length, Segment Number, More Follows, Data) /LR_State[AREP]==W-LR-REQ &AREP.AR_Type=MS2 => Current Extended Function Num[AREP]:=Push Current Slot Number[AREP]:=Slot Number Index:=255 Length:=6+Segment Length Data.PUSH.Extended_Function_Num:=Push Data.PUSH.LR_Index:=LR Index Data.PUSH.Segment_Number:=Segment Number Data.PUSH.Option.More_Follows:=More Follows Data.PUSH.Data:=Data MSAC2M_Write.req(CREP, Slot Number, Index, Length, Data) LR_State[AREP]:=W-LR-CNF	RUN
95	RUN	MSAC2M_Write.cnf(+)(CREP, Length) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Push => Current Extended Function Num[AREP]=No_Service Segment Number:=Data.PUSH.Sequence Number More Follows:=Data.PUSH.Options.More Follows Data:=Data.PUSH.Region Data Segment Length:=Length-6 FSPMM2_Push_Segment.cnf(+) (AREP) LR_State[AREP]:=W-LR-REQ	RUN
96	RUN	MSAC2M_Write.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Push => Current Extended Function Num[AREP]=No_Service Error Code:=Error_Code_1 FSPMM2_Push_Segment.cnf(-) (AREP, Error Code) LR_State[AREP]:=W-LR-REQ	RUN

#	Current state	Event / condition => action	Next state
97	RUN	FSPMM2_Terminate_Load.req (AREP, Slot Number, LR Index) /LR_State[AREP]==W-LR-REQ &AREP.AR_Type=MS2 => Current Extended Function Num[AREP]:=Terminate_Load Current Slot Number[AREP]:=Slot Number Index:=255 Length:=6 Data.TL.Extended_Function_Num:=Terminate_Load Data.TL.LR_Index:=LR Index MSAC2M_Write.req(CREP, Slot Number, Index, Length, Data) LR_State[AREP]:=W-LR-CNF	RUN
98	RUN	MSAC2M_Write.cnf(+)(CREP, Length) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Terminate_Load => Current Extended Function Num[AREP]=No_Service FSPMM2_Terminate_Load.cnf(+) (AREP) LR_State[AREP]:=W-LR-REQ	RUN
99	RUN	MSAC2M_Write.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Terminate_Load => Current Extended Function Num[AREP]=No_Service Error Code:=Error_Code_1 FSPMM2_Terminate_Load.cnf(-) (AREP, Error Code) LR_State[AREP]:=W-LR-REQ	RUN
100	RUN	FSPMM2_Call.req (AREP, Slot Number, Entity Number, FI Index, Execution Argument) /LR_State[AREP]==W-LR-REQ &AREP.AR_Type=MS2 => Current Extended Function Num[AREP]:=Call Current Slot Number[AREP]:=Slot Number Index:=255 Length:=4+sizeof(Execution Argument) Data.w.CALL.Extended_Function_Num:=Call Data.w.CALL. Entity_Number:= Entity_Number Data.w.CALL.FI_Index:=FI Index Data.w.CALL.Execution_Argument:=Execution Argument MSAC2M_Write.req(CREP, Slot Number, Index, Length, Data) LR_State[AREP]:=W-LR-CNF	RUN
101	RUN	MSAC2M_Write.cnf(+)(CREP, Length) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Call => Slot_Number:=Current Slot Number[AREP] Index:=255 Length:=AREP.Max_PDU_Length MSAC2M_Read.req(CREP, Slot Number, Index, Length) LR_State[AREP]:=W-LR-CNF	RUN
102	RUN	MSAC2M_Read.cnf(+)(CREP, Length, Data) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Call => Current Extended Function Num[AREP]=No_Service Result Argument:=Data.r.CALL.Result_Argument FSPMM2_Call.cnf(+) (AREP, Result Argument) LR_State[AREP]:=W-LR-REQ	RUN

#	Current state	Event / condition => action	Next state
103	RUN	MSAC2M_Write.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Call => Current Extended Function Num[AREP]=No_Service Error Code:=Error_Code_1 FSPMM2_Call.cnf(-) (AREP, Error Code) LR_State[AREP]:=W-LR-REQ	RUN
104	RUN	FSPMM2_Start.req (AREP, Slot Number, FI Index, Execution Argument) /LR_State[AREP]==W-LR-REQ &AREP.AR_Type=MS2 => Current Extended Function Num[AREP]:=Start Current Slot Number[AREP]:=Slot Number Index:=255 Length:=4+sizeof(Execution Argument) Data.FIS.Extended_Function_Num:=Start Data.FIS.FI_Index:=FI Index Data.FIS.Execution_Argument:=Execution Argument MSAC2M_Write.req(CREP, Slot Number, Index, Length, Data) LR_State[AREP]:=W-LR-CNF	RUN
105	RUN	MSAC2M_Write.cnf(+)(CREP, Length) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Start => Current Extended Function Num[AREP]=No_Service FSPMM2_Start.cnf(+) (AREP) LR_State[AREP]:=W-LR-REQ	RUN
106	RUN	MSAC2M_Write.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Start => Current Extended Function Num[AREP]=No_Service Error Code:=Error_Code_1 FSPMM2_Start.cnf(-) (AREP, Error Code) LR_State[AREP]:=W-LR-REQ	RUN
107	RUN	FSPMM2_Stop.req (AREP, Slot Number, FI Index) /LR_State[AREP]==W-LR-REQ &AREP.AR_Type=MS2 => Current Extended Function Num[AREP]:=Stop Current Slot Number[AREP]:=Slot Number Index:=255 Length:=4 Data.FIS.Extended_Function_Num:=Stop Data.FIS.FI_Index:=FI Index MSAC2M_Write.req(CREP, Slot Number, Index, Length, Data) LR_State[AREP]:=W-LR-CNF	RUN
108	RUN	MSAC2M_Write.cnf(+)(CREP, Length) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Stop => Current Extended Function Num[AREP]=No_Service FSPMM2_Stop.cnf(+) (AREP) LR_State[AREP]:=W-LR-REQ	RUN
109	RUN	MSAC2M_Write.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Stop => Current Extended Function Num[AREP]=No_Service Error Code:=Error_Code_1 FSPMM2_Stop.cnf(-) (AREP, Error Code) LR_State[AREP]:=W-LR-REQ	RUN

#	Current state	Event / condition => action	Next state
110	RUN	FSPMM2_Resume.req (AREP, Slot Number, FI Index, Execution Argument) /LR_State[AREP]==W-LR-REQ &AREP.AR_Type=MS2 => Current Extended Function Num[AREP]:=Resume Current Slot Number[AREP]:=Slot Number Index:=255 Length:=4+sizeof(Execution Argument) Data.FIS.Extended_Function_Num:=Resume Data.FIS.FI_Index:=FI Index Data.FIS.Execution_Argument:=Execution Argument MSAC2M_Write.req(CREP, Slot Number, Index, Length, Data) LR_State[AREP]:=W-LR-CNF	RUN
111	RUN	MSAC2M_Write.cnf(+)(CREP, Length) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Resume => Current Extended Function Num[AREP]=No_Service FSPMM2_Resume.cnf(+) (AREP) LR_State[AREP]:=W-LR-REQ	RUN
112	RUN	MSAC2M_Write.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Resume => Current Extended Function Num[AREP]=No_Service Error Code:=Error_Code_1 FSPMM2_Resume.cnf(-) (AREP, Error Code) LR_State[AREP]:=W-LR-REQ	RUN
113	RUN	FSPMM2_Reset.req (AREP, Slot Number, FI Index) /LR_State[AREP]==W-LR-REQ &AREP.AR_Type=MS2 => Current Extended Function Num[AREP]:=Reset Current Slot Number[AREP]:=Slot Number Index:=255 Length:=4 Data.FIS.Extended_Function_Num:=Reset Data.FIS.FI_Index:=FI Index MSAC2M_Write.req(CREP, Slot Number, Index, Length, Data) LR_State[AREP]:=W-LR-CNF	RUN
114	RUN	MSAC2M_Write.cnf(+)(CREP, Length) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Reset => Current Extended Function Num[AREP]=No_Service FSPMM2_Reset.cnf(+) (AREP) LR_State[AREP]:=W-LR-REQ	RUN
115	RUN	MSAC2M_Write.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Reset => Current Extended Function Num[AREP]=No_Service Error Code:=Error_Code_1 FSPMM2_Reset.cnf(-) (AREP, Error Code) LR_State[AREP]:=W-LR-REQ	RUN

#	Current state	Event / condition => action	Next state
116	RUN	FSPMM2_Get_FI_State.req (AREP, Slot Number, FI Index) /LR_State[AREP]==W-LR-REQ &AREP.AR_Type=MS2 => Current Extended Function Num[AREP]:=Get_State Current Slot Number[AREP]:=Slot Number Index:=255 Length:=4 Data.w.STATE.Extended_Function_Num:=Get_State Data.w.STATE.FI_Index:=FI Index MSAC2M_Write.req(CREP, Slot Number, Index, Length, Data) LR_State[AREP]:=W-LR-CNF	RUN
117	RUN	MSAC2M_Write.cnf(+)(CREP, Length) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Get_State => Slot_Number:=Current Slot Number[AREP] Index:=255 Length:=5 MSAC2M_Read.req(CREP, Slot Number, Index, Length) LR_State[AREP]:=W-LR-CNF	RUN
118	RUN	MSAC2M_Read.cnf(+)(CREP, Length, Data) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Get_State => Current Extended Function Num[AREP]=No_Service FI State:=Data.r.STATE.State FSPMM2_Get_FI_State.cnf (+) (AREP, FI State) LR_State[AREP]:=W-LR-REQ	RUN
119	RUN	MSAC2M_Write.cnf(-)(CREP, Error Decode, Error Code 1, Error Code 2) /LR_State[AREP]==W-LR-CNF &SetContext(CREP, Service)=TRUE && Current Extended Function Num[AREP]=Get_State => Current Extended Function Num[AREP]=No_Service Error Code:=Error_Code_1 FSPMM2_Get_State.cnf(-) (AREP, Error Code) LR_State[AREP]:=W-LR-REQ	RUN

8.3.4 Functions

Table 60 contains the functions used by the FSPMM2, their arguments and their descriptions.

Table 60 – Functions used by the FSPMM2

Function name	Description
SetContext(Rem_Add/CREP, Service)	This function checks if there exist an entry for the inputs Rem_Add/CREP and Service in the ARL and related CRL of the DP-master (Class 2). A) If there exists an entry then it returns TRUE and sets the local context according to the current CREP and AREP. B) Otherwise it returns FALSE.
IsLastWaitingReset()	This function returns TRUE if all Reset service confirmation are received otherwise it returns FALSE.
StoreNumberIssuedResets()	This function checks stores the number of issued Reset service request in a local variable.
IsLastWaitingMInit()	This function returns TRUE if all MInit service confirmation are received otherwise it returns FALSE.
StoreNumberIssuedMInit()	This function checks stores the number of issued MInit service request in a local variable.

9 Application relationship protocol machines (ARPMs)

9.1 MSCY1S

9.1.1 Primitive definitions

9.1.1.1 Primitives exchanged between MSCY1S and FSPMS

Table 61 shows the service primitives including their associated parameters issued by FSPMS and received by the MSCY1S.

Table 61 – Primitives issued by FSPMS to MSCY1S

Primitive name	Source	Associated parameters	Functions
SInit MS0.req	FSPMS	(none)	Refer to FAL Service Definition in IEC 61158-5-3
Reset.req	FSPMS	(none)	
Abort.req	FSPMS	(none)	
Application Ready.req	FSPMS	(none)	
Check User Prm Result.req	FSPMS	Prm_OK	
Check Ext User Prm Result.req	FSPMS	Ext_Prm_OK	
Check Cfg Result.req	FSPMS	Cfg_OK, Input Data Len, Output Data Len	
Set Cfg.req	FSPMS	Cfg Data	
Set Slave Diag.req	FSPMS	Ext Diag Flag, Ext Diag Overflow, Ext Diag Data	
Set Input.req	FSPMS	Input Data	
Get Output.req	FSPMS	(none)	

Table 62 shows the service primitives including their associated parameters issued by MSCY1S and received by the FSPMS.

Table 62 – Primitives issued by MSCY1S to FSPMS

Primitive name	Source	Associated parameters	Functions
SInit MS0.cnf	MSCY1S	(none)	Refer to FAL Service Definition in IEC 61158-5-3
Reset.cnf	MSCY1S	(none)	
Check User Prm Result.cnf(+)	MSCY1S	(none)	
Check User Prm Result.cnf(-)	MSCY1S	Status	
Check Ext User Prm Result.cnf(+)	MSCY1S	(none)	
Check Ext User Prm Result.cnf(-)	MSCY1S	Status	
Check Cfg Result.cnf(+)	MSCY1S	(none)	
Check Cfg Result.cnf(-)	MSCY1S	Status	
Set Cfg.cnf(+)	MSCY1S	(none)	
Set Cfg.cnf(-)	MSCY1S	(none)	

Primitive name	Source	Associated parameters	Functions
Set Slave Diag.cnf(+)	MSCY1S	(none)	
Set Slave Diag.cnf(-)	MSCY1S	(none)	
Set Input.cnf(+)	MSCY1S	(none)	
Set Input.cnf(-)	MSCY1S	(none)	
Get Output.cnf(+)	MSCY1S	Output Data, Clear Flag, New Flag	
Get Output.cnf(-)	MSCY1S	(none)	
Start.ind	MSCY1S	Actual Enabled Alarms, Alarm Sequence, Alarm Limit	
Stop.ind	MSCY1S	(none)	
Fault.ind	MSCY1S	(none)	
Set Slave Add.ind	MSCY1S	New Slave Add, Ident Number, No Add Chg, Rem Slave Data	
Check User Prm.ind	MSCY1S	User Prm Data	
Check Ext User Prm.ind	MSCY1S	Ext User Prm Data	
Check Cfg.ind	MSCY1S	Check Cfg Mode, Cfg Data	
New Output.ind	MSCY1S	Clear Flag	
SYNCH Event.ind	MSCY1S	Status	
Global Control.ind	MSCY1S	Clear Command, Sync Command, Freeze Command, Group Select	

9.1.1.2 Parameters of MSCY1S primitives

The parameters used with the primitives exchanged between the FSPMS and the MSCY1S are described in FAL Service Definition in IEC 61158-5-3.

9.1.2 State machine description

The MSCY1S State Machine handles the following services issued by the DP-master (Class 1):

diagnostic,

parameterisation,

configuration,

(cyclic) data exchange.

The MSCY1S State Machine starts (Start.req/cnf) and stops (Stop.req/cnf) the MSAC1S State Machine that handles acyclic services. The MSAC1S State Machine sends an Abort.ind to the MSCY1S State Machine that closes also the cyclic connection. The MSCY1S State Machine informs the FSPMS that the MSAC1S and the MSCY1S have been started (Start.ind) or stopped (Stop.ind).

If a Slave does not implement the parameterization structures DPV1_Status 1-3, the MSAL1S and MSAC1S State Machines, the following macros should be left empty. In this case the variable Operation_Mode remains in state V0:

PV0V0 shall return TRUE

OPERATION_MODE_OK shall return TRUE

NOPRMCMD shall return TRUE

ISO_MODE_OK shall return TRUE

STOP_ISOM

STOP_ASM

CHECK_ALARM_START

SET_OPERATION_MODE

SET_ALARM_CHKCFGM

EXE_PRM_CMD

START_C1

STOP_C1

START_C1_CON

STOP_C1_CON

Rules for checking parameterization data

The rules according to Table 63 shall be used to check the octets DPV1_Status_1 to DPV1_Status_3 from the Set_Prm-REQ-PDU with local capabilities of the DP-slave.

Table 63 – Rules for DPV1_Status_1, DPV1_Status_2 and DPV1_Status_3 check

Bit(s) in Set_Prm-REQ-PDU	Rules for checking	
Reserved bits in field DPV1_Status_1 (Bit 0, Bit 1, Bit 3, Bit 4)	These bits shall not be checked (don't care). => okay for any combination	
Bit WD_Base_1ms in field DPV1_Status_1	= FALSE(0): => okay	= TRUE (1) & WD_Base_1ms_supp = 1: => okay = TRUE (1) & WD_Base_1ms_supp = 0: => okay, but the DP-slave shall load its local WD-Timer with a corrected value (based on the next possible value with 10 ms base)
Bit Publisher_Enable in field DPV1_Status_1	= FALSE(0): => okay	= TRUE(1) & Publisher_supp = 1: => okay = TRUE(1) & Publisher_supp = 0: => Prm_Fault
Bit Fail_Safe in field DPV1_Status_1	= FALSE(0) & Fail_Safe_required = 1: => Prm_Fault = FALSE(0) & Fail_Safe_required = 0: => okay	= TRUE(1) & Fail_Safe_supp = 1: => okay = TRUE(1) & Fail_Safe_supp = 0: => Prm_Fault
Bit DPV1_Enable in field DPV1_Status_1	= FALSE(0) & C1_Read_Write_required=0: => okay = FALSE(0) & C1_Read_Write_required=1: => Prm_Fault	= TRUE(1) & C1_Read_Write_supp = 0: => Prm_Fault = TRUE(1) & C1_Read_Write_supp = 1: => okay, open MS1 application relationship
Bit Check_Cfg_Mode in field DPV1_Status_2	= 0 (strict check): => okay The application process has to check the data of the following Chk_Cfg-REQ-PDU in a strict way (equality).	= 1 (more flexible check): => okay; The application process has to check the data of the following Chk_Cfg-REQ-PDU in a more flexible way (compatibility).
Reserved Bit 1 in field DPV1_Status_2	= 0: => okay	= 1: => Prm_Fault

Bit(s) in Set_Prm-REQ-PDU	Rules for checking	
Enable_xx_Alarm Bits in field DPV1_Status_2	= FALSE(0) & xx_Alarm_required = 1 & DPV1_Status_1.DPV1_Enable = FALSE (0): => Prm_Fault = FALSE(0) & xx_Alarm_required = 0 & DPV1_Status_1.DPV1_Enable = FALSE (0): => okay = FALSE(0) & xx_Alarm_required = 1 & DPV1_Status_1.DPV1_Enable = TRUE (1): => Prm_Fault = FALSE(0) & xx_Alarm_required = 0 & DPV1_Status_1.DPV1_Enable = TRUE (1): => okay	=TRUE(1) & DPV1_Status_1.DPV1_Enable = FALSE(0): => Prm_Fault =TRUE(1) & DPV1_Status_1.DPV1_Enable = TRUE (1): => okay
Bits Alarm_Mode in field DPV1_Status_3	0 => okay	<>0 & Alarm_Sequence_Mode_count<>0: => okay <>0 & Alarm_Sequence_Mode_count=0: => Prm_Fault
Bit Prm_Structure in field DPV1_Status_3	= FALSE(0) & Prm_Structure_required=FALSE: =>okay = FALSE(0) & Prm_Structure_required=TRUE: => Prm_Fault	= TRUE(1) & Prm_Structure_supp=TRUE: =>okay = TRUE(1) & Prm_Structure_supp=FALSE: => Prm_Fault
Bit IsoM_Req in field DPV1_Status_3	= FALSE(0) & Isochronous_Mode_required=FALSE: =>okay = FALSE(0) & Isochronous_Mode_required=TRUE: => Prm_Fault	= TRUE(1) & (Isochronous_Mode_supp = TRUE & DPV1_Status_1.Fail_Safe = TRUE & Group_Ident< 0x80 & Freeze_Req=0 & Sync_Req=0): => okay TRUE(1) & !(Isochronous_Mode_supp = TRUE & DPV1_Status_1.Fail_Safe = TRUE & Group_Ident< 0x80 & Freeze_Req=0 & Sync_Req=0): => Prm-Fault
Bit PrmCmd in field DPV1_Status_3	= FALSE(0) & PrmCmd_required=FALSE: =>okay = FALSE(0) & PrmCmd_required=TRUE: => Prm_Fault	= TRUE(1) & (PrmCmd_supp = TRUE): => okay TRUE(1) & (PrmCmd_supp = FALSE): => Prm-Fault
Reserved-Bits (Bit 5 and Bit 6) in field DPV1_Status_3	= 0: => okay	1: => Prm_Fault
NOTE The table above gives a comprehensive overview on all checks that belong to the parameter. However, some of them may also checked on other places within the state machine.		

Isochronous behaviour

Devices that support isochronous operation shall set the concerning AR attributes according to the configuration. The block for isochronous parameters belong to the application and are part of user data within the Set_Prm-REQ-PDU or Set_Ext_Prm-REQ-PDU. The DP-master sets the DP-slave into isochronous operation by setting the IsoM_Req-Bit TRUE. When the MSCY1S receives the Set_Prm-REQ-PDU it has to check the following condition:

If(Isochronous Mode = TRUE) then Failsafe shall be set and any Group shall not be set concerning the Set_Prm-REQ-PDU otherwise the DP-slaves shall generate a Prm_Fault and shall leave the Data Exchange state. DP-slaves that do not support isochronous operation may be integrated in an isochronous working system by selecting “Group_8” for only this DP-slaves and using Sync and Freeze commands.

If the condition above is fulfilled the DP-slave operates isochronous. Additionally to normal cyclic data exchange the MSCY1S generates a SYNCH Event.ind with Status “IsoM Start”

with receiving the first Global_Control-REQ-PDU, “IsoM SYNCH” with subsequent Global_Control-REQ-PDU’s, and “IsoM Stop” when receiving Global_Control-REQ-PDU with Control Command “Clear” or leaving the data exchange cycle. This service triggers the PLL of the application process.The right access with Get Input and Set Output from the application point of view is monitored with the help of the PLL.

Redundancy behaviour

Devices that support Slave redundancy shall support the PrmCmd. This allows a DP-master (Class 1) to use a DP-slave as Primary or Backup. Additionally the DP-master (Class 1) can stop and start MS1 after a swichover of the DP-master. MSCY1S provides the basic mechanism to support a redundancy structure in the Application.

Local variables

Act_Ref
Counter for Reference Numbers for Diagnosis Updates.

Act_Cnt
Storage for Users Reference number

Ref_Cnt
Storage of Act_Ref at users Diagnosis.

B Sync
(Octet-String)

Intermediate storage for outputs if the DP-slave is operated in Sync Mode.

B Output
(Octet-String)

Output data of the DP-slave that can be fetched by the Get Output service. The requirements for the consistency as described in the Cfg_Data shall be taken into consideration.

B Freeze
(Octet-String)

Intermediate storage for input data.

B Input
(Octet-String)

Depending on the type (see description of Cfg_Data), the data shall be consistently loaded by the Set Input service, or may be captured freely.

B Real_Cfg
(Octet-String)

The configuration which was previously defined for the device, or the configuration which is determined in the start-up by the device, respectively. The structure of the individual octets corresponds to the structure of Cfg_Data (->Chk_Cfg).

B User Prm
(Octet-String)

Intermediate storage for User Prm data.

B Ext User Prm
(Octet-String)

Intermediate storage for Ext User Prm data.

B Cfg
(Octet-String)

Intermediate storage for Cfg data.

Diag
(Octet-String)

Intermediate storage for Diag data.

Ext Diag Data
(Octet-String)

Intermediate storage for Ext Diag Data, the diagnosis setup by the user.

Ext Diag Flag
(Boolean)

Intermediate storage for Ext Diag Flag, which indicates important user diagnosis.

Clear Command
Sync Command
Freeze Command
Intermediate storage for global control commands.

Station_Address
(Unsigned8)

Own station address which may be set by the service "Set_Slave_Add" or which is locally stored.

Check_Prm_Add
(Unsigned8)

Masters station address which has invoked the last service "Set_Prm".

Check_Ext_Prm_Add
(Unsigned8)

Masters station address which has invoked the last service "Set_Ext_Prm".

Check_Cfg_Add
(Unsigned8)

Masters station address which has invoked the last service "Chk_Cfg".

Output Data Len
(Unsigned8)

Indicates the number of output octets that are actually used.

Input Data Len
(Unsigned8)

Indicates the number of input octets that are actually used.

Real_No_Add_Chg
(Boolean)

Indicates if an address change is possible (FALSE) or not. Can be statically defined or stored.

Active_Groups
(Unsigned8)

Contains the Group_Ident as set via Set_Prm. This is used at Global_Control for the decision if the station is addressed.

Diag_Flag
(Boolean)

This flag indicates that the Master shall be informed by a high prior Reply-Update that a change in the diagnostic information occurred.

Sync_Supp
(Boolean)

Indicates that the DP-slave does support the Sync-Mode.

Freeze_Supp
(Boolean)

Indicates that the DP-slave does support the Freeze-Mode.

Operation_Mode
(Boolean)

Local flag that stores the Operation_Mode set with the Set_Prm service. Operation_Mode=V0 means that the Slave does not support acyclic services on MSAC1 and alarms.

Prm_Pending
(Unsigned8)

Local flag that indicates that the User is still processing the last call of the function “Check UserPrm".

Ext_Prm_Pending
(Unsigned8)

Local flag that indicates that the User is still processing the last call of the function “Check Ext_Prm".

Cfg_Pending
(Unsigned8)

Local flag that indicates that the User is still processing the last call of the function “Check Cfg".

Input_Pending
(Unsigned8)

Local flag that indicates that the User has not yet set the Input data

New Output
(Unsigned8)

Local flag that indicates that there are Output data available not yet fetched with "Get Output".

DX_Entered
(Boolean)

Local flag which is set after the MSCY1S State Machine has entered the DATA-EXCH mode and at least one Data_Exchange service has been completed.

Start_State
(Unsigned8)

Local variable which is used to store the state of the MSAC1S-State Machine.

Possible values for Start_State:

Idle MSAC1S is closed

Run MSAC1S is open

B Start_Operation is initiated

E Stop_Operation is initiated

BE Start_Operation is initiated, Stop_Operation is queued

EB Stop_Operation is initiated, Start _Operation is queued

BEB Start_Operation is initiated, Stop_Operation (first) and Start_Operation(second) are queued

Safe_State
(Boolean)

Indicates whether the outputs has to be switched to the safe state.

Chk Cfg Mode
(Boolean)

Indicates how to check the Cfg Data.

First Synch
(Boolean)

This local variable is FALSE if the Isochronous mode has been activated and at least one Global_Control-REQ-PDU with Group Select (Group_8) has been received.

9.1.3 MSCY1S state table

Table 64 contains the complete description of the MSCY1S state machine.

Table 64 – MSCY1S state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	MSCY1S SInit MS0.req () => Diag.Ext_Diag_Data:= NIL Diag.Ext Diag Flag:= FALSE B Real Cfg:= Real Cfg Data Chk Cfg Mode:= FALSE Sync Supp:= Sync Supported Freeze Supp:= Freeze Supported Diag.Ident Number:= Ident Number Real No Add Chg:= No Add Chg Act_Ref:= 0 Act_Cnt:= 0 Ref_Cnt:= 0 DMPMS Slave Init.req ()	WAIT-INI-CON
2	POWER-ON	MSCY1S Abort.req () => ignore	POWER-ON
3	POWER-ON	MSCY1S Set Slave Diag.req (Ext Diag Flag, Ext Diag Overflow, Ext Diag Data, Reference) => Act_Cnt:= Reference MSCY1S Set Slave Diag.cnf(-) (Reference:= Act_Cnt)	POWER-ON
4	POWER-ON	MSCY1S Set Cfg.req (Cfg Data) => MSCY1S Set Cfg.cnf(-) ()	POWER-ON

#	Current State	Event /Condition =>Action	Next State
5	POWER-ON	MSCY1S Get Output.req () => MSCY1S Get Output.cnf(-) ()	POWER-ON
6	POWER-ON	MSCY1S Set Input.req (Input Data) => MSCY1S Set Input.cnf(-) ()	POWER-ON
7	POWER-ON	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) => Status:= Reset MSCY1S Check Cfg Result.cnf(-) (Status)	POWER-ON
8	POWER-ON	MSCY1S Application Ready.req() => ignore	POWER-ON
9	POWER-ON	MSCY1S Check User Prm Result.req (Prm_OK) => Status:= Reset MSCY1S Check User Prm Result.cnf(-) (Status)	POWER-ON
10	POWER-ON	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) => Status:= Reset MSCY1S Check Ext User Prm Result.cnf(-) (Status)	POWER-ON
11	WAIT-INI-CON	DMPMS Slave Init.cnf () => Operation Mode:= V0 Start_State:= Idle Safe_State:=TRUE Diag.Prm_Req:= TRUE Diag.Cfg_Fault:= FALSE Diag.Prm_Fault:= FALSE Diag.Not_Supported:= FALSE Diag.Master_Add:= Invalid Diag.WD_On:= FALSE Diag.Station_Not_Ready:= TRUE Diag.Sync_Mode:= FALSE Diag.Freeze_Mode:= FALSE Diag.Stat_Diag:= FALSE Diag.Ext_Diag_Overflow:= FALSE B User Prm:= NIL B Cfg:= NIL Diag_Flag:= FALSE Diag Data:= Diag Act_Ref:= Act_Ref + 1 Prm Structure = FALSE DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) DMPMS Get Cfg Upd.req (Cfg Data) MSCY1S SInit MS0.cnf ()	WAIT-PRM
12	WAIT-PRM	MSCY1S Abort.req () => ignore	WAIT-PRM
13	WAIT-PRM	MSCY1S Set Slave Diag.req (Ext Diag Flag, Ext Diag Overflow, Ext Diag Data, Reference) => Diag.Ext Diag Data:= Ext Diag Data Diag.Ext Diag Flag:= Ext Diag Flag Diag.Ext Diag Overflow:= Ext Diag Overflow Diag Data:= Diag Act_Ref:= Act_Ref + 1 Act_Cnt:= Reference Ref_Cnt:= Act_Ref DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) MSCY1S Set Slave Diag.cnf(+) (Reference:= Act_Cnt)	WAIT-PRM
14	WAIT-PRM	MSCY1S Set Cfg.req (Cfg Data) => DMPMS Get Cfg Upd.req (Cfg Data) MSCY1S Set Cfg.cnf(+) ()	WAIT-PRM

#	Current State	Event /Condition =>Action	Next State
15	WAIT-PRM	MSCY1S Get Output.req () => MSCY1S Get Output.cnf(-) ()	WAIT-PRM
16	WAIT-PRM	MSCY1S Set Input.req (Input Data) => MSCY1S Set Input.cnf(-) ()	WAIT-PRM
17	WAIT-PRM	MSCY1S Check User Prm Result.req (Prm_OK) /Prm_Pending = 0 => Status:= Not pending MSCY1S Check User Prm Result.cnf(-) (Status)	WAIT-PRM
18	WAIT-PRM	MSCY1S Check User Prm Result.req (Prm_OK) /Prm_Pending = 2 => Status:= New Prm Prm_Pending:= 1 User Parameter Data:= B User Prm MSCY1S Check User Prm Result.cnf(-) (Status) MSCY1S Check User Prm.ind (Prm Structure, User Parameter Data)	WAIT-PRM
19	WAIT-PRM	MSCY1S Check User Prm Result.req (Prm_OK) /Prm_Pending = 1 => Prm_Pending:= 0 MSCY1S Check User Prm Result.cnf(+)()	WAIT-PRM
20	WAIT-PRM	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 0 => Status:= Not pending MSCY1S Check Cfg Result.cnf(-) (Status)	WAIT-PRM
21	WAIT-PRM	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 2 => Status:= New Cfg Cfg_Pending:= 1 Cfg Data:= B Cfg MSCY1S Check Cfg Result.cnf(-) (Status) MSCY1S Check Cfg.ind (Cfg Data)	WAIT-PRM
22	WAIT-PRM	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 1 => Cfg_Pending:= 0 MSCY1S Check Cfg Result.cnf(+) ()	WAIT-PRM
23	WAIT-PRM	MSCY1S Application Ready.req() => ignore	WAIT-PRM
24	WAIT-PRM	MSAC1S Abort.ind() => ignore	WAIT-PRM
25	WAIT-PRM	MSAC1S Start.cnf() => START_C1_CON	WAIT-PRM
26	WAIT-PRM	MSAC1S Stop.cnf() => Start_State:= Idle	WAIT-PRM
27	WAIT-PRM	DMPMS Set Slave Add.ind(New Slave Add, Ident Number, No Add Chg, Rem Slave Data) /Real_No_Add_Chg = FALSE && Diag.Ident_Number = Ident_Number && New_Slave_Address <= 125 => MSCY1S Set Slave Add.ind (New Slave Add, Ident Number, No Add Chg, Rem Slave Data)	POWER-ON

#	Current State	Event /Condition =>Action	Next State
28	WAIT-PRM	DMPMS Set Slave Add.ind(New Slave Add, Ident Number, No Add Chg, Rem Slave Data) /Real_No_Add_Chg = TRUE || Diag.Ident_Number <> Ident_Number || New_Slave_Add > 125 => ignore	WAIT-PRM
29	WAIT-PRM	DMPMS Slave Diag.ind(Req Add, Reference) => ignore	WAIT-PRM
30	WAIT-PRM	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Lock_Req = FALSE && Unlock_Req = FALSE => DMPMS Set minTsdr.req (MinTsdr)	WAIT-PRM
31	WAIT-PRM	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Unlock_Req = TRUE => ignore	WAIT-PRM
32	WAIT-PRM	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Lock_Req = TRUE && Unlock_Req = FALSE && (Ident_Number <> Diag.Ident_Number || !OPERATION_MODE_OK || WD_On = TRUE && (WD_Fact_1=0 || WD_Fact_2=0)) => Diag.Prm_Fault:= TRUE Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
33	WAIT-PRM	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Lock_Req = TRUE && Unlock_Req = FALSE && Ident_Number = Diag.Ident_Number && OPERATION_MODE_OK && (WD_On = FALSE || (WD_Fact_1 > 0 && WD_Fact_2 > 0)) && (Freeze_Req = TRUE && Freeze Supp = FALSE || Sync_Req = TRUE && Sync Supp = FALSE || Prm_Data[1].0, .1, .2 = TRUE) => Diag.Not_Supported:= TRUE Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
34	WAIT-PRM	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len < 7 => ignore	WAIT-PRM
35	WAIT-PRM	DMPMS Set Prm.ind(Req Add, Prm Data) /PRM_OK && Prm_Pending = 0 => Diag.Master_Add:= Req Add Check_Prm_Add:= Req Add FirstSynch:= TRUE SET_OPERATION_MODE SET_WD Active_Groups:= Group_Ident Diag.Prm_Fault:= FALSE Diag.Prm_Req:= FALSE Diag.Not_Supported:= FALSE Prm_Pending:= 1 User Parameter Data:= Prm Data.User Prm SET_ALARM_CHKCFGM Input_Pending:= FALSE Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) DMPMS Set minTsdr.req (minTsdr) MSCY1S Check User Prm.ind (Prm Structure, User Parameter Data)	WAIT-CFG

#	Current State	Event /Condition =>Action	Next State
36	WAIT-PRM	DMPMS Set Prm.ind(Req Add, Prm Data) /PRM_OK && Prm_Pending > 0 => Diag.Master_Add:= Req Add Check_Prm_Add:= Req Add FirstSynch:= TRUE SET_OPERATION_MODE SET_WD Active_Groups:= Group_Ident Diag.Prm_Fault:= FALSE Diag.Prm_Req:= FALSE Diag.Not_Supported:= FALSE Prm_Pending:= 2 B User Prm:= Prm Data.User Prm SET_ALARM_CHKCFGM Input_Pending:= FALSE Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) DMPMS Set minTsdr.req (minTsdr)	WAIT-CFG
37	WAIT-PRM	DMPMS Chk Cfg.ind(Req Add, Cfg Data) => ignore	WAIT-PRM
38	WAIT-PRM	DMPMS Set Ext Prm.ind(Req_Add, Ext Prm Data) /Diag.Master_Add = Req_Add => ignore	WAIT-PRM
39	WAIT-PRM	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) /Ext_Prm_Pending = 0 => Status:= Not pending MSCY1S Check Ext User Prm Result.cnf(-) (Status)	WAIT-PRM
40	WAIT-PRM	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) /Ext_Prm_Pending = 2 => Status:= New Prm Prm_Pending:= 1 Ext User Parameter Data:= B Ext User Prm MSCY1S Check Ext User Prm Result.cnf(-) (Status) MSCY1S Check Ext User Prm.ind (Ext User Parameter Data)	WAIT-PRM
41	WAIT-PRM	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) /Ext_Prm_Pending = 1 => Ext_Prm_Pending:= 0 MSCY1S Check Ext User Prm Result.cnf(+)()	WAIT-PRM
42	WAIT-CFG	MSCY1S Abort.req () => Operation_Mode:= V0 Diag.Master_Add:= invalid Diag.Prm_Req:= TRUE Diag.Station_Not_Ready:= TRUE StopTimer(WD) Diag.WD_On:= FALSE STOP_C1 Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM

#	Current State	Event /Condition =>Action	Next State
43	WAIT-CFG	MSCY1S Set Slave Diag.req (Ext Diag Flag, Ext Diag Overflow, Ext Diag Data, Reference) => Diag.Ext Diag Data:= Ext Diag Data Diag.Ext Diag Flag:= Ext Diag Flag Diag.Ext Diag Overflow:= Ext Diag Overflow Diag Data:= Diag Act_Ref:= Act_Ref + 1 Act_Cnt:= Reference Ref_Cnt:= Act_Ref DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) MSCY1S Set Slave Diag.cnf(+) (Reference:= Act_Cnt)	WAIT-CFG
44	WAIT-CFG	MSCY1S Set Cfg.req (Cfg Data) => DMPMS Get Cfg Upd.req (Cfg Data) MSCY1S Set Cfg.cnf(+) ()	WAIT-CFG
45	WAIT-CFG	MSCY1S Get Output.req () => MSCY1S Get Output.cnf(-) ()	WAIT-CFG
46	WAIT-CFG	MSCY1S Set Input.req (Input Data) /Input_Pending = FALSE => MSCY1S Set Input.cnf(-) ()	WAIT-CFG
47	WAIT-CFG	MSCY1S Set Input.req (Input Data) /Input_Pending = TRUE && Operation_Mode = V0 => Inp Data:= Input Data Diag.Cfg_Fault:= FALSE Diag_Flag:= TRUE Diag.Stat_Diag:= TRUE Diag.Station_Not_Ready:= FALSE DX_Entered:= FALSE Diag Data:= Diag Act_Ref:= Act_Ref + 1 MSCY1S Set Input.cnf(+) () DMPMS Enter.req (Master Add, Input Data Len, Output Data Len) DMPMS RD Inp Upd.req (Inp Data) DMPMS Data Exchange Upd.req (Diag Flag, Inp Data) DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	DATA-EXCH
48	WAIT-CFG	MSCY1S Set Input.req (Input Data) /Input_Pending = TRUE && Operation_Mode = V1 => B Input:= Input Data START_C1 MSCY1S Set Input.cnf(+) ()	WAIT-CFG
49	WAIT-CFG	MSCY1S Check User Prm Result.req (Prm_OK) /Prm_Pending = 0 => Status:= Not pending MSCY1S Check User Prm Result.cnf(-) (Status)	WAIT-CFG
50	WAIT-CFG	MSCY1S Check User Prm Result.req (Prm_OK) /Prm_Pending = 2 => Status:= New Prm Prm_Pending:= 1 User Parameter Data:= B User Prm MSCY1S Check User Prm Result.cnf(-) (Status) MSCY1S Check User Prm.ind (Prm Structure, User Parameter Data)	WAIT-CFG
51	WAIT-CFG	MSCY1S Check User Prm Result.req (Prm_OK) /Prm_Pending = 1 && Diag.Master_Add <> Check_Prm_Add => Status:= Inv Master Add Prm_Pending:= 0 MSCY1S Check User Prm Result.cnf(-) (Status)	WAIT-CFG

#	Current State	Event /Condition =>Action	Next State
52	WAIT-CFG	MSCY1S Check User Prm Result.req (Prm_OK) /Prm_Pending = 1 && Diag.Master_Add = Check_Prm_Add && Prm_OK = TRUE => Prm_Pending:= 0 MSCY1S Check User Prm Result.cnf(+)()	WAIT-CFG
53	WAIT-CFG	MSCY1S Check User Prm Result.req (Prm_OK) /Prm_Pending = 1 && Diag.Master_Add = Check_Prm_Add && Prm_OK = FALSE => Prm_Pending:= 0 Diag.Prm_Fault:= TRUE Diag.Master_Add:= Invalid Diag.Prm_Req:= TRUE StopTimer(WD) Diag.WD_On:= FALSE STOP_C1 Diag Data:= Diag Act_Ref:= Act_Ref + 1 MSCY1S Check User Prm Result.cnf(+)() DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
54	WAIT-CFG	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 0 => Status:= Not pending MSCY1S Check Cfg Result.cnf(-) (Status)	WAIT-CFG
55	WAIT-CFG	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 2 => Status:= New Cfg Cfg_Pending:= 1 Cfg Data:= B Cfg MSCY1S Check Cfg Result.cnf(-) (Status) MSCY1S Check Cfg.ind (Cfg Data)	WAIT-CFG
56	WAIT-CFG	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 1 && Diag.Master_Add <> Check_Cfg_Add => Status:= Inv Master Add Cfg_Pending:= 0 MSCY1S Check Cfg Result.cnf(-) (Status)	WAIT-CFG
57	WAIT-CFG	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 1 && Diag.Master_Add = Check_Cfg_Add && Cfg_OK = TRUE && Input Data Len > 0 => Inp Data Len:= Input Data Len Outp Data Len:= Output Data Len Cfg_Pending:= 0 Input_Pending:= TRUE MSCY1S Check Cfg Result.cnf(+) ()	WAIT-CFG
58	WAIT-CFG	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 1 && Diag.Master_Add = Check_Cfg_Add && Cfg_OK = TRUE && Operation_Mode = V1 && Input Data Len = 0 => Inp Data Len:= 0 Outp Data Len:= Output Data Len Inp Data:= NIL Cfg_Pending:= 0 START_C1 MSCY1S Check Cfg Result.cnf(+) ()	WAIT-CFG

#	Current State	Event /Condition =>Action	Next State
59	WAIT-CFG	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 1 && Diag.Master_Add = Check_Cfg_Add && Cfg_OK = TRUE && Operation_Mode = V0 && Input Data Len = 0 => Inp Data Len:= 0 Outp Data Len:= Output Data Len Inp Data:= NIL Cfg_Pending:= 0 Diag.Cfg_Fault:= FALSE Diag_Flag:= TRUE Diag.Stat_Diag:= TRUE Diag.Station_Not_Ready:= FALSE DX_Entered:= FALSE Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Enter.req DMPMS Data Exchange Upd.req (Diag Flag, Inp Data) DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) MSCY1S Check Cfg Result.cnf(+) ()	DATA-EXCH
60	WAIT-CFG	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 1 && Diag.Master_Add = Check_Cfg_Add && Cfg_OK = FALSE => Cfg_Pending:= 0 Diag.Cfg_Fault:= TRUE Diag.Prm_Req:= TRUE Diag.Master_Add:= Invalid StopTimer(WD) Diag.WD_On:= FALSE STOP_C1 Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) MSCY1S Check Cfg Result.cnf(+) ()	WAIT-PRM
61	WAIT-CFG	MSCY1S Application Ready.req() => ignore	WAIT-CFG
62	WAIT-CFG	MSAC1S Abort.ind() => ignore	WAIT-CFG
63	WAIT-CFG	MSAC1S Start.cnf() /Start_State <> B => START_C1_CON	WAIT-CFG
64	WAIT-CFG	MSAC1S Start.cnf() /Start_State = B => Diag.Cfg_Fault:= FALSE Diag_Flag:= TRUE Diag.Stat_Diag:= TRUE Diag.Station_Not_Ready:= FALSE DX_Entered:= FALSE START_C1_CON Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Enter.req DMPMS RD Inp Upd.req (Inp Data) DMPMS Data Exchange Upd.req (Diag Flag, Inp Data) DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	DATA-EXCH
65	WAIT-CFG	MSAC1S Stop.cnf() => STOP_C1_CON	WAIT-CFG

#	Current State	Event /Condition =>Action	Next State
66	WAIT-CFG	MSAC1S Abort.ind() => Operation_Mode:= V0 Diag.Master_Add:= invalid Diag.Prm_Req:= TRUE StopTimer(WD) Diag.WD_On:= FALSE STOP_C1 Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
67	WAIT-CFG	DMPMS Set Slave Add.ind(New Slave Add, Ident Number, No Add Chg, Rem Slave Data) => ignore	WAIT-CFG
68	WAIT-CFG	DMPMS Slave Diag.ind(Req Add, Reference) /Diag.Master_Add = Req_Add => TRIG_WD	WAIT-CFG
69	WAIT-CFG	DMPMS Slave Diag.ind(Req Add, Reference) /Diag.Master_Add <> Req_Add => ignore	WAIT-CFG
70	WAIT-CFG	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Unlock_Req = TRUE && Diag.Master_Add = Req_Add => Diag.Master_Add:= invalid Diag.Prm_Req:= TRUE StopTimer(WD) Diag.WD_On:= FALSE STOP_C1 Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
71	WAIT-CFG	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Unlock_Req = TRUE && Diag.Master_Add <> Req_Add => ignore	WAIT-CFG
72	WAIT-CFG	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Unlock_Req = FALSE && Lock_Req = TRUE && Diag.Master_Add = Req_Add && (Ident_Number <> Diag.Ident_Number || OPERATION_MODE_OK = FALSE || WD_On = TRUE && (WD_Fact_1=0 || WD_Fact_2=0)) => Diag.Prm_Fault:= TRUE Diag.Master_Add:= Invalid Diag.Prm_Req:= TRUE StopTimer(WD) Diag.WD_On:= FALSE STOP_C1 Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM

#	Current State	Event /Condition =>Action	Next State
73	WAIT-CFG	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Unlock_Req = FALSE && Lock_Req = TRUE && Diag.Master_Add = Req_Add && Ident_Number = Diag.Ident_Number && OPERATION_MODE_OK && (WD_On = FALSE || WD_Fact_1>0 && WD_Fact_2>0) && (Freeze_Req = TRUE && Freeze Supp = FALSE || Sync_Req = TRUE && Sync Supp = FALSE || Prm_Data[1].0, .1, .2 = TRUE) => Diag.Not_Supported:= TRUE Diag.Master_Add:= Invalid Diag.Prm_Req:= TRUE StopTimer(WD) Diag.WD_On:= FALSE STOP_C1 Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
74	WAIT-CFG	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Unlock_Req = FALSE && Lock_Req = TRUE && Diag.Master_Add <> Req_Add && (Ident_Number <> Diag.Ident_Number || OPERATION_MODE_OK = FALSE || WD_On = TRUE && (WD_Fact_1=0 || WD_Fact_2=0) || Freeze_Req = TRUE && Freeze Supp = FALSE || Sync_Req = TRUE && Sync Supp = FALSE || Prm_Data[1].0, .1, .2 = TRUE) => ignore	WAIT-CFG
75	WAIT-CFG	DMPMS Set Prm.ind(Req Add, Prm Data) /PRM_OK && Prm_Pending = 0 => Diag.Master_Add:= Req_Add Check_Prm_Add:= Req Add SET_OPERATION_MODE SET_WD Active_Groups:= Group_Ident Prm_Pending:= 1 User Parameter Data:= Prm Data.User Prm SET_ALARM_CHKCFGM Input_Pending:= FALSE STOP_C1 Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) DMPMS Set minTsdr.req (minTsdr) MSCY1S Check User Prm.ind (Prm Structure, User Parameter Data)	WAIT-CFG
76	WAIT-CFG	DMPMS Set Prm.ind(Req Add, Prm Data) /PRM_OK && Prm_Pending > 0 => Diag.Master_Add:= Req_Add Check_Prm_Add:= Req Add SET_OPERATION_MODE SET_WD Active_Groups:= Group_Ident Prm_Pending:= 2 B User Prm:= Prm Data.User Prm SET_ALARM_CHKCFGM Input_Pending:= FALSE STOP_C1 Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) DMPMS Set minTsdr.req (minTsdr)	WAIT-CFG
77	WAIT-CFG	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Unlock_Req = FALSE && Lock_Req = FALSE => DMPMS Set minTsdr.req (MinTsdr)	WAIT-CFG

#	Current State	Event /Condition =>Action	Next State
78	WAIT-CFG	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len < 7 => ignore	WAIT-CFG
79	WAIT-CFG	DMPMS Chk Cfg.ind(Req Add, Cfg Data) /Diag.Master_Add <> Req_Add => ignore	WAIT-CFG
80	WAIT-CFG	DMPMS Chk Cfg.ind(Req Add, Cfg Data) /Diag.Master_Add = Req_Add && Cfg_Pending = 0 => Cfg_Pending:= 1 Check_Cfg_Add:= Req Add TRIG_WD MSCY1S Check Cfg.ind (Check Cfg Mode, Cfg Data)	WAIT-CFG
81	WAIT-CFG	DMPMS Chk Cfg.ind(Req Add, Cfg Data) /Diag.Master_Add = Req_Add && Cfg_Pending > 0 => Cfg_Pending:= 2 Check_Cfg_Add:= Req Add B Cfg:= Cfg Data TRIG_WD	WAIT-CFG
82	WAIT-CFG	WDTimer expired => Diag.Master_Add:= invalid Diag.Prm_Req:= TRUE StopTimer(WD) Diag.WD_On:= FALSE STOP_C1 Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
83	WAIT-CFG	DMPMS Set Ext Prm.ind(Req_Add, Ext Prm Data) /Diag.Master_Add = Req_Add &&Ext_Prm_Pending = 0 => TRIG_WD Ext User Parameter Data:= Ext User Prm Data Ext_Prm_Pending:= 1 Check_Ext_Prm_Add:= Req_Add MSCY1S Check Ext User Prm.ind (Ext User Parameter Data)	WAIT-CFG
84	WAIT-CFG	DMPMS Set Ext Prm.ind(Req_Add, Ext Prm Data) /Diag.Master_Add = Req_Add && Ext_Prm_Pending > 0 => TRIG_WD B Ext User Prm:= Ext User Prm Data Ext_Prm_Pending:= 2 Check_Ext_Prm_Add:= Req_Add	WAIT-CFG
85	WAIT-CFG	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) /Ext_Prm_Pending = 0 => Status:= Not pending MSCY1S Check Ext User Prm Result.cnf(-) (Status)	WAIT-CFG
86	WAIT-CFG	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) /Ext_Prm_Pending = 2 => Status:= New Prm Ext_Prm_Pending:= 1 Ext User Parameter Data:= B Ext User Prm MSCY1S Check Ext User Prm Result.cnf(-) (Status) MSCY1S Check Ext User Prm.ind (Ext User Parameter Data)	WAIT-CFG

#	Current State	Event /Condition =>Action	Next State
87	WAIT-CFG	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) /Ext_Prm_Pending = 1 && Diag.Master_Add <> Check_Ext_Prm_Add => Status:= Inv Master Add Ext_Prm_Pending:= 0 MSCY1S Check Ext User Prm Result.cnf(-) (Status)	WAIT-CFG
88	WAIT-CFG	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) /Ext_Prm_Pending = 1 && Diag.Master_Add = Check_Ext_Prm_Add && Ext_Prm_OK = TRUE => Ext_Prm_Pending:= 0 MSCY1S Check Ext User Prm Result.cnf(+)()	WAIT-CFG
89	WAIT-CFG	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) /Ext_Prm_Pending = 1 && Diag.Master_Add = Check_Prm_Add && Ext_Prm_OK = FALSE => Ext_Prm_Pending:= 0 Diag.Cfg_Fault:= TRUE Diag.Master_Add:= Invalid Diag.Prm_Req:= TRUE StopTimer(WD) Diag.WD_On:= FALSE STOP_C1 Diag Data:= Diag Act_Ref:= Act_Ref + 1 MSCY1S Check Ext User Prm Result.cnf(+)() DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
90	DATA-EXCH	MSCY1S Abort.req () => LEAVE-MASTER Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
91	DATA-EXCH	MSCY1S Set Slave Diag.req (Ext Diag Flag, Ext Diag Overflow, Ext Diag Data, Reference) => Diag.Ext Diag Data:= Ext Diag Data Diag.Ext Diag Flag:= Ext Diag Flag Diag.Ext Diag Overflow:= Ext Diag Overflow Diag Data:= Diag Diag Flag:= TRUE Act_Ref:= Act_Ref + 1 Act_Cnt:= Reference Ref_Cnt:= Act_Ref DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) DMPMS Data Exchange Upd.req(Diag Flag, Inp Data)	DATA-EXCH
92	DATA-EXCH	MSCY1S Set Cfg.req (Cfg Data) => Diag.Cfg_Fault:=TRUE LEAVE-MASTER Act_Ref:= Act_Ref + 1 DMPMS Get Cfg Upd.req (Cfg Data) MSCY1S Set Cfg.cnf(+) () DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
93	DATA-EXCH	MSCY1S Get Output.req () /DX_Entered = TRUE => Clear Flag:= Safe_State New Flag:= New Output New Output:= FALSE Outp Data, Output Data:= B Output DMPMS RD Outp Upd.req (Outp Data) MSCY1S Get Output.cnf(+) (Output Data, Clear Flag, New Flag)	DATA-EXCH
94	DATA-EXCH	MSCY1S Get Output.req () /DX_Entered = FALSE => MSCY1S Get Output.cnf(-) ()	DATA-EXCH

#	Current State	Event /Condition =>Action	Next State
95	DATA-EXCH	MSCY1S Set Input.req (Input Data) /Diag.Freeze_Mode = FALSE => Inp Data, B Input:= Input Data DMPMS Data Exchange Upd.req (Diag Flag, Inp Data) DMPMS RD Inp Upd.req (Inp Data) MSCY1S Set Input.cnf(+) ()	DATA-EXCH
96	DATA-EXCH	MSCY1S Set Input.req (Input Data) /Diag.Freeze_Mode = TRUE => B Freeze:= Input Data MSCY1S Set Input.cnf(+) ()	DATA-EXCH
97	DATA-EXCH	MSCY1S Check User Prm Result.req (Prm_OK) /Prm_Pending = 0 => Status:= Not pending MSCY1S Check User Prm Result.cnf(-) (Status)	DATA-EXCH
98	DATA-EXCH	MSCY1S Check User Prm Result.req (Prm_OK) /Prm_Pending = 2 => Status:= New Prm Prm_Pending:= 1 User Parameter Data:= B User Prm MSCY1S Check User Prm Result.cnf(-) (Status) MSCY1S Check User Prm.ind (Prm Structure, User Parameter Data)	DATA-EXCH
99	DATA-EXCH	MSCY1S Check User Prm Result.req (Prm_OK) /Prm_Pending = 1 && Diag.Master_Add # Check_Prm_Add => Status:= Inv Master Add Prm_Pending:= 0 MSCY1S Check User Prm Result.cnf(-) (Status)	DATA-EXCH
100	DATA-EXCH	MSCY1S Check User Prm Result.req (Prm_OK) /Prm_Pending = 1 && Diag.Master_Add = Check_Prm_Add && Prm_OK = TRUE => Prm_Pending:= 0 MSCY1S Check User Prm Result.cnf(+)()	DATA-EXCH
101	DATA-EXCH	MSCY1S Check User Prm Result.req (Prm_OK) /Prm_Pending = 1 && Diag.Master_Add = Check_Prm_Add && Prm_OK = FALSE => Diag.Prm_Fault:= TRUE LEAVE-MASTER Prm_Pending:= 0 Act_Ref:= Act_Ref + 1 MSCY1S Check User Prm Result.cnf(+)() DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
102	DATA-EXCH	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 0 => Status:= Not pending MSCY1S Check Cfg Result.cnf(-) (Status)	DATA-EXCH
103	DATA-EXCH	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 2 => Status:= New Cfg Cfg_Pending:= 1 Cfg Data:= B Cfg MSCY1S Check Cfg Result.cnf(-) (Status) MSCY1S Check Cfg.ind (Cfg Data)	DATA-EXCH
104	DATA-EXCH	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 1 && Diag.Master_Add # Check_Cfg_Add => Status:= Inv Master Add Cfg_Pending:= 0 MSCY1S Check Cfg Result.cnf(-) (Status)	DATA-EXCH

#	Current State	Event /Condition =>Action	Next State
105	DATA-EXCH	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 1 && Diag.Master_Add = Check_Cfg_Add && Cfg_OK = TRUE && Input Data Len = Inp Data Len && Output Data Len = Outp Data Len => Cfg_Pending:= 0 MSCY1S Check Cfg Result.cnf(+) ()	DATA-EXCH
106	DATA-EXCH	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 1 && Diag.Master_Add = Check_Cfg_Add && Cfg_OK = TRUE && Input Data Len <> Inp Data Len && Output Data Len <> Outp Data Len => Diag.Cfg_Fault:= TRUE LEAVE-MASTER Act_Ref:= Act_Ref + 1 MSCY1S Check Cfg Result.cnf(+) () DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
107	DATA-EXCH	MSCY1S Check Cfg Result.req (Cfg_OK, Input Data Len, Output Data Len) /Cfg_Pending = 1 && Diag.Master_Add = Check_Cfg_Add && Cfg_OK = FALSE => Diag.Cfg_Fault:= TRUE LEAVE-MASTER Act_Ref:= Act_Ref + 1 MSCY1S Check Cfg Result.cnf(+) () DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
108	DATA-EXCH	MSCY1S Application Ready.req() /Diag.Stat_Diag = TRUE => Diag.Stat_Diag:= FALSE Diag_Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) DMPMS Data Exchange Upd.req(Diag Flag, Inp Data)	DATA-EXCH
109	DATA-EXCH	MSCY1S Application Ready.req() /Diag.Stat_Diag = FALSE => ignore	DATA-EXCH
110	DATA-EXCH	MSAC1S Abort.ind() => LEAVE-MASTER Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
111	DATA-EXCH	MSAC1S Start.cnf() /Start_State <> B => START_C1_CON	DATA-EXCH
112	DATA-EXCH	MSAC1S Start.cnf() /Start_State = B => START_C1_CON MSCY1S_Start.ind (Actual_Enabled_Alarms, Alarm_Sequence, Alarm_Limit)	DATA-EXCH
113	DATA-EXCH	MSAC1S Stop.cnf() /Start_State <> E => STOP_C1_CON	DATA-EXCH
114	DATA-EXCH	MSAC1S Stop.cnf() /Start_State = E => STOP_C1_CON MSCY1S_Stop.ind	DATA-EXCH
115	DATA-EXCH	DMPMS Set Slave Add.ind(New Slave Add, Ident Number, No Add Chg, Rem Slave Data) => ignore	DATA-EXCH

#	Current State	Event /Condition =>Action	Next State
116	DATA-EXCH	DMPMS Slave Diag.ind(Req Add, Reference) /Diag.Master_Add = Req_Add && (Diag.Stat_Diag = TRUE || Reference < Ref_Cnt) => TRIG_WD	DATA-EXCH
117	DATA-EXCH	DMPMS Slave Diag.ind(Req Add, Reference) /Diag.Master_Add = Req_Add && Diag.Stat_Diag = FALSE && Reference >= Ref_Cnt => Diag Flag:= FALSE TRIG_WD DMPMS Data Exchange Upd.req (Diag Flag, Inp Data) DMPMS RD Inp Upd.req (Inp Data) MSCY1S Set Slave Diag.cnf(+) (Reference:= Act_Cnt)	DATA-EXCH
118	DATA-EXCH	DMPMS Slave Diag.ind(Req Add, Reference) /Diag.Master_Add <> Req_Add => ignore	DATA-EXCH
119	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Lock_Req = FALSE && Unlock_Req = FALSE => DMPMS Set minTsdr.req (MinTsdr)	DATA-EXCH
120	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Unlock_Req = TRUE && Diag.Master_Add <> Req_Add => ignore	DATA-EXCH
121	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Unlock_Req = TRUE && Diag.Master_Add = Req_Add => LEAVE-MASTER Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
122	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Lock_Req = TRUE && Unlock_Req = FALSE && Diag.Master_Add = Req_Add && (Ident_Number <> Diag.Ident_Number || !OPERATION_MODE_OK || WD_On && (WD_Fact_1=0 || WD_Fact_2=0)) => Diag.Prm_Fault:= TRUE LEAVE-MASTER Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
123	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Lock_Req = TRUE && Unlock_Req = FALSE && Diag.Master_Add = Req_Add && Ident_Number = Diag.Ident_Number && OPERATION_MODE_OK && !WD_On || (WD_Fact_1>0 && WD_Fact_2>0) && (Freeze_Req && Freeze Supp = FALSE || Sync_Req && Sync Supp = FALSE || Prm_Data[1].0, .1, .2 = TRUE) => Diag.Not_Supported:= TRUE LEAVE-MASTER Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
124	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len >= 7 && Lock_Req = TRUE && Unlock_Req = FALSE && Diag.Master_Add <> Req_Add && (Ident_Number <> Diag.Ident_Number || !OPERATION_MODE_OK || WD_On && (WD_Fact_1=0 || WD_Fact_2=0) || Freeze_Req && !Freeze Supp || Sync_Req && !Sync Supp || Prm_Data[1].0, .1, .2 = TRUE) => ignore	DATA-EXCH

#	Current State	Event /Condition =>Action	Next State
125	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Diag.Master_Add = Req_Add && PRM_OK &&!PV0V0 && NOPRMCMD => LEAVE-MASTER Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
126	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Diag.Master_Add = Req_Add && PRM_OK && !NOPRMCMD && Prm_Pending = 0 => SET_WD Active_Groups:= Group_Ident EXE_PRM_CMD Prm_Pending:= 1 Check_Prm_Add:= Req Add User Parameter Data:= Prm Data.User Prm Diag_Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) DMPMS Set minTsdr.req (minTsdr) MSCY1S Check User Prm.ind (Prm Structure, User Parameter Data)	DATA-EXCH
127	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Diag.Master_Add = Req_Add && PRM_OK && !NOPRMCMD && Prm_Pending > 0 => SET_WD Active_Groups:= Group_Ident EXE_PRM_CMD Prm_Pending:= 2 Check_Prm_Add:= Req Add B User Prm:= Prm Data.User Prm Diag_Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) DMPMS Set minTsdr.req (minTsdr)	DATA-EXCH
128	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Diag.Master_Add = Req_Add && PRM_OK && NOPRMCMD && PV0V0 && Prm_Pending = 0 => SET_WD Active_Groups:= Group_Ident Prm_Pending:= 1 Check_Prm_Add:= Req Add User Parameter Data:= Prm Data.User Prm Diag_Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) DMPMS Set minTsdr.req (minTsdr) MSCY1S Check User Prm.ind (Prm Structure, User Parameter Data)	DATA-EXCH
129	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Diag.Master_Add = Req_Add && PRM_OK && NOPRMCMD && PV0V0 && Prm_Pending > 0 =>SET_WD Active_Groups:= Group_Iden tPrm_Pending:= 2 Check_Prm_Add:= Req Add B User Prm:= Prm Data.User Prm Diag_Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) DMPMS Set minTsdr.req (minTsdr)	DATA-EXCH

#	Current State	Event /Condition =>Action	Next State
130	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Diag.Master_Add <> Req_Add && PRM_OK && Prm_Pending = 0 => Diag.Sync_Mode:= FALSE Diag.Freeze_Mode:= FALSE Diag.Station_Not_Ready:= TRUE Diag.Stat_Diag:= FALSE SET_OPERATION_MODE SET_WD Active_Groups:= Group_Ident Safe_State:= TRUE Diag.Master_Add:= Req_Add Check_Prm_Add:= Req Add STOP_ASM STOP_C1 Prm_Pending:= 1 User Parameter Data:= Prm Data.User Prm SET_ALARM_CHKCFGM Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Leave.req DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) DMPMS Set minTsdr.req (minTsdr) MSCY1S Check User Prm.ind (Prm Structure, User Parameter Data)	WAIT-CFG
131	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Diag.Master_Add <> Req_Add && PRM_OK && Prm_Pending > 0 => Diag.Sync_Mode:= FALSE Diag.Freeze_Mode:= FALSE Diag.Station_Not_Ready:= TRUE Diag.Stat_Diag:= FALSE SET_OPERATION_MODE SET_WD Active_Groups:= Group_Ident Safe_State:= TRUE Diag.Master_Add:= Req_Add Check_Prm_Add:= Req Add STOP_ASM STOP_C1 Prm_Pending:= 2 B User Prm:= Prm Data.User Prm SET_ALARM_CHKCFGM Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Leave.req DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) DMPMS Set minTsdr.req (minTsdr)	WAIT-CFG
132	DATA-EXCH	DMPMS Set Prm.ind(Req Add, Prm Data) /Prm_Data.len < 7 => ignore	DATA-EXCH
133	DATA-EXCH	DMPMS Chk Cfg.ind(Req Add, Cfg Data) /Diag.Master_Add <> Req_Add => ignore	DATA-EXCH
134	DATA-EXCH	DMPMS Chk Cfg.ind(Req Add, Cfg Data) /Diag.Master_Add = Req_Add && Cfg_Pending = 0 => Cfg_Pending:= 1 Check_Cfg_Add:= Req Add TRIG_WD MSCY1S Check Cfg.ind (Check Cfg Mode, Cfg Data)	DATA-EXCH
135	DATA-EXCH	DMPMS Chk Cfg.ind(Req Add, Cfg Data) /Diag.Master_Add = Req_Add && Cfg_Pending > 0 => Cfg_Pending:= 2 Check_Cfg_Add:= Req Add B Cfg:= Cfg Data TRIG_WD	DATA-EXCH

#	Current State	Event /Condition =>Action	Next State
136	DATA-EXCH	DMPMS Data Exchange.ind(Outp Data) /Outp_Data.len > 0 && Outp_Data.len = Outp Data Len && Diag.Sync_Mode = FALSE => B Output:= Outp Data TRIG_WD Safe_State:= FALSE Clear Flag:= FALSE New Output:= TRUE CHECK_ALARM_START MSCY1S New Output.ind (Clear Flag)	DATA-EXCH
137	DATA-EXCH	DMPMS Data Exchange.ind(Outp Data) /Outp_Data.len > 0 && Outp_Data.len = Outp Data Len && Diag.Sync_Mode = TRUE => B Sync:= Outp Data TRIG_WD Safe_State:= FALSE CHECK_ALARM_START	DATA-EXCH
138	DATA-EXCH	DMPMS Data Exchange.ind(Outp Data) /Outp_Data.len = 0 && Outp Data Len = 0 => TRIG_WD CHECK_ALARM_START	DATA-EXCH
139	DATA-EXCH	DMPMS Data Exchange.ind(Outp Data) /Outp_Data.len <> Outp Data Len && (Outp_Data.len = 0 && Fail_Safe_supp) => TRIG_WD Safe_State:= TRUE Clear Flag:= TRUE New Output:= TRUE CHECK_ALARM_START MSCY1S New Output.ind (Clear Flag)	DATA-EXCH
140	DATA-EXCH	DMPMS Data Exchange.ind(Outp Data) /Outp_Data.len <> Outp Data Len && (Outp_Data.len <> 0 || Fail_Safe_supp) => LEAVE-MASTERAct_Ref:= Act_Ref + 1 Diag.Cfg_Fault:= TRUE DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
141	DATA-EXCH	WDTimer expired => LEAVE-MASTER Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
142	DATA-EXCH	DMPMS Global Control.ind(Control Command, Group Select) / (Control_Command.0, .6, .7 = TRUE) => LEAVE-MASTER Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
143	DATA-EXCH	DMPMS Global Control.ind(Control Command, Group Select) /Group_Select <> 0 && ((Active_Groups & Group_Select) = 0) && ((Group_Select < 0x80) || Isochronous Mode = FALSE) && (Control_Command.0, .6, .7 = FALSE) => ignore	DATA-EXCH
144	DATA-EXCH	DMPMS Global Control.ind(Control Command, Group Select) /Control Command.Clear Command = FALSE && (Group_Select = 0 || (Active_Groups & Group_Select) <> 0 || ((Group_Select >= 0x80) && Isochronous Mode = TRUE)) && (Control_Command.0, .6, .7 = FALSE) => Clear Command:= FALSE	CHECK-ISOM

#	Current State	Event /Condition =>Action	Next State
145	DATA-EXCH	DMPMS Global Control.ind(Control Command, Group Select) /Control Command.Clear Command = TRUE && (Group_Select = 0 || (Active_Groups & Group_Select) <> 0 || ((Group_Select >= 0x80) && Isochronous Mode = TRUE)) && (Control_Command.0, .6, .7 = FALSE) => Clear Command:= TRUE Clear Flag:= TRUE New Output:= TRUE Safe_State:= TRUE	CHECK-ISOM
146	DATA-EXCH	DMPMS Set Ext Prm.ind(Req_Add, Ext Prm Data) /Diag.Master_Add = Req_Add => Diag.Prm_Fault:= TRUE LEAVE-MASTER Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
147	DATA-EXCH	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) /Ext_Prm_Pending = 0 => Status:= Not pending MSCY1S Check Ext User Prm Result.cnf(-) (Status)	DATA-EXCH
148	DATA-EXCH	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) /Ext_Prm_Pending = 2 => Status:= New Prm Ext_Prm_Pending:= 1 Ext_User Parameter Data:= B Ext User Prm MSCY1S Check Ext User Prm Result.cnf(-) (Status) MSCY1S Check Ext User Prm.ind (Ext User Parameter Data)	DATA-EXCH
149	DATA-EXCH	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) /Ext_Prm_Pending = 1 && Diag.Master_Add <> Check_Ext_Prm_Add => Status:= Inv Master Add Ext_Prm_Pending:= 0 MSCY1S Check Ext User Prm Result.cnf(-) (Status)	DATA-EXCH
150	DATA-EXCH	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) /Ext_Prm_Pending = 1 && Diag.Master_Add = Check_Ext_Prm_Add && Ext_Prm_OK = TRUE => Ext_Prm_Pending:= 0 MSCY1S Check Ext User Prm Result.cnf(+)()	DATA-EXCH
151	DATA-EXCH	MSCY1S Check Ext User Prm Result.req (Ext_Prm_OK) /Ext_Prm_Pending = 1 && Diag.Master_Add = Check_Ext_Prm_Add && Ext_Prm_OK = FALSE => Diag.Cfg_Fault:= TRUE LEAVE-MASTER Ext_Prm_Pending:= 0 Act_Ref:= Act_Ref + 1 MSCY1S Check Ext User Prm Result.cnf(+)() DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM
152	CHECK-SYNC	/NOSYNC => Sync Command:= 0	CHECK-FREEZE
153	CHECK-SYNC	/UNSYNC && Sync Supp = FALSE => Sync Command:= 2	CHECK-FREEZE
154	CHECK-SYNC	/SYNC && Sync Supp = FALSE => LEAVE-MASTER Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM

#	Current State	Event /Condition =>Action	Next State
155	CHECK-SYNC	/UNSYNC && Sync Supp = TRUE && B Sync <> Nil => B Output:= B Sync B Sync:= Nil Sync Command:= 2 Clear Flag:= Safe_State New Outpu t:= TRUE Diag.Sync_Mode:= FALSE Diag Data:= Diag Act_Ref:= Act_Ref + 1 MSCY1S New Output.ind (Clear Flag) DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	CHECK-FREEZE
156	CHECK-SYNC	/UNSYNC && Sync Supp = TRUE && B Sync = Nil => Sync Command:= 2 Clear Flag:= Safe_State Diag.Sync_Mode:= FALSE Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	CHECK-FREEZE
157	CHECK-SYNC	/SYNC && Sync Supp = TRUE && Diag.Sync_Mode = FALSE => B Sync:= Nil Sync Command:= 1 Diag.Sync_Mode:= TRUE Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	CHECK-FREEZE
158	CHECK-SYNC	/SYNC && Sync Supp = TRUE && Diag.Sync_Mode = TRUE && B Sync <> Nil => B Output:= B Sync B Sync:= Nil Clear Flag:= Safe State New Output:= TRUE Sync Command:= 1 MSCY1S New Output.ind (Clear Flag)	CHECK-FREEZE
159	CHECK-SYNC	/SYNC && Sync Supp = TRUE && Diag.Sync_Mode = TRUE && B Sync = Nil => Sync Command:= 1	CHECK-FREEZE

160	CHECK-ISOM	/(Group_Select >= 0x80) && Isochronous Mode = TRUE
		=>
		if (FirstSynch = TRUE)
		Status:= IsoM Start
		FirstSynch:= FALSE
		else
		Status:=IsoM SYNCH
		endif
		MSCY1S SYNCH Event.ind(Status)
			CHECK-SYNC

161	CHECK-ISOM	/(Group_Select < 0x80) || Isochronous Mode = FALSE =>	CHECK-SYNC
162	CHECK-FREEZE	/NOFREEZE => Freeze Command:= 0 MSCY1S Global Control.ind (Clear Command, Sync Command, Freeze Command)	DATA-EXCH
163	CHECK-FREEZE	/UNFREEZE && Freeze Supp = FALSE => Freeze Command:= 2 MSCY1S Global Control.ind (Clear Command, Sync Command, Freeze Command)	DATA-EXCH
164	CHECK-FREEZE	/FREEZE && Freeze Supp = FALSE => LEAVE-MASTER Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref)	WAIT-PRM

#	Current State	Event /Condition =>Action	Next State
165	CHECK-FREEZE	/UNFREEZE && Freeze Supp = TRUE && B Freeze <> Nil => Inp Data:= B Freeze B Freeze:= Nil Freeze Command:= 2 Diag.Freeze_Mode:= FALSE Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Data Exchange Upd.req (Diag Flag, Inp Data) DMPMS RD Inp Upd.req (Inp Data) DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) MSCY1S Global Control.ind (Clear Command, Sync Command, Freeze Command)	DATA-EXCH
166	CHECK-FREEZE	/UNFREEZE && Freeze Supp = TRUE && B Freeze = Nil => Freeze Command:= 2 Diag.Freeze_Mode:= FALSE Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) MSCY1S Global Control.ind (Clear Command, Sync Command, Freeze Command)	DATA-EXCH
167	CHECK-FREEZE	/FREEZE && Freeze Supp = TRUE && Diag.Freeze_Mode = FALSE => B Freeze:= Nil Freeze Command:= 1 Diag.Freeze_Mode:= FALSE Diag Data:= Diag Act_Ref:= Act_Ref + 1 DMPMS Slave Diag Upd.req (Diag Data, Reference:= Act_Ref) MSCY1S Global Control.ind (Clear Command, Sync Command, Freeze Command)	DATA-EXCH
168	CHECK-FREEZE	/FREEZE && Freeze Supp = TRUE && Diag.Freeze_Mode = TRUE && B Freeze <> Nil => Inp Data:= B Freeze B Freeze:= Nil Freeze Command:= 1 DMPMS Data Exchange Upd.req (Diag Flag, Inp Data) DMPMS RD Inp Upd.req (Inp Data) MSCY1S Global Control.ind (Clear Command, Sync Command, Freeze Command)	DATA-EXCH
169	CHECK-FREEZE	/FREEZE && Freeze Supp = TRUE && Diag.Freeze_Mode = TRUE && B Freeze = Nil => Freeze Command:= 1 MSCY1S Global Control.ind (Clear Command, Sync Command, Freeze Command)	DATA-EXCH
170	any state	MSCY1S Reset.req () => DMPMS Slave Deact.req ()	SL-DEACT
171	any state	DMPMS Set Ext Prm.ind(Req_Add, Ext Prm Data) /Diag.Master_Add <> Req_Add => ignore	SAME
172	SL-DEACT	DMPMS Slave Deact.cnf () => MSCY1S Reset.cnf ()	

9.1.4 Functions

Table 65 contains the functions used by the MSCY1S and their descriptions.

Table 65 – Functions used by the MSCY1S

Name	Function
LEAVE-MASTER	Diag.Master_Add=invalid Diag.Sync_Mode=FALSE Diag.Freeze_Mode=FALSE Diag.Prm_Req=TRUE Diag.Station_Not_Ready=TRUE StopTimer(WD), Diag.WD_On=FALSE Diag.Stat_Diag=FALSE, Operation_Mode=V0 Safe_State=TRUE STOP_ASM, STOP_C1, STOP_ISOM DMPMS_Leave.Req Diag_Data=Diag
PRM_OK	Prm_Data.len>=7 && Lock_Req=TRUE && Unlock_Req=FALSE && Ident_Number=Diag.Ident_Number && (WD_On=FALSE || (WD_On=TRUE && WD_Fact_1>0 && WD_Fact_2>0)) && (Freeze_Req=FALSE || Freeze_Supported) && (Sync_Req=FALSE || Sync_Supported) && (Prm_Data[1].0, .1, .2=FALSE) && OPERATION_MODE_OK

SET_WD	if (WD_On=TRUE)
	StartTimer(WD),
	Diag.WD_On=TRUE)
	else
	StopTimer(WD),
	Diag.WD_On=FALSE
	endif

TRIG_WD	if (Diag.WD_On=TRUE)
	StartTimer(WD)
	endif

NOSYNC	Control_Command.Sync=0 && Control_Command.UnSync=0
SYNC	Control_Command.Sync=1 && Control_Command.UnSync=0
UNSYNC	Control_Command.UnSync=1
NOFREEZE	Control_Command.Freeze=0 && Control_Command.UnFreeze=0
FREEZE	Control_Command.Freeze=1 && Control_Command.UnFreeze=0
UNFREEZE	Control_Command.UnFreeze=1

STOP_ISOM	if(Isochronous Mode Supp = TRUE
	&& Isochronous Mode =TRUE
	&& First Synch = FALSE)
	MSCY1S SYNCH Event.ind(Status:= IsoM Stop)
	First Synch = TRUE
	endif

STOP_ASM	if (DX_Entered=TRUE)
	MSCY1S_Stop.ind
	endif

CHECK_ALARM_START	if (DX_Entered=FALSE && Diag.Stat_Diag=FALSE)
	DX_Entered=TRUE,
	MSCY1S_Start.ind(Actual_Enabled_Alarms, Alarm_Sequence, Alarm_Limit)
	endif

SET_OPERATION_MODE	if (Prm_Data.len >= 10 && DPV1_Enable=TRUE)
	Operation_Mode=V1
	else
	Operation_Mode=V0
	endif
	if (Prm_Data.len >= 10 && IsoM_Req =TRUE)
	Isochron_Mode=True
	else
	Isochron_Mode=FALSE
	endif

Name	Function
PV0V0	Operation_Mode = V0 && (Prm_Data.len < 10 || DPV1_Enable = FALSE)
OPERATION_MODE_OK	if (IsARexistent(MS1) && DPV1_Enable) further checks according the table “Rules for DPV1_Status_1, DPV1_Status_2, and DPV1_Status_3 check”
NOPRMCMD	!(DPV1_Status_3.PrmCmd = 1 && Prm_Data.len = 18 && PrmCmd_supp)

SET_ALARM_CHKCFGM	if (Prm_Data.len >=10 && DPV1_Enable=TRUE)
	Enabled_Alarms=DPV1_Status_2.2 – DPV1_Status_2.7
	else
	Enabled_Alarms=0
	endif
	if (Prm_Data.len >=10)
	Alarm_Mode_Master=DPV1_Status_3.0 – DPV1_Status_3.2,
	Chk Cfg Mode = DPV1_Status_2.0,
	Prm Structure = DPV1_Status_3.4,
	if (DPV1_Status3.PrmCmd&& Prm_Data.PrmCmd exist)
	EXE_PRM_CMD
	else
	Alarm_Mode_Master=0,
	Chk Cfg Mode = FALSE,
	Prm Structure = FALSE
	endif

EXE_PRM_CMD	if (PrmCmd.Function.Primary)
	Primary=TRUE,
	if(Start_State=B||Start_State=BEB||Start_State=EB||Start_State=Run)
	STOP_C1, START_C1
	else
	START_C1
	endif
	else
	Primary=FALSE
	endif
	if (PrmCmd.Function.Stop_MS1)
	if(Start_State=B||Start_State=BEB||Start_State=EB||Start_State=Run)
	STOP_C1
	endif
	endif
	if (PrmCmd.Function.Start_MS1)
	if(Start_State=E||Start_State=BE||Start_State=Idle)
	START_C1
	endif
	endif

START_C1	if (Start_State=Idle)
	MSAC1S_Start.req(Diag.Master_Add),
	Start_State=B
	endif
	if (Start_State=E)
	Start_State=EB
	endif
	if (Start_State=BE)
	Start_State=BEB
	endif

STOP_C1	if (Start_State=B)
	Start_State=BE
	endif
	if (Start_State=Run)
	MSAC1S_Stop.req,
	Start_State=E
	endif
	if (Start_State=EB)
	Start_State=E
	endif
	if (Start_State=BEB)
	Start_State=BE
	endif

Name	Function

START_C1_CON	if (Start_State=B)
	  Start_State=Run
	endif
	if (Start_State=BE)
	  MSAC1S_Stop.req, Start_State=E
	endif
	if (Start_State=BEB)
	  MSAC1S_Stop.req, Start_State=EB
	endif

STOP_C1_CON	if (Start_State=E)
	  Start_State=Idle
	endif
	if (Start_State=EB)
	  MSAC1S_Start.req(Diag.Master_Add),
	  Start_State=B
	endif

LEAVE-MASTER	Diag.Master_Add=invalid Diag.Sync_Mode=FALSE Diag.Freeze_Mode=FALSE Diag.Prm_Req=TRUE Diag.Station_Not_Ready=TRUE StopTimer(WD), Diag.WD_On=FALSE Diag.Stat_Diag=FALSE, Operation_Mode=V0 Safe_State=TRUE STOP_ASM, STOP_C1, STOP_ISOM DMPMS_Leave.Req Diag_Data=Diag

9.2 MSAC1S

9.2.1 Primitive definitions

9.2.1.1 Primitives exchanged between MSAC1S and FSPMS

Table 66 shows the service primitives including their associated parameters issued by the FSPMS and received by the MSAC1S.

Table 66 – Primitives issued by FSPMS to MSAC1S

Primitive name	Source	Associated parameters	Functions
SInit MS1.req	FSPMS	(none)	Refer to FAL Service Definition in IEC 61158-5-3
Reset.req	FSPMS	(none)	
Read.rsp(+)	FSPMS	Length, Data	
Read.rsp(-)	FSPMS	Error Decode, Error Code 1 Error Code 2	
Write.rsp(+)	FSPMS	Length	
Write.rsp(-)	FSPMS	Error Decode, Error Code 1 Error Code 2	
Alarm Ack.rsp(+)	FSPMS	Slot Number, Alarm Type, Seq Nr	
Alarm Ack.rsp(-)	FSPMS	(none)	

Table 67 shows the service primitives including their associated parameters issued by MSAC1S and received by the FSPMS.

Table 67 – Primitives issued by MSAC1S to FSPMS

Primitive name	Source	Associated parameters	Functions
SInit MS1.cnf	MSAC1S	(none)	Refer to FAL Service Definition in IEC 61158-5-3
Reset.cnf	MSAC1S	(none)	
Fault.ind	MSAC1S	(none)	
Read.ind	MSAC1S	Slot Number, Index, Length	
Write.ind	MSAC1S	Slot Number, Index, Length, Data	
Alarm Ack.ind	MSAC1S	Slot Number, Alarm Type, Seq Nr	

9.2.1.2 Primitives exchanged between MSAC1S and MSCY1S

Table 68 shows the service primitives including their associated parameters issued by MSCY1S and received by the MSAC1S.

Table 68 – Primitives issued by MSCY1S to MSAC1S

Primitive name	Source	Associated parameters	Functions
Start.req	MSCY1S	Master Add	Start Processing of acyclic services
Stop.req	MSCY1S	(none)	Stop Processing of acyclic services

Table 69 shows the service primitives including their associated parameters issued by the MSAC1S and received by the MSCY1S.

Table 69 – Primitives issued by MSAC1S to MSCY1S

Primitive name	Source	Associated parameters	Functions
Start.cnf	MSAC1S	(none)	MSAC1 is ready to execute acyclic services
Stop.cnf	MSAC1S	(none)	MSAC1 is disabled
Abort.ind	MSAC1S	(none)	Termination of MSAC1 due to a sequence error or wrong protocol elements

9.2.1.3 Parameter of MSAC1S primitives

The parameters used with the primitives exchanged between the FSPMS and the MSAC1S are described in FAL Service Definition in IEC 61158-5-3. Additional parameters are described in Table 70.

Table 70 – Parameter used with primitives exchanged between MSAC1S and MSCY1S

Parameter name	Description
MasterAdd	This parameter conveys the DL address of the assigned DP-master.

9.2.2 State machine description

After Power-On the MSAC1S waits for initiation by means of SInit MS1.

In the state CLOSED the service Start called from the MSCY1S is expected. Then the access protection for the SAP 51 (50) will be activated by means of the function RSAP_ACTIVATE.

When receiving the confirmation the State Machine enters the OPEN state. In this state the services Read, Write and Alarm_Ack are allowed. Only one request from the DP-master is processed simultaneously on each SAP. Additionally to Read/Write at SAP 51 a Alarm_Ack may be executed at SAP 50.

When a valid service is invoked by the DP-master, the MSAC1S will indicate this primitive to the AP-Context (Read/Write or Alarm_Ack) and waits for response. After receiving the corresponding response MSAC1S will provide this PDU in the corresponding SAP update-buffer. The service is finished after DP-master has fetched the response.

The local service Stop called from the MSCY1S is closing the connection. The SAP 51 (50) is deactivated and a possibly stored response in the DL is deleted.

The abort of the connection is indicated to the MSCY1S with the local service Abort. If the User is still processing a response while the connection is aborted this response shall be discarded by the User.

Local variables of the MSAC1S

Server_SAP
(Unsigned8)

This SAP (51) is used for the MSAC1S_Read/_Write and MSAC1S_Alarm_Ack services.

Alarm_SAP
(Unsigned8)

This SAP (50) is reserved for MSAC1S_Alarm_Ack services.

Res_SAP
(Unsigned8)

Variable holding the SAP for the MSAC1S_Alarm_Ack -Response.

Service_Header
(Unsigned8)

The actual service header of the service processed. Used to check whether the User generates the correct response.

AA_State
MSAC1S_Alarm_Ack services may be executed simultaneously with the MSAC1S_Read-/_Write-services. Thus, the actual state of the MSAC1S_Alarm_Ack processing is stored in the variable AA_State:

Values of AA_State:

Idle No MSAC1S_Alarm_Ack in progress

WRes MSAC1S_Alarm_Ack.ind is sent to the MSAL1S waiting for response

Upd Response of the MSAL1S will be sent to the Update-Buffer

SRes Response is stored – waiting for the next poll of the DP-master

VS_Upd
Indicates that a Response will be sent to the Update-Buffer.

9.2.3 MSAC1S state table

Table 71 contains the complete description of the MSAC1S state machine.

Table 71 – MSAC1S state table

#	Current state	Event / condition => action	Next state
1	POWER-ON	MSAC1S_SInit_MS1.req => Server_SAP:=51 Alarm_SAP:=50 MSAC1S_SInit_MS1.cnf	CLOSED
2	POWER-ON	MSAC1S_Reset.req => MSAC1S_Reset.cnf	POWER-ON
3	POWER-ON	MSAC1S_Alarm_Ack.rsp(+) (Alarm_Type, Slot_Number, Seq_Nr) => MSAC1S_Fault.ind	POWER-ON
4	POWER-ON	MSAC1S_Alarm_Ack.rsp(-) () => MSAC1S_Fault.ind	POWER-ON
5	CLOSED	MSAC1S_Start.req(Master_Add) => DMPMS_RSAP_ACTIVATE.req(SSAP=Server_SAP, Access=Master_Add, L_sdu_length_list=Nil,Indication_Mode=Unchanged)	SET-ACC51
6	CLOSED	MSAC1S_Stop.req => MSAC1S_Stop.cnf	CLOSED
7	SET-ACC51	DMPMS_RSAP_ACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status = OK => DMPMS_RSAP_ACTIVATE.req(SSAP=Alarm_SAP, Access=Master_Add, L_sdu_length_list=Nil,Indication_Mode=Unchanged)	SET-ACC50
8	SET-ACC51	DMPMS_RSAP_ACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status=NO/IV => MSAC1S_Fault.ind	POWER-ON
9	SET-ACC50	DMPMS_RSAP_ACTIVATE.cnf(SSAP=Alarm_SAP,M_status) /M_status = OK => AA_State:=Idle VS_Upd:=False MSAC1S_Start.cnf	OPEN
10	SET-ACC50	DMPMS_RSAP_ACTIVATE.cnf(SSAP=Alarm_SAP,M_status) /M_status=NO/IV => MSAC1S_Fault.ind	POWER-ON
11	OPEN	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /L_sdu.len=0 && Update_status=NO => MSAC1S_Fault.ind	POWER-ON

#	Current state	Event / condition => action	Next state
12	OPEN	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /L_sdu.len=0 && Update_status=NO => MSAC1S_Fault.ind	POWER-ON
13	OPEN	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Read-REQ-PDU( ) => Service_Header:=Function_Num MSAC1S_Read.ind(Req_Add=Loc_add, Slot_Number,Index,Length)	VS-WRES
14	OPEN	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Write-REQ-PDU( ) && L_sdu.len >= (Length+4) => Service_Header:=Function_Num MSAC1S_Write.ind(Req_Add=Loc_add, Slot_Number,Index,Length,Data)	VS-WRES
15	OPEN	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Alarm_Ack-REQ-PDU( ) => AA_State:=WRes Res_SAP:=Server_SAP MSAC1S_Alarm_Ack.ind(Alarm_Type, Slot_Number,Seq_Nr)	AA-WRES
16	OPEN	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Alarm_Ack-REQ-PDU( ) => AA_State:=WRes Res_SAP:=Alarm_SAP MSAC1S_Alarm_Ack.ind(Alarm_Type, Slot_Number,Seq_Nr)	AA-WRES
17	OPEN	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /NOT(Read-REQ-PDU( ) || Write-REQ-PDU( ) && (L_sdu.len >= Length+4) || Alarm_Ack-REQ-PDU( )) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
18	OPEN	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /NOT(Alarm_Ack-REQ-PDU( )) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
19	OPEN	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP,Serv_class,L_status) => MSAC1S_Fault.ind	POWER-ON
20	OPEN	MSAC1S_Read.rsp(+)(Length, Data) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
21	OPEN	MSAC1S_Read.rsp(-)(Error_Decode, Error_Code_1, Error_Code_2) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
22	OPEN	MSAC1S_Write.rsp(+)(Length) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
23	OPEN	MSAC1S_Write.rsp(-)(Error_Decode, Error_Code_1, Error_Code_2) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51

#	Current state	Event / condition => action	Next state
24	OPEN	MSAC1S_Alarm_Ack.rsp(+) (Alarm_Type, Slot_Number, Seq_Nr) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
25	OPEN	MSAC1S_Alarm_Ack.rsp(-) () => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
26	OPEN	MSAC1S_Stop.req => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP)	STOP-DEA51
27	OPEN	MSAC1S_Start.req(Master_Add) => MSAC1S_Fault.ind	POWER-ON
28	OPEN	MSAC1S_SInit_MS1.req => MSAC1S_Fault.ind	POWER-ON
29	OPEN	MSAC1S_Reset.req => MSAC1S_Reset.cnf	POWER-ON
30	VS-WRES	MSAC1S_Read.rsp(+)(Length, Data) /Service_Header=0x5E && Max_DLSDU_Length_Req_Low>=(Length+4) => VS_Upd:=True L_sdu:=Read-RES-PDU DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low,Transmit=Single)	VS-SRES
31	VS-WRES	MSAC1S_Read.rsp(+)(Length, Data) /Service_Header=0x5E && Max_DLSDU_Length_Req_Low<(Length+4) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
32	VS-WRES	MSAC1S_Write.rsp(+)(Length) /Service_Header=0x5F => VS_Upd:=True L_sdu:=Write-RES-PDU DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low,Transmit=Single)	VS-SRES
33	VS-WRES	MSAC1S_Read.rsp(-)(Error_Decode, Error_Code_1, Error_Code_2) /Service_Header=0x5E => VS_Upd:=True L_sdu:=Read-NRS-PDU DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low,Transmit=Single)	VS-SRES
34	VS-WRES	MSAC1S_Write.rsp(-)(Error_Decode, Error_Code_1, Error_Code_2) /Service_Header=0x5F => VS_Upd:=True L_sdu:=Write-NRS-PDU DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low,Transmit=Single)	VS-SRES
35	VS-WRES	MSAC1S_Read.rsp(+)(Length, Data) /Service_Header<>0x5E => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
36	VS-WRES	MSAC1S_Read.rsp(-)(Error_Decode, Error_Code_1, Error_Code_2) /Service_Header<>0x5E => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51

#	Current state	Event / condition => action	Next state
37	VS-WRES	MSAC1S_Write.rsp(+)(Length) /Service_Header<>0x5F => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
38	VS-WRES	MSAC1S_Write.rsp(-)(Error_Decode, Error_Code_1, Error_Code_2) /Service_Header<>0x5F => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
39	VS-WRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /L_sdu.len=0 && Update_status=NO => MSAC1S_Fault.ind	POWER-ON
40	VS-WRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /L_sdu.len=0 && Update_status=NO => MSAC1S_Fault.ind	POWER-ON
41	VS-WRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
42	VS-WRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Alarm_Ack-REQ-PDU( ) && AA_State=Idle => AA_State:=WRes Res_SAP:=Alarm_SAP MSAC1S_Alarm_Ack.ind(Alarm_Type, Slot_Number,Seq_Nr)	VS-WRES
43	VS-WRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Update_status=LO && L_sdu.len=0 && AA_State=SRes => AA_State:=Idle	VS-WRES
44	VS-WRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /NOT((Alarm_Ack-REQ-PDU( ) && AA_State=Idle) || (Update_status=LO && L_sdu.len=0 && AA_State=SRes)) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
45	VS-WRES	MSAC1S_Start.req(Master_Add) => MSAC1S_Fault.ind	POWER-ON
46	VS-WRES	MSAC1S_Stop.req => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP)	STOP-DEA51
47	VS-WRES	MSAC1S_Alarm_Ack.rsp(+) (Alarm_Type, Slot_Number, Seq_Nr) /AA_State<>WRes => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
48	VS-WRES	MSAC1S_Alarm_Ack.rsp(-) () /AA_State<>WRes => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51

#	Current state	Event / condition => action	Next state
49	VS-WRES	MSAC1S_Alarm_Ack.rsp(+) (Alarm_Type, Slot_Number, Seq_Nr) /AA_State=WRes => AA_State:=Upd DMPMS_REPLY_UPDATE.req(SSAP=Res_SAP, L_sdu=Alarm_Ack-RES-PDU, Serv_class=Low,Transmit=Single)	VS-WRES
50	VS-WRES	MSAC1S_Alarm_Ack.rsp(-) () /AA_State=WRes => AA_State:=Upd DMPMS_REPLY_UPDATE.req(SSAP=Res_SAP, L_sdu=Alarm_Ack-NRS-PDU, Serv_class=Low,Transmit=Single)	VS-WRES
51	VS-WRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Alarm_SAP,Serv_class,L_status) /L-status=OK && AA_State=Upd => AA_State:=SRes	VS-WRES
52	VS-WRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Alarm_SAP,Serv_class,L_status) /L-status=OK && AA_State<>Upd => MSAC1S_Fault.ind	POWER-ON
53	VS-WRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP,Serv_class,L_status) /L_status=LS/IV/LR => MSAC1S_Fault.ind	POWER-ON
54	VS-WRES	MSAC1S_SInit_MS1.req => MSAC1S_Fault.ind	POWER-ON
55	VS-WRES	MSAC1S_Reset.req => MSAC1S_Reset.cnf	POWER-ON
56	VS-SRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP,Serv_class,L_status) /L-status=OK && VS_Upd=True => VS_Upd:=False	VS-SRES
57	VS-SRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP,Serv_class,L_status) /L_status=LS/IV/LR => MSAC1S_Fault.ind	POWER-ON
58	VS-SRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP,Serv_class,L_status) /L-status=OK && VS_Upd=False => MSAC1S_Fault.ind	POWER-ON
59	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /L_sdu.len=0 && Update_status=NO => MSAC1S_Fault.ind	POWER-ON
60	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /L_sdu.len=0 && Update_status=NO => MSAC1S_Fault.ind	POWER-ON
61	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Update_status=LO && L_sdu.len=0 && AA_State=Idle&& VS_Upd=False	OPEN
62	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Update_status=LO && L_sdu.len=0 && AA_State=WRes && VS_Upd=False	AA-WRES
63	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Update_status=LO && L_sdu.len=0 && (AA_State=SRes|| AA_State=Upd) && VS_Upd=False	AA-SRES

#	Current state	Event / condition => action	Next state
64	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /L_sdu.len=0 && VS_Upd=True => MSAC1S_Fault.ind	POWER-ON
65	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /L_sdu.len<>0 => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
66	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Alarm_Ack-REQ-PDU( ) && AA_State=Idle => AA_State:=WRes Res_SAP:=Alarm_SAP MSAC1S_Alarm_Ack.ind(Alarm_Type, Slot_Number,Seq_Nr)	VS-SRES
67	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Update_status=LO && L_sdu.len=0 && AA_State=SRes => AA_State:=Idle	VS-SRES
68	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /NOT((Alarm_Ack-REQ-PDU( ) && AA_State=Idle) || (Update_status=LO && L_sdu.len=0 && AA_State=SRes)) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
69	VS-SRES	MSAC1S_Start.req(Master_Add) => MSAC1S_Fault.ind	POWER-ON
70	VS-SRES	MSAC1S_Stop.req => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP)	STOP-DEA51
71	VS-SRES	MSAC1S_Read.rsp(+)(Length, Data) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
72	VS-SRES	MSAC1S_Read.rsp(-)(Error_Decode, Error_Code_1, Error_Code_2) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
73	VS-SRES	MSAC1S_Write.rsp(+)(Length) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
74	VS-SRES	MSAC1S_Write.rsp(-)(Error_Decode, Error_Code_1, Error_Code_2) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
75	VS-SRES	MSAC1S_Alarm_Ack.rsp(+) (Alarm_Type, Slot_Number, Seq_Nr) /AA_State<>WRes => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
76	VS-SRES	MSAC1S_Alarm_Ack.rsp(-) () /AA_State<>WRes => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51

#	Current state	Event / condition => action	Next state
77	VS-SRES	MSAC1S_Alarm_Ack.rsp(+) (Alarm_Type, Slot_Number, Seq_Nr) /AA_State=WRes => AA_State:=Upd DMPMS_REPLY_UPDATE.req(SSAP=Res_SAP, L_sdu=Alarm_Ack-RES-PDU, Serv_class=Low,Transmit=Single)	VS-SRES
78	VS-SRES	MSAC1S_Alarm_Ack.rsp(-) () /AA_State=WRes => AA_State:=Upd DMPMS_REPLY_UPDATE.req(SSAP=Res_SAP, L_sdu=Alarm_Ack-NRS-PDU, Serv_class=Low,Transmit=Single)	VS-SRES
79	VS-SRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Alarm_SAP,Serv_class,L_status) /L-status=OK && AA_State=Upd => AA_State:=SRes	VS-SRES
80	VS-SRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Alarm_SAP,Serv_class,L_status) /L-status=OK && AA_State<>Upd => MSAC1S_Fault.ind	POWER-ON
81	VS-SRES	MSAC1S_SInit_MS1.req => MSAC1S_Fault.ind	POWER-ON
82	VS-SRES	MSAC1S_Reset.req => MSAC1S_Reset.cnf	POWER-ON
83	AA-WRES	MSAC1S_Alarm_Ack.rsp(+) (Alarm_Type, Slot_Number, Seq_Nr) => AA_State:=Upd L_sdu:=Alarm_Ack-RES-PDU DMPMS_REPLY_UPDATE.req(SSAP=Res_SAP, L_sdu, Serv_class=Low,Transmit=Single)	AA-SRES
84	AA-WRES	MSAC1S_Alarm_Ack.rsp(-) () => AA_State:=Upd L_sdu:=Alarm_Ack-NRS-PDU DMPMS_REPLY_UPDATE.req(SSAP=Res_SAP, L_sdu, Serv_class=Low,Transmit=Single)	AA-SRES
85	AA-WRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP,Serv_class,L_status) => MSAC1S_Fault.ind	POWER-ON
86	AA-WRES	MSAC1S_Read.rsp(+)(Length, Data) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
87	AA-WRES	MSAC1S_Read.rsp(-)(Error_Decode, Error_Code_1, Error_Code_2) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
88	AA-WRES	MSAC1S_Write.rsp(+)(Length) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
89	AA-WRES	MSAC1S_Write.rsp(-)(Error_Decode, Error_Code_1, Error_Code_2) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
90	AA-WRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /L_sdu.len=0 && Update_status=NO => MSAC1S_Fault.ind	POWER-ON

#	Current state	Event / condition => action	Next state
91	AA-WRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /L_sdu.len=0 && Update_status=NO => MSAC1S_Fault.ind	POWER-ON
92	AA-WRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
93	AA-WRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Read-REQ-PDU( ) && Res_SAP=Alarm_SAP => Service_Header:=Function_Num MSAC1S_Read.ind(Req_Add=Loc_add, Slot_Number,Index,Length)	VS-WRES
94	AA-WRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Write-REQ-PDU( ) && Res_SAP=Alarm_SAP => Service_Header:=Function_Num MSAC1S_Write.ind(Req_Add=Loc_add, Slot_Number,Index,Length,Data)	VS-WRES
95	AA-WRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /NOT(Write-REQ-PDU( ) || Read-REQ-PDU( )) || Res_SAP=Server_SAP => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
96	AA-WRES	MSAC1S_Start.req(Master_Add) => MSAC1S_Fault.ind	POWER-ON
97	AA-WRES	MSAC1S_Stop.req => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP)	STOP-DEA51
98	AA-WRES	MSAC1S_Reset.req => MSAC1S_Reset.cnf	POWER-ON
99	AA-WRES	MSAC1S_SInit_MS1.req => MSAC1S_Fault.ind	POWER-ON
100	AA-SRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Alarm_SAP,Serv_class,L_status) /L-status=OK && AA_State=Upd && Res_SAP=Alarm_SAP => AA_State:=SRes	AA-SRES
101	AA-SRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP,Serv_class,L_status) /L_status=LS/IV/LR => MSAC1S_Fault.ind	POWER-ON
102	AA-SRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Alarm_SAP,Serv_class,L_status) /L-status=OK &&(AA_State<>Upd || Res_SAP<>Alarm_SAP) => MSAC1S_Fault.ind	POWER-ON
103	AA-SRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP,Serv_class,L_status) /L-status=OK && AA_State=Upd && Res_SAP=Server_SAP => AA_State:=SRes	AA-SRES
104	AA-SRES	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP,Serv_class,L_status) /L-status=OK && (AA_State<>Upd || Res_SAP<>Server_SAP) => MSAC1S_Fault.ind	POWER-ON

#	Current state	Event / condition => action	Next state
105	AA-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /L_sdu.len=0 && Update_status=NO => MSAC1S_Fault.ind	POWER-ON
106	AA-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /L_sdu.len=0 && Update_status=NO => MSAC1S_Fault.ind	POWER-ON
107	AA-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Update_status=LO && L_sdu.len=0 && AA_State=SRes => AA_State:=Idle	OPEN
108	AA-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Alarm_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /(Update_status<>LO || AA_State<>SRes) && Res_SAP=Alarm_SAP => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
109	AA-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Update_status=LO && L_sdu.len=0 && AA_State=SRes => AA_State:=Idle	OPEN
110	AA-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /(Update_status<>LO || L_sdu.len<>0 || AA_State<>SRes) && Res_SAP=Server => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
111	AA-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Read-REQ-PDU( ) && RES_SAP=ALARM_SAP => Service_Header:=Function_Num MSAC1S_Read.ind(Req_Add=Loc_add, Slot_Number,Index,Length)	VS-WRES
112	AA-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /Write-REQ-PDU( ) && RES_SAP=ALARM_SAP => Service_Header:=Function_Num MSAC1S_Write.ind(Req_Add=Loc_add, Slot_Number,Index,Length,Data)	VS-WRES
113	AA-SRES	DMPMS_DATA_REPLY.ind(SSAP,DSAP=Server_SAP,Loc_add,Rem_add,L_sdu,Serv_class, Update_status) /NOT(Write-REQ-PDU( ) || Read-REQ-PDU( )) && RES_SAP=ALARM_SAP => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
114	AA-SRES	MSAC1S_Start.req(Master_Add) => MSAC1S_Fault.ind	POWER-ON
115	AA-SRES	MSAC1S_Stop.req => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP)	STOP-DEA51
116	AA-SRES	MSAC1S_Read.rsp(+)(Length, Data) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
117	AA-SRES	MSAC1S_Read.rsp(-)(Error_Decode, Error_Code_1, Error_Code_2) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51

#	Current state	Event / condition => action	Next state
118	AA-SRES	MSAC1S_Write.rsp(+)(Length) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
119	AA-SRES	MSAC1S_Write.rsp(-)(Error_Decode, Error_Code_1, Error_Code_2) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
120	AA-SRES	MSAC1S_Alarm_Ack.rsp (Alarm_Type, Slot_Number, Seq_Nr) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
121	AA-SRES	MSAC1S_Alarm_Ack.rsp (Alarm_Type, Slot_Number, Seq_Nr) => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC1S_Abort.ind	ERR-DEA51
122	AA-SRES	MSAC1S_SInit_MS1.req => MSAC1S_Fault.ind	POWER-ON
123	AA-SRES	MSAC1S_Reset.req => MSAC1S_Reset.cnf	POWER-ON
124	STOP-DEA51	DMPMS_SAP_DEACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status = OK => DMPMS_SAP_DEACTIVATE.req(SSAP=Alarm_SAP)	STOP-DEA50
125	STOP-DEA51	DMPMS_SAP_DEACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status=NO/IV => MSAC1S_Fault.ind	POWER-ON
126	STOP-DEA50	DMPMS_SAP_DEACTIVATE.cnf(SSAP=Alarm_SAP,M_status) /M_status = OK => MSAC1S_Stop.cnf	CLOSED
127	STOP-DEA50	DMPMS_SAP_DEACTIVATE.cnf(SSAP=Alarm_SAP,M_status) /M_status=NO/IV => MSAC1S_Fault.ind	POWER-ON
128	ERR-DEA51	DMPMS_SAP_DEACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status = OK => DMPMS_SAP_DEACTIVATE.req(SSAP=Alarm_SAP)	ERR-DEA50
129	ERR-DEA51	DMPMS_SAP_DEACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status=NO/IV => MSAC1S_Fault.ind	POWER-ON
130	ERR-DEA50	DMPMS_SAP_DEACTIVATE.cnf(SSAP=Alarm_SAP,M_status) /M_status = OK	CLOSED
131	ERR-DEA50	DMPMS_SAP_DEACTIVATE.cnf(SSAP=Alarm_SAP,M_status) /M_status=NO/IV => MSAC1S_Fault.ind	POWER-ON

9.2.4 Functions

Table 72 contains the functions used by the MSAC1S, their arguments and their descriptions.

Table 72 – Functions used by the MSAC1S

Function name	Description
Read-REQ-PDU ( )	This function will return TRUE if the corresponding L_sdu is a valid Read-REQ-PDU. (L_sdu.len >= 4 && L_sdu[1] = 0x5E)
Write-REQ-PDU ( )	This function will return TRUE if the corresponding L_sdu is a valid Write-REQ-PDU. (L_sdu.len >= 4 && L_sdu[1] = 0x5F)
Alarm_Ack-REQ-PDU ( )	This function will return TRUE if the corresponding L_sdu is a valid Alarm_Ack-REQ-PDU. (L_sdu.len >= 4 && L_sdu[1] = 0x5C)

9.3 SSCY1S

9.3.1 Primitive definitions

9.3.1.1 Primitives exchanged between SSCY1S and FSPMS

Table 73 shows the service primitives including their associated parameters issued by the FSPMS and received by the SSCY1S.

Table 73 – Primitives issued by FSPMS to SSCY1S

Primitive name	Source	Associated parameters	Functions
Get Publisher Data.req	FSPMS	Rem_Add	Refer to FAL Service Definition in IEC 61158-5-3
Start Subscriber.req	FSPMS	Rem_Add	
Stop Subscriber.req	FSPMS	Rem_Add	

Table 74 shows the service primitives including their associated parameters issued by the SSCY1S and received by the FSPMS.

Table 74 – Primitives issued by SSCY1S to FSPMS

Primitive name	Source	Associated parameters	Functions
Get Publisher Data.cnf(+)	SSCY1S	Rem_Add, Data, New Flag	Refer to FAL Service Definition in IEC 61158-5-3
Get Publisher Data.cnf(-)	SSCY1S	Rem_Add	
Start Subscriber.cnf(+)	SSCY1S	Rem_Add	
Start Subscriber.cnf(-)	SSCY1S	Rem_Add	
Stop Subscriber.cnf(+)	SSCY1S	Rem_Add	
Stop Subscriber.cnf(-)	SSCY1S	Rem_Add	
New Publisher Data.ind	SSCY1S	Rem_Add	
Publisher Active.ind	SSCY1S	Rem_Add, Status	

9.3.1.2 Parameter of SSCY1S primitives

The parameters used with the primitives exchanged between the FSPMS and the SSCY1S are described in FAL Service Definition in IEC 61158-5-3.

9.3.2 State machine description

After Power-On the SSCY1S waits for initiation by means of DMPMS DX Entered.ind with the Status TRUE. It indicates that the MSCY1S is opened for data exchange by the associated DP-master.

In the state W-START the service Start Subscriber called from the FSPMS is expected. When receiving this event the transition initializes all local variables and enters the RUN state. Furthermore, the local monitoring timer with the WD value (MSCY1S) is started.

When receiving a DMPMS DX Broadcast.ind with publisher data in the OPEN state the sample data according to the filter table are stored. Furthermore, the local flags New Data and New Stat will be set to TRUE and a New Publisher Data.ind will be issued to the FSPMS. The application retrieves the sample data with the Get Publisher Data request service primitive.

When the monitoring timer expires the status is checked and the timer will be started again. If the current status (New Stat) differs from status (Status) of the last interval the change is indicated to the application by means of the Publisher Active indication service primitive. Therefore, every change of status from active to passive is reported to the application. The application may stop the subscribing procedure by sending a Stop Subscriber service request that sets the state machine back to W-START.

In any case, when a DMPMS DX Entered .ind with status FALSE (MSCY1S leaves the data exchange) is invoked the state machine is set back to POWER ON.

Local variables of the SSCY1S

Status
(Boolean)

This local variable is TRUE if the publisher was active during the last WD period. That means the data exchange mode was entered, the subscriber was started and at least one DXB had been received during the previous WD interval. Otherwise it is FALSE.

New_Stat
(Boolean)

This local variable is TRUE if the publisher is active. That means the Data Exchange is entered, the subscriber is started and at least one DXB has been received during the current WD interval. Otherwise it is FALSE

New Data
(Boolean)

This variable is TRUE if new data have arrived and are not retrieved by the Get Publisher Data service primitive. Otherwise it is FALSE.

Sample Data
(Octet-String)

The data buffer for publisher data.

9.3.3 SSCY1S state table

Table 75 contains the complete description of the SSCY1S state machine.

Table 75 – SSCY1S state table

#	Current state	Event / condition => action	Next state
1	POWER-ON	SSCY1S_Start_Subscriber.req (Rem_Add) => SSCY1S_Start_Subscriber.cnf (-) (Rem_Add)	POWER-ON
2	POWER-ON	SSCY1S_Stop_Subscriber.req (Rem_Add) => SSCY1S_Stop_Subscriber.cnf (-) (Rem_Add)	POWER-ON
3	POWER-ON	SSCY1S_Get_Publisher_Data.req (Rem_Add) => SSCY1S_Get_Publisher_Data.cnf (-) (Rem_Add)	POWER-ON
4	POWER-ON	DMPMS_DX_Broadcast.ind (Rem_Add, Data) => ignore	POWER-ON
5	POWER-ON	DMPMS_DX_Entered.ind (Rem_Add, Status) /Status =>	W-START
6	W-START	SSCY1S_Start_Subscriber.req (Rem_Add) => SET_WD, New_Stat:= FALSE New_Data:= FALSE, Sample_Data:= 0 SSCY1S_Start_Subscriber.cnf (+) (Rem_Add)	RUN
7	W-START	SSCY1S_Stop_Subscriber.req (Rem_Add) => SSCY1S_Stop_Subscriber.cnf (-) (Rem_Add)	W-START
8	W-START	SSCY1S_Get_Publisher_Data.req (Rem_Add) => SSCY1S_Get_Publisher_Data.cnf (-) (Rem_Add)	W-START
9	W-START	DMPMS_DX_Broadcast.ind (Rem_Add, Data) => ignore	W-START
10	RUN	SSCY1S_Start_Subscriber.req (Rem_Add) => SSCY1S_Start_Subscriber.cnf (-) (Rem_Add)	RUN
11	RUN	SSCY1S_Stop_Subscriber.req (Rem_Add) => StopTimer(WD) SSCY1S_Stop_Subscriber.cnf (+) (Rem_Add)	W-START
12	RUN	SSCY1S_Get_Publisher_Data.req (Rem_Add) => Data:= Sample_Data, New_Flag:= New_Data, New_Data:= FALSE SSCY1S_Get_Publisher_Data.cnf (+) (Rem_Add, Data, New_Flag)	RUN
13	RUN	DMPMS_DX_Broadcast.ind (Rem_Add, Data) => New_Stat:= TRUE, New_Data:= TRUE, Sample_Data:= Filter (Data) SSCY1S_New_Publisher_Data.ind (Rem_Add)	RUN
14	RUN	WDTimer expired /New_Stat => TRIG_WD, New_Stat:= FALSE SSCY1S_Publisher_Active.ind (Rem_Add, Status)	RUN

#	Current state	Event / condition => action	Next state
15	RUN	WDTimer expired /!New_Stat => TRIG_WD, New_Stat:= FALSE	RUN
16	any	DMPMS_DX_Entered.ind (Rem_Add, Status) /!Status => StopTimer(WD), Publisher Address:= 127, Status:= FALSE, New_Stat:= FALSE SSCY1S_Publisher_Active.ind (Rem_Add, Status)	POWER-ON

9.3.4 Functions

Table 76 contains the functions used by the SSCY1S, their arguments and their descriptions.

Table 76 – Functions used by the SSCY1S

Function name	Description
Filter(Data)	This function will return sample data (Octet-String) extracted from Data according to the offset and length from the DXB link attributes.

SET_WD	if (WD Enabled = TRUE)
	  StartTimer (WD)
	else
	  StopTimer (WD)

TRIG_WD	if (WD Enabled = TRUE) StartTimer (WD)

9.4 MSRM2S

9.4.1 Primitive definitions

9.4.1.1 Primitives exchanged between MSRM2S and FSPMS

Table 77 shows the service primitives including their associated parameters issued by the FSPMS and received by the MSRM2S.

Table 77 – Primitives issued by FSPMS to MSRM2S

Primitive name	Source	Associated parameters	Functions
SInit MS2.req	FSPMS	(none)	Refer to FAL Service Definition in IEC 61158-5-3
Reset.req	FSPMS	(none)	

Table 78 shows the service primitives including their associated parameters issued by the MSRM2S and received by the FSPMS.

Table 78 – Primitives issued by MSRM2S to FSPMS

Primitive name	Source	Associated parameters	Functions
SInit MS2.cnf	MSRM2S	(none)	Refer to FAL Service Definition in IEC 61158-5-3
Reset.cnf	MSRM2S	(none)	
Fault.ind	MSRM2S	(none)	

9.4.1.2 Parameter of MSRM2S primitives

There are no parameters associated to the primitives.

9.4.2 State machine description

For acyclic communication the DSAPs are allocated dynamically by the so called Resource Manager (RM) of the DP-slave.

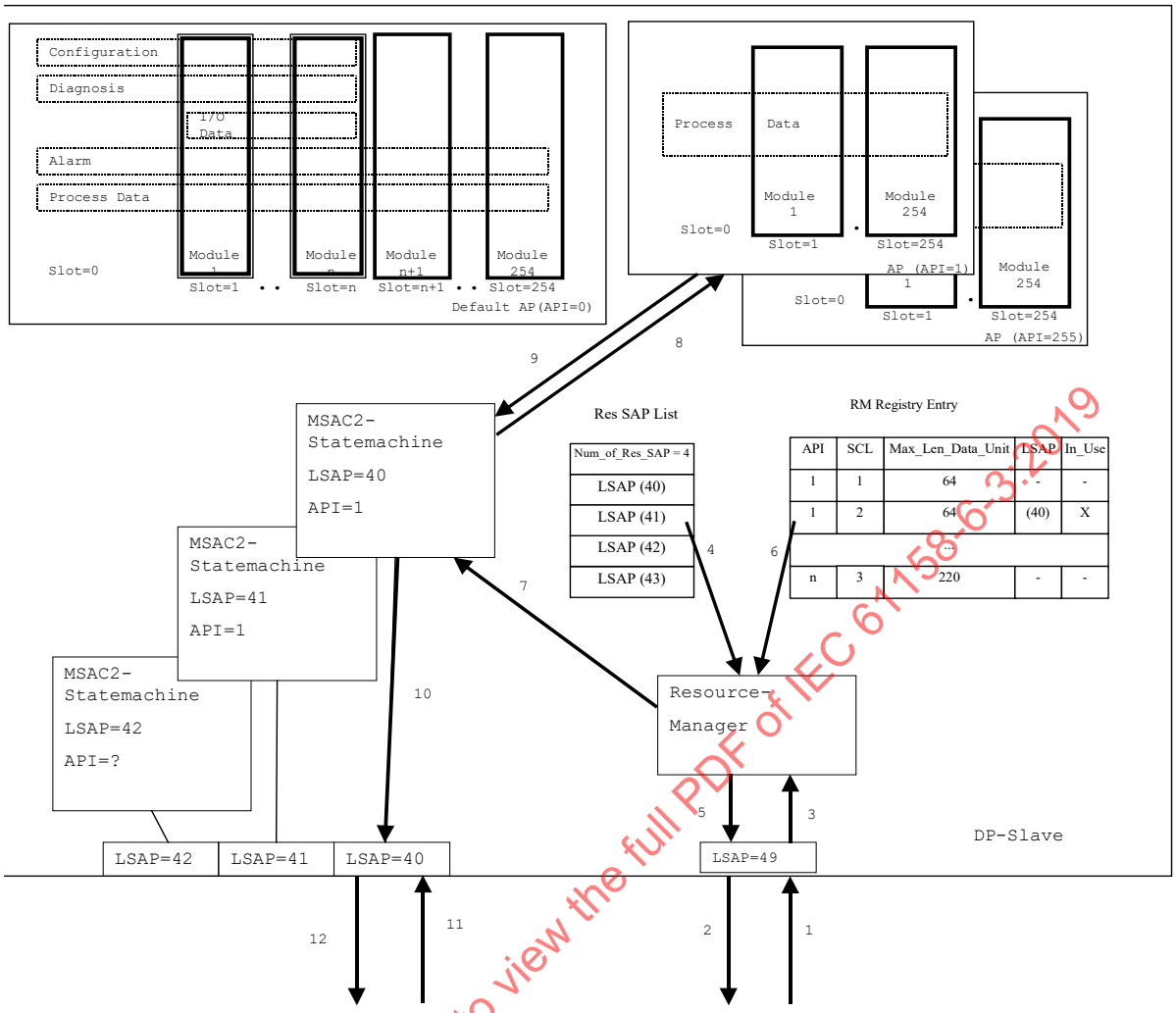
The Resource Manager (RM) co-ordinates the communication resources for MS2 AR of the DP-slave. The main features of the RM are described in the following sequence, see Figure 22.

– The DP-master requests a communication connection by means of the service Initiate (DSAP 49). If multiple application processes are used in the DP-slave it is possible to address these application processes through an Application Process Identifier (API).

– The RM of the DP-slave provides an unused SAP (SAP 0 – 48). The possible SAPs are defined locally in the Res_SAP_List.

– The RM responds the Initiate.req immediately with the RM_REQ-PDU containing the corresponding unused SAP.

– Further steps following the Initiate.req (Initiate.res, Read, Write, Abort) are handled by the corresponding MSAC2S State Machine.



1 Initiate-REQ-PDU (DSAP=49)

2 RM-REQ-PDU (Server_SAP=next SAP not In_Use)

3 Initiate-REQ-PDU

4 Search of the next SAP not In_Use

5 Reply_Update.req(L_sdu=RM-REQ-PDU with next SAP not In_Use)

6 Check RM_Registry_Entry for API, In_Use=FALSE

7 MSAC2S_Initiate.req (Server_SAP=40)

8 MSAC2S_Initiate.ind (Server_SAP=40)

9 MSAC2S_Initiate.res (Server_SAP=40)

10 Reply_Update.req(L_sdu=Initiate-RES-PDU)

11 Polling with (L_sdu.len=0)

12 Initiate-RES-PDU

Figure 22 – Example for connection establishment on MS2(server-side)

The Resource Manager is responsible for the allocation of SAPs to incoming requests for a MS2 AR communication.

If a Res_SAP is available when an Initiate-REQ-PDU is received on the RM_SAP, the Slave responds with a RM-REQ-PDU.

The only PDU which is accepted is an Initiate-REQ-PDU. The Resource Manager checks for an unused API/SCL in the RM_Registry and passes the Initiate-PDU with an Initiate to a MSAC2S State Machine.

If none of the entries in the RM_Registry fits to the requested API/SCL, the last SAP sent in the RM-REQ-PDU is temporarily used to send an Abort-REQ-PDU to the Master.

If no Res_SAP is available then the RM_SAP is deactivated.

If an MSAC2S State Machine indicates with anAbort.req that a connection has been closed, the corresponding entry in the RM_Registry is marked as unused and the corresponding DLSAP is released and can be used for further requests.

Local variables of the MSRM2S

List of RM entries

Figure 23 shows the structure of the RM entries as they are listed in the RM_Registry.

static part			dynamic part	
API	SCL	Max_Len_Data_Unit	SAP	In_Use

Figure 23 – Structure of RM entries in the RM_Registry

List of SSAPs

SSAP_Entry
(Unsigned8)

0..48

Stored_Req_Add,
Stored_Send_Timeout,
Stored_Features_Supported,
Stored_Profile_Features_Supported,
Stored_Profile_Ident_Number,
Stored_Add_Addr_Param
Storage of values of the last Initiate function (see part DP-Services).

RM_SAP
(constant)

Set to 49

Last_SAP
(Unsigned8)

0..48

SAP for previous established connection.

Server_SAP
(Unsigned8)

0..48

SAP which will be allocated to the next connection.

Macros

INITIATE-REQ-PDU-LEN
(

48 {64 for linking device}

)

RMREQ-PDU-LEN
(

4

)

9.4.3 MSRM2S state table

Table 79 contains the complete description of the MSRM2S state machine.

Table 79 – MSRM2S state table

#	Current state	Event / condition => action	Next state
1	POWER-ON	MSRM2S_SInit.req =>RM_SAP:=49 Res_SAP_entry_Index:=1 Server_SAP:=Get_from_List_of_SSAPs() DMPMS_RSAP_ACTIVATE.req(SSAP=RM_SAP, Access=All, L_sdu_length_list=(<INITIATE-REQ-PDU-LEN>,0,<RMREQ-PDU-LEN>,0) Indication_Mode=Data)	WAIT-ACT
2	WAIT-ACT	DMPMS_RSAP_ACTIVATE.cnf(SSAP,M_status) /M_status=OK =>RM_SAP:=49 Res_SAP_entry_Index:=1 Server_SAP:=Get_from_List_of_SSAPs() DMPMS_REPLY_UPDATE.req(SSAP=RM_SAP, L_sdu=RM-REQ-PDU, Serv_class=Low, Transmit=Single)	INIT-WUPD
3	WAIT-ACT	DMPMS_RSAP_ACTIVATE.cnf(SSAP,M_status) /M_status=NO/IV =>MSRM2S_Fault.ind	POWER-ON
4	INIT-WUPD	DMPMS_REPLY_UPDATE.cnf (SSAP=RM_SAP, Serv_class=Low, L_status) /L_status=OK =>MSRM2S_SInit.cnf	OPEN
5	INIT-WUPD	DMPMS_REPLY_UPDATE.cnf (SSAP=RM_SAP, Serv_class=Low, L_status) /L_status= LS/IV/LR =>MSRM2S_Fault.ind	POWER-ON
6	OPEN	DMPMS_DATA_REPLY.ind(SSAP, DSAP=RM_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && L_sdu=Initiate-REQ-PDU && List_of_SSAPs<>empty =>Last_SAP:= Server_SAP Stored_Req_Add:=Loc_add, Stored_Send_Timeout:= Send_Timeout Stored_Features_Supported:=Features_Supported Stored_Profile_Features_Supported:= Profile_Features_Supported Stored_Profile_Ident_Number:=Profile_Ident_Number Stored_Add_Addr_Param:= Add_Addr_Param Server_SAP:=Get_from_List_of_SSAPs() DMPMS_REPLY_UPDATE.req(SSAP=RM_SAP, L_sdu=RM-REQ-PDU, Serv_class=Low, Transmit=Single)	OW-UPDATE

#	Current state	Event / condition => action	Next state
7	OPEN	DMPMS_DATA_REPLY.ind(SSAP, DSAP=RM_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && L_sdu=Initiate-REQ-PDU && List_of_SSAPs=empty =>Stored_Req_Add:=Loc_add, Stored_Send_Timeout:= Send_Timeout Stored_Features_Supported:=Features_Supported Stored_Profile_Features_Supported:= Profile_Features_Supported Stored_Profile_Ident_Number:=Profile_Ident_Number Stored_Add_Addr_Param:= Add_Addr_Param DMPMS_SAP_DEACTIVATE.req(SSAP=RM_SAP)	OW-DEACT
8	OPEN	DMPMS_DATA_REPLY.ind(SSAP, DSAP=RM_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && (L_sdu<>Initiate-REQ-PDU) =>ignore Lsdu DMPMS_REPLY_UPDATE.req(SSAP=RM_SAP, L_sdu=RM-REQ-PDU, Serv_class=Low, Transmit=Single)	INIT-WUPD
9	OPEN	DMPMS_DATA_REPLY.ind(SSAP, DSAP=RM_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=NO =>ignore Lsdu	OPEN
10	OPEN	MSRM2S_Reset.req =>for (all CRL entries with AR-Type=MS2) MSAC2S_Reset.req(Res_SAP=SSAP) endfor	RESET-AC2
11	OPEN	MSAC2S_Closed.ind(Res_SAP) =>Index:= Search_in_List_of_RM_Entries(SAP:=Res_SAP, In_Use:=True)	AW-SEARCH
12	AW-SEARCH	/Index <> 0 =>List_of_RM_Entries.In_Use[Index]:=False Put_to_List_of_SSAPs(Stored_Res_SAP)	OPEN
13	AW-SEARCH	/Index = 0 =>MSRM2S_Fault.ind	POWER-ON
14	OW-UPDATE	DMPMS_REPLY_UPDATE.cnf (SSAP=RM_SAP, Serv_class=Low, L_status) /L_status=OK =>Index:= Search_in_List_of_RM_Entries(API:=Add_Addr_Param.D_Addr.API, SCL:= Add_Addr_Param.D_Addr.SCL, In_Use:=False)	OW-SEARCH
15	OW-UPDATE	DMPMS_REPLY_UPDATE.cnf (SSAP=RM_SAP, Serv_class=Low, L_status) /L_status=LS/IV/LR =>MSRM2S_Fault.ind	POWER-ON
16	OW-SEARCH	/Index <> 0 =>List_of_RM_Entries.In_Use[Index]:=True List_of_RM_Entries.SAP[Index]:=Last_SAP Max_Len_Data_Unit:=List_of_RM_Entries.Max_Len_Data_Unit[Index] Send_Timeout:=Max(Stored_Send_Timeout, Stored_Min_Send_Timeout) MSAC2S_Initiate.req(Res_SAP=Last_SAP, Req_Add=Stored_Req_Add, Max_Len_Data_Unit, Send_Timeout, Features_Supported= Stored_Features_Supported, Profile_Features_Supported= Stored_Profile_Features_Supported, Profile_Ident_Number= Stored_Profile_Ident_Number, Add_Addr_Param= Stored_Add_Addr_Param)	OPEN
17	OW-SEARCH	/Index= 0 =>Send_Timeout:=Max(Stored_Send_Timeout, Stored_Min_Send_Timeout) MSAC2S_Abort.req(Res_SAP=Last_SAP, Req_Add, Subnet=NO, Instance=MSAC2, Reason_Code=ABT_IA, Send_Timeout)	OPEN
18	OW-DEACT	DMPMS_SAP_DEACTIVATE.cnf(SSAP,M_status) /M_status=OK =>Index:= Search_in_List_of_RM_Entries(API:=Add_Addr_Param.D_Addr.API, SCL:= Add_Addr_Param.D_Addr.SCL, In_Use:=False)	CW-SEARCH
19	OW-DEACT	DMPMS_SAP_DEACTIVATE.cnf(SSAP,M_status) /M_status=NO/IV =>MSRM2S_Fault.ind	POWER-ON

#	Current state	Event / condition => action	Next state
20	CW-SEARCH	/Index <> 0 =>List_of_RM_Entries.In_Use[Index]:=True List_of_RM_Entries.SAP[Index]:=Last_SAP Max_Len_Data_Unit:=List_of_RM_Entries.Max_Len_Data_Unit[Index] Send_Timeout:=Max(Stored_Send_Timeout, Stored_Min_Send_Timeout) MSAC2S_Initiate.req(Res_SAP=Last_SAP, Req_Add=Stored_Req_Add, Max_Len_Data_Unit, Send_Timeout, Features_Supported= Stored_Features_Supported, Profile_Features_Supported= Stored_Profile_Features_Supported, Profile_Ident_Number= Stored_Profile_Ident_Number, Add_Addr_Param= Stored_Add_Addr_Param)	CLOSED
21	CW-SEARCH	/Index = 0 =>Send_Timeout:=Max(Stored_Send_Timeout, Stored_Min_Send_Timeout) MSAC2S_Abort.req(Res_SAP=Last_SAP, Req_Add, Subnet=NO, Instance=MSAC2, Reason_Code=ABT_IA, Send_Timeout)	CLOSED
22	CLOSED	MSRM2S_Reset.req =>for (all CRL entries with AR-Type=MS2) MSAC2S_Reset.req(Res_SAP=SSAP) endfor	RESET-AC2
23	CLOSED	MSAC2S_Closed.ind(Res_SAP) =>Stored_Res_SAP:=Res_SAP Index:= Search_in_List_of_RM_Entries(SAP:=Res_SAP, In_Use:=True)	REOW-SEARCH
24	REOW-SEARCH	/Index <> 0 =>List_of_RM_Entries.In_Use[Index]:=False Put_to_List_of_SSAPs(Stored_Res_SAP)	WAIT-ACT
25	REOW-SEARCH	/Index = 0 =>MSRM2S_Fault.ind	POWER-ON
26	RESET-AC2	for (all CRL entries with AR-Type=MS2) MSAC2S_Reset.cnf(Res_SAP=SSAP) endfor /M_status=OK =>MSRM2S_Reset.cnf	POWER-ON
27	ANY-STATE	MSAC2S_Fault.ind(Res_SAP) =>MSRM2S_Fault.ind	POWER-ON
28	ANY-STATE	unexpected DL reaction =>MSRM2S_Fault.ind	POWER-ON

9.5 MSAC2S

9.5.1 Primitive definitions

9.5.1.1 Primitives exchanged between MSAC2S and FSPMS

Table 80 shows the service primitives including their associated parameters issued by the FSPMS and received by the MSAC2S.

Table 80 – Primitives issued by FSPMS to MSAC2S

Primitive name	Source	Associated parameters	Functions
Initiate.rsp(+)	FSPMS	Res SAP, Max Len Data Unit, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param	Refer to FAL Service Definition in IEC 61158-5-3
Initiate.rsp(-)	FSPMS	Res SAP, Error Decode, Error Code 1 Error Code 2	
Read.rsp(+)	FSPMS	Res SAP, Length, Data	
Read.rsp(-)	FSPMS	Res SAP, Error Decode, Error Code 1 Error Code 2	
Write.rsp(+)	FSPMS	Res SAP, Length	
Write.rsp(-)	FSPMS	Res SAP, Error Decode, Error Code 1 Error Code 2	
Data Transport.rsp(+)	FSPMS	Res SAP, Length, Data	
Data Transport.rsp(-)	FSPMS	Res SAP, Error Decode, Error Code 1 Error Code 2	
Abort.req	FSPMS	Res SAP, Subnet, Instance, Reason Code	

Table 81 shows the service primitives including their associated parameters issued by the MSAC2S and received by the FSPMS.

Table 81 – Primitives issued by MSAC2S to FSPMS

Primitive name	Source	Associated parameters	Functions
Initiate.ind	MSAC2S	Res SAP, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param	Refer to FAL Service Definition in IEC 61158-5-3
Abort.ind	MSAC2S	Res SAP, Locally Generated, Subnet, Instance, Reason Code, Additional Detail	
Read.ind	MSAC2S	Res SAP, Slot Number, Index, Length	
Write.ind	MSAC2S	Res SAP, Slot Number, Index, Length, Data	
Data Transport.ind	MSAC2S	Res SAP, Slot Number, Index, Length, Data	

9.5.1.2 Primitives exchanged between MSAC2S and MSRM2S

Table 82 shows the service primitives including their associated parameters issued by MSAC2S and received by the MSRM2S.

Table 82 – Primitives issued by MSRM2S to MSAC2S

Primitive name	Source	Associated parameters	Functions
Reset.req	MSRM2S	Res SAP	Refer to FAL Service Definition in IEC 61158-5-3
Initiate.req	MSRM2S	Res SAP, Features Supported, Profile Features Supported, Profile Ident Number, Add Addr Param	
RM2_Abort.req	MSRM2S	Res SAP, Subnet, Instance, Reason Code	

Table 83 shows the service primitives including their associated parameters issued by MSAC2S and received by the MSRM2S.

Table 83 – Primitives issued by MSAC2S to MSRM2S

Primitive name	Source	Associated parameters	Functions
Reset.cnf	MSAC2S	Res SAP	Refer to FAL Service Definition in IEC 61158-5-3
Fault.ind	MSAC2S	Res SAP	
Closed.ind	MSAC2S	Res SAP	Signals the Termination of a MS2 Communication

9.5.1.3 Parameters of MSAC2S primitives

The parameters used with the primitives exchanged between the MSAC2S and FSPMS,MSRM2S are described in FAL Service Definition in IEC 61158-5-3. Additional parameters are described in Table 84.

Table 84 – Parameter used with primitives exchanged with MSAC2S

Parameter name	Description
Res SAP	This parameter conveys the SSAP for addressing of the various MS2 AR.

9.5.2 State machine description

In the state CLOSED the Initiate service is expected from the Resource Manager. Then the corresponding DLSAP is activated and the User gets the Initiate.ind. The User shall generate a MSAC2S_Initiate.rsp and the connection is established after the Master has fetched the response.

In the OPEN-state valid services (see macro <VALID_SERVICE>) are allowed. Only one request from the DP-master is processed simultaneously, otherwise the connection is aborted.

Both the DP-master and the DP-slave can close the connection using the Abort. The remote User gets an Abort.ind.

The connection monitoring is based on the U-Timer, F-Timer and I-Timer (see 6.9).

Local variables of the MSAC2S

Server_SAP
(Unsigned8)

The used SAP for the MS2 connection.

U-Timer
(Unsigned16)

The U-Timer controls the reaction of the User after delivering an indication. If the U-Timer expires an Idle-PDU is created and transferred to the local DLL. The U-Timer is stopped when the User provides the response.

F-Timer
(Unsigned16)

The F-Timer controls the Master fetching the previously provided PDU. If the F-Timer expires an Abort is generated, transferred to the local DLL and the F-Timer starts again. The F-Timer is stopped as soon as the Master fetches the PDU.

I-Timer
(Unsigned16)

The I-Timer controls the next activity of the client after the Master has fetched the previous PDU. When the I-Timer expires an Abort is generated, transferred to the local DLL and the F-Timer starts. The I-Timer is stopped when the Master has sent the next request.

NOTE Only one of the timers described above is running at the same time.

Service_Header
(Unsigned8)

Contains the actual service header to check whether the User generates the correct response.

Return-State
INITIATE, VALID-SERVICE

This variable is used to store the state where to return after an idle cycle has been processed.

Go_Abort
(Boolean)

Local flag that indicates an abort reaction after finishing the present service response.

Service_Buffer
(Octet-String[244])

Local buffer that stores one complete Response-PDU (Initiate.rsp or <VALID_SERVICE>.rsp) during an outstanding Idle-RES-PDU.

Stored_Slot_Number
Actual value of the slot number

Stored_Index
Stored values of the parameters of Read/Write-services

Stored_Instance
Actual value of the instance (e.g. MSAC2)

Stored_Reason
Stored values of the parameters of MSAC2S_Abort-service

INITIATE-REQ-PDU-LEN
(local constant)

PDU-size (the 4 Octet Header is excluded) set to 48 (64 respectively for links)

ABORT_PDU_LEN
(local constant)

Length of an Abort-REQ-PDU = 4

Macros

Read-REQ-PDU
This function will return True if the corresponding L_sdu is a valid Read-REQ-PDU

Write-REQ-PDU
This function will return True if the corresponding L_sdu is a valid Write-REQ-PDU

Data_Transport-REQ-PDU
This function will return True if the corresponding L_sdu is a valid Data_Transport-REQ-PDU

Abort-REQ-PDU
This function will return True if the corresponding L_sdu is a valid Abort-REQ-PDU

Idle-REQ-PDU
This function will return True if the corresponding L_sdu is a valid Idle-REQ-PDU

STORE_ABORT_PARAMETER
(

Stored-Instance=Instance

Stored-Reason_Code=Reason_Code

)

<VALID-SERVICE>
<

Service_Header=0x5E: Read

Service_Header=0x5F: Write

Service_Header=0x51: Data_Transport

>

VALID-SERVICE-REQ-PDU
(

(Read-REQ-PDU || Write-REQ-PDU || OR Data_Transport-REQ-PDU)

&& L_sdu.len >= 4

&& (L_sdu[1] = 0x5e || (L_sdu.len >= Length+4

&& Max_DLSDU_length_ind_low-4 >= Length)

)

9.5.3 MSAC2S state table

Table 85 contains the complete description of the MSAC2S state machine.

Table 85 – MSAC2S state table

#	Current state	Event / condition => action	Next state
1	POWER-ON	=> Server_SAP:=Res_SAP	CLOSED
2	CLOSED	MSAC2S_Initiate.req(Res_SAP, Req_Add, Max_Len_Data_Unit, Send_Timeout, Features_Supported, Profile_Features_Supported, Profile_Ident_Number, Add_Addr_Param) => Service_Header:=0x57 F-Timer:=Send_Timeout U-Timer:=Send_Timeout I-Timer:=2*Send_Timeout Stored_Max_Len_Data_Unit:= Max(Max_Len_Data_Unit, INITIATE-REQ-PDU-LEN) clear Service_Buffer Go_Abort:=FALSE MSAC2_Initiate.ind(Req_Add, Res_SAP, Features_Supported, Profile_Features_Supported, Profile_Ident_Number, Add_Addr_Param) DMPMS_RSAP_ACTIVATE.req(SSAP=Server_SAP, Access=Req_Add, L_sdu_length_list=(Stored_Max_Len_Data_Unit+4,0, Stored_Max_Len_Data_Unit+4,0) , Indication_Mode=Data)	SET-ACC-POS
3	CLOSED	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) => ignore	CLOSED
4	CLOSED	MSAC2S_RM2_Abort.req(Res_SAP, Req_Add, Subnet, Instance, Reason_Code, Send_Timeout) => F-Timer:=Send_Timeout STORE_ABORT_PARAMETER DMPMS_RSAP_ACTIVATE.req(SSAP=Server_SAP, Access=Req_add, L_sdu_length_list=(ABORT_PDU_LEN,0, ABORT_PDU_LEN,0) Indication_Mode=Data)	SET-ACC-FE
5	CLOSED	MSAC2S_XXX.rsp(+/-)(Res_SAP) => Reason_Code:=ABT_SE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	CLOSED
6	CLOSED	MSAC2S_Initiate.rsp(+/-)(Res_SAP) => Reason_Code:=ABT_SE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	CLOSED
7	CLOSED	MSAC2S_Abort.req(Res_SAP, Subnet, Instance, Reason_Code) => ignore	CLOSED
8	SET-ACC-POS	DMPMS_RSAP_ACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status = OK Start U-Timer	INITI-WRES
9	SET-ACC-POS	DMPMS_RSAP_ACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status=NO/IV => MSAC2S_Fault.ind	POWER-ON
10	SET-ACC-FE	DMPMS_RSAP_ACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status = OK => L_sdu:=Abort-REQ-PDU(with stored parameter) Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	SABORT-WUPD
11	SET-ACC-FE	DMPMS_RSAP_ACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status=NO/IV => MSAC2S_Fault.ind	POWER-ON

#	Current state	Event / condition => action	Next state
12	INITI-WRES	MSAC2S_Initiate.rsp(+)(Res_SAP) => L_sdu:=Initiate-RES-PDU(With Stored_MAx_Len_Data_Unit) Stop U-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	INITI-WUPD
13	INITI-WRES	MSAC2S_Initiate.rsp(-)(Res_SAP) => L_sdu:=Initiate-NRS-PDU Stop U-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	SABORT-WUPD
14	INITI-WRES	MSAC2S_XXX.rsp(+/-)(Res_SAP) => Reason_Code:=ABT_SE Stop U-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
15	INITI-WRES	MSAC2S_Abort.req(Res_SAP, Subnet, Instance, Reason_Code) => L_sdu:=Abort-REQ-PDU Stop U-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	SABORT-WUPD
16	INITI-WRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Abort-REQ-PDU => Stop U-Timer DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=FALSE, (Subnet, Instance, Reason_Code) from Abort-REQ-PDU)	WAIT-DEACT
17	INITI-WRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /NOT(Abort-REQ-PDU) => Reason_Code:=ABT_FE Stop U-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
18	INITI-WRES	U-Timer expired => L_sdu:=Idle-REQ-PDU Return-State:=INITIATE Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	IDLE-WUPD
19	INITI-WUPD	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP, Serv_class=Low, L_status) /L_status=OK => ignore	INITI-SRES

#	Current state	Event / condition => action	Next state
20	INITI-WUPD	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP, Serv_class=Low, L_status) /L_status= LS/IV/LR => Stop F-Timer Stop I-Timer MSAC2S_Fault.ind	POWER-ON
21	INITI-SRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && L_sdu.len=0 && Go_Abort = FALSE => Stop F-Timer Start I-Timer	OPEN
22	INITI-SRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && L_sdu.len=0 && Go_Abort = TRUE => L_sdu:=Abort-REQ-PDU(With Stored Parameter) Stop F-Timer Start I-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	SABORT-WUPD
23	INITI-SRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO/NO && L_sdu.len<>0 && Abort-REQ-PDU => Stop F-Timer DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=FALSE, (Subnet, Instance, Reason_Code) from Abort-REQ-PDU)	WAIT-DEACT
24	INITI-SRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && L_sdu.len<>0 && NOT(Abort-REQ-PDU) => Reason_Code:=ABT_FE Stop F-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
25	INITI-SRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=NO && L_sdu.len<>0 && NOT(Abort-REQ-PDU) => Go_Abort:=TRUE STORE_ABORT_PARAMETER Reason_Code:=ABT_FE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	INITI-SRES
26	INITI-SRES	MSAC2S_XXX.rsp(+/-)(Res_SAP) => Go_Abort:=TRUE STORE_ABORT_PARAMETER Reason_Code:=ABT_SE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	INITI-SRES
27	INITI-SRES	MSAC2S_Initiate.rsp(+/-)(Res_SAP) => Go_Abort:=TRUE STORE_ABORT_PARAMETER Reason_Code:=ABT_SE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	INITI-SRES

#	Current state	Event / condition => action	Next state
28	INITI-SRES	MSAC2S_Abort.req(Res_SAP, Subnet, Instance, Reason_Code) => Go_Abort:=TRUE STORE_ABORT_PARAMETER	INITI-SRES
29	INITI-SRES	F-Timer expired => Reason_Code:=ABT_TO Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
30	IDLE-WUPD	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP, Serv_class=Low, L_status) /Status=OK	IDLE-SREQ
31	IDLE-WUPD	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP, Serv_class=Low, L_status) /L_status= LS/IV/LR => Stop F-Timer Stop I-Timer MSAC2S_Fault.ind	POWER-ON
32	IDLE-SREQ	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && L_sdu.len=0 && Go_Abort = FALSE => Stop F-Timer Start I-Timer	W-IDLE-CON
33	IDLE-SREQ	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && L_sdu.len=0 && Go_Abort = TRUE => L_sdu:=Abort-REQ-PDU(With Stored Parameter) Stop F-Timer Start I-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	SABORT-WUPD
34	IDLE-SREQ	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO/NO && L_sdu.len<>0 && Abort-REQ-PDU DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC2_Abort.ind(Res_SAP=Server_SAP, Locally-Generated=FALSE, (Subnet, Instance, Reason_Code) from Abort-REQ-PDU) Stop F-Timer	WAIT-DEACT
35	IDLE-SREQ	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && L_sdu.len<>0 && NOT(Abort-REQ-PDU) => Reason_Code:=ABT_FE Stop F-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
36	IDLE-SREQ	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=NO && L_sdu.len<>0 && NOT(Abort-REQ-PDU) => Go_Abort:=TRUE STORE_ABORT_PARAMETER Reason_Code:=ABT_FE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	IDLE-SREQ

#	Current state	Event / condition => action	Next state
37	IDLE-SREQ	MSAC2S_XXX.rsp(+)(Res_SAP) /Service-Response fits stored Service_Header && Stored_Max_Len_Data_Unit >= Length && Service_Buffer empty => Service_Buffer:= <VALID-SERVICE>-RES-PDU(with Stored_Slot_Number,Stored_Index)	IDLE-SREQ
38	IDLE-SREQ	MSAC2S_XXX.rsp(-)(Res_SAP) /Service-Response fits stored Service_Header && Stored_Max_Len_Data_Unit >= Length && Service_Buffer empty => Service_Buffer:= <VALID-SERVICE>-NRS-PDU(with Stored_Slot_Number,Stored_Index)	IDLE-SREQ
39	IDLE-SREQ	MSAC2S_Initiate.rsp(+)(Res_SAP) /Service-Response fits stored Service_Header && Service_Buffer empty => Service_Buffer:= Initiate-RES-PDU	IDLE-SREQ
40	IDLE-SREQ	MSAC2S_Initiate.rsp(-)(Res_SAP) /Service-Response fits stored Service_Header && Service_Buffer empty => Service_Buffer:= Initiate-NRS-PDU	IDLE-SREQ
41	IDLE-SREQ	MSAC2S_XXX.rsp(+/-)(Res_SAP) /(NOT(Service-Response fits stored Service_Header && Stored_Max_Len_Data_Unit >= Length) || NOT(Service_Buffer empty)) => Go_Abort:=TRUE STORE_ABORT_PARAMETER Reason_Code:=ABT_RE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	IDLE-SREQ
42	IDLE-SREQ	MSAC2S_Initiate.rsp(+/-)(Res_SAP) /(NOT(Service-Response fits stored Service_Header) || NOT(Service_Buffer empty)) => Go_Abort:=TRUE STORE_ABORT_PARAMETER Reason_Code:=ABT_RE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	IDLE-SREQ
43	IDLE-SREQ	MSAC2S_Abort.req(Res_SAP, Subnet, Instance, Reason_Code) => Go_Abort:=TRUE STORE_ABORT_PARAMETER	IDLE-SREQ
44	IDLE-SREQ	F-Timer expired => Reason_Code:=ABT_TO Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
45	W-IDLE-CON	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Idle-RES-PDU && Return-State=INITIATE && Go_Abort=FALSE && (Service_Buffer empty) => Stop I-Timer Start U-Timer	INITI-WRES

#	Current state	Event / condition => action	Next state
46	W-IDLE-CON	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Idle-RES-PDU && Return-State=INITIATE && Go_Abort=FALSE && NOT(Service_Buffer empty) && Service_Buffer=Initiate-NRS-PDU => L_sdu:=Stored L_sdu from Service_Buffer clear Service_Buffer Stop I-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	SABORT-WUPD
47	W-IDLE-CON	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Idle-RES-PDU && Return-State=INITIATE && Go_Abort=FALSE && NOT(Service_Buffer empty) && Service_Buffer=Initiate-RES-PDU => L_sdu:=Stored L_sdu from Service_Buffer clear Service_Buffer Stop I-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	INITI-WUPD
48	W-IDLE-CON	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Idle-RES-PDU && Return-State=VALID-SERVICE && Go_Abort=FALSE && (Service_Buffer empty) => Stop I-Timer Start U-Timer	VS-WRES
49	W-IDLE-CON	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Idle-RES-PDU && Return-State=VALID-SERVICE && Go_Abort=FALSE && NOT(Service_Buffer empty) => L_sdu:=Stored L_sdu from Service_Buffer clear Service_Buffer Stop I-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	VS-WUPD
50	W-IDLE-CON	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Idle-RES-PDU && Go_Abort=TRUE => L_sdu:=Abort-REQ-PDU(with stored Parameter) DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	SABORT-WUPD
51	W-IDLE-CON	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Abort-REQ-PDU => Stop I-Timer DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=FALSE, (Subnet, Instance, Reason_Code) from Abort-REQ-PDU)	WAIT-DEACT
52	W-IDLE-CON	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /NOT(Abort-REQ-PDU || Idle-RES-PDU) => Reason_Code:=ABT_FE Stop I-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD

#	Current state	Event / condition => action	Next state
53	W-IDLE-CON	MSAC2S_XXX.rsp(+)(Res_SAP) /Service-Response fits stored Service_Header && Stored_Max_Len_Data_Unit >= Length && Service_Buffer empty => Service_Buffer:= <VALID-SERVICE>-RES-PDU(with Stored_Slot_Number,Stored_Index)	W-IDLE-CON
54	W-IDLE-CON	MSAC2S_XXX.rsp(-)(Res_SAP) /Service-Response fits stored Service_Header && Stored_Max_Len_Data_Unit >= Length && Service_Buffer empty => Service_Buffer:= <VALID-SERVICE>-NRS-PDU (with Stored_Slot_Number,Stored_Index)	W-IDLE-CON
55	W-IDLE-CON	MSAC2S_XXX.rsp(+/-)(Res_SAP) /(NOT(Service-Response fits stored Service_Header && Stored_Max_Len_Data_Unit >= Length) || NOT(Service_Buffer empty)) => Go_Abort:=TRUE STORE_ABORT_PARAMETER Reason_Code:=ABT_RE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	W-IDLE-CON
56	W-IDLE-CON	MSAC2S_Initiate.rsp(+/-)(Res_SAP) /(NOT(Service-Response fits stored Service_Header) || NOT(Service_Buffer empty)) => Go_Abort:=TRUE STORE_ABORT_PARAMETER Reason_Code:=ABT_RE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	W-IDLE-CON
57	W-IDLE-CON	MSAC2S_Abort.req(Res_SAP, Subnet, Instance, Reason_Code) => Go_Abort:=TRUE STORE_ABORT_PARAMETER	W-IDLE-CON
58	W-IDLE-CON	I-Timer expired => Reason_Code:=ABT_TO Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
59	SABORT-WUPD	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP, Serv_class=Low, L_status) /Status=OK	SABORT-SRES
60	SABORT-WUPD	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP, Serv_class=Low, L_status) /L_status= LS/IV/LR => Stop F-Timer Stop I-Timer MSAC2S_Fault.ind	POWER-ON
61	SABORT-SRES	MSAC2S_Abort.req(Res_SAP, Subnet, Instance, Reason_Code) => ignore	SABORT-SRES
62	SABORT-SRES	MSAC2S_XXX.rsp(+/-)(Res_SAP) => Reason_Code:=ABT_SE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-SRES
63	SABORT-SRES	MSAC2S_Initiate.rsp(+/-)(Res_SAP) => Reason_Code:=ABT_SE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-SRES

#	Current state	Event / condition => action	Next state
64	SABORT-SRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && L_sdu.len=0 => Stop I-Timer Stop F-Timer DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP)	WAIT-DEACT
65	SABORT-SRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=NO && L_sdu.len<>0 => ignore	SABORT-SRES
66	SABORT-SRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && L_sdu.len<>0 => ignore L_sdu Stop I-Timer Stop F-Timer DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP)	WAIT-DEACT
67	SABORT-SRES	I-Timer expired => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code=ABT_TO)	WAIT-DEACT
68	SABORT-SRES	F-Timer expired => DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code=ABT_TO)	WAIT-DEACT
69	WAIT-DEACT	DMPMS_SAP_DEACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status = OK => MSAC2S_Closed.ind(Res_SAP)	CLOSED
70	WAIT-DEACT	DMPMS_SAP_DEACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status=NO/IV => MSAC2S_Fault.ind	POWER-ON
71	OPEN	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /VALID-SERVICE-REQ-PDU => Service_Header:=Function_Num Stored_Slot_Number:=Slot_Number Stored_Index:=Index Stop I-Timer Start U-Timer MSAC2S_<VALID-SERVICE>.ind(Res_SAP=Server_SAP)	VS-WRES
72	OPEN	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Idle-REQ-PDU {IDLE-PDU} => L_sdu:=Idle-RES-PDU Stop I-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	VS-WUPD
73	OPEN	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Abort-REQ-PDU => Stop I-Timer DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=FALSE, (Subnet, Instance, Reason_Code) from Abort-REQ-PDU)	WAIT-DEACT

#	Current state	Event / condition => action	Next state
74	OPEN	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /NOT(VALID-SERVICE-REQ-PDU || (Abort-REQ-PDU) || (Idle-REQ-PDU)) => Reason_Code:=ABT_FE Stop I-Timer Start I-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
75	OPEN	MSAC2S_Abort.req(Res_SAP, Subnet, Instance, Reason_Code) => L_sdu:=Abort-REQ-PDU Stop I-Timer Start I-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	SABORT-WUPD
76	OPEN	MSAC2S_Initiate.rsp(+/-)(Res_SAP) => Reason_Code:=ABT_SE Stop I-Timer Start I-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
77	OPEN	MSAC2S_XXX.rsp(+/-)(Res_SAP) => L_sdu:=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code=ABT_TO) Stop I-Timer Start I-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	SABORT-WUPD
78	OPEN	I-Timer expired => Reason_Code:=ABT_TO Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
79	VS-WRES	MSAC2S_XXX.rsp(+)(Res_SAP) /Service-Response fits stored Service_Header && Stored_Max_Len_Data_Unit >= Length => L_sdu:=<VALID-SERVICE>-RES-PDU(with Stored_Slot_Number,Stored_Index) Stop U-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	VS-WUPD
80	VS-WRES	MSAC2S_XXX.rsp(-)(Res_SAP) /Service-Response fits stored Service_Header && Stored_Max_Len_Data_Unit >= Length => L_sdu:=<VALID-SERVICE>-NRS-PDU Stop U-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	VS-WUPD

#	Current state	Event / condition => action	Next state
81	VS-WRES	MSAC2S_Initiate.rsp(+/-)(Res_SAP) => Reason_Code:=ABT_SE Stop U-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
82	VS-WRES	MSAC2S_XXX.rsp(+/-)(Res_SAP) /NOT(Service-Response fits stored Service_Header && Stored_Max_Len_Data_Unit >= Length) => Reason_Code:=ABT_RE Stop U-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
83	VS-WRES	MSAC2S_Abort.req(Res_SAP, Subnet, Instance, Reason_Code) => L_sdu:=Abort-REQ-PDU Stop U-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	SABORT-WUPD
84	VS-WRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Abort-REQ-PDU => Stop U-Timer DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=FALSE, (Subnet, Instance, Reason_Code) from Abort-REQ-PDU)	WAIT-DEACT
85	VS-WRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /NOT(Abort-REQ-PDU) => Reason_Code:=ABT_FE Stop U-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
86	VS-WRES	U-Timer expired => L_sdu:=Idle-REQ-PDU Return-State:=VALID-SERVICE Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	IDLE-WUPD
87	VS-WUPD	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP, Serv_class=Low, L_status) /Status=OK	VS-SRES
88	VS-WUPD	DMPMS_REPLY_UPDATE.cnf(SSAP=Server_SAP, Serv_class=Low, L_status) /L_status= LS/IV/LR => Stop F-Timer Stop I-Timer MSAC2S_Fault.ind	POWER-ON

#	Current state	Event / condition => action	Next state
89	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && L_sdu.len=0 && Go_Abort = FALSE Stop F-Timer Start I-Timer	OPEN
90	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && L_sdu.len=0 && Go_Abort = TRUE => L_sdu:=Abort-REQ-PDU with stored Parameter Stop F-Timer Start I-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu, Serv_class=Low, Transmit=Single)	SABORT-WUPD
91	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO/NO && L_sdu.len<>0 && Abort-REQ-PDU => Stop F-Timer DMPMS_SAP_DEACTIVATE.req(SSAP=Server_SAP) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=FALSE, (Subnet, Instance, Reason_Code) from Abort-REQ-PDU)	WAIT-DEACT
92	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=LO && L_sdu.len<>0 && NOT (Abort-REQ-PDU) => Reason_Code:=ABT_FE Stop F-Timer Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
93	VS-SRES	DMPMS_DATA_REPLY.ind(SSAP, DSAP=Server_SAP, Loc_add, Rem_add,L_sdu, Serv_class=Low, Update_status) /Update_status=NO && L_sdu.len<>0 && NOT(Abort-REQ-PDU) => Go_Abort:=TRUE STORE_ABORT_PARAMETER Reason_Code:=ABT_FE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	VS-SRES
94	VS-SRES	MSAC2S_XXX.rsp(+/-)(Res_SAP) => Go_Abort:=TRUE STORE_ABORT_PARAMETER Reason_Code:=ABT_SE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	VS-SRES
95	VS-SRES	MSAC2S_Initiate.rsp(+/-)(Res_SAP) => Go_Abort:=TRUE STORE_ABORT_PARAMETER Reason_Code:=ABT_SE MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	VS-SRES
96	VS-SRES	MSAC2S_Abort.req(Res_SAP, Subnet, Instance, Reason_Code) => Go_Abort:=TRUE STORE_ABORT_PARAMETER	VS-SRES

#	Current state	Event / condition => action	Next state
97	VS-SRES	F-Timer expired => Reason_Code:=ABT_TO Start F-Timer DMPMS_REPLY_UPDATE.req(SSAP=Server_SAP, L_sdu=Abort-REQ-PDU(Subnet=NO, Instance=MSAC2, Reason_Code), Serv_class=Low, Transmit=Single) MSAC2S_Abort.ind(Res_SAP=Server_SAP, Locally_Generated=TRUE, Subnet=NO, Instance=MSAC2, Reason_Code)	SABORT-WUPD
98	ANY-STATE	unexpected DMPMS reaction => Stop U-Timer Stop F-Timer Stop I-Timer MSAC2S_Fault.ind	POWER-ON
99	DEACT-PON	DMPMS_SAP_DEACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status = OK	POWER-ON
100	DEACT-PON	DMPMS_SAP_DEACTIVATE.cnf(SSAP=Server_SAP,M_status) /M_status=NO/IV => MSAC2S_Fault.ind	POWER-ON
101	any state	MSAC2S_Reset.req(Res_SAP) => MSAC2S_Reset.cnf(Res_SAP)	POWER-ON

9.6 MSCS1S

9.6.1 Primitive definitions

9.6.1.1 Primitives exchanged between MSCS1S and FSPMS

Table 86 shows the service primitives including their associated parameters issued by MSCS1S and received by the FSPMS.

Table 86 – Primitives issued by MSCS1S to FSPMS

Primitive name	Source	Associated parameters	Functions
Set Time.ind	MSCS1S	AREP, Time Value, Local Time Diff, Summertime, Accuracy, Synchronisation Active, Announcement Hour	Refer to FAL Service Definition in IEC 61158-5-3
Sync Interval Violation.ind	MSCS1S	AREP	

9.6.1.2 Parameter of MSCS1S primitives

The parameters used with the primitives exchanged between the FSPMS and the MSCS1S are described in FAL Service Definition in IEC 61158-5-3.

9.6.2 State machine description

This machine always remains in IDLE state. It receives Clock Value indications from the DMPMS. A Set Time indication will be issued on detection of the end of a valid Clock Synchronization sequence. Otherwise a Sync Interval Violation indication will be issued.

Local variables of the MSCS1S

Time Last Rcvd
(Network Time)

This local variable contains the Clock_Value_Time_Event of the last valid received Clock Value indication.

Error
(Unsigned8)

This local variable is a counter for invalid Clock Synchronisation sequences. It will be reset to 0, when a Sync Interval Violation is indicated.

Macros of the MSCS1S:
SET_TIME_ATTRIBUTES
(

Local Time Diff:=CS_list.Clock_value_status.C*(-1)^CS_list.Clock_value_status.CV

Summertime:= CS_list.Clock_value_status.SWT

Accuracy:= CS_list.Clock_value_status.CR

Synchronisation Active:= CS_list.Clock_value_status.SYF

Announcement Hour:=CS_list.Clock_value_status.ANH

)

9.6.3 MSCS1S state table

Table 87 contains the complete description of the MSCS1S state machine.

Table 87 – MSCS1S state table

#	Current state	Event / condition => action	Next state
1	Idle	CS_CLOCK_VALUE.ind (Time_Master_Addr, CS_list, CS_status, Receive_Delay_Time) /CS_status == SV && Error < 2 => Error:= Error+1	Idle
2	Idle	CS_CLOCK_VALUE.ind (Time_Master_Addr, CS_list, CS_status, Receive_Delay_Time) /CS_status == OK && CS_list.Clock_Value_previous_TE != Time Last Rcvd && Error < 2 => Error:= Error+1	Idle
3	Idle	CS_CLOCK_VALUE.ind (Time_Master_Addr, CS_list, CS_status, Receive_Delay_Time) /CS_status == SV && Error >= 2 => Error:= 0 Sync Interval Violation.ind	Idle
4	Idle	CS_CLOCK_VALUE.ind (Time_Master_Addr, CS_list, CS_status, Receive_Delay_Time) /CS_status == OK && CS_list.Clock_Value_previous_TE != Time Last Rcvd && Error >= 2 => Error:= 0 Sync Interval Violation.ind	Idle

#	Current state	Event / condition => action	Next state
5	Idle	CS_CLOCK_VALUE.ind (Time_Master_Addr, CS_list, CS_status, Receive_Delay_Time) /CS_status == OK && CS_list.Clock_Value_previous_TE == Time Last Rcvd => Error:= 0 Time Last Rcvd:= CS_list.Clock_Value_Time_Event Time Value:= Time Last Rcvd + Receive Delay Time SET_TIME_ATTRIBUTES Set_Time.ind(Time Value, Local Time Diff, Summertime, Accuracy, Synchronisation Active, Announcement Hour)	Idle

9.7 MSCY1M

9.7.1 Primitive definitions

9.7.1.1 Primitives exchanged between FSPMM1 and MSCY1M

Table 88 shows the service primitives including their associated parameters issued by the FSPMM1 and received by the MSCY1M.

Table 88 – Primitives issued by FSPMM1 to MSCY1M

Primitive name	Source	Associated parameters	Functions
MInit MS0.req	FSPMM1	Rem Add	Refer to FAL Service Definition in IEC 61158-5-3
Reset.req	FSPMM1	(none)	
Abort.req	FSPMM1	Rem Add, Subnet, Instance, Reason Code	
Start Slave Handler.req	FSPMM1	Rem Add	Start first Cycle of MSCY1M
Stop Slave Handler.req	FSPMM1	Rem Add	Stop processing of MSCY1M
Cont Slave Handler.req	FSPMM1	Rem Add, Output Clear	Start new Cycle of MSCY1M
Get Slave Diag.req	FSPMM1	Rem Add	Refer to FAL Service Definition in IEC 61158-5-3
Set Output.req	FSPMM1	Rem Add, Slot_Number Output Data	
Get Input.req	FSPMM1	Rem Add, Slot Number	

Table 89 shows the service primitives including their associated parameters issued by the MSCY1M and received by the FSPMM1.

Table 89 – Primitives issued by MSCY1M to FSPMM1

Primitive name	Source	Associated parameters	Functions
MInit MS0.cnf	MSCY1M	Rem Add	Refer to FAL Service Definition in IEC 61158-5-3
Reset.cnf	MSCY1M	(none)	
Start Slave Handler.cnf	MSCY1M	Rem Add	Start first Cycle of MSCY1M
Stop Slave Handler.cnf	MSCY1M	Rem Add	Stop processing of MSCY1M
Cont Slave Handler.cnf	MSCY1M	Rem Add, Diag, No AClr	Start new Cycle of MSCY1M
Get Slave Diag.cnf(+)	MSCY1M	Rem Add, CREP, Diag Data	Refer to FAL Service Definition in IEC 61158-5-3
Get Slave Diag.cnf(-)	MSCY1M	Rem Add	
Set Output.cnf(+)	MSCY1M	Rem Add, Slot Number	
Set Output.cnf(-)	MSCY1M	Rem Add, Slot Number	
Get Input.cnf(+)	MSCY1M	Rem Add, Slot_Number, Input Data	
Get Input.cnf(-)	MSCY1M	Rem Add	
New Slave Diag.ind	MSCY1M	Rem Add	
New Input.ind	MSCY1M	Rem Add	
Started.ind	MSCY1M	Rem Add, Alarm Limit	
Stopped.ind	MSCY1M	Rem Add	
Fault.ind	MSCY1M	Rem Add	

9.7.1.2 Parameters of MSCY1M primitives

The parameters used with the primitives exchanged between the FSPMM1 and MSCY1M are described in Table 90.

Table 90 – Parameters used with primitives exchanged between FSPMM1 and MSCY1M

Parameter name	Description
Rem Add	This parameter is used to identify the DP-slave which is assigned to that communication relation.
Output Clear	This parameter indicates to the MSCY1M state machine that the Outputs shall be cleared.
Diag	This parameter indicates to the FSPMM1 state machine that no Data Exchange was processed in the preceding cycle or the Slave was deactivated.
No Aclr	This parameter indicates to the FSPMM1 state machine that the corresponding DP-slave is operated in the non Autoclear-Mode (this DP-slave has no influence to the Masters State (OPERATE or CLEAR).

Other parameters used with the primitives exchanged between the FSPMM1 and the MSCY1M are described in FAL Service Definition in IEC 61158-5-3.

9.7.2 State machine description

For each possible DP-slave a State Machine MSCY1M (Slave-Handler) is established and started by a superior State Machine, the FSPMM1. The remote address of the Slave (Rem_Add) is used as a local reference for MSCY1M.

The Slave-Handler manages the individual states for every DP-slave. The following main states will be distinguished:

– diagnostic,
– parameterisation,
– configuration,
– data exchange.

SI_Para_Exist
Access: -r

Local information indicating whether or not a Slave parameter set of the associated Rem_Add exists.

SPara
Access:-r/-w

Any Information from ARL/CRL related to this DP-slave.

BOutput
Access: -r

Local storage of Output data for the DP-slave.

BInput
Access: -w

Local storage of Input data for the DP-slave.

BDiag
Access: -r/-w

Local storage of Diagnostic information of the DP-slave.

MSCY1M changes the following flags of the diagnostic information:

Invalid_Slave_Response

Station_Non_Existent

Deactivated

DTrans
This variable is mapped on the Data Transfer List attribute of CR-List.

Access: -r/-w

The user data exchange mode between the DP-master (Class 1) and its assigned DP-slaves is supervised on the Master side. This means it is checked whether a user data transfer to the associated DP-slaves was executed within the last DP cycle or a diagnostic cycle, after which the directly following Slave state is again the data transfer state. Any other following state forces the associated bit of the Slave to be cleared.

SDiag
This variable is mapped on the System Diagnosis List attribute of CR-List.

Access: -r/-w

It is set while the MSCY1M state machine of an activated Slave is not able to establish the connection to the Slave without errors. E. g. it is set if a Slave does not respond, if he responds with a negative DL confirmation or if he does not respond with the correct number of input data during Data_Exchange. The bit is reset after the connection could be established without any error. It is also reset if the Slave is deactivated by the User or if the MSCY1M State Machine is stopped

Local variables

Delay_Count
(Unsigned8)

The Delay_Count is used to count the number of Slave_Diag.cnf in the state DIAG2 while Diag_Data.Prm_Req is still set (Slow Slave). The Delay_Count is initialized with the value of SPara.Diag_Upd_Delay.

Range: 0 to 15 (extendible up to 255)

MSAL1M_Started
(Boolean)

This variable is used to store the information whether the MSAL1M State Machine is started (True) or not (False).

Wait_for_Start_con
(Boolean)

This variable is used to store the information whether a Start.req has not yet been confirmed by the MSAL1M State Machine (True) or not (False).

Go-Abort
(Boolean)

This variable is used to store the information that an error occurred forcing the MSCY1M State Machine to abort the connection.

Macros

CHANGE_DIAG_IND
(

MSCY1M_New Slave Diag.ind(Rem_Add)

if (MSAL1M_Started = TRUE)

MSAL1M_Change Diag.req(Rem_Add)

)

GO_ABORT
(

DTrans = 0

Go_Abort = TRUE

BInp = Nil

)

START_MSAL1M
(

```
if ((SPara.DPV1_Supported= TRUE)
 && (Wait_for_Start_con = FALSE)
  && (MSAL1M_Started = FALSE))
    MSAL1M_Start.req(Rem_Add),
    MSAL1M_Started = TRUE,
    MSAC1M_Start.req(Rem_Add)
)
```

9.7.3 MSCY1M state table

Table 91 contains the complete description of the MSCY1M state machine.

Table 91 – MSCY1M state table

#	Current state	Event / condition => action	Next state
1	POWER-ON	MSCY1M MInit MS0.req (Rem Add) => BDiag:= NIL, SDiag:= 0, DTrans:= 0 Go_Abort:= FALSE MSAL1M_Started:= FALSE ExtPrmBsy:= FALSE MSCY1M MInit MS0.cnf(Rem Add)	STOP
2	STOP	MSCY1M Abort.req (Rem Add) => ignore	STOP
3	STOP	MSAL1 Abort.ind (Rem_Add) => ignore	STOP
4	STOP	MSCY1M Start Slave Handler.req (Rem Add) /SI_Para_Exist = FALSE || SPara.Active = FALSE => MSCY1M Start Slave Handler.cnf(Rem Add)	DEACT
5	STOP	MSCY1M Start Slave Handler.req (Rem Add) /SI_Para_Exist = TRUE && SPara.Active = TRUE => SDiag:= 1, BInput:= Nil, BOutput:= Nil BDiag.Deactivated:= FALSE BDiag.Station_Non_Existent:= TRUE MSCY1M Start Slave Handler.cnf(Rem Add)	DIAG1
6	STOP	MSCY1M Get Slave Diag.req (Rem Add) => MSCY1M Get Slave Diag.cnf(-) (Rem Add)	STOP
7	STOP	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => MSCY1M Set Output.cnf(-) (Rem Add, Slot Number)	STOP
8	STOP	MSCY1M Get Input.req (Rem Add, Slot_Number) => MSCY1M Get Input.cnf(-) (Rem Add)	STOP
9	DEACT	MSCY1M Abort.req (Rem Add) => ignore	DEACT
10	DEACT	MSAL1 Abort.ind (Rem_Add) => ignore	DEACT

#	Current state	Event / condition => action	Next state
11	DEACT	MSCY1M Stop Slave Handler.req (Rem Add) => MSCY1M Stop Slave Handler.cnf(Rem Add)	STOP
12	DEACT	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) /SI_Para_Exist = FALSE || SPara.Active = FALSE => Diag:=FALSE, No_AClr:=TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DEACT
13	DEACT	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) /SI_Para_Exist = TRUE && SPara.Active = TRUE => SDiag:= 1, BInput:= Nil, BOutput:= Nil BDiag.Deactivated:= FALSE BDiag.Station_Non_Existent:= TRUE DMPMM1 Slave Diag.req (Rem Add)	WDIAG1
14	DEACT	MSCY1M Get Slave Diag.req (Rem Add) => MSCY1M Get Slave Diag.cnf(-) (Rem Add)	DEACT
15	DEACT	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => MSCY1M Set Output.cnf(-) (Rem Add, Slot Number)	DEACT
16	DEACT	MSCY1M Get Input.req (Rem Add, Slot_Number) => MSCY1M Get Input.cnf(-) (Rem Add)	DEACT
17	DIAG1	MSCY1M Abort.req (Rem Add) => ignore	DIAG1
18	DIAG1	MSAL1 Abort.ind (Rem_Add) => ignore	DIAG1
19	DIAG1	MSCY1M Stop Slave Handler.req (Rem Add) => BDiag.Deactivated:= TRUE, SDiag:= 0 MSCY1M Stop Slave Handler.cnf(Rem Add)	STOP
20	DIAG1	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) => ExtPrmBsy:=FALSE DMPMM1 Slave Diag.req (Rem Add)	WDIAG1
21	DIAG1	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	DIAG1
22	DIAG1	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	DIAG1
23	DIAG1	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	DIAG1
24	WDIAG1	MSCY1M Abort.req (Rem Add) => ignore	WDIAG1
25	WDIAG1	MSAL1 Abort.ind (Rem_Add) => ignore	WDIAG1
26	WDIAG1	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	WDIAG1

#	Current state	Event / condition => action	Next state
27	WDIAG1	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	WDIAG1
28	WDIAG1	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	WDIAG1
29	WDIAG1	DMPMM1 Slave Diag.cnf(+) (Rem Add, Diag Data) /SPara.Active = FALSE => Diag:=FALSE, No_AClr:=TRUE BDiag.Deactivated:= TRUE, SDiag:= 0 MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DEACT
30	WDIAG1	DMPMM1 Slave Diag.cnf(-) (Rem Add, Status) /SPara.Active = FALSE => Diag:=FALSE, No_AClr:=TRUE BDiag.Deactivated:= TRUE, SDiag:= 0 MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DEACT
31	WDIAG1	DMPMM1 Slave Diag.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status=NA => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Station_Non_Existent:= TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG1
32	WDIAG1	DMPMM1 Slave Diag.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status=RE/RS/RR/UE/NR => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Invalid_Slave_Response:= TRUE BDiag.Station_Non_Existent:= FALSE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG1
33	WDIAG1	DMPMM1 Slave Diag.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status=DS => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG1
34	WDIAG1	DMPMM1 Slave Diag.cnf(+) (Rem Add, Diag Data) /SPara.Active = TRUE && Diag_Data.Master_Lock = TRUE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag:= Diag_Data MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG1
35	WDIAG1	DMPMM1 Slave Diag.cnf(+) (Rem Add, Diag Data) /SPara.Active = TRUE && Diag_Data.Master_Lock = FALSE && Diag_Data.Master_Add <> invalid && SPara.Prm_Data.DPV1_Enable=TRUE && SPara.DPV1_Supported=TRUE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag:= Diag_Data MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	UNLCK
36	WDIAG1	DMPMM1 Slave Diag.cnf(+) (Rem Add, Diag Data) /SPara.Active = TRUE && Diag_Data.Master_Lock = FALSE && (Diag_Data.Master_Add = invalid || SPara.Prm_Data.DPV1_Enable=FALSE || SPara.DPV1_Supported=FALSE) => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr SPara.New_Prm:= FALSE BDiag:= Diag_Data, CHANGE_DIAG_IND MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	PRM
37	PRM	MSCY1M Abort.req (Rem Add) => ignore	PRM

#	Current state	Event / condition => action	Next state
38	PRM	MSAL1 Abort.ind (Rem_Add) => ignore	PRM
39	PRM	MSCY1M Stop Slave Handler.req (Rem Add) => BDiag.Deactivated:= TRUE, SDiag:= 0 MSCY1M Stop Slave Handler.cnf(Rem Add)	STOP
40	PRM	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) / ExtPrmBsy=FALSE => Prm_Data:= SPara .Prm_Data Prm_Data.Lock_Req:= TRUE Prm_Data.Unlock_Req:= FALSE DMPMM1 Set Prm.req (Rem Add, Prm Data)	WPRM
41	PRM	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	PRM
42	PRM	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	PRM
43	PRM	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	PRM
44	WPRM	MSCY1M Abort.req (Rem Add) => ignore	WPRM
45	WPRM	MSAL1 Abort.ind (Rem_Add) => ignore	WPRM
46	WPRM	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	WPRM
47	WPRM	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	WPRM
48	WPRM	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	WPRM
49	WPRM	DMPMM1 Set Prm.cnf(+) (Rem Add) /SPara.Active = FALSE => Diag:=FALSE, No_AClr:=TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	UNLCK
50	WPRM	DMPMM1 Set Prm.cnf(-) (Rem Add, Status) /SPara.Active = FALSE => Diag:=FALSE, No_AClr:=TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	UNLCK
51	WPRM	DMPMM1 Set Prm.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = DS => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	PRM

#	Current state	Event / condition => action	Next state
52	WPRM	DMPMM1 Set Prm.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = NA => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Station_Non_Existent:= TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG1
53	WPRM	DMPMM1 Set Prm.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = RE/RS/RR/UE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Invalid_Slave_Response:= TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG1
54	WPRM	DMPMM1 Set Prm.cnf(+) (Rem Add) /SPara.Active = TRUE && ExtPrmFlag = FALSE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	CFG
55	CFG	MSCY1M Abort.req (Rem Add) => ignore	CFG
56	CFG	MSAL1 Abort.ind (Rem_Add) => ignore	CFG
57	CFG	MSCY1M Stop Slave Handler.req (Rem Add) => Prm_Data:= SPara .Prm_Data Prm_Data.Unlock_Req:= TRUE DMPMM1 Set Prm.req (Rem Add, Prm Data)	SUNLCK
58	CFG	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) => Cfg_Data:=SPara .Cfg_Data DMPMM1 Chk Cfg.req (Rem Add, Cfg Data)	WCFG
59	CFG	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	CFG
60	CFG	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	CFG
61	CFG	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	CFG
62	WCFG	MSCY1M Abort.req (Rem Add) => ignore	WCFG
63	WCFG	MSAL1 Abort.ind (Rem_Add) => ignore	WCFG
64	WCFG	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	WCFG
65	WCFG	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	WCFG
66	WCFG	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	WCFG

#	Current state	Event / condition => action	Next state
67	WCFG	DMPMM1 Chk Cfg.cnf(+) (Rem Add) /SPara.Active = FALSE => Diag:=FALSE, No_AClr:=TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	UNLCK
68	WCFG	DMPMM1 Chk Cfg.cnf(-) (Rem Add, Status) /SPara.Active = FALSE => Diag:=FALSE, No_AClr:=TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	UNLCK
69	WCFG	DMPMM1 Chk Cfg.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = DS => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	CFG
70	WCFG	DMPMM1 Chk Cfg.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = NA => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Station_Non_Existent:= TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG1
71	WCFG	DMPMM1 Chk Cfg.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = RE/RS/RR/UE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Invalid_Slave_Response:= TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG1
72	WCFG	DMPMM1 Chk Cfg.cnf(+) (Rem Add) /SPara.Active = TRUE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr MSAL1M_Started:= FALSE Delay_Count:= SPara .Diag_Upd_Delay Wait_for_Start_con:= FALSE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2
73	DIAG2	MSCY1M Abort.req (Rem Add) => GO_ABORT	DIAG2
74	DIAG2	MSAL1 Start.cnf (Rem Add) => Wait_for_Start_con:=FALSE MSCY1M_Started.ind(Rem_Add)	DIAG2
75	DIAG2	MSAL1 Abort.ind (Rem_Add) => GO_ABORT	DIAG2
76	DIAG2	MSCY1M Stop Slave Handler.req (Rem Add) /SPara.DPV1_Supported = FALSE => Go_Abort:= FALSE, BInput:= Nil DTrans:= 0 Prm_Data:= SPara .Prm_Data Prm_Data.Unlock_Req:= TRUE DMPMM1 Set Prm.req (Rem Add, Prm Data)	SUNLCK
77	DIAG2	MSCY1M Stop Slave Handler.req (Rem Add) /SPara.DPV1_Supported = TRUE => Go_Abort:= FALSE, BInput:= Nil DTrans:= 0 MSAL1 Stop.req (Rem Add)	WSTPCS
78	DIAG2	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) /Go_Abort = FALSE && Wait_for_Start_con = FALSE => DMPMM1 Slave Diag.req (Rem Add)	WDIAG2

#	Current state	Event / condition => action	Next state
79	DIAG2	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) /Go_Abort = FALSE && Wait_for_Start_con = TRUE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2
80	DIAG2	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) /Go_Abort=TRUE && SPara.DPV1_Supported = FALSE => Go_Abort:= FALSE No_AClr:= SPara_.Ignore_AClr OR NOT SPara.Active Prm_Data:= SPara .Prm_Data Prm_Data.Unlock_Req:= TRUE DMPMM1 Set Prm.req (Rem Add, Prm Data)	WUNLCK
81	DIAG2	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) /Go_Abort=TRUE && SPara.DPV1_Supported = TRUE => Go_Abort:= FALSE, Diag:= TRUE No_AClr:= SPara_.Ignore_AClr OR NOT SPara.Active MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr) MSAL1 Stop.req (Rem Add)	WSTPCC
82	DIAG2	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	DIAG2
83	DIAG2	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	DIAG2
84	DIAG2	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	DIAG2
85	WDIAG2	MSCY1M Abort.req (Rem Add) => GO_ABORT	WDIAG2
86	WDIAG2	MSAL1 Abort.ind (Rem_Add) => GO_ABORT	WDIAG2
87	WDIAG2	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	WDIAG2
88	WDIAG2	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	WDIAG2
89	WDIAG2	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	WDIAG2
90	WDIAG2	DMPMM1 Slave Diag.cnf(+) (Rem Add, Diag Data) /SPara.Active = FALSE => Diag:=FALSE, No_AClr:=FALSE BDiag:= Diag_Data, GO_ABORT MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2
91	WDIAG2	DMPMM1 Slave Diag.cnf(-) (Rem Add, Status) /SPara.Active = FALSE => GO_ABORT, Diag:=FALSE, No_AClr:=FALSE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2

#	Current state	Event / condition => action	Next state
92	WDIAG2	DMPMM1 Slave Diag.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = DS => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2
93	WDIAG2	DMPMM1 Slave Diag.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = NA && SPara_.NA_To_Abort = FALSE => Diag:=TRUE,No_AClr:=SPara_.Ignore_AClr BDiag.Station_Non_Existent:= TRUE SDiag:= 1, DTrans:= 0 MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2
94	WDIAG2	DMPMM1 Slave Diag.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = NA && SPara_.NA_To_Abort = TRUE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr, BDiag.Station_Non_Existent:= TRUE, SDiag:= 1, GO_ABORT MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2
95	WDIAG2	DMPMM1 Slave Diag.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = RE/RS/RR/UE/NR => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr, BDiag.Station_Non_Existent:= TRUE, BDiag.Invalid_Slave_Response:= TRUE, SDiag:= 1, GO_ABORT MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2
96	WDIAG2	DMPMM1 Slave Diag.cnf(+) (Rem Add, Diag Data) /SPara.Active = TRUE && (Diag_Data.Prm_Req = TRUE || Diag_Data.Master_Lock = TRUE) && Delay_Count > 0 => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr Delay_Count:= Delay_Count-1, BDiag:= Diag_Data BInput:= 0, SDiag:= 1, DTrans:= 0 MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2
97	WDIAG2	DMPMM1 Slave Diag.cnf(+) (Rem Add, Diag Data) /SPara.Active = TRUE && (Diag_Data.Prm_Req = TRUE || Diag_Data.Master_Lock = TRUE) && Delay_Count = 0 => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag:= Diag_Data, SDiag:= 1, GO_ABORT MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2
98	WDIAG2	DMPMM1 Slave Diag.cnf(+) (Rem Add, Diag Data) /SPara.Active = TRUE && Diag_Data.Prm_Req = FALSE && Diag_Data.Master_Lock = FALSE && Diag_Data.Station_Not_Ready = TRUE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr SDiag:= 1, DTrans:= 0, BInput:= Nil BDiag:= Diag_Data MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2
99	WDIAG2	DMPMM1 Slave Diag.cnf(+) (Rem Add, Diag Data) /SPara.Active = TRUE && Diag_Data.Prm_Req = FALSE && Diag_Data.Master_Lock = FALSE && Diag_Data.Station_Not_Ready = FALSE && Diag_Data.Stat_Diag = TRUE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr START_MSAL1M, SDiag:= 1, DTrans:= 0 BInput:= Nil, BDiag:= Diag_Data MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2
100	WDIAG2	DMPMM1 Slave Diag.cnf(+) (Rem Add, Diag Data) /SPara.Active = TRUE && Diag_Data.Prm_Req = FALSE && Diag_Data.Master_Lock = FALSE && Diag_Data.Station_Not_Ready = FALSE && Diag_Data.Stat_Diag=FALSE && DTrans = 0 => Diag:=TRUE, No_AClr:=TRUE SDiag:= Diag_Data.Ext_Diag, Delay_Count:= 0 BDiag:= Diag_Data, START_MSAL1M, CHANGE_DIAG_IND MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA

#	Current state	Event / condition => action	Next state
101	WDIAG2	DMPMM1 Slave Diag.cnf(+) (Rem Add, Diag Data) /SPara.Active = TRUE && Diag_Data.Prm_Req = FALSE && Diag_Data.Master_Lock = FALSE && Diag_Data.Station_Not_Ready = FALSE && Diag_Data.Stat_Diag = FALSE && DTrans = 1 => Diag:=TRUE, No_AClr:=TRUE SDIAG:= Diag_Data.Ext_Diag, CHANGE_DIAG_IND BDiag:= Diag_Data MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
102	DATA	MSCY1M Abort.req (Rem Add) => GO_ABORT	DATA
103	DATA	MSAL1 Start.cnf (Rem Add) => Wait_for_Start_con:= FALSE, START_MSAL1M MSCY1M_Started.ind(Rem_Add)	DATA
104	DATA	MSAL1 Abort.ind (Rem_Add) => GO_ABORT	DATA
105	DATA	MSCY1M Stop Slave Handler.req (Rem Add) /SPara.DPV1_Supported = FALSE => Go_Abort:= FALSE, BInput:= Nil DTrans:= 0 Prm_Data:= SPara .Prm_Data Prm_Data.Unlock_Req:= TRUE DMPMM1 Set Prm.req (Rem Add, Prm Data)	SUNLCK
106	DATA	MSCY1M Stop Slave Handler.req (Rem Add) /SPara.DPV1_Supported = TRUE => Go_Abort:= FALSE, BInput:=Nil, DTrans:= 0 MSAL1 Stop.req (Rem Add)	WSTPCS
107	DATA	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) /Go_Abort = FALSE && Wait_for_Start_con = FALSE && Output_Clear = TRUE && ((SPara.DPV1_Supported = FALSE && SPara.New_Prm = FALSE) || SPara.DPV1_Supported = TRUE) && SPara.Fail_Safe = FALSE => Outp_Data:=0 DMPMM1 Data Exchange.req (Rem Add, Outp Data)	WDATA
108	DATA	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) /Go_Abort = FALSE && Wait_for_Start_con = FALSE && Output_Clear = TRUE && ((SPara.DPV1_Supported = FALSE && SPara.New_Prm = FALSE) || SPara.DPV1_Supported = TRUE) && SPara.Fail_Safe = TRUE => Outp_Data.len:=0 DMPMM1 Data Exchange.req (Rem Add, Outp Data)	WDATA
109	DATA	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) /Go_Abort = FALSE && Wait_for_Start_con = FALSE && Output_Clear = FALSE && ((SPara.DPV1_Supported = FALSE && SPara.New_Prm = FALSE) || SPara.DPV1_Supported = TRUE) => Outp_Data:=BOutput DMPMM1 Data Exchange.req (Rem Add, Outp Data)	WDATA
110	DATA	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) /Go_Abort = FALSE && SPara.DPV1_Supported = FALSE && SPara.New_Prm = TRUE => Prm_Data:= SPara .Prm_Data Prm_Data.Lock_Req:= TRUE Prm_Data.Unlock_Req:= FALSE DMPMM1 Set Prm.req (Rem Add, Prm Data)	WDATA
111	DATA	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) /Go_Abort = FALSE && Wait_for_Start_con = TRUE => Diag:=TRUE, No_AClr:=TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA

#	Current state	Event / condition => action	Next state
112	DATA	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) /Go_Abort=TRUE && (SPara.DPV1_Supported = FALSE|| SPara_PrmCmd_Supported = TRUE) => Go_Abort:= FALSE, Prm_Data:= SPara .Prm_Data Prm_Data.Unlock_Req:= TRUE DMPMM1 Set Prm.req (Rem Add, Prm Data)	WUNLCK
113	DATA	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) /Go_Abort=TRUE && SPara.DPV1_Supported = TRUE && SPara_PrmCmd_Supported = FALSE => Go_Abort:= FALSE, Diag:= TRUE No_AClr:= SPara_.Ignore_AClr OR NOT SPara.Active MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr) MSAL1 Stop.req (Rem Add)	WSTPCC
114	DATA	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	DATA
115	DATA	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	DATA
116	DATA	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	DATA
117	WDATA	MSCY1M Abort.req (Rem Add) => GO_ABORT	WDATA
118	WDATA	MSAL1 Abort.ind (Rem_Add) => GO_ABORT	WDATA
119	WDATA	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	WDATA
120	WDATA	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	WDATA
121	WDATA	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	WDATA
122	WDATA	DMPMM1 Data Exchange.cnf(+) (Rem Add, Diag Flag, Inp Data) /SPara.Active = FALSE => GO_ABORT, Diag:=FALSE, No_AClr:=TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
123	WDATA	DMPMM1 Data Exchange.cnf(-) (Rem Add, Status) /SPara.Active = FALSE => GO_ABORT, Diag:=FALSE, No_AClr:=TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
124	WDATA	DMPMM1 Data Exchange.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = DS => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr DTrans:= 0, BInput:= Nil MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA

#	Current state	Event / condition => action	Next state
125	WDATA	DMPMM1 Data Exchange.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = NA && SPara_.NA_To_Abort = FALSE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Station_Non_Existent:= TRUE, SDiag:= 1 MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
126	WDATA	DMPMM1 Data Exchange.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status=NA && SPara_.NA_To_Abort = TRUE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Station_Non_Existent:= TRUE, SDiag:= 1 GO_ABORT MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
127	WDATA	DMPMM1 Data Exchange.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = RE/RS/RR/UE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Invalid_Slave_Response:= TRUE BDiag.Station_Non_Existent:= FALSE SDiag:= 1, GO_ABORT MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
128	WDATA	DMPMM1 Data Exchange.cnf(+) (Rem Add, Diag Flag, Inp Data) /SPara.Active = TRUE && Diag_Flag = FALSE && Inp_Data.len = Exp_Inp_Len => Diag:=FALSE, No_AClr:=TRUE BDiag.Station_Non_Existent:= FALSE BInput:= Inp_Data, DTrans:= 1 SDiag:= BDiag.Ext_Diag MSCY1M_New_Input.ind(Rem_Add) MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
129	WDATA	DMPMM1 Data Exchange.cnf(+) (Rem Add, Diag Flag, Inp Data) /SPara.Active = TRUE && Inp_Data.len <> Exp_Inp_Len && (Inp_Data.len <> 1 ‖ Exp_Inp_Len <> 0 ‖ Diag_Flag = FALSE) => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Invalid_Slave_Response:= TRUE BDiag.Station_Non_Existent:= FALSE SDiag:= 1, GO_ABORT MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
130	WDATA	DMPMM1 Data Exchange.cnf(+) (Rem Add, Diag Flag, Inp Data) /SPara.Active = TRUE && Diag_Flag = TRUE && Inp_Data.len = Exp_Inp_Len => Diag:=FALSE, No_AClr:=TRUE BDiag.Station_Non_Existent:= FALSE BInput:= Inp_Data, DTrans:= 1 SDiag:= BDiag.Ext_Diag MSCY1M_New_Input.ind(Rem_Add) MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2
131	WDATA	DMPMM1 Data Exchange.cnf(+) (Rem Add, Diag Flag, Inp Data) /SPara.Active = TRUE && Diag_Flag = TRUE && Inp_Data.len = 1 && Exp_Inp_Len = 0 => Diag:=FALSE, No_AClr:=TRUE BDiag.Station_Non_Existent:= FALSE, DTrans:= 1 SDiag:= BDiag.Ext_Diag MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG2
132	WDATA	DMPMM1 Set Prm.cnf(+) (Rem Add) /SPara.Active = FALSE => GO_ABORT, Diag:=FALSE, No_AClr:=TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
133	WDATA	DMPMM1 Set Prm.cnf(-) (Rem Add, Status) /SPara.Active = FALSE => GO_ABORT, Diag:=FALSE, No_AClr:=TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA

#	Current state	Event / condition => action	Next state
134	WDATA	DMPMM1 Set Prm.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = DS => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
135	WDATA	DMPMM1 Set Prm.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = NA && SPara_.NA_To_Abort = FALSE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Station_Non_Existent:= TRUE, SDiag:= 1 MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
136	WDATA	DMPMM1 Set Prm.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = NA && SPara_.NA_To_Abort = TRUE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Station_Non_Existent:= TRUE, SDiag:= 1 GO_ABORT MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
137	WDATA	DMPMM1 Set Prm.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = RE/RS/RR/UE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Invalid_Slave_Response:= TRUE BDiag.Station_Non_Existent:= FALSE SDiag:= 1, GO_ABORT MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
138	WDATA	DMPMM1 Set Prm.cnf(+) (Rem Add) /SPara.Active = TRUE => Diag:=TRUE, No_AClr:=TRUE SPara.New_Prm:= FALSE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DATA
139	UNLCK	MSCY1M Abort.req (Rem Add) => ignore	UNLCK
140	UNLCK	MSAL1 Abort.ind (Rem_Add) => ignore	UNLCK
141	UNLCK	MSCY1M Stop Slave Handler.req (Rem Add) => Prm_Data:= SPara .Prm_Data Prm_Data.Unlock_Req:= TRUE DMPMM1 Set Prm.req (Rem Add, Prm Data)	SUNLCK
142	UNLCK	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) => Prm_Data:= SPara .Prm_Data Prm_Data.Unlock_Req:= TRUE DMPMM1 Set Prm.req (Rem Add, Prm Data)	WUNLCK
143	UNLCK	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	UNLCK
144	UNLCK	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	UNLCK
145	UNLCK	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	UNLCK
146	WUNLCK	MSCY1M Abort.req (Rem Add) => ignore	WUNLCK

#	Current state	Event / condition => action	Next state
147	WUNLCK	MSAL1 Abort.ind (Rem_Add) => ignore	WUNLCK
148	WUNLCK	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	WUNLCK
149	WUNLCK	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	WUNLCK
150	WUNLCK	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	WUNLCK
151	WUNLCK	DMPMM1 Set Prm.cnf(+) (Rem Add) /SPara.Active = TRUE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG1
152	WUNLCK	DMPMM1 Set Prm.cnf(+) (Rem Add) /SPara.Active = FALSE => Diag:=FALSE, No_AClr:=TRUE, SDiag:= 0 BDiag.Deactivated:= TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DEACT
153	WUNLCK	DMPMM1 Set Prm.cnf(-) (Rem Add, Status) /SPara.Active = TRUE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG1
154	WUNLCK	DMPMM1 Set Prm.cnf(-) (Rem Add, Status) /SPara.Active = FALSE => Diag:=FALSE, No_AClr:=TRUE, SDiag:= 0 BDiag.Deactivated:= TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DEACT
155	SUNLCK	MSCY1M Abort.req (Rem Add) => ignore	SUNLCK
156	SUNLCK	MSAL1 Abort.ind (Rem_Add) => ignore	SUNLCK
157	SUNLCK	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	SUNLCK
158	SUNLCK	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	SUNLCK
159	SUNLCK	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	SUNLCK
160	SUNLCK	DMPMM1 Set Prm.cnf(+) (Rem Add) => SDiag:= 0, BDiag.Deactivated:= TRUE MSCY1M Stop Slave Handler.cnf(Rem Add)	STOP
161	SUNLCK	DMPMM1 Set Prm.cnf(-) (Rem Add, Status) => SDiag:= 0, BDiag.Deactivated:= TRUE MSCY1M Stop Slave Handler.cnf(Rem Add)	STOP

#	Current state	Event / condition => action	Next state
162	WSTPCC	MSAL1 Start.cnf (Rem Add) => MSAL1M_Started:= FALSE	WSTPCC
163	WSTPCC	MSAL1 Stop.cnf (Rem Add) /MSAL1M_Started = TRUE => MSCY1M Stopped.ind (Rem Add)	UNLCK
164	WSTPCC	MSAL1 Stop.cnf (Rem Add) /MSAL1M_Started = FALSE => ignore	UNLCK
165	WSTPCC	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr OR NOT SPara.Active MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	WSTPCC
166	WSTPCC	MSCY1M Abort.req (Rem Add) => ignore	WSTPCC
167	WSTPCC	MSAL1 Abort.ind (Rem_Add) => ignore	WSTPCC
168	WSTPCC	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	WSTPCC
169	WSTPCC	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	WSTPCC
170	WSTPCC	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	WSTPCC
171	WSTPCS	MSAL1 Start.cnf (Rem Add) => MSAL1M_Started:= FALSE	WSTPCS
172	WSTPCS	MSAL1 Stop.cnf (Rem Add) /MSAL1M_Started = TRUE => Prm_Data:= SPara .Prm_Data Prm_Data.Unlock_Req:= TRUE MSCY1M Stopped.ind (Rem Add) DMPMM1 Set Prm.req (Rem Add, Prm Data)	SUNLCK
173	WSTPCS	MSAL1 Stop.cnf (Rem Add) /MSAL1M_Started = FALSE => Prm_Data:= SPara .Prm_Data Prm_Data.Unlock_Req:= TRUE DMPMM1 Set Prm.req (Rem Add, Prm Data)	SUNLCK
174	WSTPCS	MSCY1M Abort.req (Rem Add) => ignore	WSTPCS
175	WSTPCS	MSAL1 Abort.ind (Rem_Add) => ignore	WSTPCS
176	WSTPCS	MSCY1M Get Slave Diag.req (Rem Add) => Diag_Data:= BDiag MSCY1M Get Slave Diag.cnf(+) (Rem Add, Diag Data)	WSTPCS

#	Current state	Event / condition => action	Next state
177	WSTPCS	MSCY1M Set Output.req (Rem Add, Slot Number, Output Data) => BOutput(Slot_Number):= Output_Data MSCY1M Set Output.cnf(+) (Rem Add, Slot Number)	WSTPCS
178	WSTPCS	MSCY1M Get Input.req (Rem Add, Slot_Number) => Input_Data:= BInput(Slot_Number) MSCY1M Get Input.cnf(+) (Rem Add, Input Data)	WSTPCS
179	ANY-STATE	MSCY1M Reset.req (Rem Add) => MSCY1M Reset.cnf (Rem Add)	POWER-ON
180	WPRM	DMPMM1 Set Prm.cnf(+) (Rem Add) /SPara.Active = TRUE && ExtPrmFlag = TRUE => ExtPrmBsy:=TRUE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	PRM
181	PRM	MSCY1M Cont Slave Handler.req (Rem Add, Output Clear) / ExtPrmBsy=TRUE => Prm_Data:= SPara .ExtPrm_Data DMPMM1 Set ExtPrm.req (Rem Add, Prm Data)	WPRM
182	WPRM	DMPMM1 Set ExtPrm.cnf(+) (Rem Add) /SPara.Active = TRUE => Diag:=TRUE, No_AClr:=SPara_.Ignore_Aclr, ExtPrmBsy:=FALSE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	CFG
183	WPRM	DMPMM1 Set ExtPrm.cnf(+) (Rem Add) /SPara.Active = FALSE => Diag:=FALSE, No_AClr:=TRUE, ExtPrmBsy:=FALSE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	UNLCK
184	WPRM	DMPMM1 Set ExtPrm.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = DS => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	PRM
185	WPRM	DMPMM1 Set ExtPrm.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = NA => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Station_Non_Existent:= TRUE, ExtPrmBsy:=FALSE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG1
186	WPRM	DMPMM1 Set ExtPrm.cnf(-) (Rem Add, Status) /SPara.Active = TRUE && Status = RE/RS/RR/UE => Diag:=TRUE, No_AClr:=SPara_.Ignore_AClr BDiag.Invalid_Slave_Response:= TRUE, ExtPrmBsy:=FALSE MSCY1M Cont Slave Handler.cnf(Rem Add, Diag, No AClr)	DIAG1

9.8 MSAL1M

9.8.1 Primitive definitions

9.8.1.1 Primitives exchanged between FSPMM1 and MSAL1M

Table 92 shows the service primitives including their associated parameters issued by the FSPMM1 and received by the MSAL1M.

Table 92 – Primitives issued by FSPMM1 to MSAL1M

Primitive name	Source	Associated parameters	Functions
MInit.req	FSPMM1	(none)	Refer to FAL Service Definition in IEC 61158-5-3
Reset.req	FSPMM1	(none)	—
Abort.req	FSPMM1	(none)	—

Table 93 shows the service primitives including their associated parameters issued by the MSAL1M and received by the FSPMM1.

Table 93 – Primitives issued by MSAL1M to FSPMM1

Primitive name	Source	Associated parameters	Functions
MInit.cnf	MSAL1M	(none)	Refer to FAL Service Definition in IEC 61158-5-3
Reset.cnf	MSAL1M	(none)	—
Fault.ind	MSAL1M	(none)	—
Started.ind	MSAL1M	Alarm Limit	—
Stopped.ind	MSAL1M	(none)	—
Alarm Notification.ind	MSAL1M	Slot Number, Alarm Type, Seq Nr, Add Ack, Alarm Specifier, Alarm Data	—

9.8.1.2 Primitives exchanged between MSCY1M and MSAL1M

Table 94 shows the service primitives including their associated parameters issued by the MSCY1M and received by the MSAL1M.

Table 94 – Primitives issued by MSCY1M to MSAL1M

Primitive name	Source	Associated parameters	Functions
Start.req	MSCY1M	Rem Add	Start MS1-processing
Stop.req	MSCY1M	Rem Add	Stop MS1-processing
Change Diag.req	MSCY1M	Rem Add, Diag Data	Indicates change of diagnosis

Table 95 shows the service primitives including their associated parameters issued by the MSAL1M and received by the MSCY1M.

Table 95 – Primitives issued by MSAL1M to MSCY1M

Primitive name	Source	Associated parameters	Functions
Abort.ind	MSAL1M	Rem Add	MSAL1 detected an error
Start.cnf	MSAL1M	Rem Add	Start completed
Stop.cnf	MSAL1M	Rem Add	Stop completed

9.8.1.3 Parameters of MSAL1M primitives

The parameters used with the primitives exchanged between the MSAL1M and MSCY1M are described in Table 96.

Table 96 – Parameter used with primitives exchanged between MSAL1M and MSCY1M

Parameter name	Description
Rem Add	This parameter conveys the DL address of the assigned DP-slave.
Diag Data	This parameter conveys the new diagnosis information of a DP-slave

9.8.2 State machine description

The Alarm State Machine of a DP-master (Class 1) handles alarm messages, which have been indicated by a DP-slave.

When started the State Machine waits for being started by MSCY1M. The Start.req follows, when the assigned DP-slave has reached the state DATA-EXCH. MSAL1M will be stopped by MSCY1M if the corresponding DP-slave leaves the state DATA-EXCH.

The service Slave_Diag.cnf provides new Diag_Data to MSCY1M. MSCY1M copies the diagnosis information into the User data interface. Additionally MSCY1M hands over an Alarm in Ext_Diag_Data to MSAL1M.

Alarm-Requests are indicated to the User. The User has to confirm the Alarm-Indications with the service Alarm_Ack.

The MSAC1 State Machine indicates the termination of the alarm sequence with the service primitive Alarm_Ack.cnf.

Two types of alarm handling are provided by a DP-master (Class 1) the type mode and the sequence mode. Depending on the chosen mode the number of outstanding alarms per DP-slave differs. In type mode one pending alarm per type and per DP-slave is allowed. 6 different alarm types are defined. In sequence mode the alarm type is irrelevant. Only the number of simultaneously processed alarms for each DP-slave is limited to a value between 2 and 32.

Local variables

Alarm_Sequence
(Boolean)

This variable indicates whether:

only one alarm of a specific alarm type can be active at one time (type mode: Alarm_Sequence=False) or

several alarms (2 to 32) of any type can be active at one time (sequence mode: Alarm_Sequence=True).

Alarm_State_Table
(Unsigned8)

The Alarm_State_Table is a two dimensional array with 7 * 32 elements. Each element within the array stores information about the actual state of any alarm. The array is indexed with the alarm class calculated from the Alarm_Type and the Seq_Nr of the alarm.

Range
Table 97 shows the possible values for the actual states of alarms in the Alarm_State_Table.

Table 97 – Possible values in the Alarm_State_Table

Value	Meaning
idle	No service is being processed according to the actual index.
w_res	Wait for MSAL1_Alarm.res according to the actual index.
w_req	For this index (Alarm_Type,Seq_Nr) an Alarm_Acknowledge is stored in the Alarm_Ack_FIFO and MSAL1M waits for another MSAC1M_Alarm_Ack.cnf.
w_con	Wait for MSAC1M_Alarm_Ack.cnf according to the actual index.

Alarm_Limit
(Unsigned8)

This variable indicates the maximum number of parallel alarms of each Slave.

Range: 7..Bus_Para.Alarm_Max

Alarm_Decode
(Array 0..7 of Unsigned8)

Table for Decoding the number of parallel alarms.

Range:

Alarm_Decode[0]= 1
Alarm_Decode[1]= 2
Alarm_Decode[2]= 4
Alarm_Decode[3]= 8
Alarm_Decode[4]=12
Alarm_Decode[5]=16
Alarm_Decode[6]=24
Alarm_Decode[7]=32

Actual_Max_Alarm_Len
(Unsigned8)

Actual_Max_Alarm_Len contains the actual maximum length of an Alarm.

Range: 4..64

Alarm_Count
(Unsigned8)

This counter contains the number of alarms which have actually been sent by the Slave. The counter is only used in the sequence mode.

Range: 0..32

Waiting_For_Alarm_Ack
(Boolean)

This variable shows, whether the Alarm State Machine is actually waiting for the confirmation of any alarm acknowledge (True) or not (False).

Alarm_Ack_FIFO
The Alarm_Ack_FIFO is used to store the Alarm_Ack which are still to be sent. Each element of this alarm acknowledge FIFO has 3 Unsigned8 entries: Alarm_Type, Slot_Number and Seq_Nr. The FIFO shall be able to hold up to 32 respectively Bus_Para.Alarm_Max entries.

Alarm_Ack_FIFO_Filled
(Boolean)

This variable indicates whether the Alarm_Ack_FIFO contains at least one entry (True) or it is empty (False). The value of the variable is adapted with each access to the FIFO.

Act_Alarm_PDU
(Octet-String)

The structure Act_Alarm_PDU is temporarily used to store one Alarm while being evaluated.

Stored_Alarm_PDU
(Octet-String)

The structure Stored_Alarm_PDU is used to store exactly one Alarm-PDU. Up to one Alarm-PDU shall be saved locally during waiting for MSAC1M_Alarm_Ack.cnf of the according Seq_Nr, because only one channel for alarm acknowledges exists.

Actual_Enabled_Alarms
(Bitarea[0..7])

This variable indicates the type of alarms which are actually supported by the Master.

Bit 0 = reserved

Bit 1 = reserved

Bit 2 = Update_Alarm

Bit 3 = Status_Alarm

Bit 4 = Manufacturer_Specific_Alarm

Bit 5 = Diagnostic_Alarm

Bit 6 = Process_Alarm

Bit 7 = Pull_Plug_Alarm

Functions

Fill_Alarm_State_Table(State)
This function sets each entry of the two dimensional Alarm_State_Table to the given initial value of State. Valid parameters for State are idle, w_res, w_req and w_con. The function has no return value.

Reset_Alarm_Ack_FIFO()
This function resets the FIFO for alarm acknowledges. During this function all entries are cleared and the variable Alarm_Ack_FIFO_Filled is thereby set to False. The function has no return value.

Store_To_Alarm_Ack_FIFO(Alarm_Type, Slot_Number, Seq_Nr)
By means of this function one set of the 3 alarm recognition parameters Alarm_Type, Slot_Number and Seq_Nr is stored from the given parameters into the Alarm_Ack_FIFO. The function has no return value.

Load_From_Alarm_Ack_FIFO(Alarm_Type, Slot_Number, Seq_Nr)
By means of this function one set of the 3 alarm recognition parameters Alarm_Type, Slot_Number and Seq_Nr is read from the Alarm_Ack_FIFO and stored into the given parameters. The function has no return value.

AcIs(Alarm_Type)
This function calculates the Alarm_Type related index as return value as follows:

if (Alarm_Type = 1) return 0

if (Alarm_Type = 2) return 1

if (Alarm_Type = 3) return 2

if (Alarm_Type = 4) return 3

if (Alarm_Type = 5) return 4

if (Alarm_Type = 6) return 5

if (Alarm_Type >= 32) && (Alarm_Type <= 126) return 6

Macros

ALARM_ENABLED

(

((Alarm_Type=3 OR Alarm_Type=4)

&& Actual_Enabled_Alarms[7]=TRUE)

|| ((Alarm_Type=2) && Actual_Enabled_Alarms[6]=TRUE)

|| ((Alarm_Type=1) && Actual_Enabled_Alarms[5]=TRUE)

|| ((Alarm_Type>=32 && Alarm_Type<=126)

&& Actual_Enabled_Alarms[4]=TRUE)

|| ((Alarm_Type=5) && Actual_Enabled_Alarms[3]=TRUE)

|| ((Alarm_Type=6) && Actual_Enabled_Alarms[2]=TRUE)

)

9.8.3 MSAL1M state table

Table 98 contains the complete description of the MSAL1M state machine.

Table 98 – MSAL1M state table

#	Current state	Event / condition => action	Next state
1	POWER-ON	MSAL1M_Minit_MS1.req(Rem_Add) =>MSAL1M_Minit_MS1.cnf(Rem_Add)	WSTART
2	POWER-ON	MSAL1M_Abort.req(Rem_Add) =>Ignore	POWER-ON
3	POWER-ON	MSAL1M_Alarm_Ack.req(Rem_Add, Alarm_Type, Slot_Number, Seq_Nr) =>MSAL1M_Alarm_Ack.cnf(-)(Rem_Add, Status:=Not_Initialized)	POWER-ON
4	WSTART	MSAL1M_Start.req(Rem_Add) /Alarm_Mode = 0 =>Alarm_Sequence:= False Alarm_Limit:= 7 Actual_Enabled_Alarms:= Enabled_Alarms Actual_Max_Alarm_Len:= Max_Alarm_Len Fill_Alarm_State_Table(idle) Reset_Alarm_Ack_FIFO() Alarm_Count:= 0 Clear Stored_Alarm_PDU Waiting_For_Alarm_Ack:= False MSAL1M_Started.ind(Rem_Add, Alarm_Limit) MSAL1M_Start.cnf(Rem_Add)	W-DIA-EVENT

#	Current state	Event / condition => action	Next state
5	WSTART	MSAL1M_Start.req(Rem_Add) /Alarm_Mode <> 0 =>Alarm_Sequence:= True Alarm_Limit:= min(Bus_Para.Alarm_Max, Alarm_Decode[Alarm_Mode]) Actual_Enabled_Alarms:= Enabled_Alarms Actual_Max_Alarm_Len:= Max_Alarm_Len Fill_Alarm_State_Table(idle) Reset_Alarm_Ack_FIFO() Alarm_Count:= 0 Clear Stored_Alarm_PDU Waiting_For_Alarm_Ack:= False MSAL1M_Started.ind(Rem_Add, Alarm_Limit) MSAL1M_Start.cnf(Rem_Add)	W-DIA-EVENT
6	WSTART	MSAL1M_Stop.req(Rem_Add) =>MSAL1M_Fault.ind	POWER-ON
7	WSTART	MSAL1M_Change_Diag.req(Rem_Add, Diag_Block) =>MSAL1M_Fault.ind	WSTART
8	WSTART	MSAL1M_Abort.req(Rem_Add) =>ignore	WSTART
9	WSTART	MSAL1M_Alarm_Ack.req(Rem_Add, Alarm_Type, Slot_Number, Seq_Nr) =>MSAL1M_Alarm_Ack.cnf(-)(Rem_Add, Status:=Not_Started)	WSTART
10	WSTART	MSAC1M_Alarm_Ack.cnf(+)(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr) =>MSAL1M_Fault.ind	POWER-ON
11	W-DIA-EVENT	MSAL1M_Minit_MS1.req(Rem_Add) =>MSAL1M_Fault.ind	POWER-ON
12	W-DIA-EVENT	MSAL1M_Abort.req(Rem_Add) =>MSAL1M_Abort.ind(Rem_Add)	WSTART
13	W-DIA-EVENT	MSAL1M_Start.req(Rem_Add) =>MSAL1M_Fault.ind	POWER-ON
14	W-DIA-EVENT	MSAL1M_Stop.req(Rem_Add) =>MSAL1M_Stopped.ind(Rem_Add) MSAL1M_Stop.cnf(Rem_Add)	WSTART
15	W-DIA-EVENT	MSAL1M_Change_Diag.req(Rem_Add, Diag_Block) =>Act_Alarm_PDU:= Alarm-PDU	CHK-DIA-ALARM
16	W-DIA-EVENT	MSAL1M_Alarm_Ack.req(Rem_Add, Alarm_Type, Slot_Number, Seq_Nr) /ALARM_ENABLED(Alarm_Type) = False =>MSAL1M_Alarm_Ack.cnf(-)(Rem_Add, Status:=Not_Enabled)	W-DIA-EVENT
17	W-DIA-EVENT	MSAL1M_Alarm_Ack.req(Rem_Add, Alarm_Type, Slot_Number, Seq_Nr) /ALARM_ENABLED(Alarm_Type) = True && Alarm_State_Table[Acls(Alarm_Type), Seq_Nr] <> w_res =>MSAL1M_Alarm_Ack.cnf(-)(Rem_Add, Status:=Alarm_Not_Pending)	W-DIA-EVENT
18	W-DIA-EVENT	MSAL1M_Alarm_Ack.req(Rem_Add, Alarm_Type, Slot_Number, Seq_Nr) /ALARM_ENABLED(Alarm_Type) = True && Alarm_Sequence = False && Alarm_State_Table[Acls(Alarm_Type), Seq_Nr] = w_res && Waiting_For_Alarm_Ack = False =>Alarm_State_Table[Acls(Alarm_Type), Seq_Nr]:= w_con Waiting_For_Alarm_Ack:= True MSAC1M_Alarm_Ack.req(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr)	W-DIA-EVENT

#	Current state	Event / condition => action	Next state
19	W-DIA-EVENT	MSAL1M_Alarm_Ack.req(Rem_Add, Alarm_Type, Slot_Number, Seq_Nr) /ALARM_ENABLED(Alarm-PDU.Alarm_Type) = True && Alarm_Sequence = False && Alarm_State_Table[Acls(Alarm_Type), Seq_Nr] = w_res && Waiting_For_Alarm_Ack = True =>Alarm_State_Table[Acls(Alarm_Type), Seq_Nr]:= w_req Store_To_Alarm_Ack_FIFO(Alarm_Type, Slot_Number, Seq_Nr)	W-DIA-EVENT
20	W-DIA-EVENT	MSAL1M_Alarm_Ack.req(Rem_Add, Alarm_Type, Slot_Number, Seq_Nr) /ALARM_ENABLED(Alarm_Type) = True && Alarm_Sequence = True && Alarm_State_Table[Acls(Alarm_Type), Seq_Nr] = w_res && Waiting_For_Alarm_Ack = False =>Alarm_State_Table[Acls(Alarm_Type), Seq_Nr]:= w_con Alarm_Count:= Alarm_Count-1 Waiting_For_Alarm_Ack:= True MSAC1M_Alarm_Ack.req(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr)	W-DIA-EVENT
21	W-DIA-EVENT	MSAL1M_Alarm_Ack.req(Rem_Add, Alarm_Type, Slot_Number, Seq_Nr) /ALARM_ENABLED(Alarm_Type) = True && Alarm_Sequence = True && Alarm_State_Table[Acls(Alarm_Type), Seq_Nr] = w_res && Waiting_For_Alarm_Ack = True =>Alarm_State_Table[Acls(Alarm_Type), Seq_Nr]:= w_req Store_To_Alarm_Ack_FIFO(Alarm_Type, Slot_Number, Seq_Nr)	W-DIA-EVENT
22	W-DIA-EVENT	MSAC1M_Alarm_Ack.cnf(+)(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr) /Waiting_For_Alarm_Ack = False =>MSAL1M_Fault.ind	POWER-ON
23	W-DIA-EVENT	MSAC1M_Alarm_Ack.cnf(+)(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr) /Waiting_For_Alarm_Ack = True && Alarm_State_Table[Acls(Alarm_Type), Seq_Nr] <> w_con =>MSAL1M_Abort.ind(Rem_Add) MSAL1M_Stopped.ind(Rem_Add)	WSTART
24	W-DIA-EVENT	MSAC1M_Alarm_Ack.cnf(+)(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr) /Waiting_For_Alarm_Ack = True && Alarm_State_Table[Acls(Alarm_Type), Seq_Nr] = w_con =>Alarm_State_Table[Acls(Alarm_Type), Seq_Nr]:= idle Waiting_For_Alarm_Ack:= False MSAL1M_Alarm_Ack.cnf(+)(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr)	SEND-NEXT-ALARM
25	CHK-DIA-ALARM	/Act_Alarm_PDU not present =>ignore	W-DIA-EVENT
26	CHK-DIA-ALARM	/Act_Alarm_PDU present && ALARM_ENABLED(Act_Alarm_PDU.Alarm_Type) = False =>MSAL1M_Abort.ind(Rem_Add) MSAL1M_Stopped.ind(Rem_Add)	WSTART
27	CHK-DIA-ALARM	/Act_Alarm_PDU present && ALARM_ENABLED(Act_Alarm_PDU.Alarm_Type) = True && Alarm_Sequence = False && Alarm_State_Table[Acls(Act_Alarm_PDU.Alarm_Type), Act_Alarm_PDU.Seq_Nr] = w_res/w_req =>MSAL1M_Abort.ind(Rem_Add) MSAL1M_Stopped.ind(Rem_Add)	WSTART

#	Current state	Event / condition => action	Next state
28	CHK-DIA-ALARM	/Act_Alarm_PDU present && ALARM_ENABLED(Act_Alarm_PDU.Alarm_Type) = True && Alarm_Sequence = False && Alarm_State_Table[Acls(Act_Alarm_PDU.Alarm_Type), Act_Alarm_PDU.Seq_Nr] = w_con && Stored_Alarm_PDU present =>MSAL1M_Abort.ind(Rem_Add) MSAL1M_Stopped.ind(Rem_Add)	WSTART
29	CHK-DIA-ALARM	/Act_Alarm_PDU present && ALARM_ENABLED(Act_Alarm_PDU.Alarm_Type) = True && Alarm_Sequence = False && Alarm_State_Table[Acls(Act_Alarm_PDU.Alarm_Type), Act_Alarm_PDU.Seq_Nr] = idle =>Alarm_State_Table[Acls(Act_Alarm_PDU.Alarm_Type), Act_Alarm_PDU.Seq_Nr]:= w_res MSAL1M_Alarm_Notification.ind(Rem_Add, Alarm_Type:=Act_Alarm_PDU.Alarm_Type, Slot_Number:=Act_Alarm_PDU.Slot_Number, Seq_Nr:=Act_Alarm_PDU.Seq_Nr, Alarm_Specifier:=Act_Alarm_PDU.Alarm_Specifier, Add_Ack:=Act_Alarm_PDU.Add_Ack, Alarm_Data:=Act_Alarm_PDU.Diagnostic_User_Data)	W-DIA-EVENT
30	CHK-DIA-ALARM	/Act_Alarm_PDU present && ALARM_ENABLED(Act_Alarm_PDU.Alarm_Type) = True && Alarm_Sequence = False && Alarm_State_Table[Acls(Act_Alarm_PDU.Alarm_Type), Act_Alarm_PDU.Seq_Nr] = w_con && Stored_Alarm_PDU not present =>Stored_Alarm_PDU:= Act_Alarm_PDU	W-DIA-EVENT
31	CHK-DIA-ALARM	/Act_Alarm_PDU present && ALARM_ENABLED(Act_Alarm_PDU.Alarm_Type) = True && Alarm_Sequence = True && Alarm_Count >= Alarm_Limit =>MSAL1M_Abort.ind(Rem_Add) MSAL1M_Stopped.ind(Rem_Add)	WSTART
32	CHK-DIA-ALARM	/Act_Alarm_PDU present && ALARM_ENABLED(Act_Alarm_PDU.Alarm_Type) = True && Alarm_Sequence = True && Alarm_Count < Alarm_Limit && Alarm_State_Table[Acls(Act_Alarm_PDU.Alarm_Type), Act_Alarm_PDU.Seq_Nr] = w_res/w_req =>MSAL1M_Abort.ind(Rem_Add) MSAL1M_Stopped.ind(Rem_Add)	WSTART
33	CHK-DIA-ALARM	/Act_Alarm_PDU present && ALARM_ENABLED(Act_Alarm_PDU.Alarm_Type) = True && Alarm_Sequence = True && Alarm_Count < Alarm_Limit && Alarm_State_Table[Acls(Act_Alarm_PDU.Alarm_Type), Act_Alarm_PDU.Seq_Nr] = idle =>Alarm_State_Table[Acls(Act_Alarm_PDU.Alarm_Type), Act_Alarm_PDU.Seq_Nr]:= w_res Alarm_Count:= Alarm_Count+1 MSAL1M_Alarm_Ack.ind(Rem_Add, Alarm_Type:=Act_Alarm_PDU.Alarm_Type, Slot_Number:=Act_Alarm_PDU.Slot_Number, Seq_Nr:=Act_Alarm_PDU.Seq_Nr, Alarm_Specifier:=Act_Alarm_PDU.Alarm_Specifier, Add_Ack:=Act_Alarm_PDU.Add_Ack, Alarm_Data:=Act_Alarm_PDU.Diagnostic_User_Data)	W-DIA-EVENT

#	Current state	Event / condition => action	Next state
34	CHK-DIA-ALARM	/Act_Alarm_PDU present && ALARM_ENABLED(Act_Alarm_PDU.Alarm_Type) = True && Alarm_Sequence = True && Alarm_Count < Alarm_Limit && Alarm_State_Table[Acls(Act_Alarm_PDU.Alarm_Type), Act_Alarm_PDU.Seq_Nr] = w_con && Stored_Act_Alarm_PDU present =>MSAL1M_Abort.ind(Rem_Add) MSAL1M_Stopped.ind(Rem_Add)	WSTART
35	CHK-DIA-ALARM	/Act_Alarm_PDU present && ALARM_ENABLED(Act_Alarm_PDU.Alarm_Type) = True && Alarm_Sequence = True && Alarm_Count < Alarm_Limit && Alarm_State_Table[Acls(Act_Alarm_PDU.Alarm_Type), Act_Alarm_PDU.Seq_Nr] = w_con && Stored_Alarm_PDU not present =>Stored_Alarm_PDU:= Act_Alarm_PDU Alarm_Count:= Alarm_Count+1	W-DIA-EVENT
36	SEND-NEXT-ALARM	/Stored_Alarm_PDU not present =>ignore	READ-NEXT-ACK
37	SEND-NEXT-ALARM	/Stored_Alarm_PDU present =>Alarm_State_Table[Acls(Stored_Alarm_PDU.Alarm_Type), Stored_Alarm_PDU.Seq_Nr]:= w_res Clear Stored_Alarm_PDU MSAL1M_Alarm_Notification.ind(Rem_Add, Alarm_Type:=Stored_Alarm_PDU.Alarm_Type, Slot_Number:=Stored_Alarm_PDU.Slot_Number, Seq_Nr:=Stored_Alarm_PDU.Seq_Nr, Alarm_Specifier:=Stored_Alarm_PDU.Alarm_Specifier, Add_Ack:=Stored_Alarm_PDU.Add_Ack, Alarm_Data:=Stored_Alarm_PDU.Diagnostic_User_Data)	READ-NEXT-ACK
38	READ-NEXT-ACK	/Alarm_Ack_FIFO_Filled = False =>ignore	W-DIA-EVENT
39	READ-NEXT-ACK	/Alarm_Ack_FIFO_Filled = True =>Load_From_Alarm_Ack_FIFO(New_Alarm_Type, New_Slot_Number, New_Seq_Nr)	SEND-NEXT-ACK
40	SEND-NEXT-ACK	/Alarm_State_Table[Acls(New_Alarm_Type), New_Seq_Nr] <> w_req =>MSAL1M_Abort.ind(Rem_Add) MSAL1M_Stopped.ind(Rem_Add)	WSTART
41	SEND-NEXT-ACK	/Alarm_State_Table[Acls(New_Alarm_Type), New_Seq_Nr] = w_req =>Alarm_State_Table[Acls(New_Alarm_Type), New_Seq_Nr]:= w_con Alarm_Count:= Alarm_Count-1 Waiting_For_Alarm_Ack:= True MSAC1M_Alarm_Ack.req(Rem_Add, Slot_Number:=New_Slot_Number, Alarm_Type:=New_Alarm_Type, Seq_Nr:=New_Seq_Nr)	W-DIA-EVENT
42	ANY-STATE	MSAL1M_Reset.req(Rem_Add) =>MSAL1M_Reset.cnf(Rem_Add)	POWER-ON

9.9 MSAC1M

9.9.1 Primitive definitions

9.9.1.1 Primitives exchanged between FSPMM1 and MSAC1M

Table 99 shows the service primitives including their associated parameters issued by the FSPMM1 and received by the MSAC1M.

Table 99 – Primitives issued by FSPMM1 to MSAC1M

Primitive name	Source	Associated parameters	Functions
Read.req	FSPMM1	Rem Add, Slot Number, Index, Length	
Write.req	FSPMM1	Rem Add, Slot Number, Index, Length, Data	
Alarm Ack.req	FSPMM1	Rem Add, Slot Number, Alarm Type, Seq Nr	

Table 100 shows the service primitives including their associated parameters issued by the MSAC1M and received by the FSPMM1.

Table 100 – Primitives issued by MSAC1M to FSPMM1

Primitive name	Source	Associated parameters	Functions
Reject.ind	MSAC1M	Rem Add, Status	
Read.cnf(+)	MSAC1M	Rem Add, Length, Data	
Read.cnf(-)	MSAC1M	Rem Add, Error Decode, Error Code 1 Error Code 2	
Write.cnf(+)	MSAC1M	Rem Add, Length	
Write.cnf(-)	MSAC1M	Rem Add, Error Decode, Error Code 1 Error Code 2	
Alarm Ack.cnf(+)	MSAC1M	Rem Add, Slot Number, Alarm Type, Seq Nr	
Alarm Ack.cnf(-)	MSAC1M	Rem_Add, Slot Number, Alarm Type, Seq Nr	

9.9.1.2 Primitives exchanged between MSAC1M and MSAL1M

Table 101 shows the service primitives including their associated parameters issued by MSAL1M and received by the MSAC1M.

Table 101 – Primitives issued by MSAL1M to MSAC1M

Primitive name	Source	Associated parameters	Functions
SInit MSAC1.req	MSAL1M	Rem Add	Refer to FAL Service Definition in IEC 61158-5-3
Reset.req	MSAL1M	Rem Add	
Start.req	MSAL1M	Rem Add	Start MS1-processing
Stop.req	MSAL1M	Rem Add	Stop MS1-processing

Table 102 shows the service primitives including their associated parameters issued by MSAC1M and received by the MSAL1M.

Table 102 – Primitives issued by MSAC1M to MSAL1M

Primitive name	Source	Associated parameters	Functions
SInit MSAC1.cnf	MSAC1M	Rem Add	Refer to FAL Service Definition in IEC 61158-5-3
Reset.cnf	MSAC1M	Rem Add	
Start.cnf	MSAC1M	Rem Add	Start completed
Stop.cnf	MSAC1M	Rem Add	Stop completed
Fault.ind	MSAC1M	(none)	
Abort.ind	MSAC1M	Rem Add	MSAC1 detected an error

9.9.1.3 Parameters of MSAL1M primitives

The parameters used with the primitives exchanged between the MSAL1M and MSCY1M are described in Table 103.

Table 103 – Parameter used with primitives exchanged between MSAL1M and MSCY1M

Parameter name	Description
Rem Add	This parameter conveys the DL address of the assigned DP-slave.

9.9.2 State machine description

The MSAC1M State Machine has the three states POWER-ON, CLOSED and OPEN. With the Init.req from the FSPMM1, the State Machine is initialized and changes to CLOSED. The state transition from CLOSED to OPEN is caused by the Start.req from the Slave-Handler (MSCY1M). The Slave-Handler is calling this service as soon as the DP-slave has entered the DATA-EXCH-state. When the DP-slave leaves the DATA-EXCH-state, the Stop.req service is called and the MSAC1M State Machine changes from OPEN to CLOSED. A Read/_Write request with an illegal parameter or a parallel service request while the last Read/Write is still being processed, is rejected to the User with a Reject.ind. An Alarm_Ack.req containing illegal service parameters or a parallel Alarm_Ack.req while the last Alarm_Ack.req is still being processed, is causing a state transition to POWER-ON and a Fault.ind to the FSPMM1.

The MSAC1M State Machine takes into account that the Alarm_Ack can be handled either on the SAP 51 as the Read/Write-service or on SAP 50.

The responses for Read and Write services are monitored by means of C1 Timer.

MSAC1M_Alarm_Ack is handled on the same SAP as MSAC1M_Read/Write.

If an MSAC1M_Alarm_Ack is requested by the MSAL1M State Machine or a Read/_Write is requested by the User while no service is being processed, the service request is passed directly to the DL. If an Alarm_Ack.req is requested and a Read/Write is being processed, the Alarm_Ack.req is stored and sent as soon as the confirmation of the Read/Write is received. When a Read/Write.req is requested and an Alarm_Ack is being processed, the Read/Write is stored and sent as soon as the confirmation of the Alarm_Ack is received.

When recognizing an error while a service request is being processed, the MSAC1M State Machine

– aborts the MS0 connection by indicating a Abort.ind to the MSCY1S,
– deletes an eventually stored service,
– informs the User with a Reject.ind in case of an outstanding Read/_Write service, and
– changes from state OPEN to CLOSE.
– When a Stop.req is requested by the Slave-Handler, the MSAC1M first waits for an eventually outstanding DATA-REPLY.cnf. After this the Stop.req will be processed. A Read/_Write-confirmation is passed to the User and a possibly stored Alarm_Ack.req is deleted. In case of an Alarm_Ack confirmation, this confirmation is ignored, and a possibly outstanding Read/_Write is deleted and the User is informed with a Reject.ind. After that, the Stop.cnf is passed to the Slave-Handler and the MSAC1M State Machine changes from state OPEN to CLOSE.

MSAC1M_Alarm_Ack is handled on SAP 50

When an Alarm_Ack is requested by the MSAL1M State Machine or a Read/_Write service is requested by the User, the service request is passed directly to the DL.

When recognizing an error while a service request is being processed on the other SAP, the MSAC1M State Machine

– waits for the confirmation of the service on the other SAP,
– then aborts the MSCY1M connection by indicating a Abort.ind to the Slave-Handler MSCY1S,
– informs the User with a Reject.ind in case of an outstanding _Read/_Write service, and
– changes from state OPEN to CLOSE.

When a Stop.req is requested by the Slave-Handler, the MSAC1M first waits for eventually outstanding DL-DATA-REPLY.cnf. After this the Stop.req will be processed. An outstanding MSAC1M_Read/_Write confirmation is passed to the User and an outstanding Alarm_Ack confirmation is ignored. Then the Stop.cnf is passed to the Slave-Handler and the MSAC1M State Machine changes from state OPEN to CLOSE.

Local Variables

Go_Abort
(Boolean)

This variable is set TRUE if the Stop.req is in use.

Internal_Abort
(Boolean)

This variable is set TRUE if an error in the Slave response occurs.

Max_Data_Length
(Unsigned8)

This variable defines the maximum size of MS1-PDU for all DP-slaves.

Alarm_DSAP
(Unsigned8)

This variable contains the DSAP the Alarm_Ack is sent to.

Service_Header
(Unsigned8)

This variable contains the Function_Num of the MS1-Service actually sent to the DSAP 51.

Service_Buffer
(Octet-String)

This variable contains a Read or Write Request-PDU if an Alarm_Ack is in use (DSAP 51).

Alarm_Buffer
(Octet-String)

This variable contains an Alarm_Ack if a Read or Write Request-PDU is in use (DSAP 51).

Src
(Unsigned8)

This variable contains the number of the actual processed Read or Write Services (only 0 or 1 is possible).

Arc
(Unsigned8)

This variable contains the number of the actual Alarm_Ack Services (only 0 or 1 is possible)

Stop_Pending
(Boolean)

This variable is set TRUE if a Stop.cnf is pending.

Start_C1_Monitoring
(Boolean)

This variable is set TRUE if the user issues a Read or Write request service primitive. It is used to distinguish the first polling from later ones, and to start the C1 Timer for monitoring the user response.

C1_Timer_Expired
(Boolean)

This variable is set TRUE if the C1 Timer expires. With the next confirmation it is used to abort the connection if no response is available. It is only relevant if at least one polling for the response has been issued.

C1_Timer
Timer with 10 ms base

This timer is used to monitor the response of the DP-slave according the values specified in the GSD of the DP-slave (stored in the CRL). The function Start C1 Timer starts or restarts this timer with the value C1_Response_Timeout taken from the appropriate CRL entry. The function Stop C1 Timer stops the operation no matter of it was running before.

Macros and Replacements

<REQPARAM>

<
XXX=Read: Slot_Number, Index, Length
XXX=Write: Slot_Number, Index, Length, Data
>

<VALID_SERVICE_FUNCTION_NUM>

<
XXX=Read: 0x5E
XXX=Write: 0x5F
>

<LOAD_SERVICE_BUFFER>

<
XXX=Read: Service_Buffer[1]=0x5E
Service_Buffer[2]=Slot_Number
Service_Buffer[3]=Index
Service_Buffer[4]=Length
XXX=Write: Service_Buffer[1]=0x5F
Service_Buffer[2]=Slot_Number
Service_Buffer[3]=Index
Service_Buffer[4]=Length
Service_Buffer[5..Length+4]=Data
>

<LOAD_ALARM_BUFFER>

<
Alarm_Buffer[1]=0x5C
Alarm_Buffer[2]=Slot_Number
Alarm_Buffer[3]=Alarm_Type
Alarm_Buffer[4]=Seq_Nr*8
>

IS_VALID_SERVICE_REQ

(
Length ≤ Max_Data_Length
)

IS_VALID_SERVICE_RES_PDU

(
L_sdu[1] = Service_Buffer[1]
&& L_sdu.len ≥ 4

```
    && (L_sdu[1] <> 0x5E || L_sdu.len ≥ L_sdu[4]+4)
    && Max_Data_Length ≥ L_sdu[4]
    )

IS_VALID_SERVICE_NRS_PDU
    (
    L_sdu[1] = (Service_Buffer[1]+0x80)
    && L_sdu.len = 4
    )

IS_ALARM_ACK_PDU
    (
    L_sdu[2..4] = Alarm_Buffer[2..4]
    && (L_sdu[1] AND 0x7F = (Service_Buffer[1])
    && L_sdu.len = 4
    )

<VALID_SERVICE>
    <
     Service_Header=0x5E: MSAC1M_Read
     Service_Header=0x5F: MSAC1M_Write
    >

<VALID_SERVICE_RES_PARAM>
    <
    Service_Header=0x5E: Length=L_sdu[4],
       Data=L_sdu[5..Length+4]
     Service_Header=0x5F: Length=L_sdu[4]
    >

<VALID_SERVICE_NRS_PARAM>
    <
    Error_Decode=L_sdu[2],
    Error_Code_1=L_sdu[3],
    Error_Code_2=L_sdu[4],
    >
REJECT_VALID_SERVICE
    (
    if (Stop_Pending=True)
      MSAC1M_Reject.ind(Rem_Add, Status=REJ_SE)
    else
    if.(Go_Abort=True)
      MSAC1M_Reject.ind(Rem_Add, Status=REJ_ABORT)
    else
    if (Src <> 0)
```

```
  MSAC1M_Reject.ind(Rem_Add, Status=REJ_PS)
else
if (Length<=Max_Data_Length)
  MSAC1M_Reject.ind(Rem_Add, Status=REJ_LE)
else
  MSAC1M_Reject.ind(Rem_Add, Status=REJ_IV)
)
```

CONTINUE_VALID_SERVICE

```
(
if (Src <> 0 && Alarm_Dsap=51)
  Service_Header = Service_Buffer[1]
  Start_C1_Monitoring = TRUE
  DMPMM1_DATA_REPLY.req(SSAP=51, DSAP=51,
   Rem_Add,L_sdu=Service_Buffer, Serv_class=Low)
)
```

CONTINUE_ALARM_ACK

```
(
if (Arc <> 0 && Alarm_Dsap=51)
  Service_Header = Alarm_Buffer[1]
  DMPMM1_DATA_REPLY.req(SSAP=51, DSAP=51, Rem_Add,
   L_sdu=Alarm_Buffer, Serv_class=Low)
)
```

IS_INVALID_SERVICE_RES

```
(
 (L_status = DL &&
 (L_sdu[1]<> Service_Header || L_sdu.len<4) &&
 (L_sdu[1]<> (Service_Header+0x80) || L_sdu.len<> 4)
 ) ||
 L_status = DS/UE/RS/RDL/NA/RR/LR/DH/RDH
)
```

IS_INVALID_ALARM_ACK_RES

```
(
 (L_status = DL &&
 (L_sdu[1..4]<> Alarm_Buffer[1..4] || L_sdu.len<> 4)
 ) ||
 L_status = DS/UE/RS/RDL/NA/RR/LR/DH/RDH
)
```

CHECK_STOP_PENDING

```
(
if (Stop_Pending = TRUE)
```

MSAC1M_Stop.cnf(Rem_Add)
else
MSAC1M_Abort.ind(Rem_Add)
)

9.9.3 MSAC1M state table

Table 104 contains the complete description of the MSAC1M state machine.

Table 104 – MSAC1M state table

#	Current state	Event / condition => action	Next state
1	POWER-ON	MSAC1M_MInit_MSAC1.req(Rem_Add) => MSAC1M_MInit_MSAC1.cnf(Rem_Add)	CLOSED
2	CLOSED	MSAC1M_Start.req(Rem_Add) => Alarm_Buffer:= empty Service_Buffer:= empty Stop_Pending:= False Go_Abort:= False Start_C1_Monitoring:= FALSE C1_Timer_Expired:= FALSE Src:= 0 Arc:= 0 Service_Header:= 0 Max_Data_Length:= Max_Channel_Data_Length if (Extra_Alarm_Sap=FALSE) (Alarm_Dsap:=51) ELSE (Alarm_Dsap:=50) MSAC1M_Start.cnf(Rem_Add)	OPEN
3	CLOSED	MSAC1M_Stop.req(Rem_Add) => MSAC1M_Stop.cnf(Rem_Add)	CLOSED
4	CLOSED	MSAC1M_XXX.req(Rem_Add, <REQPARAM>) => MSAC1M_Reject.ind(Rem_Add, Status:=REJ_SE)	CLOSED
5	CLOSED	MSAC1M_Alarm_Ack.req(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr) => ignore	CLOSED
6	CLOSED	DMPMM1_DATA_REPLY.cnf(SSAP, DSAP, Rem_Add, L_sdu, Serv_class, L_status) => MSAC1M_Fault.ind(Rem_Add)	POWER-ON
7	OPEN	MSAC1M_Start.req(Rem_Add) => MSAC1M_Fault.ind(Rem_Add)	POWER-ON
8	OPEN	MSAC1M_Stop.req(Rem_Add) /Arc = 0 && Src = 0 => MSAC1M_Stop.cnf(Rem_Add)	CLOSED
9	OPEN	MSAC1M_Stop.req(Rem_Add) /Arc <> 0 || Src <> 0 => Stop_Pending:=True	OPEN

#	Current state	Event / condition => action	Next state
10	OPEN	MSAC1M_XXX.req(Rem_Add, <REQPARAM>) /Src = 0 && Stop_Pending = False && Go_Abort = False && IS_VALID_SERVICE_REQ && (Arc = 0 || Alarm_Dsap <> 51) => Service_Header:= <VALID_SERVICE_FUNCTION_NUM> Src:= Src+1 Start_C1_Monitoring:=TRUE <LOAD_SERVICE_BUFFER> DMPMM1_DATA_REPLY.req(SSAP:=51, DSAP:=51, Rem_Add, L_sdu:=Service_Buffer, Serv_class:=Low)	OPEN
11	OPEN	MSAC1M_XXX.req(Rem_Add, <REQPARAM>) /Src = 0 && Stop_Pending = False && Go_Abort = False && IS_VALID_SERVICE_REQ && Arc <> 0 && Alarm_Dsap = 51 => Src:= Src+1 <LOAD_SERVICE_BUFFER>	OPEN
12	OPEN	MSAC1M_XXX.req(Rem_Add, <REQPARAM>) /Src <> 0 || Stop_Pending = True || Go_Abort = True || NOT (IS_VALID_SERVICE_REQ) => REJECT_VALID_SERVICE	OPEN
13	OPEN	MSAC1M_Alarm_Ack.req(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr) /Arc = 0 && Stop_Pending = False && Go_Abort = False && Alarm_Dsap <> 51 => Arc:= Arc+1 <LOAD_ALARM_BUFFER> DMPMM1_DATA_REPLY.req(SSAP:=51, DSAP:=Alarm_Dsap, Rem_Add, L_sdu:=Alarm_Buffer, Serv_class:=Low))	OPEN
14	OPEN	MSAC1M_Alarm_Ack.req(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr) /Arc = 0 && Stop_Pending = False && Go_Abort = False && Src = 0 && Alarm_Dsap = 51 => Arc:= Arc+1 DMPMM1_DATA_REPLY.req(SSAP:=51, DSAP:=Alarm_Dsap, Rem_Add, L_sdu:=Alarm_Buffer, Serv_class:=Low))	OPEN
15	OPEN	MSAC1M_Alarm_Ack.req(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr) /Arc = 0 && Stop_Pending = False && Go_Abort = False && Src <> 0 && Alarm_Dsap = 51 => Arc:= Arc+1 <LOAD_ALARM_BUFFER>	OPEN
16	OPEN	MSAC1M_Alarm_Ack.req(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr) /Arc=0 && (Stop_Pending = True || Go_Abort = True) => ignore	OPEN
17	OPEN	MSAC1M_Alarm_Ack.req(Rem_Add, Slot_Number, Alarm_Type, Seq_Nr) /Arc <> 0 => DMPMM1_Fault.ind(Rem_Add, Status:=MSAC1_Fault)	POWER-ON
18	OPEN	DMPMM1_DATA_REPLY.cnf(SSAP=51, DSAP=51, Rem_Add, L_sdu, Serv_class=LOW, L_status) /L_status = DL && IS_VALID_SERVICE_RES_PDU && Src <> 0 && L_sdu[1] = Service_Header && Stop_Pending = False && Go_Abort = False => Stop C1 Timer() C1_Timer_Expired:= FALSE Src:= 0 CONTINUE_ALARM_ACK MSAC1_<VALID_SERVICE>.cnf(+)(Rem_Add, <VALID_SERVICE_RES_PARAM>)	OPEN

#	Current state	Event / condition => action	Next state
19	OPEN	DMPMM1_DATA_REPLY.cnf(SSAP=51, DSAP=51, Rem_Add, L_sdu, Serv_class=LOW, L_status) /L_status = DL && IS_VALID_SERVICE_RES_PDU && Src <> 0 && L_sdu[1] = Service_Header && Stop_Pending = True && (Arc=0 || Alarm_Dsap=51) => Stop C1 Timer() C1_Timer_Expired:= FALSE Src:= 0 Arc:= 0 MSAC1_<VALID_SERVICE>.cnf(+)(Rem_Add, <VALID_SERVICE_RES_PARAM>) MSAC1M_Stop.cnf(Rem_Add)	CLOSED
20	OPEN	DMPMM1_DATA_REPLY.cnf(SSAP=51, DSAP=51, Rem_Add, L_sdu, Serv_class=LOW, L_status) /L_status = DL && IS_VALID_SERVICE_RES_PDU && Src <> 0 && L_sdu[1] = Service_Header && Stop_Pending = True && Arc <> 0 && Alarm_Dsap <> 51 => Stop C1 Timer() C1_Timer_Expired:= FALSE Src:= 0 MSAC1_<VALID_SERVICE>.cnf(+)(Rem_Add, <VALID_SERVICE_RES_PARAM>)	OPEN
21	OPEN	DMPMM1_DATA_REPLY.cnf(SSAP=51, DSAP=51, Rem_Add, L_sdu, Serv_class=LOW, L_status) /L_status = DL && IS_VALID_SERVICE_RES_PDU && Src <> 0 && L_sdu[1] = Service_Header && Stop_Pending = False && Go_Abort = True => Stop C1 Timer() C1_Timer_Expired:= FALSE Src:= 0 MSAC1_<VALID_SERVICE>.cnf(+)(Rem_Add, <VALID_SERVICE_RES_PARAM>) MSAC1M_Abort.ind(Rem_Add)	CLOSED
22	OPEN	DMPMM1_DATA_REPLY.cnf(SSAP=51, DSAP=51, Rem_Add, L_sdu, Serv_class=LOW, L_status) /L_status = DL && IS_VALID_SERVICE_NRS_PDU && Src <> 0 && L_sdu[1] = (Service_Header+0x80) && Stop_Pending = False && Go_Abort = False => Stop C1 Timer() C1_Timer_Expired:= FALSE Src:= 0 CONTINUE_ALARM_ACK MSAC1_<VALID_SERVICE>.cnf(-)(Rem_Add, <VALID_SERVICE_NRS_PARAM>)	OPEN
23	OPEN	DMPMM1_DATA_REPLY.cnf(SSAP=51, DSAP=51, Rem_Add, L_sdu, Serv_class=LOW, L_status) /L_status = DL && IS_VALID_SERVICE_NRS_PDU && Src <> 0 && L_sdu[1] = (Service_Header+0x80) && Stop_Pending = True && (Arc=0 || Alarm_Dsap=51) => Stop C1 Timer() C1_Timer_Expired:= FALSE Src:= 0 Arc:= 0 MSAC1_<VALID_SERVICE>.cnf(-)(Rem_Add, <VALID_SERVICE_NRS_PARAM>) MSAC1M_Stop.cnf(Rem_Add)	CLOSED
24	OPEN	DMPMM1_DATA_REPLY.cnf(SSAP=51, DSAP=51, Rem_Add, L_sdu, Serv_class=LOW, L_status) /L_status = DL && IS_VALID_SERVICE_NRS_PDU && Src <> 0 && L_sdu[1] = (Service_Header+0x80) && Stop_Pending = True && Arc <> 0 && Alarm_Dsap <> 51 => Stop C1 Timer() C1_Timer_Expired:= FALSE Src:= 0 MSAC1_<VALID_SERVICE>.cnf(-)(Rem_Add, <VALID_SERVICE_NRS_PARAM>)	OPEN