Base Standard
AISG v3.0
v3.0.0.10

Revision History

DATE	ISSUE	NOTES
November 5th 2018	v3.0.0.10	First public release.

1. FOREWORD

This standard has been produced by the Antenna Interface Standards Group (AISG) to introduce and define new features and enhancement of the management system for antenna line devices (ALDs) with remote control and monitoring facilities.

New functions introduced in this version of the standard include the discovery of device interconnections, site mapping capabilities and the functionality necessary to control an ALD from more than one primary. These functions adhere to the AISG interoperability requirements.

This standard is independent of previous 3GPP specifications and provides a complete description of all layers of the protocol.

2. SCOPE

AISG v3.0 specifies the standard data interface between a primary, typically a base station, and antenna line devices (ALDs) which are manageable units, usually associated with base station antenna systems.

The standard is divided into this base standard and several subunit type standards. This base standard describes the common behaviour of antenna line devices with AISG interfaces. Type-specific functionality is defined in separate subunit type standards.

This standard defines the functional behaviour of the ALD. It also specifies some recommended and some mandatory primary behaviour.

The text of the standard defines explicitly what is required or permitted. Anything that is not explicitly allowed is not permitted.

3. BACKWARD COMPATIBILITY WITH AISG v2

This standard provides tools that enable ALD vendors to build ALDs that share a bus with equipment supporting AISG v2. AISG v3.0 ALDs may be made to switch to AISG v2 mode where they can be controlled by AISG v2 primaries. AISG v3.0 ALDs operating in v3.0 mode can be used on the same bus as AISG v2 ALDs provided that the primary supports this.

Pure v2 operation is achieved by building support for AISG v2 protocol into AISG v3.0 ALDs and primaries. The v3.0 standard provides tools and methods that enable the equipment to change between AISG v2 and AISG v3.0 mode in controlled fashion.

Mixed bus operation can be achieved by separately polling v2 and v3.0 devices on a bus.

The following AISG v3.0 functionality is not available in AISG v2 mode:

- Site Mapping
- Ping functionality
- MALD Configuration

MALD operation is not defined in AISG v2. MALDs supporting AISG v3.0 can be controlled by AISG v2 primaries but will have limited functionality.

4. REFERENCES

This AISG Standard incorporates provisions from other publications. These are cited in the text and the referenced publications are listed below. Where references are listed with a specific version or release, subsequent amendments or revisions of these publications apply only when specifically incorporated by amendment or revision of this AISG standard. For references listed without a version or release, the latest edition of the publication referred to applies.

1 ISO/IEC 8482 (1993): “Information technology –Telecommunications and information exchange between systems – Twisted pair multipoint interconnections”

2 TIA/EIA TSB-89-A 2003: “Application guidelines for TIA/EIA-485-A”

3 ETSI 3GPP TS137.113: “Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; E-UTRA, UTRA and GSM/EDGE; Multi standard radio base station electromagnetic compatibility”

4 MIL-STD 461F 2007: “Requirement for the control of electromagnetic interference characteristics of subsystems and equipment”

5 IEC CISPR 16-2-1 2014: “Specification for radio disturbance and immunity measuring apparatus and methods – Part 2-1: Methods of measurement of disturbances and immunity – Conducted disturbance measurements”

6 ISO/IEC 13239 (2nd Edition, March 2000): "Information Technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures"

7 Vendor Codes list on http://www.aisg.org.uk

8 ITU-T X.733: “Data communication networks, Information Technology – Open Systems Interconnection – Systems management: Alarm reporting function”

9 RFC1549: “PPP in HDLC Framing” available from http://www.rfc-base.org

10 ITU(T) O.153-1992: “Basic parameters for the measurement of error performance at bit rates below the primary rate”

11 ISO/IEC 646:1991: “Information technology – ISO 7-bit coded character set for information interchange”

12 ETSI 3GPP TS23.003: “Digital cellular telecommunication systems (Phase 2+); Universal Mobile Telecommunication Systems (UMTS); Numbering, addressing and identification”

13 AISG: “Antenna Port Colour Coding”

5. ABBREVIATIONS

Where abbreviations or acronyms are used in this document they have the following meanings:

ACK	Acknowledgment
ADB	Antenna Database
ALD	Antenna Line Device
ANT	Antenna
BER	Bit Error Rate
CRC	Cyclic Redundancy Check
CPM	Configurable Power Monitor
CW	Continuous Wave
DC	Direct Current
DISC	Disconnect (frame type)
DM	Disconnected Mode (frame type)
FCS	Frame Check Sequence
FI	Format Identifier
FRMR	Frame Reject (frame type)
GI	Group Identifier
GL	Group Length
HDLC	High-Level Data Link Control
HW	Hardware
I	Information (frame type)
ID	Identifier
IM	Intermodulation
IM3	Third Order Intermodulation
IM5	Fifth Order Intermodulation
INFO	Information (field name)
ISB	Idle State Biasing
MALD	Multi-primary ALD
MSB	Most Significant Byte
NAK	Negative Acknowledgment
NRM	Normal Response Mode
OOK	On-Off Keying
P/F	Poll/Final

PI	Parameter Identifier
PL	Parameter Length
PV	Parameter Value
RET	Remote Electrical Tilting
RF	Radio Frequency
RNR	Receive Not Ready (frame type)
RR	Receive Ready (frame type)
RX	Receive
SALD	Single-primary ALD
SNRM	Set Normal Response Mode (frame type)
SW	Software
TCC	Time-Consuming Command
TMA	Tower Mounted Amplifier
TWA	Two Way Alternate
TX	Transmit
UA	Unnumbered Acknowledgement (frame type)
UCC	Upper Camel Case
UNC	Unbalanced Operation Normal Response Mode Class
XID	Exchange ID (frame type)
3GPP	3rd Generation Partnership Project

6. TERMINOLOGY

Where the following terms are used in this document, they have the following meanings:

AISG bus	A layer 1 bus between an AISG port on a primary and AISG port(s) on one or more ALDs. Each ALD may have one or more AISG ports connected to the same AISG bus.
AISG port	A port, either RS-485 or OOK, on a MALD, SALD or primary. An AISG port on an ALD can only support one layer 2 link. An AISG port on a primary may support multiple layer 2 links.
Alarm	An alarm is a persistent indication of a fault.
ALD enclosure	An ALD enclosure contains only one ALD with at least one connectable AISG interface. Camouflage boxes are not ALD enclosures.
ALD type	One octet identifying the type of an ALD as either SALD or MALD.
Antenna line	A group of logical devices associated with one or more antenna systems, which may include antenna actuators, amplifiers and other equipment.
Antenna line device	A generic term for an addressable physical device. An ALD can only be a SALD or MALD in this standard.
ANT RS-485 modem	External modem at the antenna end of the antenna line (for instance a smart bias-T).
Array	An array is a logical group of single or dual polarized radiators inside the antenna radome supporting a common frequency band and a common beam shape and tilt. A single physical array comprises more than one logical array when fed through multiple band-selective filters.
Array ID	A UTF-8 string identifying an antenna array as defined in [13].
ASCII character	A character forming part of the International Reference Version of the 7-bit character set defined in [11] represented as one octet.
BS RS-485 modem	External modem at the base station end of an AISG RS-485 bus (for instance a smart bias-T).
Control port	An AISG port on a MALD or SALD with a layer 2 link to a primary.
Error	A deviation of a system from normal operation.

Event	Something that happens which may be of interest. For instance a fault, a change in status, crossing a threshold or an external input to the system.
Fault	Lasting error or warning condition.
Frame	A layer 2 HDLC frame as defined in [6].
Intra frame gap	The time interval between two consecutive octets in an HDLC frame.
Layer 1 bus	A sequence of layer 1 segments carrying the same signal.
Layer 1 segment	A direct physical connection between two ports, using either the OOK or RS-485 option.
Layer 2 link	An HDLC connection between a primary and an ALD after a successful link establishment.
Listener	An ALD or primary that listens for the layer 2 Ping message.
MALD configuration transaction	An atomic sequence of MALD configuration commands, that is, either the effect of all the commands are accepted or they are all rejected.
Message	A layer 2 command or response, or a layer 7 command or response.
Modem	A circuit providing a layer 1 conversion between OOK and RS-485 or the internal interface of an ALD.
Multi-primary ALD	An ALD type capable of simultaneously supporting multiple layer 2 links on different ports.
Octet	8 bits as used in [6].
On-off keying	A modulation system in which a carrier is switched between two states, ON and OFF.
PING port	A port capable of performing OOK Ping message reception or transmission.
Ping process	The succession of commands that enables the verification of RF cabling and discovery of RF paths.
Pingee	An ALD or primary that received the layer 2 Ping message.
Pinger	An ALD that sends the layer 2 Ping message on the requested port.
Port Number	A unique 2-octet integer that identifies an RF port, AISG port or PING port within an ALD.

Primary	The entity which controls the connected ALDs using all layers.
PrimaryID	An unsigned 4-octet value identifying the AISG primary. It is defined as the leftmost 8 hexadecimal digits of the SHA1 checksum of the primary node name.
Primary node name	A UTF-8 string uniquely identifying the primary in the network. For LTE this shall be the Global eNodeB-ID (for instance enbA9F7D.enb.epc.mncEHC.mccFIN.3gppnetwork.org), see [12].
RF Path	An unordered list of ALD UniqueIDs and an antenna array ID constituting an RF signal path between a base station RF port and an antenna array.
RF Path ID	A unique 2-octet integer identifying a specific RF path.
RF Path ID Alias	A user friendly UTF-8 string identifying a specific RF Path ID.
Reset	A process by which an ALD is put in the same status that it reaches after a completed power-up. Reset can be caused by DC power-up, DC power cycle, communication timeout, an internally implemented ALD watchdog timeout or the layer 2 ResetALD command.
Single-primary ALD	An ALD type supporting only one layer 2 link.
Smart bias-T	A device combining/splitting DC power and RF signals and incorporating an OOK modem in the RF path.
Subunit	An ALD may comprise one or more functions such as RETs and TMAs. These are referred to as subunits.
Subunit type	The classification of a subunit in an ALD that describes its function, for instance RET or TMA.
Transaction	A MALD configuration transaction.
UniqueID	A concatenation of the vendor code (2 octets) and an exactly 17-octet long unit specific code (for instance serial number) exclusive to each ALD.
Vendor code	A unique ASCII 2-character code assigned to each vendor in [7].

7. DEFINITIONS

7.1. Interpretation

The word *shall* indicates mandatory requirements strictly to be followed in order to conform to this standard and from which no deviation is permitted.

The phrase *shall, if supported*, indicates a mandatory requirement strictly to be followed in order to conform to this standard and from which no deviation is permitted, if an ALD supports a functionality declared as optional in this standard.

The word *should* indicates that among several possibilities, one is recommended as particularly suitable without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (*should* equals *is recommended*).

The word *may* is used to indicate a course of action permissible within the limits of the standard.

The word *can* is used for statements of capability.

Numbers prefixed with 0x are hexa-decimal. All other numbers are decimal.

7.2. Definition of AISG coding style

This section defines the coding style for primary and ALD commands and responses which is used in this standard. The AISG coding style is inspired by the C programming language, but AISG does not require that any software is programmed in the C language. When the standard states that a variable has a specific type, the mandatory requirement is only related to the described logic and data content.

7.2.1. Keywords

The keyword “CONSTANT” is used to define that the data cannot be changed.

The keyword “persistent” is used to define that the data is stored in non-volatile memory.

7.2.2. Basic data types

The following simple integer data types are used:

```
// unsigned 8-bit integer
TYPEDEF uint8_t      INTEGER RANGE 0..255

// signed 8-bit integer
TYPEDEF int8_t       INTEGER RANGE -128..127

// unsigned 16-bit integer
TYPEDEF uint16_t     INTEGER RANGE 0..65535

// signed 16-bit integer
TYPEDEF int16_t      INTEGER RANGE -32768..32767

// unsigned 32-bit integer
TYPEDEF uint32_t     INTEGER RANGE 0..4294967295

// signed 32-bit integer
TYPEDEF int32_t      INTEGER RANGE -2147483648..2147483647
```

The following floating point data types are used:

```
float      // IEEE 754 32-bit floating point
double     // IEEE 754 64-bit floating point
```

The following layer 7 message data types are used:

```
TYPEDEF CommandCode_t      uint16_t
TYPEDEF CommandSequence_t uint16_t
TYPEDEF DataLength_t       uint16_t
TYPEDEF Subunit_t          uint16_t
```

7.2.3. String data types

Strings are not NUL terminated. The following string data types are used:

```
TYPEDEF Char_t             uint8_t
TYPEDEF TextChar_t         INTEGER RANGE 0x20..0x7E
TYPEDEF UIDChar_t          INTEGER RANGE 0x00, 0x21..0x7E
```

Strings are not NUL terminated. The following string data types are used:

```
// sequence of UTF-8 characters
TYPEDEF UTF8String_t       Char_t[]

// array of ASCII characters
TYPEDEF AsciiString_t      Char_t[]

// AsciiString with characters 0x00 or 0x21..0x7E inclusive
TYPEDEF UIDString_t        UIDChar_t[]

// AsciiString with characters between 0x20 and 0x7E inclusive
TYPEDEF TextString_t       TextChar_t[]
```

The length of a UTF8String is specified in octets, not characters.

7.2.4. Combined data types

A structure is a data type that consists of a number of parameters which may have different data types. A structure is identified by the keyword “struct” followed by its name:

```
struct Name_t {
     uint8_     parameter1
     uint8_t    parameter2
     uint16_t   parameter3
}
```

An enumeration is a data type that consists of a complete ordered listing of all the named integer constants, each with an explicitly assigned value. An enumeration is identified by the keyword “Enumeration” followed by its name, a colon and the data type of the integer constants.

```
Enumeration Count_t : uint8_t {
     One   ← 0
     Two   ← 1
     Three← 2
}
```

A bit field is a data type that consists of a complete ordered listing of all the named bits in an integer. A bit field is identified by the keyword “Bitfield” followed by its name, a colon and the data type of the integer containing the bit field. If all bits except bit number 0 are set to 0 and bit number 0 is set to 1 the integer value of the entire bit field is 1. Unused bitfield flags are reserved for future use, shall always be returned as 0 by the ALD and modification attempts shall be ignored.

```
Bitfield Bitset_t : uint8_t {
      Claudia    : Bit 0
      Kari       : Bit 1
      Nicolas    : Bit 2
      Harri      : Bit 3
      Brian      : Bit 4
      Maurice    : Bit 5
      Torbjorn   : Bit 6
      Gerry      : Bit 7
}
```

7.2.5. ALD constants

The data types and constants provide information about the ALD the code is running on.

The ALDType constant is set by design to the type of the ALD that is running the pseudocode (see Section 8.1.4. “SALD and MALD”).

```
CONSTANT ALDTypes_t ALDType[1+MaxPort]
```

The MaxPort constant is set by design to the highest port number in the ALD.

```
CONSTANT uint16_t MaxPort
```

7.2.6. Subunit information

The NrOfSubunits constant is set by design to the number of subunits in the ALD (see Section 8.1.5. “Subunits”).

```
uint16_t NrOfSubunits          // number of subunits within the ALD
```

The SubunitType_t enumeration is used to identify the type of a subunit.

```
Enumeration SubunitType_t : uint8_t {
      RET  ← 0x01
      TMA  ← 0x02
      ADB  ← 0x03
}
```

The SubunitInfo_t structure describes a subunit. Each subunit type is defined in the associated standard.

```
struct SubunitInfo_t {
      SubunitType_t Type
}
```

The Subunits array is initialised by design and describes all the subunits.

```
struct SubunitInfo_t Subunits[NrOfSubunits]
```

The SubunitTypeListElement_t structure describes a subunit and its type.

```
struct SubunitTypeListElement_t {
      Subunit_t Subunit
      SubunitType_t Type
}
```

7.2.7. Port interconnection information

The PortInterconnection_t structure describes an interconnection from a port by specifying the port it is connected to and the interconnection type.

```
struct PortInterconnection_t {
     uint16_t   PortNumber
     InterconnectionType_t Type
}
```

7.2.8. Version information

The AISGVersion_t structure describes the release; major and minor version of AISG base standard and AISG subunit type standards. For the base standard major is the number b and minor is the number c as defined Section 14.1. “Base standard versions”. For the subunit type standards major is the number b and minor is the number c as defined in Section 14.2. “Subunit type standard versions”.

```
struct AISGVersion_t {
     uint8_t    ReleaseVersion
     uint8_t    MajorVersion
     uint8_t    MinorVersion
}
```

The ConfiguredSubunitTypeVersion contains the currently configured subunit type standard version.

```
ConfiguredSubunitTypeVersion[MaxPort + 1]
```

7.2.9. Layer 7 command information

The ReturnCode_t enumeration is used in layer 7 message responses to indicate success or the cause of a failure. All ReturnCode_t values used by this AISG v3.0 standard are listed here.

```
Enumeration ReturnCode_t : uint16_t {
     OK                              ← 0x0000
     Busy                            ← 0x0005
     GeneralError                    ← 0x0011
     PortInUse                       ← 0x0012
     OutOfRange                      ← 0x0013
     TransactionInProgress           ← 0x0014
     TransactionNotInProgress        ← 0x0015
     IncorrectCommitCounter          ← 0x0017
     UploadRejected                  ← 0x0018
     UnknownCommand                  ← 0x0019
     UnsupportedFileType             ← 0x0020
     InvalidFileContent              ← 0x0022
     InUseByAnotherPrimary           ← 0x0023
     FormatError                     ← 0x0024
     NotAuthorised                   ← 0x002C
     InvalidSubunitNumber            ← 0x002D
     InvalidPortNumber               ← 0x002E
     InvalidAuthorityValue           ← 0x002F
     UnsupportedConfiguration        ← 0x003B
     InvalidSettingSource            ← 0x003C
     IncorrectPortType               ← 0x003D
     InvalidSubunitType              ← 0x003E
     InvalidRFPathID                 ← 0x003F
```

```
      IncorrectState                  ← 0x0040
      InvalidMonitorPhase             ← 0x0041
      IncorrectDirection              ← 0x0042
      TooManyArguments                ← 0x0043
      NotConfigured                   ← 0x0044
      NotCalibrated                   ← 0x0045
      CalibrationNotSupported         ← 0x0046
      InvalidArrayNumber              ← 0x0047
      UnsupportedSecuritySetting      ← 0x0048
      InvalidConfTargetPortNumber     ← 0x0049
      InvalidConfTargetSubunitNumber  ← 0x004A
      NotControlCapablePort           ← 0x004B
      NoAlarmSubscription             ← 0x004C
      DownloadFailed                  ← 0x004D
      UnsupportedValue                ← 0x004E
      CalibrationFailed               ← 0x004F
      ConfigurationNotSupported       ← 0x0050
}
```

The AlarmCode_t enumeration is used in layer 7 alarm indication message to specify which alarm is being raised or cleared.

```
Enumeration AlarmCode_t : uint16_t {
      AlarmMovementTimeout            ← 0x0000
      AlarmInternalError              ← 0x0001
      AlarmNotConfigured              ← 0x0002
      AlarmNotCalibrated              ← 0x0003
      AlarmActuatorJammed             ← 0x0004
      AlarmPingTimeoutExpired         ← 0x0005
      AlarmGeneralError               ← 0x0006
}
```

The AlarmSubscribeFlag indicates which primaries have subscribed to the alarms. These flags are set for each port.

```
Boolean AlarmSubscribeFlag[MaxPort+1]
```

The PingMonitorRFPort variable is set to the number of the port the ALD was told to monitor during the ping process (see Section 8.4.2. “Ping process cycle”).

```
uint16_t PingMonitorRFPort
```

7.2.10. Layer 2 information

The LinkState_t enumeration defines the layer 2 link states of each AISG port (see Section 8.2.1. “State models for layer 2”).

```
Enumeration LinkState_t {
      NoAddress            ← 0
      AddressAssigned      ← 1
      Connected            ← 2
}
```

The LinkState variable sets the layer 2 link state of each AISG port (see Section 8.2.1. “State models for layer 2”).

```
LinkState_t LinkState[MaxPort+1]
```

7.2.11. Layer 7 information

The ALDState_t enumeration defines the layer 7 state of the ALD (see Section 8.2.2. "State model for layer 7").

```
Enumeration ALDState_t : uint8_t {
      IdleState                                  ← 0
      OperatingState                             ← 1
      DownloadState                              ← 2
      MALDConfigState                            ← 3
      PingerRestrictedState                      ← 4
      PingerRestrictedTransmitState              ← 5
      PingerBroadcastWaitState                   ← 6
      ListenerRestrictedMonitorState             ← 7
      ListenerRestrictedPreparationState         ← 8
      ListenerBroadcastWaitState                 ← 9
}
```

The ALDState variable sets the layer 7 state of the ALD. (see Section 8.2.2. "State model for layer 7")

```
ALDState_t ALDState
```

The ConnectionState_t enumeration defines the layer 7 ConnectionState of each AISG port (see Section 8.2.2. "State model for layer 7").

```
Enumeration ConnectionState_t : uint8_t {
      NoConnectionState                          ← 0
      OperatingConnectionState                   ← 1
      MALDConfigConnectionState                  ← 2
      UploadConnectionState                      ← 3
      DownloadConnectionState                    ← 4
      DownloadFailedConnectionState              ← 5
      RestrictedConnectionState                  ← 6
      DownloadNotificationConnectionState        ← 7
      OffConnectionState                         ← 8
      PingerConnectionState                      ← 9
      ListenerConnectionState                    ← 10
}
```

The ConnectionState variable sets the layer 7 connection state of each AISG port (see Section 8.2.2. "State model for layer 7").

```
ConnectionState_t ConnectionState[MaxPort+1]
```

7.2.12. Upload information

UploadRemainingLength and UploadPosition are used during upload to keep track of what data to send next.

```
uint32_t   UploadRemainingLength[MaxPort + 1]
uint32_t   UploadPosition[MaxPort + 1]
```

7.2.13. Gain information

The type ddB_t is used to represent decibel values as deci-dB (tenths of dB).

```
TYPEDEF ddB_t   int16_t    //dB*10
```

The type GainRange_t is used represent a range of gain values with a linear step size. A single gain value is represented as min and max having same value and a zero step size.

Gain ranges with nonlinear step sizes are represented by multiple GainRange_t. Starting from the lowest unallocated gain value, the next GainRange_t shall cover as many gain steps as possible. This allocation process shall be repeated until all gain values are allocated.

Example: 2, 4, 6, 10 shall be represented as (min 2, max 6, step size 2) and (min 10, max 10, step size 0).

```
struct GainRange_t {
      ddB_t Min
      ddB_t Max
      ddB_t StepSize
}
```

7.3. Definition of layer 2 frame format

Frames in layer 2 are shown as data structures identified by the keyword “Frame” followed by its name. Frame names use Upper Camel Case (UCC) format. A frame issued by the primary shall be identified by the keyword “PrimaryFrame”. A frame issued by the ALD shall be identified by the keyword “ALDFrame”. The name of the frame is suffixed by “Command” or “Response” as appropriate.

```
PrimaryFrame <NameCommand> {
      uint8_t Address
      uint8_t Ctrl
      uint8_t Payload[]
      uint8_t FCS1
      uint8_t FCS2
}
```

```
ALDFrame <NameResponse> {
      uint8_t Address
      uint8_t Ctrl
      uint8_t Payload[]
      uint8_t FCS1
      uint8_t FCS2
}
```

If the frame is an I-frame, the Payload contains the layer 7 message. Otherwise, the Payload contains layer 2 frame data. The minimum Payload is 0 octets and the maximum is 264 octets.

7.4. Definition of layer 7 message format

There are two types of layer 7 messages: commands and responses. Layer 7 messages are defined as data structures.

Message names use UCC format. (see Section 7.4.2. “Responses”).

A single layer 7 message must fit into a single layer 2 I-frame.

7.4.1. Commands

A command requests that the receiver executes a defined procedure and returns a response.

Commands are defined as structures.

A command issued by the primary is identified by the keyword “PrimaryCommand” and command issued by an ALD is identified by the keyword “ALDCommand”.

The names of commands have the suffix “Command”.

The first parameter in a command is the command code, which specifies the procedure to execute.

The second parameter in a command is a sequence number which is used as described below. It is called PrimaryCommandSequence in a PrimaryCommand and ALDCommandSequence in an ALDCommand.

NOTE: The command sequence number is totally unrelated to the layer 2 I-frame sequence number.

The third parameter is the subunit number (see Section 8.1.5. “Subunits”). Subunit number 0 refers to the ALD and subunit number 1..65535 identifies which subunit shall execute the procedure or return the response.

The fourth parameter of a command is the data length, which states the number of octets in the message data. The length of the data is 0 to 256 octets. The details of the data are specified by the message format for each command.

The PrimaryCommandSequence is used to match ALD responses to PrimaryCommands and is used to handle cases where the responses are processed in a different order from that in which the commands were issued. Each primary only has one wraparound PrimaryCommandSequence counter, not one per ALD.

A primary shall increment its PrimaryCommandSequence counter every time it issues a PrimaryCommand and the ALD shall copy this field unchanged into the response. The ALD shall not check or verify the PrimaryCommandSequence number in any way.

The ALDCommandSequence is used to match primary responses to ALDCommands and is used to handle cases where the responses are processed in a different order from that in which the commands were issued. Each ALD only has one ALDCommandSequence counter, not one per primary.

An ALD shall increment its ALDCommandSequence counter every time it issues an ALDCommand and the primary shall copy this field unchanged into the response. The primary shall not check or verify the ALDCommandSequence number in any way.

The maximum command message size is 264 octets.

```
PrimaryCommand <Name>Command {
      CommandCode_t          Command
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit
      DataLength_t           DataLength
      uint8_t                Data[]
}

ALDCommand <Name>Command {
      CommandCode_t          Command
      CommandSequence_t      ALDCommandSequence
      Subunit_t              Subunit
      DataLength_t           DataLength
      uint8_t                Data[]
}
```

7.4.2. Responses

A response is sent by the receiver of a command.

Responses are defined as structures. A response issued by the primary is identified by the keyword “PrimaryResponse” and response issued by an ALD is identified by the keyword “ALDResponse”.

The names of responses have the suffix “Response”.

The maximum response message size is 264 octets.

7.4.2.1. Successful execution of command

Parameter 1 specifies the procedure that was executed.

Parameter 2 is the command sequence number which must be copied verbatim from the command.

Parameter 3 has the value OK to indicate that the procedure was successfully executed.

Parameter 4 is the data length, which states the number of octets in the message data for the response.

7.4.2.2. Failed execution of command

Parameter 1 specifies the procedure that failed to execute.

Parameter 2 is the command sequence number which must be copied verbatim from the command.

Parameter 3 is the return code which identifies the cause of the failure.

Parameter 4 is the data length, which states the number of octets in the message data for the response.

Parameter 5 is the ALD state of the ALD.

Parameter 6 is the connection state of the port that the command was received on. The state information is provided to help identify the detailed cause of the failure.

```
PrimaryResponse <NameResponse> {
     CommandCode_t              Command
     CommandSequence_t          ALDCommandSequence
     ReturnCode_t               ReturnCode
     DataLength_t               DataLength
     if (ReturnCode == OK) {
          uint8_t               Data[]
     }
}
```

```
ALDResponse <Name>Response> {
      CommandCode_t              Command
      CommandSequence_t          PrimaryCommandSequence
      ReturnCode_t               ReturnCode
      DataLength_t               DataLength
      if (ReturnCode == OK) {
            uint8_t              Data[]
      } else {
            ALDState_t           ALDState
            ConnectionState_t    ConnectionState
      }
}
```

7.5. Definition of UniqueID

The UniqueID is a concatenation of the vendor code (2 octets) part and an exactly 17-octet long unit specific part containing unit specific code (for instance serial number), exclusive to each ALD, provided by the vendor to whom the vendor code is assigned. The vendor code is placed in the left-most (most significant) position of the UniqueID. The vendor, to whom the vendor code is assigned, is responsible for ensuring the uniqueness of the UniqueID for each ALD. The UniqueID shall consist of ASCII characters between 0x21 and 0x7E, inclusive. If the unit specific code is shorter than 17 octets, the unit specific code is right aligned in the unit specific part and any unused octets are filled with 0x00.

```
UIDString_t UniqueID[19]
```

```
uint8_t L              // Length of unit specified code
UIDString_t U[L]       // Unit specified code
UniqueID[0..1] ← VendorCode
L ← the length of unit specific code
U[L] ← the unit specific code

FOREACH N FROM 0 TO 16-L DO
      UniqueID[N+2] ← 0x00
ENDFOR

FOREACH N FROM 0 TO L-1 DO
      UniqueID[17-L+N+2] ← U[N]
ENDFOR
```

8. GENERAL ASPECTS

8.1. General

AISG v3.0 specifies the standard interface between the primary, typically a base station, and ALDs which are units close to mobile base station antennas. ALDs include one or more subunits of different subunit types such as RET, TMA and antenna sensors.

An ALD may have one or more AISG interfaces to be controlled by one or more primaries. Therefore, AISG v3.0 defines two different types of ALDs, which are termed Single-primary ALDs (SALD) and Multi-primary ALDs (MALD).

An ALD contained within an enclosure shall provide only one UniqueID. An enclosure containing more than one ALD is not allowed.

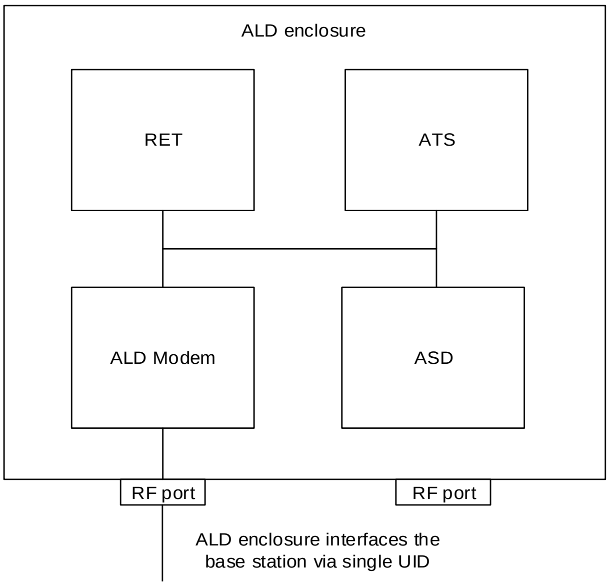


Figure 8.1-1: Example of an ALD enclosure

AISG v3.0 follows a three-layer model as a compact form of the OSI seven-layer reference model and includes only layers 1, 2 and 7:

- Layer 1 (physical layer) defines the signalling levels and basic data characteristics including data rates and OOK modem parameters.
- Layer 2 (data link layer) defines a specific class of the HDLC standard [6] used for signalling transport.
- Layer 7 (application layer) defines the data payload format and required command set. This basic functionality of the layer 7 is described in this standard and is extended by subunit type standards.

8.1.1. Layer 1

Layer 1 provides a multi-drop broadcast link between the primary and all ALDs. Any message transmitted will be received by all other ALDs. If two ALDs transmit at the same time, their messages may be garbled.

Layer 1 defines an additional type of port, which is called a PING port, which has the capability to realise OOK pinging.

8.1.2. Layer 2

Layer 2 provides:

- A data packet communication format;
- An addressing scheme;
- A master/slave relationship whereby the primary controls the half duplex timing;
- A frame checksum scheme to detect transmission errors;
- A frame sequence numbering scheme which protects layer 7 from:
  - Duplicated frames;
  - Deleted frames;
  - Receiving frames in the wrong order;
  - A flow control mechanism protecting each ALD frame receiver from being overrun by frames.

These functions provide layer 7 with a safe virtual full-duplex connection between the primary and each ALD. This virtual full-duplex connection allows both the primary and the ALD to transmit layer 7 messages between the primary and the connected ALD whenever required. Actual delivery time on layer 7 depends on the layer 2 polling frequency, which is chosen by the primary.

Each layer 2 link belongs to one primary and a primary may have multiple layer 2 links.

8.1.3. Layer 7

The function of the layer 7 is to support:

- Control of ALD subunits (for instance RET subunit, TMA subunit)
- Common commands
- Software and configuration download
- Alarm reporting
- Site mapping
- OOK pinging
- MALD Configuration.

Common commands, which include Site Mapping and MALD configuration, are generic and are defined in this document.

ALD functionalities are provided by subunits, each having their own subunit type (for instance RET, TMA).

8.1.4. SALD and MALD

A SALD is controlled by a single primary, it can have only one control port. A SALD may have multiple AISG ports. Each port can simultaneously have an assigned ALDAddress. At any time, only one primary can establish a layer 2 link and activate layer 7 to a SALD.

Any port that is connected to a primary by a layer 2 link is termed a control port.

A MALD shall be able to support multiple control ports, each of which is independently connected by a layer 2 link and each control port can accept layer 7 activation from one primary at any time.

Every AISG port can become a control port.

For a MALD, the mapping of subunits to control ports is carried out by the process of configuration. Access to a specific ALD subunit may be limited on certain ports as determined by the active MALD configuration.

The port numbering scheme is vendor specific. Port numbers shall start from 1 and it is not allowed to have gaps in the port numbering sequence.

Primaries supporting AISG v3.0 (for example a handheld controller supporting AISG v3.0) can be used to configure AISG v3.0 MALDs which can then work even in configurations that do not contain any AISG v3.0 primaries.

8.1.5. Subunits

The functionalities of an ALD are provided by one or more subunits. Each subunit has a subunit-type such as RET, TMA or ADB as defined in separate subunit type standards.

Subunits are identified by a unique subunit number incrementing sequentially from 1. A fully equipped ALD shall have no gaps in the subunit numbers. An ALD product version not fully equipped may omit certain subunit number in that sequence.

A configured MALD may make any subset of its subunits visible on any AISG port. MALD configuration does not renumber subunits. If a subunit is visible on multiple AISG buses, it shall have the same subunit number on each bus and shall have the same behaviour regarding controlling, alarming and configuration.

A single functionality may be presented as multiple subunits, each with its own independent settings, to allow for example two primaries to set different alarm thresholds of a sensor. This behaviour is defined in the subunit type standard.

Subunit number 0 refers to the ALD. Entity. Subunit number 1..65535 identifies a specific subunit.

8.1.6. Subunit type

Each subunit has a dedicated subunit type which represents its functionality (for instance RET, TMA). Subunit types are identified by a 1-octet unsigned integer which is defined in the corresponding subunit type standard.

8.1.7. Ports

A port is a signal interface. Several ports may be contained in a single multi-coupling connector system. Ports are described by port properties.

An ALD shall only support link establishment on ports that supply the ALD with DC power.

8.1.7.1. Interconnections

Generally, signals pass via interconnections within an ALD from one port to one or more other port(s). Some ports, for example those on antennas and sensors, may have no interconnection to any other port.

Each interconnection is between two ports. Information about all interconnections from one port to other ports and their properties can be retrieved from the ALD. This information is primarily used during site mapping.

8.1.7.2. Subunit association

A port may be associated with one or more subunits. Multiple ports can be associated with the same subunit. The term *associated* is used to describe that there is a functional relationship between port(s) and subunit.

A subunit may be associated with an interconnection and the term used is that the subunit has functional relationship with this interconnection. In that case the associated subunit is associated with both ports of this interconnection. There may be subunits (for example temperature sensor), without any associated port. (see Figure 8.1.7.2-1: “Subunit association”).

As an example, a TMA subunit is associated with an interconnection between the RF input and output ports of the TMA.

A subunit may be associated with several ports without any interconnection between these ports. As an example, a RET subunit is associated with one or more RF ports of an antenna without any interconnection.

In the case of a MALD, all subunits are always reported to all connected primaries in Site Mapping command responses, regardless of the configuration of the MALD.

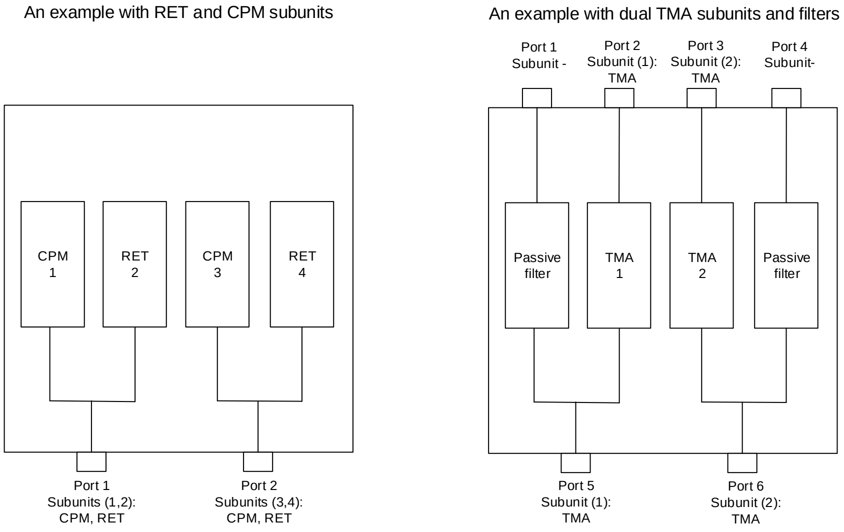


Figure 8.1.7.2-1: Subunit association

8.2. State models

The state model diagrams contain only the transitions from one state to another.

8.2.1. State models for layer 2

8.2.1.1. Layer 2 LinkState model of a SALD

The layer 2 LinkState model of a SALD (one per port) is shown in Figure 8.2.1.1-1: “State model for a SALD control port”. Events are written in italic and layer 2 and layer 7 commands are written in bold.

The state model is valid for all AISG input ports of a SALD with the following limitations:

- The Connected LinkState only applies to the AISG input port that first received an SNRM command. This input port is thereafter known as the control port (see Figure 8.2.1.1-1: “State model for a SALD control port”).
- SNRM commands received on any port except the control port shall be rejected and the response shall be DM.
- SNRM commands received on any port except the control port shall be rejected and after link establishment on the control port the response on any other port shall be DM.
- The ResetPort XID command shall be accepted:
    - On all AISG input ports while the SALD is not in Connected LinkState.
    - Only on the control port when the SALD is in Connected LinkState.

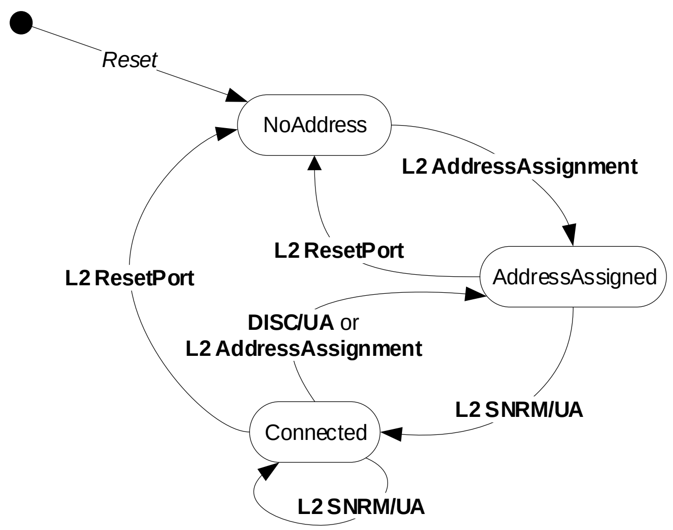


Figure 8.2.1.1-1: State model for a SALD control port

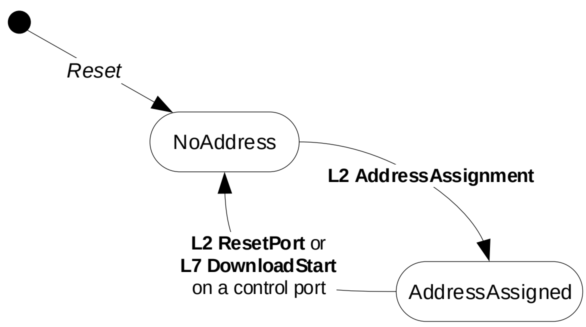


Figure 8.2.1.1-2: State model for a SALD non-control port

8.2.1.2. Layer 2 LinkState model of a MALD

The layer 2 LinkState model for the layer 2 of the MALD is shown in Figure 8.2.1.2-1: "LinkState model for a MALD AISG port". Each AISG port has a LinkState. Events are written in italic and layer 2 and layer 7 commands are written bold.

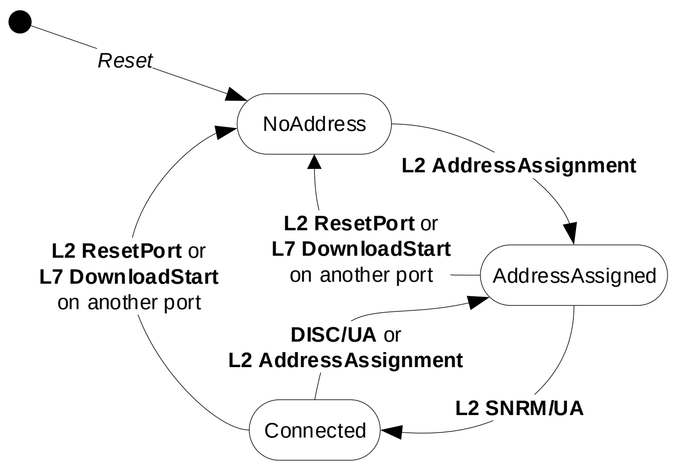


Figure 8.2.1.2-1: LinkState model for a MALD AISG port

8.2.1.3. Layer 2 LinkState model of a primary

The layer 2 LinkState model of a primary is not defined in this document. The behaviour of a primary shall be based on the LinkState models of the ALDs.

8.2.2. State model for layer 7

The state model in Figure 8.2.2-1: "ALDState state model" shows the relationship between different states of the whole ALD.

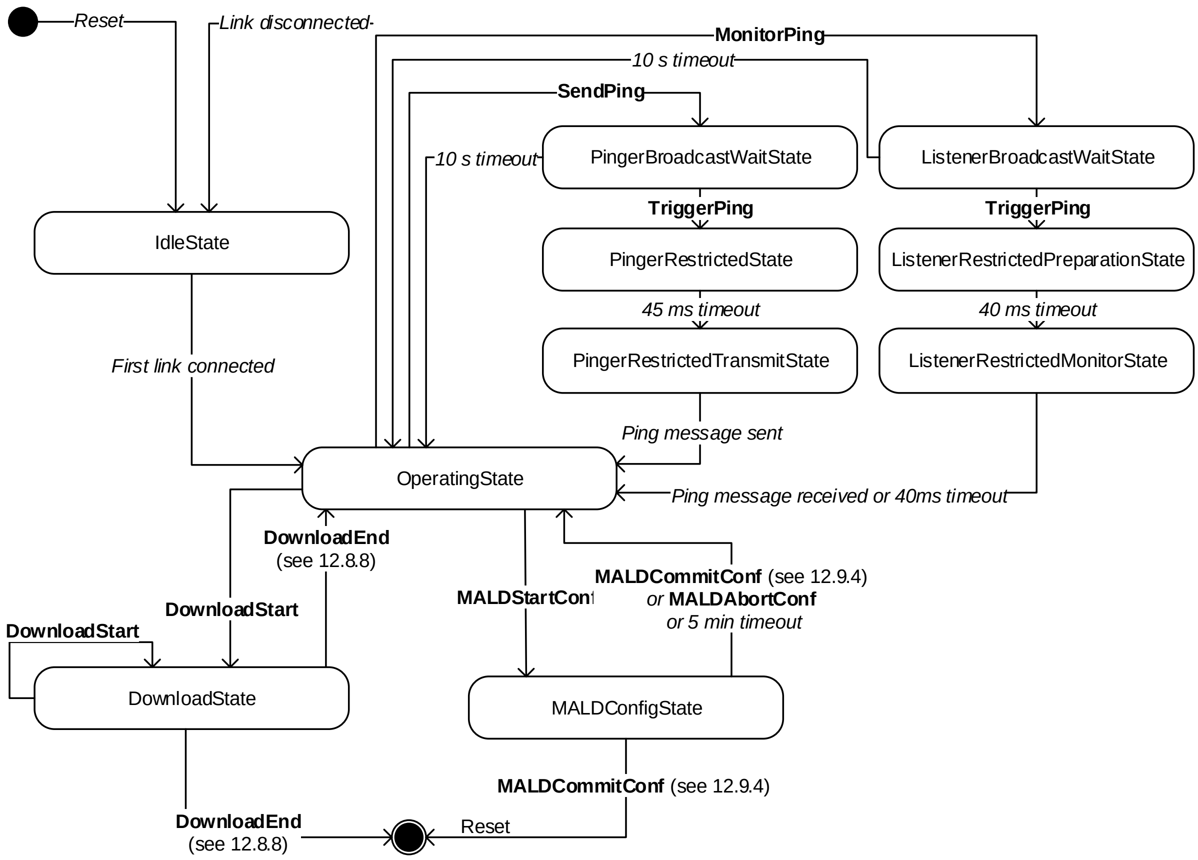


Figure 8.2.2-1 ALDState state model

The relationship between different ConnectionState states is shown in Figure 8.2.2-2: “ConnectionState state model”.

Each control port has a ConnectionState.

A MALD may operate AISG v2 and AISG v3.0 simultaneously on different control ports.

This document does not define state models for ALDs in AISG v2 mode.

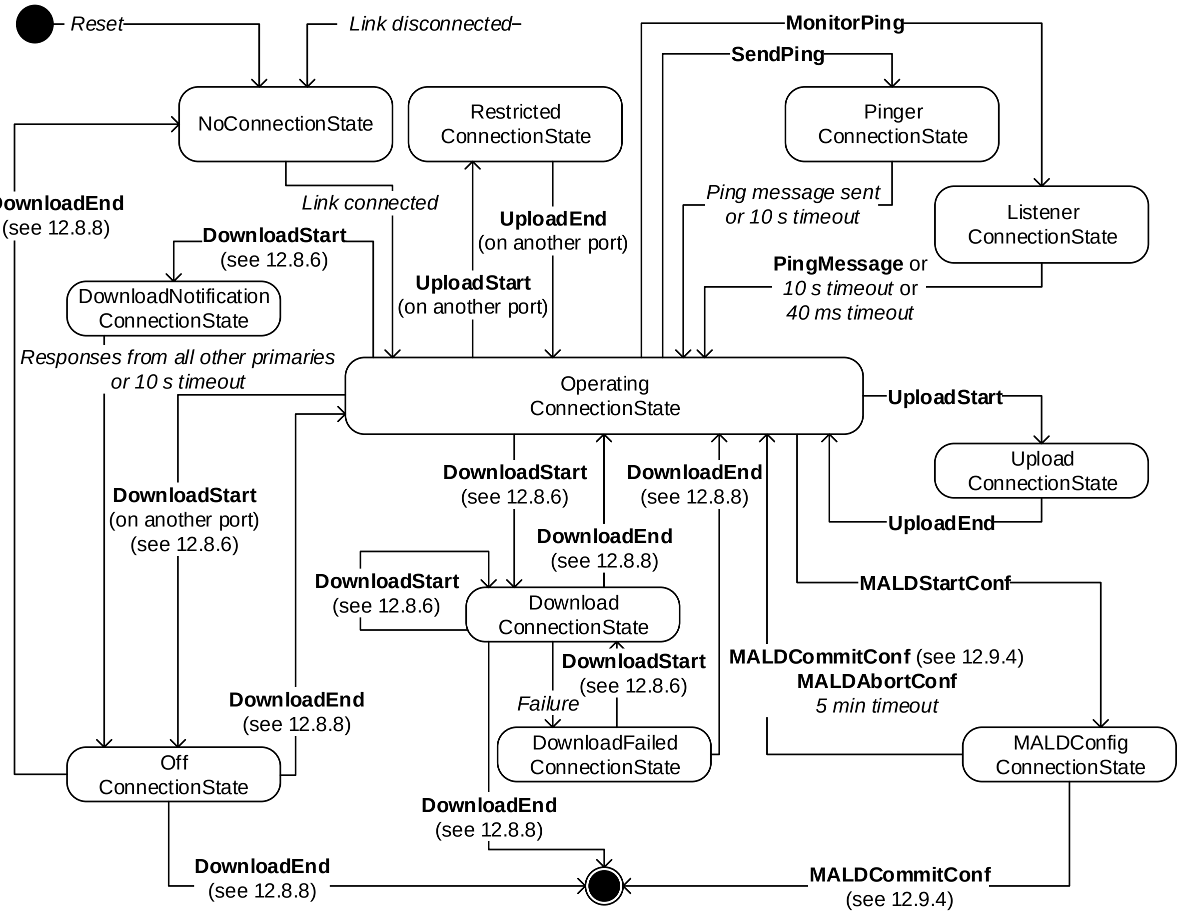


Figure 8.2.2-2: ConnectionState state model

8.3. Site mapping process

Site mapping is deduced by the primary based on information reported by each ALD. If the site has more than one AISG primary, the entire site view may be assembled from each primary's map.

To generate the complete map, the primary requires that all ALDs support AISG 3.0 and all antennas include an Antenna Database (ADB). An ADB is a specific subunit type that contains antenna array properties. There shall exist a maximum of one ADB subunit within an ALD.

To generate the site mapping, the primary shall execute the following steps:

1. For each ALD, retrieve the ALD input/output port details including subunit association;
2. For each ALD, retrieve the type of interconnection between its ports;
3. For each port of each antenna, retrieve the list of Array IDs connected to it;
4. If available, perform the ping process to assign RF Path ID(s) to the RF ports of the ALD;

5. Assign RF Path ID(s) to the RF ports of the ALD. If pinging is not available, input this data manually;
6. For each array, retrieve the array ID, position within the sector, array position, polarization, array frequency range, sector ID, mechanical bearing and mechanical tilt;
7. For each array ID collected on the network, assign the RF Path ID and optionally the RF Path ID alias;
8. For each RF Path ID, store the sector ID;
9. For each RF Path ID assign a user-friendly RF path ID Alias.

The MALD shall provide separate RF Path ID and RF Path ID Alias tables for each of its control ports. Each connected primary can only set and read its associated RF Path ID and RF Path ID Alias tables.

NOTE: Step 3 allows a primary to know how many antennas and antenna ports are in its AISG network.

NOTE: Step 4 allows a primary to count the number of RF paths in its antenna line.

8.4. Pinging

AISG v3.0 enables verification of the RF-cable connections in the antenna system by sending Ping messages between the RF ports of antennas, ALDs and the primary. The method can be used to diagnose crossed, missing, stolen or broken RF cables by comparing the actual connections with the site installation plan. This ping process can be used in conjunction with Site Mapping commands to automatically discover the RF paths.

During a ping process cycle a MonitorPing command is sent on one port and every ALD that receive this command listens for a subsequent Ping message.

When a listener receives the Ping message, an RF cable connection has been identified.

The ping process repeats these ping process cycles over and over for all port combinations in order to identify all cabling in the antenna system.

Pinging is initiated and controlled by the primary.

Pinging is only possible via OOK connections.

The ALD that sends the Ping message is called the pinger.

The primary and the ALD(s) that listen for the Ping message are called listeners.

The listener that received the Ping message is called the pingee.

The ping process commands are:

- MonitorPing : A layer 7 primary command sent to the ALDs selected to listen to the Ping message. The ALD listeners monitor the requested RF port.
- SendPing : A layer 7 primary command sent to the ALD selected to be the pinger.
- TriggerPing : A layer 2 primary command broadcast to the pinger and ALD listeners. Upon reception, the pinger sends the Ping message and the listeners monitor the RF port for the Ping message.

- AbortPing : A layer 7 primary command sent to the pinger and ALD listeners aborting the ping process cycle.
- A Ping message is a layer 2 message send by the pinger when it receives the TriggerPing command.

Figure 8.4-1: “Pinger and listener” depicts a simple example of a single ping process cycle where a primary uses pinging to identify the RF path between antenna port 6 and TMA port 4. In this case the pinger is ANT-1 and the listener is the TMA-1. The Ping message is sent from ANT-1 port labelled 6 and received on the TMA-1 port labelled 4.

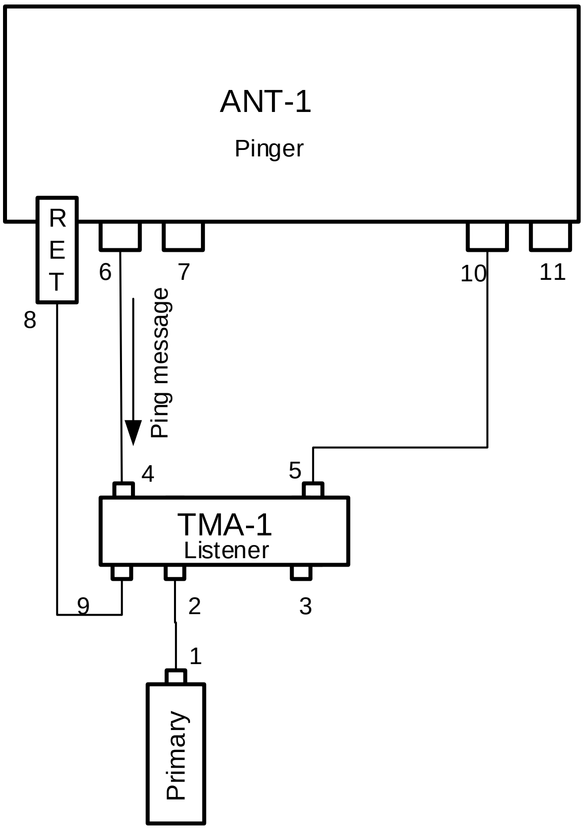


Figure 8.4-1: Pinger and Listener

8.4.1. Rules for ping process

The following rules shall apply to complete the ping process:

1. The Ping message is transmitted from a pinger (a SALD or a MALD), and monitored by listeners (SALD, MALD or primary).
2. The Ping message is transmitted towards the primary. The ping process starts from a ping-capable antenna or from the ping-capable ALD closest to an antenna. The order of the ALDs is found during the device scan.
3. Both OOK and PING ports can transmit and receive Ping messages.

4. Pinging is allowed:
   - On ports without an active layer 2 link.
   - On ports, with an active layer 2 link, that belongs to the primary running the ping process.
5. The Ping message is an OOK layer 2 message containing the 4-octet Primary ID of the primary initiating the ping process. The Ping message cannot be transmitted via an RS-485 connection.
6. During a ping process cycle the primary shall send MonitorPing commands to all ALDs, the primary selects to listen.
7. During a ping process cycle the primary shall send the SendPing command to the ALD it selects as the pinger.
8. During a ping process cycle the ALD (pinger and listener(s)) shall deactivate all internal OOK paths not belonging to another primary.

NOTE: A MALD does not inform other connected primaries about the start of the ping process cycle.

9. An ALD shall enter the ping process cycle only when it is in OperatingState.
10. If there is a ping process cycle in progress, a MALD shall reject any request to start a new ping process cycle from any other primary.
11. If there is a ping process cycle in progress, an ALD shall only accept the AbortPing command from the primary that started the ping process cycle when in ListenerBroadcastWaitState and PingerBroadcastWaitState.
12. The pinger shall wait up to 10 seconds for the TriggerPing. If the TriggerPing has not been received within the 10 seconds, the pinger switches ALDState to OperatingState.
13. The listener(s) shall wait up to 10 seconds for the TriggerPing. If the TriggerPing has not been received within the 10 seconds, the ALD listener switches ALDState to OperatingState.
14. The primary shall broadcast the TriggerPing command.
15. The primary shall not send any message for 95 ms after it has broadcast TriggerPing command.
16. Upon receiving the TriggerPing command, the pinger shall wait 45 ms before sending the layer 2 Ping message.
17. The pinger switches ALDState to PingerRestrictedTransmitState immediately after queueing the Ping message for transmission.
18. The pinger switches ALDState to OperatingState when the transmission complete event occurs.
19. The listeners(s) shall switch ALDState to ListenerRestrictedTransmitState immediately after receiving the TriggerPing command and wait 40 ms before monitoring the RF Port.
20. After the 40 ms wait, the listener switches ALDState to ListenerRestrictedMonitorState, the listener monitors the RF port for up to 40 ms.

21. If the listener receives the Ping message it switches ALDState to the OperatingState. It is now designated the pingee.
22. If no Ping message is received by a listener after the 40ms, it switches ALDState to the OperatingState.
23. If a primary receives an IncorrectState rejection (because a second primary connected to the same MALD has already initiated a ping process) the primary shall send the AbortPing command to all other ALDs.

The primary may apply a random delay before executing another ping process cycle. This reduces the possibility of deadlock between multiple primaries controlling the same ALD.

8.4.2. Ping process cycle

The Ping process cycle steps are based on PingTimers, whose accuracy shall be better than ± 1 ms.

1. The primary sends a layer 7 MonitorPing command to all the listeners, specifying the port on which they shall listen.
2. Each listener stores the requested RF port as PingMonitorRFPort, switches ALDState to ListenerBroadcastWaitState and initiates its 10-second PingTimer.
3. The primary sends a layer 7 SendPing command to the pinger, specifying the port on which the Ping message shall be transmitted.
4. The pinger stores the requested RF port as PingSendRFPort, switches ALDState to PingerBroadcastWaitState and initiates its 10-second PingTimer.
5. If a listener’s PingTimer expires (after 10 seconds), it sends a PingTimeout command and switches ALDState to OperatingState.
6. If a pinger’s PingTimer expires (after 10 seconds), it sends a PingTimeout command and switches ALDState to OperatingState.
7. The primary broadcasts a layer 2 TriggerPing command and initiates a 95-ms PingTimer.
8. When a listener receives the layer 2 TriggerPing command, it switches ALDState to ListenerRestrictedPreparationState and initiates a 40-ms PingTimer. It selects its PingMonitorRFPort and deactivates all OOK paths associated with the primary that initiated the ping process cycle.
9. When the pinger receives the layer 2 TriggerPing command, it switches ALDState to PingerRestrictedState, initiates its 45-ms PingTimer, selects its PingSendRFPort and deactivates all OOK paths associated with the primary that initiated the ping process cycle.
10. When each listener’s PingTimer expires (after 40 ms), it sets its PingReceivedFlag to 0, clears its receive buffer, switches ALDState to ListenerRestrictedMonitorState and initiates its 40-ms PingTimer.
11. When the pinger’s PingTimer expires (after 45 ms), it queues a Ping message for transmission, switches ALDState to PingerRestrictedTransmitState and deactivates all OOK paths associated with the primary that initiated the ping process cycle.

12. When the pinger's serial port has transmitted the stop bit of the closing flag of the Ping message, the ALD switches back to the AISG port on which it received the SendPing command, switches ALDState to OperatingState and activates all previously deactivated OOK paths.
13. When a listener receives the Ping message, it stores the primary's ID as PrimaryID, sets its PingReceivedFlag to 1, switches back to the AISG port on which it received the MonitorPing command, switches ALDState to OperatingState and activates all previously deactivated OOK paths.
14. If a listener's PingTimer expires, it switches back to the AISG port on which it received the MonitorPing command, switches ALDState to OperatingState and activates all previously deactivated OOK paths.
15. When the primary's 95-ms PingTimer expires, it may continue with the next ping process cycle.

8.4.3. Flow diagrams

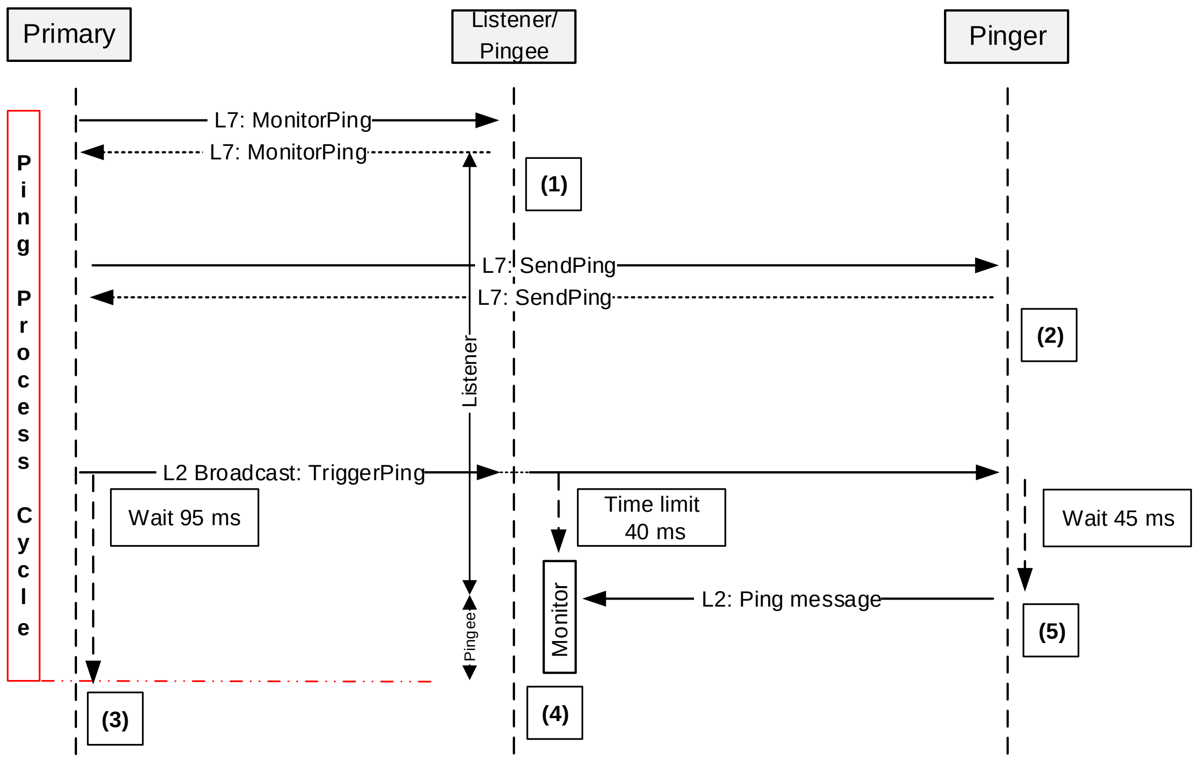


Figure 8.4.3-1: Sequence diagram for a ping process cycle

NOTE 1: Once the response is sent, the listener sets its ListenerBroadcastWaitState timeout to 10 seconds and switches back to OperatingState when the timer expires.

NOTE 2: Once the response is sent, the pinger sets its PingerBroadcastWaitState timeout to 10 seconds and switches back to OperatingState when the timer expires.

NOTE 3: Once TriggerPing is sent, the primary waits for 95 ms and may continue with the ping process cycle.

NOTE 4: Once TriggerPing is received, the listener waits for 40 ms during which time it deactivates all OOK paths associated with the primary that initiated the ping process cycle, monitors the port for up to 40 ms and returns to the OperatingState. (see Section 8.4.2. "Ping process cycle" item 14).

NOTE 5: Once TriggerPing is received, the pinger waits 45 ms, sends the Ping message and returns to the OperatingState.

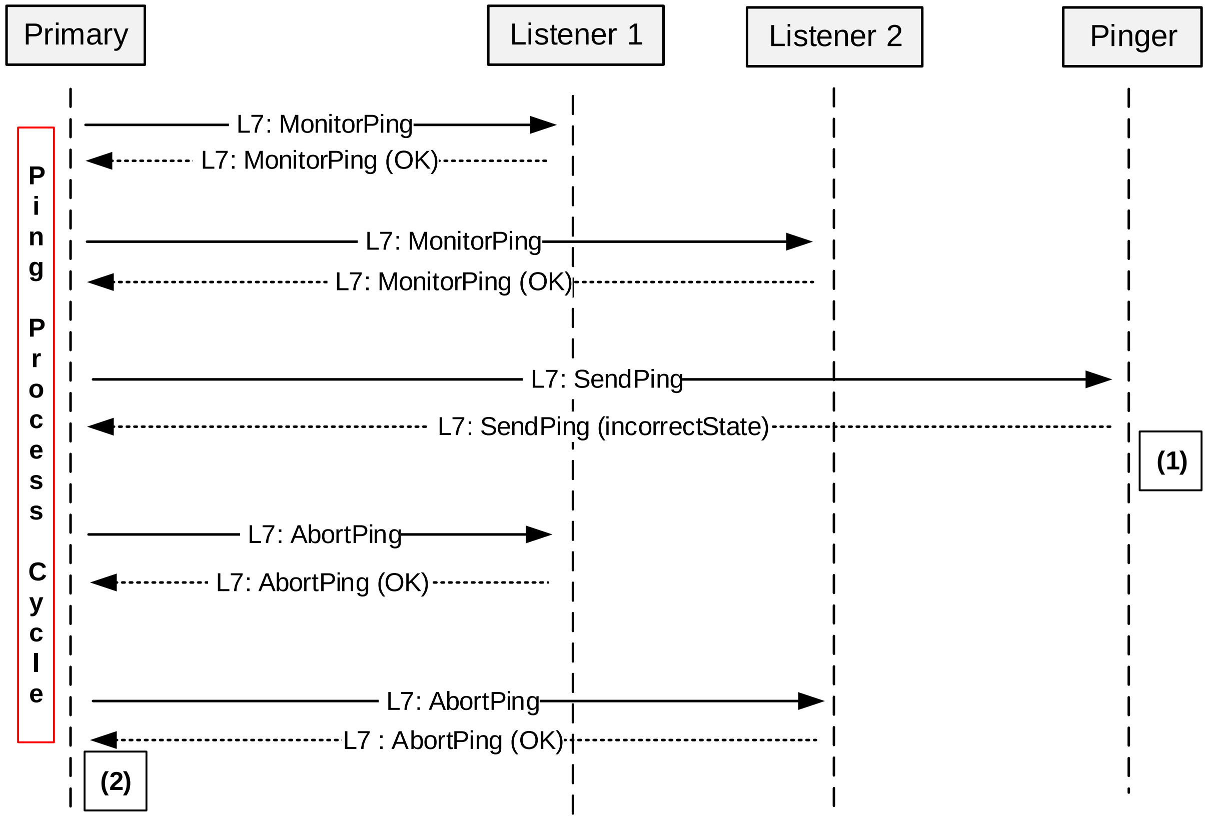


Figure 8.4.3-2: Sequence diagram when the MALD rejects the Ping Process

NOTE 1: The MALD rejects the SendPing command with the ReturnCode_t IncorrectState because another primary has already initiated the ping process.

NOTE 2: The primary may apply a random delay before retrying and start the ping process again.

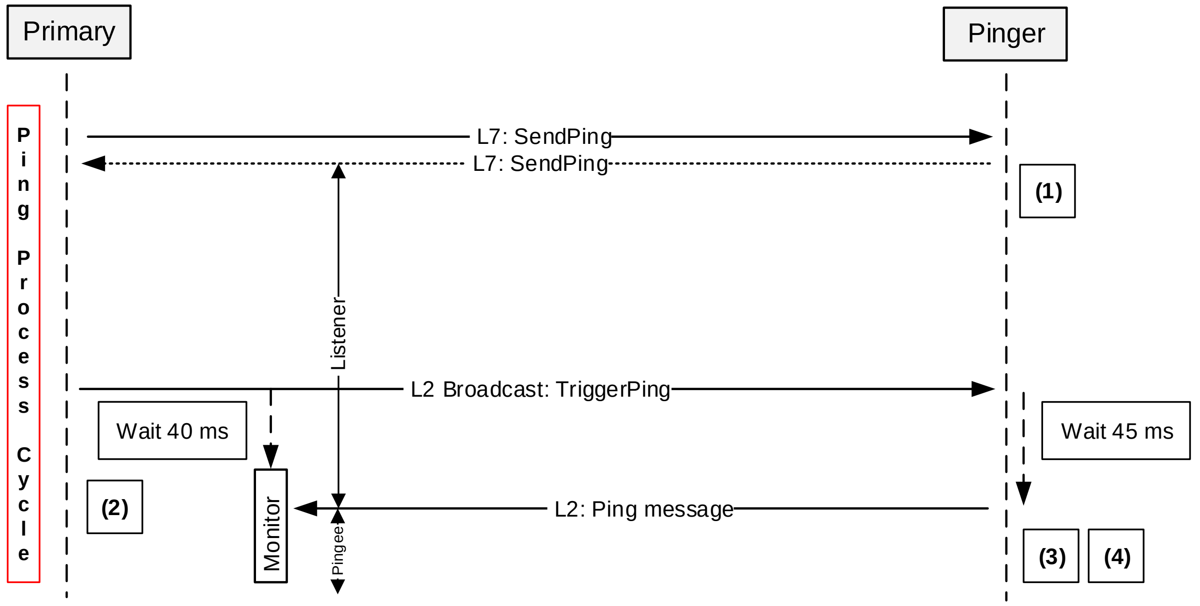


Figure 8.4.3-3: Sequence diagram when the primary receives the Ping message

NOTE 1: Once the response is sent, the pinger sets its PingerBroadcastWaitState timeout to 10 seconds and returns to OperatingState if it expires.

NOTE 2: Once TriggerPing is received, the primary listener waits for 40 ms and then monitors the port for up to 40 ms.

NOTE 3: Once TriggerPing is received, the pinger waits 45 ms, sends the Ping message and switches ALDState to OperatingState.

NOTE 4: If the primary does not receive the Ping message within 95 ms, it may continue the ping process cycle.

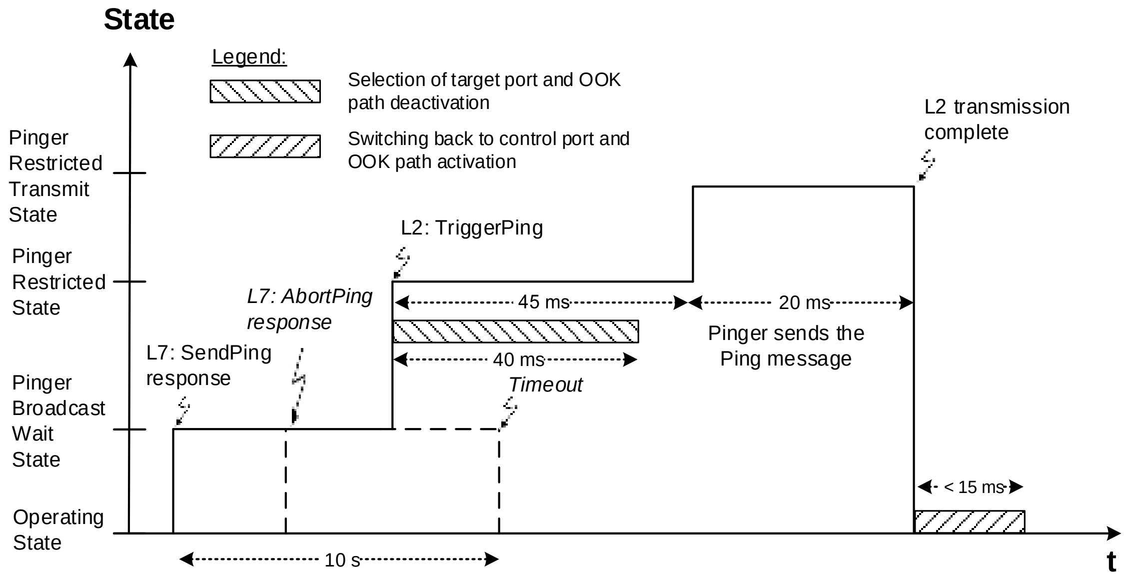


Figure 8.4.3-4: Pinger ALDState timing diagram

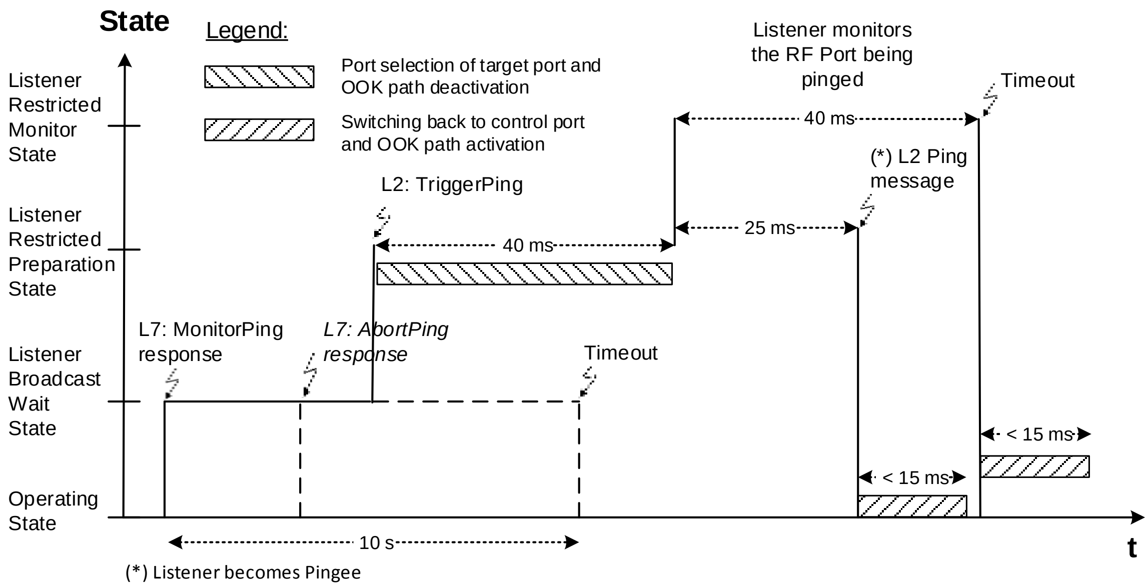


Figure 8.4.3-5: Listener ALDState timing diagram

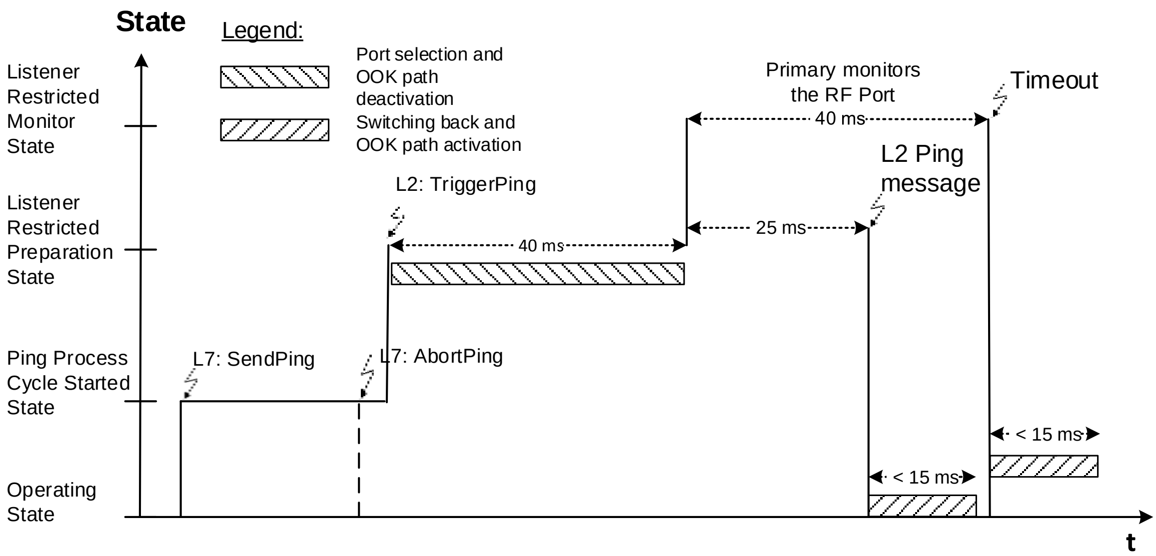


Figure 8.4.3-6: State diagram when the primary receives the Ping message

NOTE: The states in Figure 8.4.3-6: "State diagram when the primary receives the Ping message" apply to the primary only.

8.5. MALD configuration

8.5.1. Introduction

MALD configuration is performed to control the access of the connected primaries. It assigns the control of different subunits to the primaries connected to the MALD.

The concept of MALD configuration consists of two domains: MALD authority and MALD configuration security.

MALD configuration provides a method for a primary to configure the control authorities (ReadWrite, ReadOnly or NoAccess) of each primary in respect of the subunits of a MALD.

This is achieved by configuring the authorities of all the ports that may be used to control the MALD; these ports have PortPropertiesType RS485ControlPort or RFControlPort. Any primary connected to a MALD port and having appropriate MALD security setting can configure all authorities within the MALD.

Subunit	Control Port 1 (Primary 1)	Control Port 2 (Primary 2)
1	ReadWrite	ReadOnly
2	ReadOnly	ReadWrite
3	NoAccess	ReadWrite

Table 8.5.1-1: Example of authority settings in MALD configuration (MALD with 3 subunits and 2 control ports)

MALD security provides a method for a primary to configure the security settings of each primary in respect of MALD configuration and MALD SW Download. This means that MALD security controls which primaries are allowed to perform MALD configuration and SW download.

Security type	Control Port 1 (Primary 1)	Control Port 2 (Primary 2)
MALD Configuration	Allowed	Not Allowed
SW Download	Allowed	Not Allowed

Table 8.5.1-2: Example of MALD security settings (MALD with 2 control ports)

These methods are not applicable to a SALD.

```
Enumeration AuthorityType_t : uint8_t {
      NoAccess                ← 0
      ReadOnly                ← 1
      ReadWrite               ← 2
}

Enumeration SecurityType_t : uint8_t {
      MALDConfiguration       ← 0
      SWDownload              ← 1
}

Enumeration SecuritySettingType_t : uint8_t {
      NotAllowed              ← 0
      Allowed                 ← 1
}

Enumeration SettingSourceType_t : uint8_t {
      Active                  ← 0
      Volatile                ← 1
}

// Subunits are indexed from 0, but index 0 is never used here
struct MALDAuthType_t {
      AuthorityType_t         Authority[NrOfSubunits+1]
      SecurityType_t          Config
      SecurityType_t          SWDownload
}

struct MALDAuthType_t VolatileAuth[MaxPort+1]
uint16_t VolatileCommitCounter
persistent struct MALDAuthType_t ActiveAuth[MaxPort+1]
persistent uint16_t ActiveCommitCounter
```

8.5.2. MALD configuration transactions

The MALD configuration and security settings are changed using MALD configuration transactions. These are used to prevent a second transaction from being started until the previous transaction has been completed.

All MALD configuration commands belonging to the same transaction must be initiated and performed on the same AISG port. For the period of the transaction, this port shall be referred to as the transaction port.

MALDStartConf copies the active configuration to the volatile copy where it can be edited without affecting the current configuration or security settings. It then puts the MALD into MALDConfigState and starts the 5-minute timer.

MALDAbortConf can be used to discard the changes, end the MALD configuration transaction and enter the OperatingState.

MALDSetSubunitConf is used to edit the MALD configuration in volatile memory.

MALDSetSecurityConf is used to edit the MALD configuration security setting in volatile memory.

MALDCommitConf checks the validity of the volatile copy and if it is valid, it saves the contents of the volatile copy as the active configuration and finally performs a reset of the MALD (which in effect makes the MALD enter the IdleState).

The process to modify a configuration is to issue a MALDStartConf, a sequence of MALDSetSubunitConf commands to edit the current configuration and finally a MALDCommitConf to apply the changes.

The process to modify the security settings is to issue a MALDStartConf, a sequence of MALDSetSecuritySetting commands to edit the current configuration and finally a MALDCommitConf to apply the changes.

Both MALDSetSubunitConf and MALDSetSecuritySetting commands can be used within the same transaction.

MALD configuration and security settings are edited in the volatile copy. Changes in the volatile copy have no effect on the MALD configuration or security settings. Once all desired settings are completed, the content of the volatile copy is validated, and if valid it is saved as the active configuration and the MALD resets (in effect applying the new configuration).

The MALDConfigState has a 5-minute timeout, which is cleared every time a MALDSetSubunitConf or MALDSetSecuritySetting command is sent. If the timeout expires the transaction is aborted, any uncommitted changes are discarded and the MALD enters the OperatingState.

The MALD maintains a persistent 16-bit wrap-around MALDConfCommit counter, incremented by one on each successful commit. The counter starts with a value of 0 and wraps around from 65535 to 1 (not to 0). The counter is not allowed to wrap around to 0 since this represents the default configuration. If this counter has not changed, the primaries can assume that the MALD configuration has not been changed. If the MALDConfCommit counter equals 0, the primary knows that the MALD is in the default configuration.

MALDResetConf can be used to re-initialise the MALD to its default configuration (see Section 8.5.3.3. “MALD default configuration”). This command also sets the MALDCommitCounter to 0.

After the MALD operation has resumed, primaries can discover the reset cause using the GetResetCause command. If the reset cause is MALDConfigChanged all the previous information about the MALD configuration may have become invalid and each primary should now discover the current MALD configuration.

In the default configuration all control ports have ReadWrite authorities for all subunits within the MALD. MALDResetConf returns a MALD to the default configuration.

NOTE: The default configuration is provided to allow an AISG v2 primary to access a non-configured MALD.

MALDGetInformation command provides information about the physical organisation of the MALD. It provides:

- configuration commit counter value
- number of control ports within the MALD
- list of control port numbers
- number of subunits within the MALD
- list of subunit number and type tuples

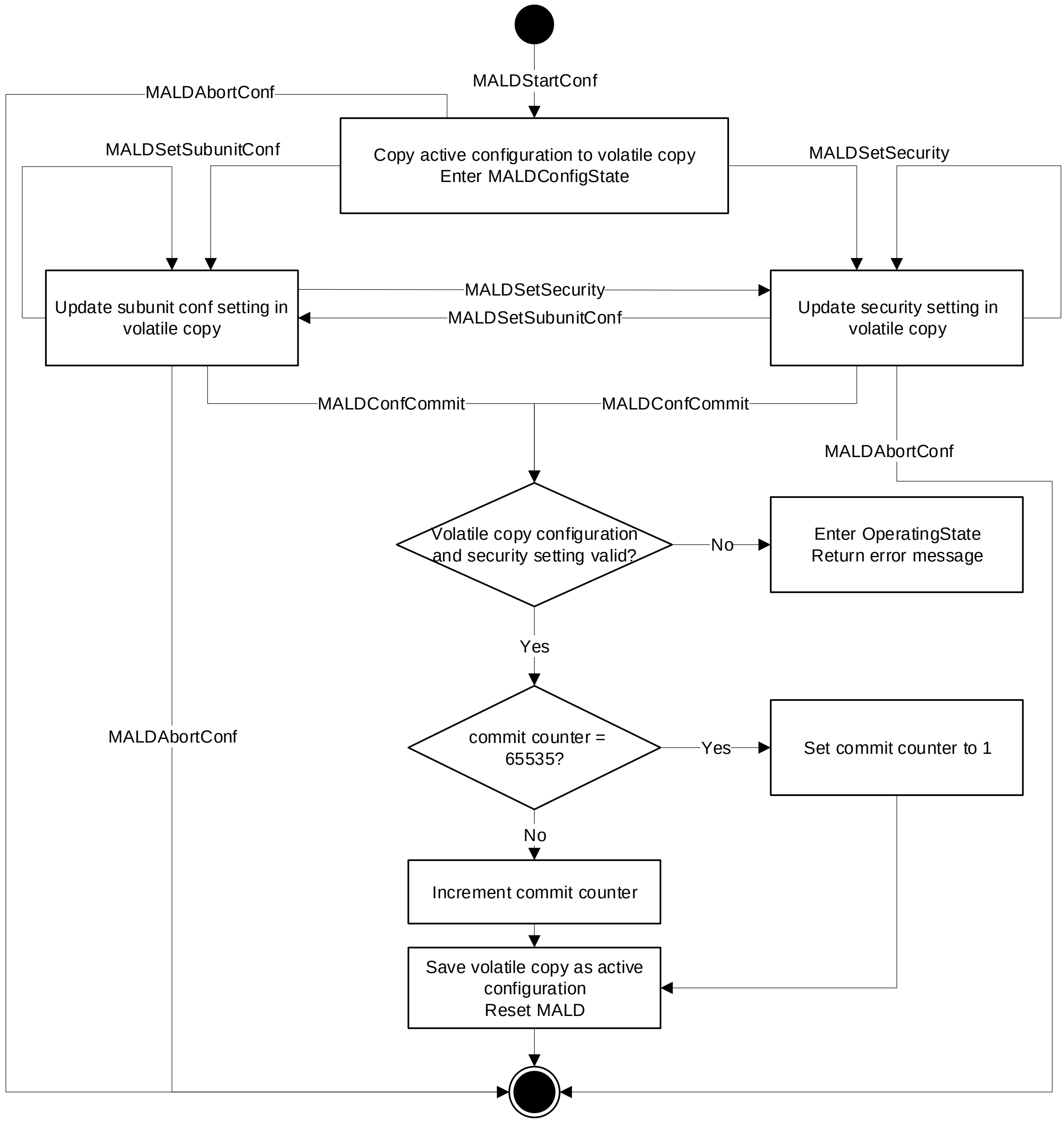


Figure 8.5.2-1: MALD Configuration Flow Chart

8.5.3. Authority control

8.5.3.1. Subunit authorities

Each primary has the following possible authorities: ReadWrite (RW), ReadOnly (RO) or NoAccess (NA).

If a primary has RW authority over a subunit, that subunit shall execute all commands addressed to it from that primary.

If a primary has RO authority over a subunit, that subunit shall execute only those commands which are designated for read-only.

If a primary has NA authority over a subunit, that subunit shall not be visible from that primary on layer 7.

If a primary connected to a MALD has NA authority to all the subunits of the MALD, the MALD shall execute commands not related to subunits (for instance MALDGetInformation or GetInformation).

The setting of authority for subunit number 0 (that is the entire MALD) is not permitted. Every MALD configuration command addressed to subunit number 0 shall be rejected.

Commands that may have limitation in authority are identified in this standard and relevant subunit type standards.

8.5.3.2. Subunit authorities configuration

The authority of each primary to control or monitor the subunits within the MALD is configured using the MALD configuration command set.

MALDSetSubunitConf edits the configuration in the volatile copy.

MALDGetSubunitConf can be used to read the contents of the active and volatile configuration. The volatile configuration can only be read when a MALD is in the MALDConfigState.

The MALD configuration commands are sent to subunit 0. They include parameters for the subunit whose authority is being configured and to which control port this authority applies.

8.5.3.3. MALD default configuration

Before a MALD is configured for the first time, it is in the default configuration. After the first configuration transaction is successfully completed, the MALD will no longer be in the default configuration. The only way to return the MALD to the default configuration is to use the MALDResetConf command.

In the MALD default configuration all primaries have ReadWrite authorities over all subunits. After the MALD is configured one primary can have ReadWrite authority over a subunit, it is also allowed to have no primary with authority over a subunit, effectively hiding the subunit.

The purpose of the MALD default configuration is to allow a primary to control a MALD without having to configure it. This is beneficial for AISG v2 primaries which do not support AISG v3.0, and therefore cannot configure a MALD. Since in default configuration more than one primary can control the same subunit, a risk of conflicting commands exists (e.g. two primaries commanding different tilts to same RET subunit). Users must exercise extra care when using the default configuration as AISG v3.0 authority control that prevents conflicting write type commands is not in effect.

NOTE: Using MALD in the default configuration is only recommended in situations where configuring it is not feasible.

8.5.3.4. MALD configuration security

MALD security settings define the ability of each connected primary to configure the MALD or to perform Software downloads.

The security parameter that controls the ability of the primaries connected to a MALD to configure the MALD is called MALDConfiguration and the security parameter that controls the ability of the primaries connected to a MALD to update the MALD SW is called SWDownload. These parameters can have two values, Allowed and NotAllowed.

MALDSetSecuritySetting enables editing of the security settings in the volatile copy.

MALDGetSecuritySetting is used to read the security parameters from the active security setting or volatile copy. The volatile security settings copy can only be read when the MALD is in the MALDConfigState.

A scenario in which no primary has the right to do MALD configuration is not allowed. This means that at least one primary shall have the right to configure the MALD at all times. To ensure that this is the case, the primary is not allowed to remove its right to configure the MALD.

If the right to configure the MALD needs to be transferred to another primary, the transfer of the right has to happen in two steps. In the first step the primary, that currently has the right to configure the MALD, gives the other primary the right to control the MALD by modifying the security setting accordingly. In the second step, the primary that just received the right to configure the MALD then removes the right to configure the MALD from the original primary.

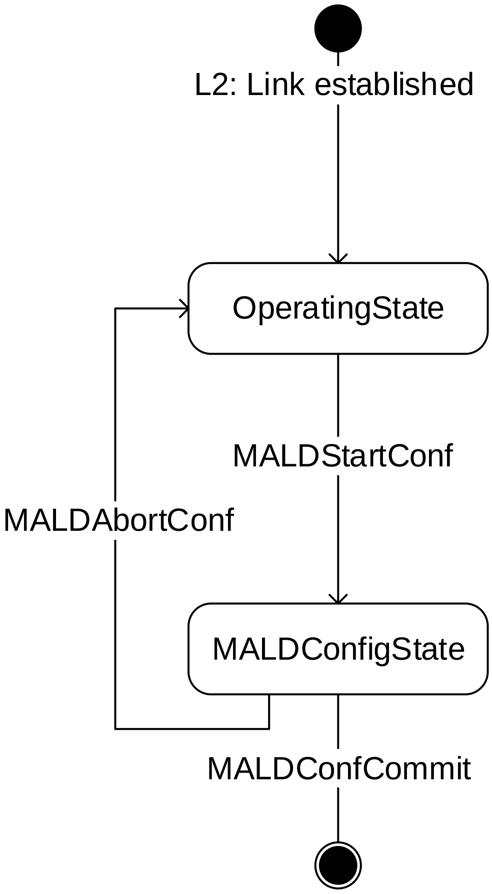


Figure 8.5.3.4-1: MALDConfigState

8.6. Download

The download process provides the capability of downloading files of certain file types to the ALD. The architecture allows only one file of each file type to exist within the ALD or any subunit. Since only one file per file type is permitted, there is no need for file names or indexes. Download can be performed to the ALD itself, indicated by subunit number 0, or to an individual subunit. The file to be downloaded is indicated by its file type. The supported file types and their allowable destinations (ALD or subunit) are defined in Section 12.8.6. “Download Start”.

The command DownloadStart selects the file type to be downloaded and starts the download process.

The file is downloaded by the primary issuing a series of DownloadFile commands. With each command a block of 256 octets in size is sent from the primary to the ALD until the complete file has been transferred. The last command in a sequence may transfer a block of less than 256 octets.

To indicate the successful completion of the Download process the primary sends the DownloadEnd command with the OptionCode Complete.

After a successful firmware download the ALD will restart. After a successful configuration file download the ALD will apply the changes, but not restart. After a successful information file download the ALD does not restart.

To terminate the Download prematurely the primary sends the DownloadEnd command with the OptionCode Cancel or the DownloadStart command (to re-start a new download immediately after cancelling the previous one). Regardless, the ALD does not restart.

If DownloadFile detects an invalid file or some hardware problems or is unable to store the data, it returns an error code and will reject further DownloadFile commands. The primary must cancel the download as described in previous paragraph.

Parallel execution of another download process or any other layer 7 command is not allowed even from different subunits within the same ALD However, the ALD may send AlarmIndication commands during a download.

The data content of the DownloadFile command is vendor specific, but it is recommended to implement an application software validation feature that prevents the risk of downloading faulty or invalid application software. After a failed SW download the ALD shall not be left into a state where it has no working software.

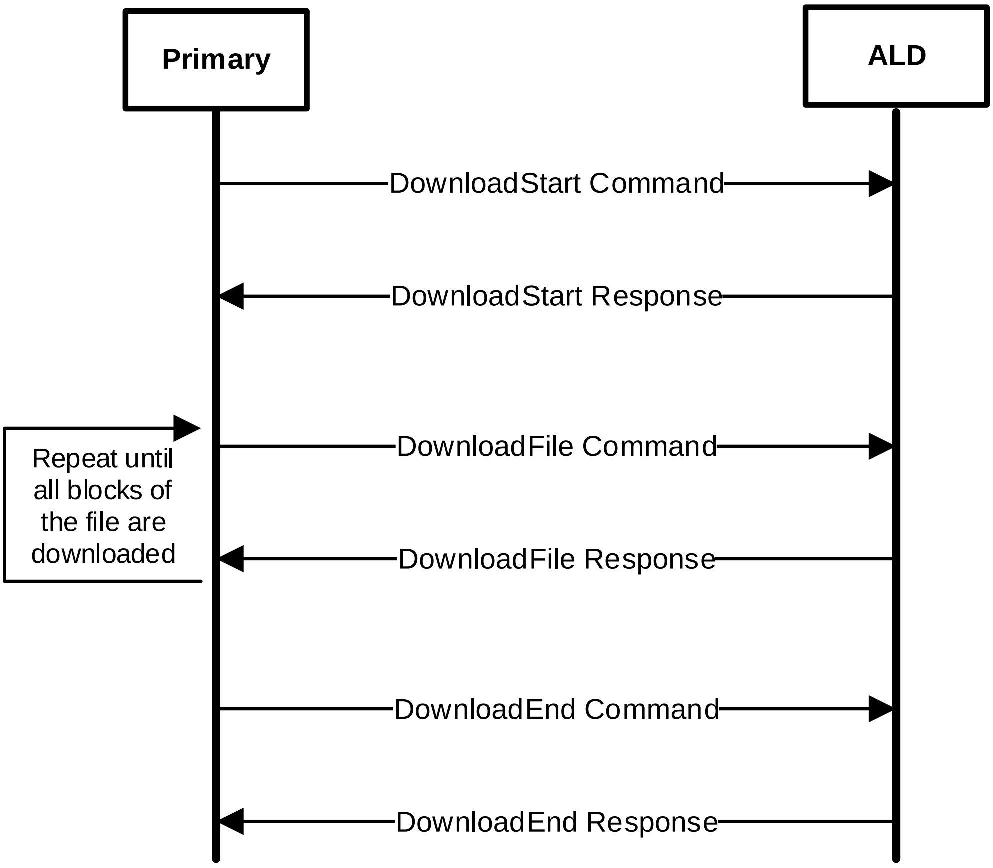


Figure 8.6-1: Command sequence for Download

8.7. Upload

The upload process provides the capability of uploading files of certain file types from the ALD. The architecture allows only one file of each file type to exist within the ALD or any subunit. Since only one file per file type is permitted, there is no need for file names or indexes. Upload can be performed from the ALD itself, indicated by subunit number 0, or from an individual subunit. The file to be uploaded is indicated by its file type. The supported file types and their allowable sources (ALD or subunit) are defined in Section 12.8.6. “Download Start”.

Using the command UploadInfo, the primary can query the size of the file to be uploaded. The command returns the size of the file in octets. If the file does not exist, an error is returned.

The command UploadStart selects the file to be uploaded and starts the upload process.

The file is uploaded by the ALD issuing a series of UploadFile commands. With each command a block of 256 octets in size is sent from the ALD to the primary until the complete file has been transferred. The last command in a sequence may transfer a block of less than 256 octets.

The ALD will send the UploadEnd command to the primary to indicate the successful completion of the Upload process.

Parallel execution of another upload process or any other Layer 7 command on the same port is not allowed (even from different subunits). However, the ALD may send AlarmIndication commands during an upload.

8.8. Resumption of operation

The following data shall be retained after reset:

- Firmware
- MALD Config Security settings
- MALD SW Download Security settings
- MALD Configuration settings
- RF Path IDs
- RF Path ID Aliases

Each subunit type standard may have additional requirements for retaining information.

9. AISG PSEUDOCODE

9.1. Global AISG code definitions

The following definitions are required for the pseudocode environment.

9.1.1. Port information

The following variables are required to provide information about the ALD ports.

```
uint16_t CurrentPort // The port number the command was received on
uint16_t DownloadPort
```

9.1.2. ALD information

ALDType_t is set by design to the type of the ALD.

```
Enumeration ALDType_t {
      SALD ← 0
      MALD ← 1
}
```

9.1.3. Subunit information

NrOfSubunits is initialised during start-up to the number of subunits within the ALD.

```
uint16_t NrOfSubunits          // number of subunits within the ALD
```

9.1.4. Diagnostic information

RAISE is a function that, for an alarmcode, sets the alarm severity and stores a descriptive string for an alarmcode.

CLEAR is a function that, for an alarmcode, sets the alarm severity to Cleared and sets the descriptive string to an empty string.

9.1.5. Ping process

```
Boolean PingReceivedFlag
```

9.1.6. Array definitions

Following the reset MaxArray is set by the ALD to the highest array number in the antenna.

```
uint16_t MaxArray
```

9.1.7. Filetype definitions

```
Enumeration FileType_t: uint8_t {
      FirmwareFile          ← 0
      ConfigurationFile     ← 1
      LogFile               ← 2
      InformationFile       ← 3
}
FileType_t ReceivedFileType
```

9.1.8. Primary IDs

PrimaryIDs contains the PrimaryID for each port.

```
uint32_t PrimaryIDs[1+MaxPort]  // PrimaryID of each port
```

10. LAYER 1

10.1. General

There are two layer 1 connectivity options:

- RS-485 option: A screened multicore cable, which supports a conventional RS-485 serial multi-drop bus.
- OOK option: A coaxial cable, which is shared with DC and RF signals.

Both layer 1 options support the transmission of two-way serial data and DC power to a connected ALD. At least one of these options shall be supported by any primary or ALD.

Data rate: 9.6 kbps ±3%.

An ALD shall not communicate through an AISG port that has AISG communication but no applied DC power. The ping process is allowed on AISG OOK ports whether or not DC power is applied on that port.

10.1.1. One / zero relationship

The relationship between an idle bus, 1, 0, the RS-485 differential voltages and OOK levels shall be according to Figure 10.1.1-1: “One / zero relationship”.

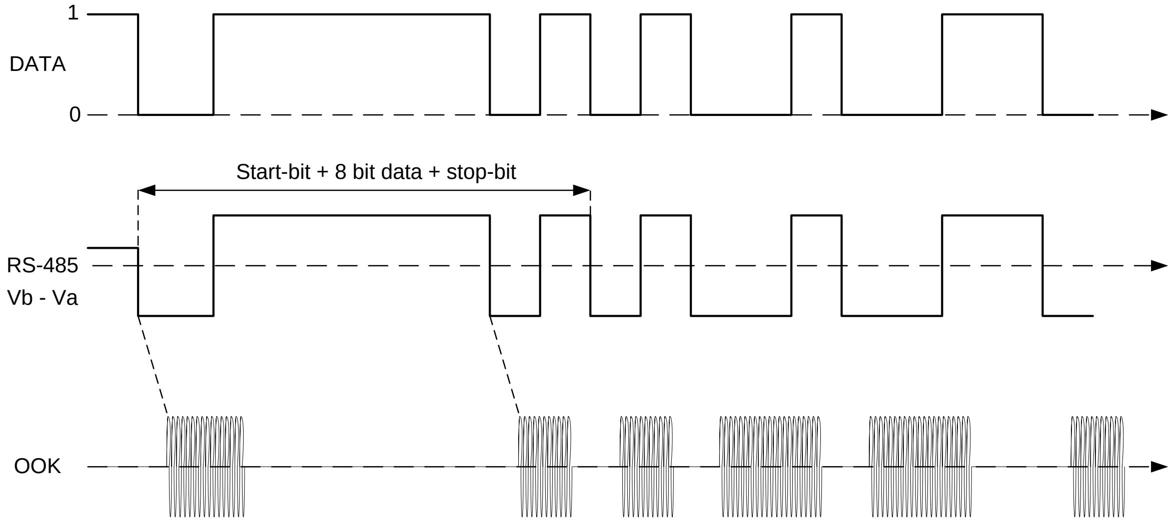


Figure 10.1.1-1: One / zero relationship

10.2. RS-485 option

The RS-485 bus used in AISG is a 2-wire half duplex bus supporting multi-drop. The bus shall be used together with separate wires for DC supply and DC return. The mapping of mark/space to logical one and zero as referred in [1] shall be according to Figure 10.2-1: “Format and order of transmitted data”.

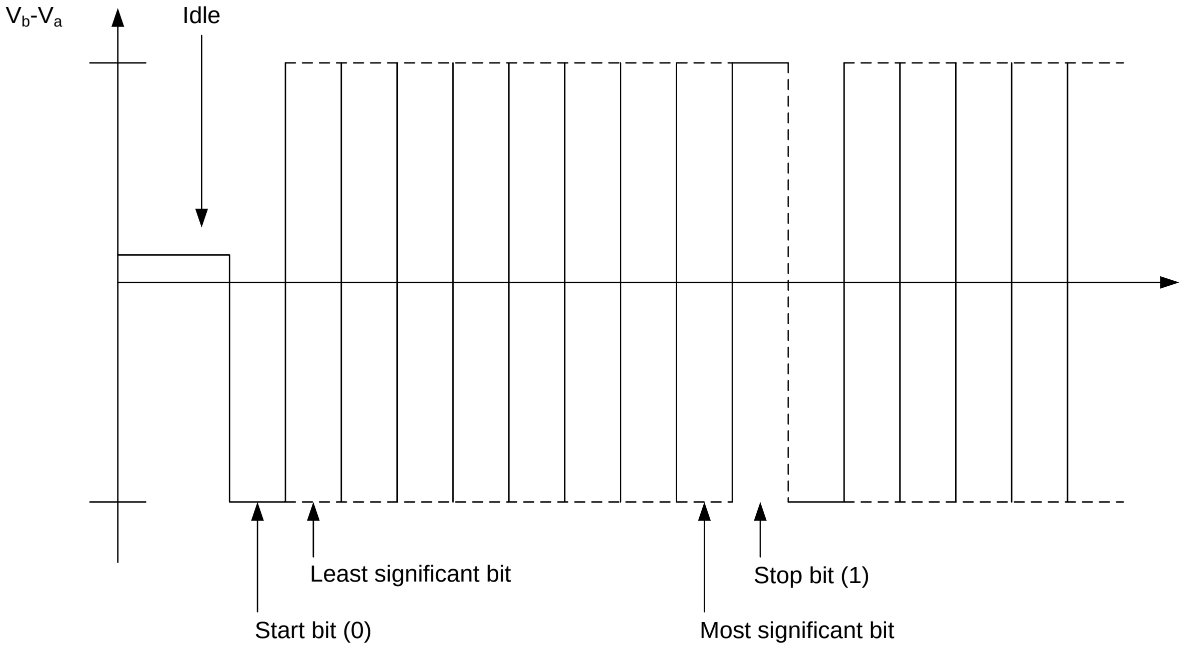


Figure 10.2-1: Framing and order of transmitted data

10.2.1. RS-485 bus load

An RS-485 bus interface shall present a bus load less than or equal to a unit bus load defined in [1]. One unit load is approximately 12 kohm.

10.2.2. RS-485 bus termination

It is not necessary to provide an external termination for the RS-485 bus.

10.2.3. RS-485 idle state biasing

Idle state bias is mandatory. Within one RS-485 bus only one ISB source is allowed.

Primaries and ANT RS-485 modems shall provide ISB. (see [2])

The ISB circuits shall provide approximately 120-ohm source impedance to the bus.

The polarity of the idle-state bias shall be Vb > Va.

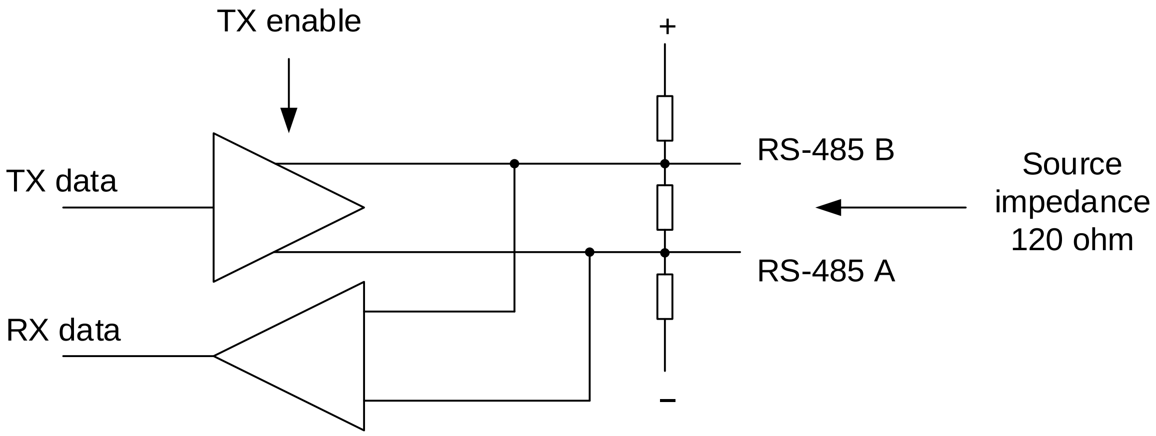


Figure 10.2.3-1: RS-485 transceiver

The ISB source shall be sufficient to guarantee at least 220 mV bias voltage when the bus is externally terminated with a 120-ohm load termination resistor.

10.2.4. Bus collisions

The RS-485 bus in AISG may be subject to bus data collisions. The ALD shall survive any type of bus collisions within the operating conditions allowed by [1] and [2].

10.2.5. Voltages

The AISG RS-485 bus shall accommodate different bus driver voltages provided that the line voltages are within the RS-485 standard common mode voltage range [1]. The bus shall operate correctly if 3.3 volt and 5-volt RS-485 circuits are mixed on the same bus.

10.2.6. RS-485 timing

The RS-485 transmitter shall be set to drive the bus not later than the leading edge of the first start bit, and held active until the last stop bit is sent. The RS-485 transmitter shall stop driving the bus not later than 2 ms after the last stop bit is sent.

10.3. OOK Option

The OOK option is a signal connection via modems via a coaxial cable which is shared with DC supply and RF signals.

10.3.1. Modem configurations

The connection between a base station and an ALD is provided by 2 modems, a BS RS-485 modem or a BS modem on the primary side and an ANT RS-485 modem or an ALD modem on the secondary side. A modem is not an ALD.

A BS RS-485 modem shall be connected to the antenna connector of the BS. The BS modem is integrated in the BS.

A ANT RS-485 modem is located between the antenna feeder cable and an ALD or an antenna, which in this context should be understood to contain an ALD. An ANT RS-485 modem can be integrated into an ALD or the antenna.

A ALD Modem can be integrated into an ALD. An ALD modem does not provide an RS-485 connection.

A modem may contain a PING port for mapping of RF ports in the antenna line system. A PING port is a special version of the RF port of an ALD or a BS modem. It is only capable of delivering RF and OOK signals. It is not capable of sourcing or consuming DC power. A PING port is only allowed to transmit or receive the Ping messages. All specifications for an ALD or BS modem also apply to a modem with a PING port.

Figure 10.3.1-1: "Reference planes for typical configurations at the antenna end" and Figure 10.3.1-2: "Reference planes for typical configurations at the BTS end" show different modem configurations in the antenna line. The reference planes, DC and RS-485 reference points are defined as reference points for the modem specifications. The antenna feeder cable shall transmit DC power, RF and OOK signals between the modems. In the case of an antenna with an integrated modem, the reference plane 4 is between the modem (with a Bias-T) and the antenna.

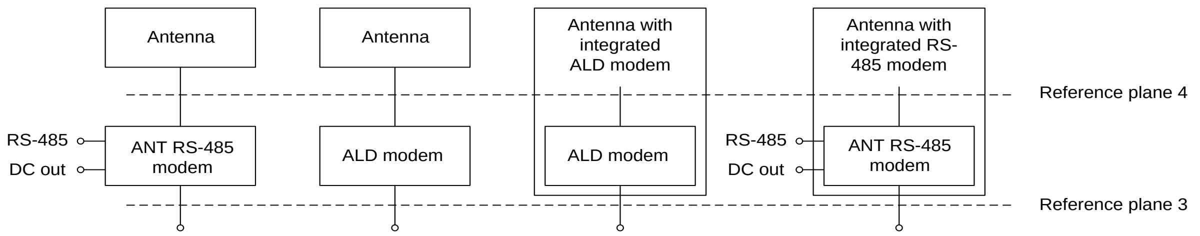


Figure 10.3.1-1: Reference planes for typical configurations at the antenna end

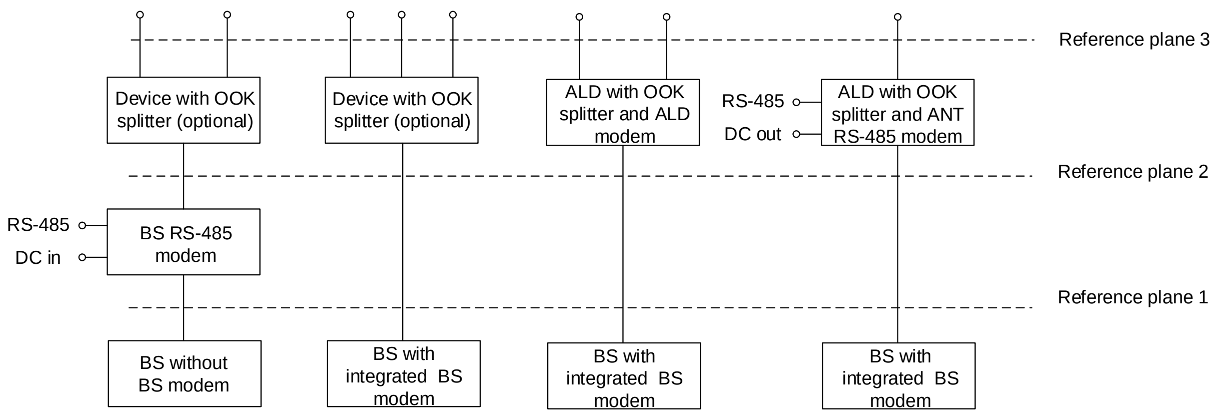


Figure 10.3.1-2: Reference planes for typical configuration options at the BS end

10.3.2. Modem operating frequency band

The modem is designed to operate in one or several uplink and downlink operating bands. The operating frequency bands of the BS RS-485 modem, ANT RS-485 modem or ALD modem shall be declared by the manufacturer in the datasheet. In the case of an ALD modem, the ALD shall report the operating frequency band according to binary frequency coding. The operating frequency band(s) shall be reported for all RF ports of an ALD. The reporting is detailed in Chapter 11 “Binary based frequency coding”.

10.3.3. Modem attenuation

The BS RS-485 modem shall provide less than attenuation between reference plane 2 and referece plane 1 not less than that shown in Figure 10.3.3-1: “Modem attenuation” to protect the BS from emissions of the antenna or modem.

The BS RS-485 modem emissions at reference plane 1 shall be attenuated at least as shown in Figure 10.3.3-1: “Modem attenuation” with respect to the levels specified for the modem spectrum emission mask in Figure 10.3.11.3-2: “BS RS-485 Modem spectrum emission mask at reference plane 1” to protect the BS from emissions of the BS modem.

The ANT RS-485 modem or ALD modem shall provide attenuation between reference plane 3 and reference plane 4 1 not less that shown in Figure 10.3.3-1: “Modem attenuation” to protect other radio systems from emissions of the modem.

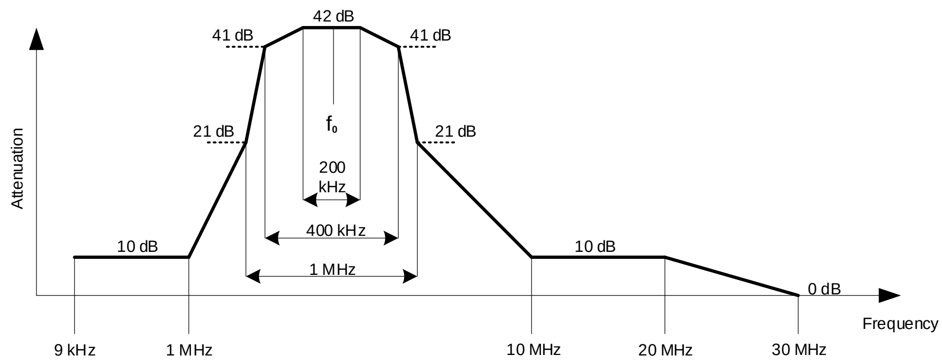


Figure 10.3.3-1: Modem attenuation

10.3.4. DC port isolation

For an ANT RS-485 modem the minimum isolation between the DC-out reference point and reference planes 3 and 4 shall comply with the values shown in Figures 10.3.4-1 “Isolation between DC in and RF port” and 10.3.4-2 “Detailed isolation requirement around the OOK carrier frequency”.

For a BS RS-485 modem without an integrated power supply, the isolation between the DC-in reference point and reference planes 1 and 2 shall comply with the values shown in Figures 10.3.4-1 “Isolation between DC in and RF port” and 10.3.4-2 “Detailed isolation requirement around the OOK carrier frequency”.

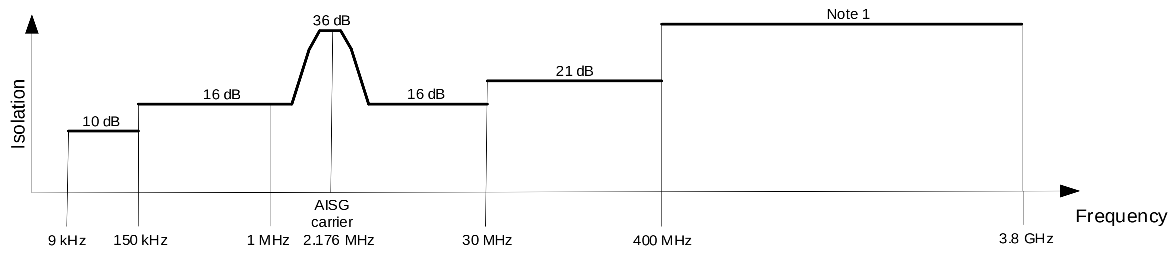


Figure 10.3.4-1: Minimum isolation between DC-in and RF port

NOTE: 38 dB, except for uplink and downlink operating bands where it is 65 dB.

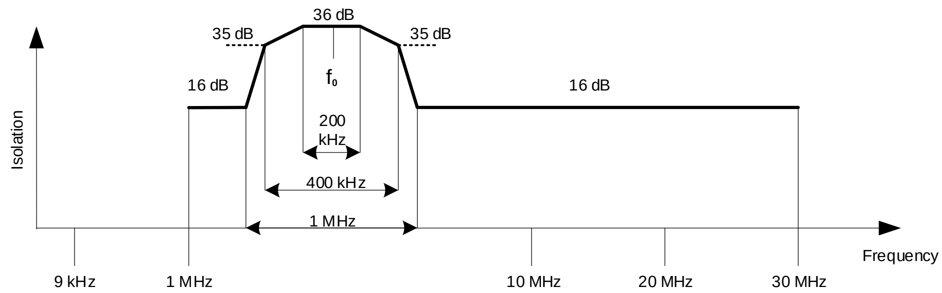


Figure 10.3.4-2: Detailed minimum isolation requirement around the OOK carrier frequency

10.3.5. Modem intermodulation attenuation

The modem intermodulation attenuation is specified in terms of the power in intermodulation products of WCDMA modulated carriers present at reference plane 1 or reference plane 3.

For two downlink carriers of 43 dBm the power of third order intermodulation products in the defined operating uplink frequency band for the BS RS-485 modem, ANT RS-485 modem and ALD modem shall not exceed:

–130 dBm/100 kHz for frequencies <1 GHz

–120 dBm/1 MHz for frequencies ⩾1 GHz

For the worst input configuration of power and number of carriers declared by the modem manufacturer the power of any intermodulation product for BS RS-485 modem, ANT RS-485 modem and ALD modem shall not exceed:

–98 dBm/100 kHz

In addition, for the worst input configuration of power and number of carriers declared by the modem manufacturer the power of fifth or higher order intermodulation products in the defined operating frequency band for the BS RS-485 modem, ANT RS-485 modem and ALD modem shall not exceed:

–135 dBm/100 kHz for frequencies <1 GHz

–125 dBm/1 MHz for frequencies ⩾1 GHz

The conversion between modulated and CW signals shall be as follows:

- The requirement for IM3 below 1 GHz shall be relaxed by 15 dB and tested with CW interferers at the specified levels.
- The requirement for IM3 above 1 GHz shall be relaxed by 5 dB and tested with CW interferers at the specified levels.
- The requirement for IM5 or higher below 1 GHz shall be relaxed by 10 dB and tested with CW interferers at the specified levels.
- The requirement for IM5 or higher above 1 GHz shall be relaxed by 0 dB and tested with CW interferers at the specified levels.

10.3.5.1. Emission requirement below noise floor

As a general rule, the resolution bandwidth of the measuring equipment should be equal to the measurement bandwidth. However, to improve measurement accuracy and sensitivity when measuring close to or below the noise floor, the resolution bandwidth may be smaller than the measurement bandwidth. When the resolution bandwidth is smaller than the measurement bandwidth, the result should be integrated over the measurement bandwidth in order to obtain the equivalent noise bandwidth of the measurement bandwidth.

10.3.5.2. Conversion between modulated and CW for IM measurement

The requirement for IM3 below 1 GHz shall be relaxed by 15 dB and tested with CW interferers at the specified levels.

The requirement for IM3 above 1 GHz shall be relaxed by 5 dB and tested with CW interferers at the specified levels.

The requirement for IM5 or higher below 1 GHz shall be relaxed by 10 dB and tested with CW interferers at the specified levels.

The requirement for IM5 or higher above 1 GHz shall be relaxed by 0 dB and tested with CW interferers at the specified levels.

Example: A -130 dBm/100 kHz requirement below 1 GHz with two WCDMA-modulated carriers at 43 dBm is converted to a -115 dBm requirement with two CW carriers at 43 dBm.

10.3.6. Modem impedance

The RF ports of the BS RS-485, ANT RS-485 and ALD modems that support OOK signals shall provide constant impedance:

- Nominal impedance Zo: 50 ohms;
- Return loss at modem carrier frequency ±0.1 MHz >10 dB;
- Return loss of RF ports in the defined operating frequency bands >20 dB;
- DC impedance of PING port: > 1 kohm.

10.3.7. Modem insertion loss in RF bands

The maximum insertion loss of the BS RS-485 and ANT RS-485 modem in the RF operating frequency bands shall be ≤0.3dB.

The actual insertion loss of BS RS-485 and ANT RS-485 modem shall be declared in the manufacturer's datasheet.

10.3.8. Modem power consumption

All modems shall be able to operate with a supply voltage range of 10 V – 30 V. The maximum power consumption of BS RS-485 and ANT RS-485 modems shall each be less than 2 W. A BS RS-485 modem shall cause a voltage drop less than 2 V between reference points DC-in and 2. An ANT RS-485 modem shall cause a voltage drop less than 2 V between reference points 2 and DC-out and less than 2 V between reference points 3 and DC-out. These voltage drops shall be measured at the declared maximum operating current.

A modem shall fulfil the DC power-up characteristics specified in Section 10.4.3 "DC power-up and steady state mode".

If a modem is integrated in an ALD, the maximum voltage drop between RF port and RS-485 port caused by the external load is undefined. The maximum current supported by the RS-485 interface and the voltage drop at that current, including the internal highest current consumption, shall be declared by the manufacturer in the datasheet.

10.3.9. Modem RF time delay and accuracy

The BS RS-485 and ANT RS-485 modem RF time delays and their accuracy in the operating bands, shall be declared in the manufacturer's datasheet.

10.3.10. Modem timing

Modem timing shall comply with the requirements of Para 10.2.6.

Modem data delay shall be less than or equal to 0.2 ms in each direction.

10.3.11. Modulator characteristics

10.3.11.1. Carrier frequency and accuracy

The following carrier frequency shall be used:

2.176 MHz ±100 ppm

10.3.11.2. Levels

ON-Level: +3 dBm ±2 dB

OFF-Level: ≤-40 dBm

The modulator signal levels are referred to the RF port of the modem or ALD.

10.3.11.3. Spectrum emission mask

The modem spectrum emission mask is specified in Figure 10.3.11.3-1: "Modem spectrum emission mask". Intermediate values may be obtained by linear interpolation between the points shown. The corresponding measurement bandwidths are specified in Table 10.3.11.3-1: "Modem spectrum emission mask".

For modem configurations according to Figure 10.3.1-2: "Reference planes for typical configuration options at the BS end" the BS RS-485 modem emissions shall not exceed the limits of the spectrum emission mask at reference plane 2. For modem configurations according to Figure 10.3.1-2: "Reference planes for typical configuration options at the BS end" the modem emissions from a BS with integrated BS modem shall not exceed the limits of the spectrum emission mask at reference plane 1 for frequencies below 20 MHz. ANT RS-485 or ALD modem emissions shall not exceed the limits of the spectrum emission mask at reference planes 2 and 3 according to Figure 10.3.1-1: ": Reference planes for typical configurations at the antenna end" and Figure 10.3.1-2: "Reference planes for typical configuration options at the BS end".

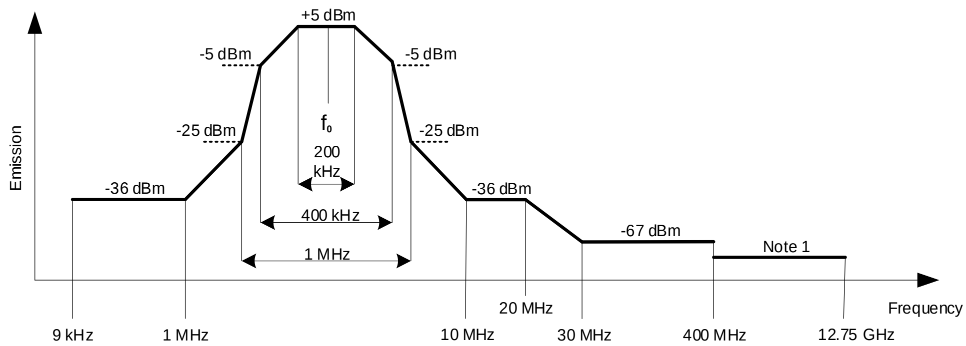


Figure 10.3.11.3-1: Modem spectrum emission mask

NOTE: For frequencies below 1 GHz the general emission limit is -108 dBm, except at modem operating band uplink frequencies where the emission limit is -135 dBm. For frequencies above 1 GHz the general emission limit is -98 dBm, except at modem operating band uplink frequencies where the emission limit is -125 dBm.

Band	Measurement Bandwidth
9 kHz - 150 kHz	1 kHz
150 kHz - 30 MHz	10 kHz
30 MHz - 1 GHz	100 kHz
1 GHz - 12.75 GHz	1 MHz

Table 10.3.11.3-1: Modem spectrum emission mask measurement bandwidth

For modem configurations according to Figure 10.3.1-2: "Reference planes for typical configuration options at the BS end" the BS RS-485 modem emissions shall not exceed the limits of the spectrum emission mask at reference plane 1 according to Figure 10.3.11.3-2: "BS RS-485 Modem spectrum emission mask at reference plane 1".

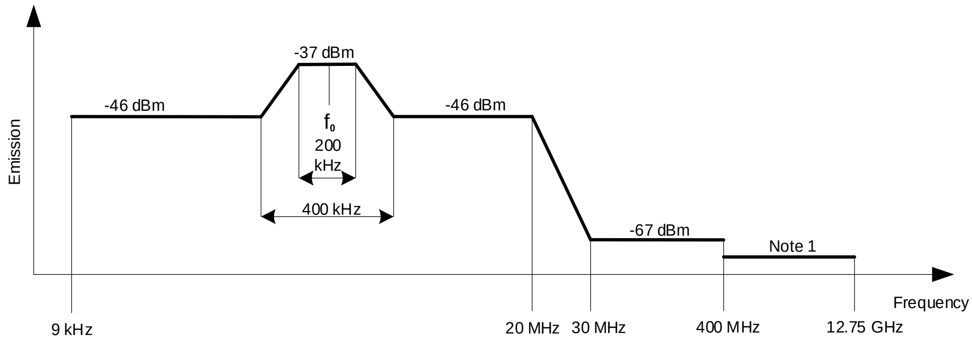


Figure 10.3.11.3-2: BS RS-485 Modem spectrum emission mask at reference plane 1

NOTE: For frequencies below 1 GHz the general emission limit is -108 dBm, except modem operating band uplink frequencies where the emission limit is -135 dBm. For frequencies above 1 GHz the general emission limit is -98 dBm, except at modem operating band uplink frequencies where the emission limit is -125 dBm.

10.3.11.4. Spectrum mask and emission testing

The spectrum mask and emission requirement shall be tested both with a consecutive series of “0” and an alternating sequence of “0” and “1”.

10.3.12. Demodulator characteristics

The demodulator shall fulfil the following requirements for selectivity and duty cycle variation.

10.3.12.1. Demodulator selectivity

The following signals at the RF port of ALD shall not result in detection of the ON-state:

Centre frequency of interfering CW signal	Interfering CW signal level	OOK signal level at 2.176 MHz
9 kHz – 1.25 MHz	-13 dBm	< -18 dBm
3.7 MHz – 12.75 GHz	-13 dBm	< -18 dBm
The defined TX carrier frequency band of ALD RF port	The defined maximum acceptable TX carrier signal level of ALD RF port	< -18 dBm

Table 10.3.12.1-1: The definitions of signal levels for ON-state

The following signals at the RF port of ALD shall not result in detection of the OFF-state:

Centre frequency of interfering CW signal	Interfering CW signal level	OOK signal level at 2.176 MHz
9 kHz – 1.25 MHz	-13 dBm	-12 dBm – +5 dBm
3.7 MHz – 12.75 GHz	-13 dBm	-12 dBm – +5 dBm
The defined TX carrier frequency band of ALD RF port	The defined maximum acceptable TX carrier signal level of ALD RF port	-12 dBm – +5 dBm

Table 10.3.12.1-2: The definitions of signal levels for OFF-state

10.3.12.2. Duty cycle variation

For transmission through a coaxial cable, two modems are required, one converting from a bit stream to OOK and one from OOK back to a bit stream. In order to guarantee proper transmission of data bits through the processes of modulation and demodulation of BS RS-485 and ANT RS-485 modems, the following system duty cycle limits shall be met for a carrier ON-Level between +5 dBm and -12 dBm and a carrier OFF-Level less than -18 dBm. Levels between -12 dBm and -18 dBm are undefined.

$\Delta DC_{SYSTEM} = |DC_{RX} – DC_{TX}| \leq 10\ \%$

Where: ΔDC_{SYSTEM} is the difference between the duty cycles of the transmitted and received bit streams,

DC_{TX} = Duty cycle for the input bit stream, and

DC_{RX} = Duty cycle for the output bit stream.

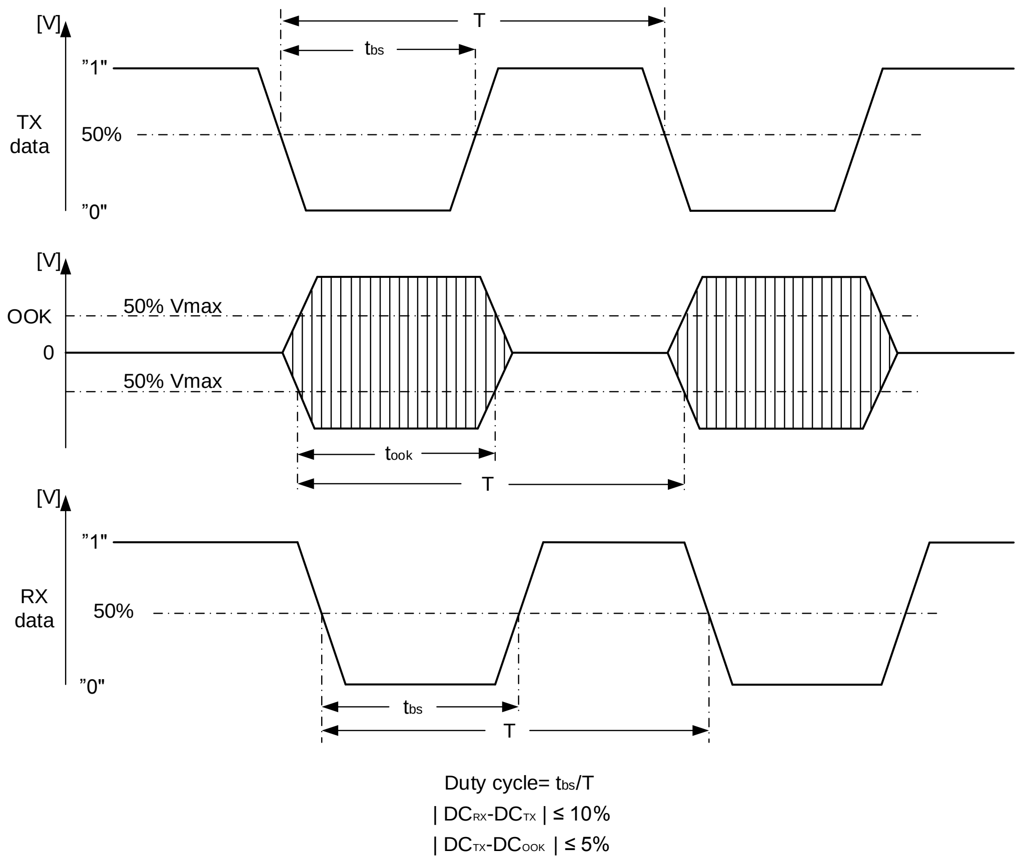


Figure 10.3.12.2-1: Duty cycles of the bit stream and OOK modulated subcarrier

For an input bit stream with a duty ratio of 50%, the cascaded modulator and demodulator shall provide an output bit stream with a duty ratio within the limits 40% – 60%, measured in each case at 0.5 times peak amplitude (see Figure 10.3.12.2-1. "Duty cycles of the bit stream and OOK modulated subcarrier").

The permitted duty cycle limit for a single BS RS-485 or ANT RS-485 modem is 45% – 55%. The duty cycle of a single modem can be measured by testing it both as modulator and demodulator, paired with another known modem. The duty cycle of a single modem can also be determined by measuring the time between the points at 50% of maximum voltage of the OOK signal (see Figure 10.3.12.2-1. "Duty cycles of the bit stream and OOK modulated subcarrier").

10.3.13. OOK combiners and splitters

It is permissible to combine and split the DC+RF+OOK signal.

All external inputs and outputs on splitting devices must be matched to 50 ohms, both for the OOK band and the specified RF band must meet the following requirements:

1. Return loss: ⩾14 dB at 2.176 MHz ±100 kHz
2. Maximum insertion loss at 2.176 MHz ±100 kHz
   - Two-way split: 4.5 dB
   - Three-way split: 6.3 dB

The values for the maximum voltage drop and the maximum operating current shall be declared by the manufacturer in the datasheet.

If an OOK-combiner or an OOK-splitter is integrated into an ALD, the OOK signals generated, used and by-passed by such a device are permitted to have the same level of insertion loss as is specified for an external splitting device as above.

10.3.14. Active regeneration of the OOK signal at ALD

It is allowed to regenerate the OOK signal at an ALD to implement a bi-directional OOK repeater. The data stream between different RF port ALD modems can be implemented by any means. The maximum OOK signal delay between RF ports of the ALD is 0.2 ms.

10.3.15. OOK bypass in ALD

An OOK bypass creates a path for the OOK signal between specific RF ports of the ALD. In the case of OOK combining or splitting, the ALD modem may be common to several OOK bypass paths.

If pinging is supported, OOK bypass paths shall provide an attenuation of at least 23 dB at the OOK frequency during the PingerRestrictedTransmitState or the ListenerRestrictedMonitorState.

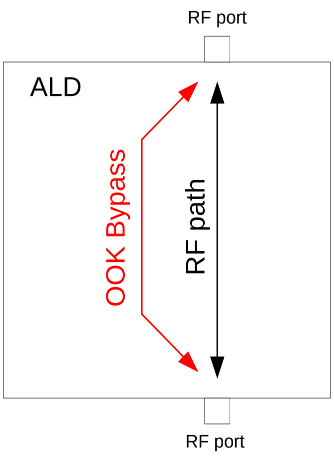


Figure 10.3.15-1: OOK bypass

10.3.16. Conducted emissions

The levels of generated conducted noise and ripple on DC power supply shall be within the following limits:

Item	Limit	Frequency	Remarks
ANT RS-485 modem, RF port	15 mVp-p	0.15-30 MHz	Generated noise and ripple at RF feeder (in RX mode)
ANT RS-485 modem, DC port	20 mVp-p	0.15-30 MHz	Allowed noise and ripple at external DC port (in TX mode)

Table 10.3.16-1: Noise and ripple

The noise and ripple measurement setup is defined in Section 10.5.1. “Noise and ripple”.

10.3.17. Spurious emissions at modem input

Spurious emissions at the DC input of a BS RS-485 modem shall not generate spurious emissions, at reference plane 1, above a level that will violate the spectrum emission mask requirement according to Section 10.3.11.3. “Spectrum emission mask”. The DC port isolation according to Section 10.3.4. “DC port isolation” shall be taken into account.

10.4. ALD DC power supply

10.4.1. DC supply level

An ALD shall support a DC supply operational voltage of 10.0 – 30.0 V DC.

NOTE: An ALD bypassing a layer 2 link via an OOK or RS-485 port in the ANT direction shall supply DC power on that port.

10.4.2. Definition of power modes

ALDs may have up to three different power modes: SteadyStatePowerMode, HighPowerMode and SleepPowerMode.

SteadyStatePowerMode is the normal operating condition of the ALD.

HighPowerMode is a specific condition requested by the primary for an ALD having higher DC power consumption compared to SteadyStateMode.

SleepPowerMode is a specific condition saving DC power.

On receipt of a GetAISGPortDCPowerInformationCommand, the ALD shall report the DC power consumption for these modes as integers with a resolution of 0.1 watt. All stored values shall be worst case values over all specified operating conditions.

In the case of MALD, the stored values are for the condition when DC power is supplied by only one AISG port (OOK or RS-485).

P1 is the maximum SteadyStatePowerMode consumption for a specific ALD. The stored value concerns this ALD only.

P2 is the HighPowerMode power consumption for a specific ALD. An ALD shall only switch into the HighPowerMode in response to a layer 2 or layer 7 command explicitly permitting the ALD to do so. For ALDs having no HighPowerMode, P2 is equal to P1.

P3 is the SleepPowerMode consumption for a specific ALD. An ALD shall only switch into the sleep power mode in response to a layer 2 or layer 7 command explicitly permitting the ALD to do so. For ALDs having no SleepPowerMode, P3 is equal to P1.

Subunit type standards may optionally declare an upper limit for one or more of these values.

10.4.3. DC power-up and steady state mode

DC power-up requirements shall be fulfilled at start-up, after a DC power cycle defined in Section 10.4.4. “Reset triggered by DC power cycle”, and at start-up after reset.

DC power-up requirements shall be fulfilled at any ALD voltage within the AISG specified operating voltage range.

DC power-up requirements shall be fulfilled for all AISG OOK and RS-485 DC input interfaces.

DC power-up requirements are verified using an ALD input voltage having a maximum rise time of 100 microseconds to 90% of the final voltage.

SteadyStatePowerMode starts 10 seconds after power is supplied to an ALD, at which time the ALD shall be ready to receive layer 2 commands. An ALD remains in SteadyStatePowerMode unless commanded to an alternative power mode. Once an alternative power mode is completed, the ALD shall return to SteadyStatePowerMode.

10.4.3.1. Allowed initial energy consumption at power-up

The initial consumed energy per AISG DC input port shall be less than or equal to 1 mJ during the first 0.2 milliseconds.

10.4.3.2. Allowed initial current consumption at power-up

The peak current consumption from 0.2 milliseconds to 50 milliseconds shall be less than or equal to the declared SteadyStatePowerMode consumption P1 divided by 30 volts.

The peak current consumption from 50 milliseconds to 10 seconds shall be less than or equal to the declared SteadyStatePowerMode consumption P1 divided by the ALD voltage.

10.4.3.3. Minimum DC input impedance at low voltages

AISG DC input ports shall provide a DC impedance ⩾ 1000 ohms for voltages < 3.5 volts.

10.4.4. Reset triggered by DC power cycle

If the AISG port voltage is below 3.5V for more than 3 seconds, a SALD shall execute reset when the port voltage reaches the ALD DC supply operational voltage range.

If the AISG port voltage is below 3.5V for more than 3 seconds for all its AISG ports simultaneously, a MALD shall execute reset when the port voltage of at least one of its AISG ports reaches the ALD DC supply operational voltage range.

An ALD shall be ready to receive layer 2 commands within 10 seconds after reset is initiated.

10.4.5. MALD DC power supply management

A MALD operates in redundant power supply configuration when connected to multiple primaries. SteadyStatePowerMode DC power of a MALD may be provided by any combination of AISG interface input ports. HighPowerMode DC power shall be taken from the AISG interface port that requested it. A short circuit on a single AISG OOK or RS-485 DC port shall not result in an interruption of the operation of a MALD related to the other primaries. Hot swapping (power off and on) of the primary shall be possible without an interruption of the operation of a MALD related to the other primaries.

A MALD having integrated ANT-RS-485 modem, the whole DC power delivery to an AISG RS-485 DC output port shall be taken from the OOK port having an internal connection to RS-485 bus.

If the AISG port voltage falls below 3.5V for at least 3 seconds on an AISG port, means shall be provided by which AISG layer 2 and layer 7 are informed.

10.4.6. Multi-pole connector

Connector type: Conforming to AISG C485.

Pin number	Signal	Requirement	Description
1	Not used		NOTE 1
2	Not used		NOTE 1
3	RS-485 B	Mandatory	Line voltage Vb
4	Not used		NOTE 2
5	RS-485 A	Mandatory	Line voltage Va
6	10 V–30 V DC	Mandatory	
7	DC return	Mandatory	NOTE 3
8	Not used		

Table 10.4.6-1: RS-485 interface multi-pole connector pin-out

NOTE 1: This pin has been used as a DC supply pin in earlier AISG versions.

NOTE 2: This pin has been an optional RS-485 ground pin in earlier AISG versions.

NOTE 3: DC return is preferably not grounded for any device deriving its DC power through this connector. If the DC return is grounded there is a risk of unwanted ground currents and also of higher lightning current inside the RS-485 cables. If this pin is grounded the DC feeding circuit must be mounted close to the ALD and both must have the same ground potential.

10.4.6.1. Polarity of multi-pole connectors

The polarity of the multi-pole connector pins shall follow the principle that live male connector pins are not exposed at any point, for example:

Primary:	Where the RS-485 interface is provided: Output socket(s) with female pins;
ALD or antenna:	When ALD or antenna contains an ANT RS-485 modem; Output socket(s) with female pins;
	When ALD or antenna control is to be independent of the RF cable: One input socket with male pins and optionally a second output socket with female pins;
Interconnecting cables:	Plug with male pins at one end and plug with female pins at the other end.

10.4.6.2. Daisy chaining with multi-pole connectors

At least mandatory pins shall be connected through for daisy chaining. Building a passive RS-485 splitter into an ALD is not recommended.

10.5. Emission and immunity requirements for ALDs

10.5.1. Noise and ripple

The levels of generated conducted noise and ripple on the ports of the ALD (unless other limits are defined in ALD specific standards) shall be within the following limits:

Item	Limit	Frequency	Remarks
ALD, RF port	15 mVpp	0.15-30 MHz	Generated noise and ripple at RF feeder (without OOK transmission)
ALD, DC port	20 mVpp	0.15-30 MHz	

Table 10.5.1-1: Noise and ripple

The noise and ripple measurement setup is defined in Section 10.5.1. “Noise and ripple”. The emissions and immunity for different BTS ancillary equipment ports and enclosure are defined in [3].

10.5.2. Conducted noise and ripple measurement

In order to achieve accurate, reproducible and comparable noise and ripple measurement results the following measurement guidelines shall be followed. Comparable test results are accomplished using an interface with a characteristic impedance of 50 ohms at the measurement port.

The conducted noise and ripple shall be measured with a 50-microhenry impedance stabilizing network (ISN). This device incorporates a 50-ohm impedance interface and filters the noise from the DC power supply. The conducted noise can be measured from the RF port or the RS-485 DC port of the ALD.

The measurement instrument needs to have a 50-ohm interface. The measurement instrument can be a measuring receiver, spectrum analyser or oscilloscope. The peak detector shall be used with a measuring receiver. When an oscilloscope is used, a 50-ohm shunt resistor shall be placed next to the probe and an additional low pass filter is needed to limit the measurement frequency to 30 MHz.

Test setup for the ALD port consuming DC current shall be configured as shown in Figure 10.5.2-1: “The test setup for conducted noise and ripple measurement”.

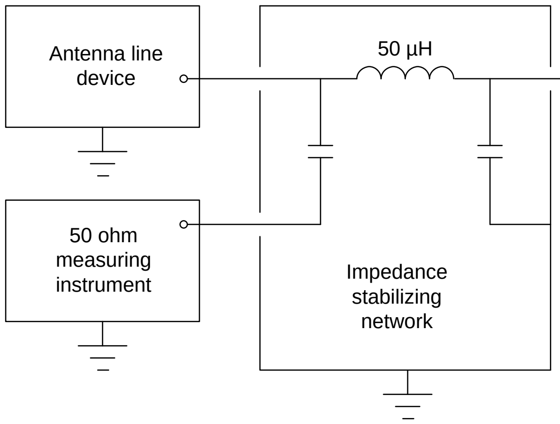


Figure 10.5.2-1: The test setup for conducted noise and ripple measurement

In the case where an ALD has RF port, the ALD power mode may be controlled by connecting the modem to the interface used by the 50-ohm measuring instrument, and disconnecting it before the noise measurement is made.

The test setup for an ALD with a DC current feed shall be as shown in Figure 10.5.2-2: “The test setu for an ALD with DC current feed”.

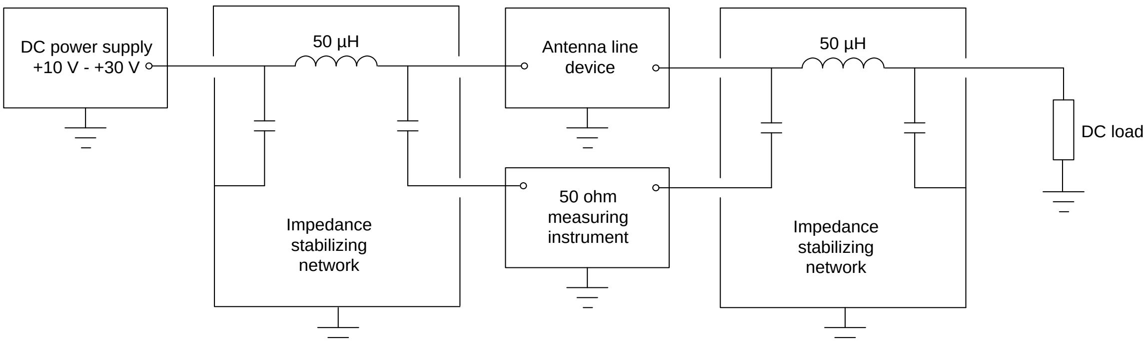


Figure 10.5.2-2: The test setup for an ALD with DC current feed

More specific information can be found in [4] and [5].

10.6. Primary DC supply

It is not allowed to establish a layer 2 link on any port without supplying DC-Power on that port. This requirement includes both cases OOK option and RS-485 option.

10.6.1. Primary DC supply for MALD

All primaries connected to MALD must be able to provide DC power simultaneously. The primary performing the MALD configuration shall be able to provide all idle mode DC power.

A primary issuing a command for a HighPowerMode command must be able to provide the additional power required by the HighPowerMode command.

A primary, which has a connection to a MALD port delivering power to an ANT RS-485 modem, must be able to provide all the DC power required by the devices connected to the connected RS-485 bus.

11. LAYER 2

11.1. General

Layer 2 is based on HDLC Class UNC1,15.1 TWA, according to Section 6.10. in [6].

This comprises the following subset of HDLC:

- Unbalanced operation (master / slave operation)
- Normal response mode (sequence numbers in data frames)
- XID negotiation
- Start and stop transmission with basic transparency
- Two Way Alternate (TWA) (half-duplex)

NOTE: Two different data stations are defined in [6], which are called primary station and secondary station. In this standard primary stations are called primary and secondary stations are called ALD.

11.2. Frame receiver

The ALD frame receiver requires a set of states per port. The frame receiver is defined to be called every time a port receives a character or other serial port event.

The term framing error is used to indicate that the stop bit had the wrong value.

```
struct AISGPortRxFrameStatus_t {
     uint8_t address             // Assigned ALDAddress,
                                 // 0 means NoAddress Linkstate
     Boolean in_frame            // Inside a frame (between HDLC flags)
     Boolean control_escape      // As defined in [6]
     uint16_t count              // Number of received octets
     uint16_t last_rx_time       // Time in ms when last octet was received
     uint16_t fcs                // Frame check sum
     uint8_t buffer[266]         // Received octets
}

// Per port state variable for frame receiver
AISGPortRxFrameStatus_t status
```

ALD Specification (Normative):

```
ON Reset DO
      status.in_frame ← false
DONE

ON ReceivedEvent DO                  // Serial port receive event
uint8_t C
uint16_t Now

      IF framing error THEN          // If stop bit had wrong value
            status.in_frame ← false
            EXIT
      ENDIF
```

```
C ← received character
Now ← current ms timer

IF Now – status.last_rx_time > 10 THEN          // Frame timeout
        status.in_frame ← false
ENDIF

status.last_rx_time ← Now                       // From a free running millisecond timer
IF C = 0x7E THEN                                // HDLC flag
        IF status.in_frame = true THEN

                IF status.control_escape = true THEN
                        status.in_frame ← false                 // HDLC abort frame
                        status.control_escape ← false
                        EXIT
                ENDIF

                IF status.count > 3 AND status.fcs = 0xF0B8 THEN  // Closing flag
                        uint8_t address
                        status.in_frame ← false
                        // Only process frames addressed to us or the all-station address
                        address ← status.buffer[0]

                        IF address = 0 THEN
                                EXIT
                                ENDIF

                        IF address = status.address OR address = 0xFF THEN
                                Queue frame for processing
                        ENDIF
                        EXIT
                ENDIF

        ENDIF

        status.fcs ← 0xFFFF                     // Opening flag
        status.count ← 0
        status.in_frame ← true
        status.control_escape ← false
        EXIT
ENDIF

IF status.in_frame= false THEN                  // Avoids processing out of frame octets
        EXIT
ENDIF

IF C = 0x7D THEN                                // HDLC transparency control escape
        status.control_escape ← true
        EXIT
ENDIF
```

```
        IF status.control_escape = true THEN
                C ← C bitwise XOR 0x20
                status.control_escape ← false
        ENDIF

        IF status.count >= Size of status.buffer THEN
                status.in_frame ← false                 // Discard excessive frame
        ENDIF

        status.fcs ← pppfcs16(status.fcs, C, 1)   // Calculate new FCS according to [9]
        status.buffer[status.count] ← C               // Store character
        status.count ← status.count + 1
DONE
```

11.3. Frame transmitter

The ALD frame transmitter needs a set of states per port. The frame transmitter is defined to be called every time a port is ready to transmit one or more characters.

```
Enumeration AISGTxFrameState_t {
      DeQueue          ← 0        // Get next message to transmit
      SendOctet        ← 1        // Send next octet
      ControlEscape    ← 2        // Send transparency modified octet
      FCS1             ← 3        // Frame check sum octet 1
      FCS2             ← 4        // Frame check sum octet 2
      ClosingFlag      ← 5        // Send closing flag
}

struct AISGPortTxFrameStatus_t {
      AISGTxFrameState_t state    // Current state of frame transmitter
      uint16_t count              // Number of octets to transmit
      uint16_t pos                // Index of next octet to transmit
      uint16_t fcs                // Frame check sum
      uint8_t buffer[266]         // Octets to transmit
}

// Per port state variable for frame transmitter
AISGPortTxFrameStatus_t status
```

ALD Specification (Normative):

```
FUNCTION SendMessage(uint8_t message[length]) IS
        IF length > Size of status.buffer THEN
                EXIT
        ENDIF

        Queue message on transmit queue
        // Enabling port transmitter will trigger TransmitterReadyEvent if serial port is idle
        // Enable serial port tx
        // Enables RS-485 transmitter. Does nothing if transmitter is active
END

ON Reset DO
        status.state ← DeQueue
DONE
```

```
ON TransmitterReadyEvent DO
    uint8_t C          // Character to send
    CASE status.state IS
        WHEN DeQueue:

            IF transmit queue is empty THEN
                Disable serial port tx        // Disables RS-485 transmitter
                EXIT
            ENDIF

            Copy message to status.buffer
            Remove message from transmit queue
            status.count ← length of message
            status.pos ← 0
            status.fcs ← 0xFFFF
            Send 0x7E                         // Opening flag
            status.state ← SendOctet
            EXIT

        WHEN SendOctet:
            C ← status.buffer[status.pos]
            status.fcs ← pppfcs16(status.fcs, C, 1)   // Calculate new FCS

            IF C = 0x7E OR C = 0x7D THEN
                status.state ← ControlEscape
                Send 0x7D
                status.buffer[pos] ← C bitwise XOR 0x20
                EXIT
            ENDIF

            Send C
            status.pos ← status.pos + 1

            IF status.pos = status.count THEN
                status.state ← FCS1
            ENDIF

            EXIT

        WHEN ControlEscape:
            C ← status.buffer[status.pos]
            Send C
            status.pos ← status.pos + 1

            IF status.pos = status.count THEN
                status.state ← FCS1
            ELSE
                status.state ← SendOctet
            ENDIF
```

```
                EXIT
        WHEN FCS1:
                status.buffer[0] ← status.fcs MOD 256
                status.buffer[1] ← status.fcs DIV 256
                C ← status.buffer[0]
                IF C = 0x7E OR C = 0x7D THEN
                        Send 0x7D
                        status.buffer[0] ← C bitwise XOR 0x20
                        EXIT
                ENDIF
                Send C
                status.state ← FCS2
                EXIT
        WHEN FCS2:
                C ← status.buffer[1]
                IF C = 0x7E OR C = 0x7D THEN
                        Send 0x7D
                        status.buffer[1] ← C bitwise XOR 0x20
                        EXIT
                ENDIF
                Send C
                status.state ← ClosingFlag
                EXIT
        WHEN ClosingFlag:
                Send 0x7E
                status.state ← DeQueue
                EXIT
    ENDCASE
DONE
```

11.4. Invalid reception

Frames shall be discarded if a framing error or data overrun occurs.

11.5. Frame lengths

All ALDs shall support HDLC frame lengths between 4 and 268 octets.

NOTE: The maximum layer 2 frame length is 4 octets plus the maximum payload length (See Section 7.3. “Definition of layer 7 message format”).

NOTE: The maximum layer 7 message size is 264 octets (see Section 7.4. “Definition of layer 7 message format”).

NOTE: The opening, closing flag and transparency are excluded from the calculation of the frame length.

11.6. Default address

After reset, an ALD shall use the no-device address (0x00). While it has the no-device address, it shall only respond to XID messages.

11.7. Window size

All ALDs shall support a window size of 1.

11.8. Frame timing

An ALD shall, after reception of a frame with the P bit set, start transmitting a response between 3 ms and 10 ms from the end of the stop bit of the closing flag.

A primary shall, after reception of a response with the F bit set, start transmitting a frame no sooner than 3 ms from the end of the stop bit of the closing flag. If no such response received, the primary may start transmitting after a 15-ms timeout.

Intra frame gap is not allowed for either primary or ALDs.

The data rate is specified in Section 10.1. “General”.

11.9. Frame completion

A frame is completed after a transmitting station (either the primary or an ALD) has sent the closing flag.

Further transmission shall not be allowed for this frame after the closing flag is sent.

11.10. ALD types

Two ALD types are defined and identified by the assigned 1-octet unsigned integer in this layer.

If the ALD is a SALD it shall use the ALD type SALD, and if the ALD is a MALD it shall use the ALD type MALD.

ALD type	1-octet unsigned integer
SALD	64
MALD	65

Table 11.10-1: ALD types and codes

NOTE: The subunit types, which are defined in subunit type standards (such as RET and TMA), shall not be used on layer 2.

11.11. XID frames

XID frames shall use the standard format (see Sections 5.5.3.1. - 5.5.3.2.3.2. in [6]). All Group Length (GL) fields have a size of 1 octet.

Any parameter 11.11.1 (AISG parameters) in an XID command shall be supported by all ALDs. XID parameters that are not defined in Section 11.11.1. "AISG parameters" shall be ignored.

The order of PIs in an XID frame is defined in the layer 2 message definition.

NOTE: In AISG v2 PIs could be sent in any order, but this not permitted in AISG v3.0.

XID Command	Primary	SALD	MALD
Device Scan	Mandatory	Mandatory	Mandatory
AISG v2 Address Assignment	Mandatory	Mandatory	Mandatory
AISG v3 Address Assignment	Mandatory	Mandatory	Mandatory
Reset Port	Mandatory	Mandatory	Mandatory
ResetALD	Mandatory	Mandatory	Mandatory
Trigger Ping	Optional	Optional	Optional
Ping Message	Optional	Optional	Optional
Disable OOK Bypass	Optional	Optional	Optional

Table 11.11-1: XID command set

11.11.1. AISG parameters

Format Identifier (FI) shall be 0x81 and Group Identifier (GI) shall be 0xF0. Table 11.11.1-1: "HDLC parameters for ALDs" provides an overview of all Parameter Identifiers which are defined in this standard.

PI	PL	Description of PV	PV Type
1	0–19	Subset of UniqueID	UIDString_t
2	1	ALD Address	uint8_t
3	0–19	Bit mask (for subset of UniqueID), indicates a device scan in AISG v2.0 mode	bit mask
4	1	ALD type (see Table 11.10-1: “ALD types and codes”)	ALDType_t
5	1	Not used	-
6	2	Vendor code as given in [7]	AsciiString_t
7	0	Reset port	-
8	0-19	Bit mask (for subset of UniqueID), indicates a device scan in AISG v3 mode	bit mask
9	1	Reserved (by prior standard)	uint8_t
10	0-2	Port number	uint16_t
11	0-2	Bit mask (for Port number)	bit mask
12	1	Disable OOK bypass	uint8_t
19	1	Device Scan Version	uint8_t
22	3-225	List of base standard version tuples	struct AISGVersion_t[PL/3]
24	0	ResetALD	-
25	0	TriggerPing	-
26	4	PrimaryID	uint32_t
27	1-255	List of subunit types	SubunitType_t[PL]
28	1	Ping Frame	-

Table 11.11.1-1: HDLC parameter for ALDs

NOTE: For multi-octet integers in XID frames, the higher-order bits shall be sent in the first octet transmitted, according to Section 5.5.3.1.2. in [6].

For AsciiStrings, the left-most characters shall be transmitted first.

Bit masks which apply to AsciiStrings shall be sent in the same order as AsciiStrings.

Bit masks which apply to multi-octet integers shall be sent in the same order as multi-octet integers.

XID parameters are presented by the following struct in the following sections:

XID parameter with a parameter length is not equal to 0:

```
struct XidParameter_t (Identifier, Value){
     uint8_t PI      ← Identifier         // parameter identifier
     uint8_t PL      ← length(Value)      // parameter length
     uint8_t PV[PL]  ← Value              // parameter value
}

XID parameters with a parameter length is equal to 0:
struct XidParameter_t (Identifier){
     uint8_t PI      ← Identifier         // parameter identifier
     uint8_t PL      ← 0                  // parameter length
}
```

11.11.2. Device scan

Description (Informative):

The device scan messages may be utilised by the primary to identify all ALDs in the NoAddress LinkState on an interface. For this purpose the primary may use PI = 8 to scan only AISG v3 compliant ALDs and may use PI = 3 to scan AISG v2.0 ALDs.

AISG release	Device scan version
2.0	Not applicable
3.0.0	1

Table 11.11.2-1: AISG releases and device scan versions

Command Specification (Normative):

Frame format for an AISG v3.0 device scan command:

```
PrimaryFrame AisgV3DeviceScanCommand {
      uint8_t address ← 0xFF              // All-station address
      uint8_t Ctrl    ← 0xBF              // Control field for XID
      uint8_t FI      ← 0x81              // Format identifier
      uint8_t GI      ← 0xF0              // Group identifier
      uint8_t GL                          // Length of the following octets
      XidParameter(1, UniqueID)
      XidParameter(8, BitMaskUniqueID)
      XidParameter(10, PortNumber)
      XidParameter(11, BitmaskPort)
      XidParameter(19, DeviceScanVersion)
}
```

Frame format for an AISG v2 device scan command:

```
PrimaryFrame AisgV2DeviceScanCommand {
      uint8_t address ← 0xFF              // All-station address
      uint8_t Ctrl    ← 0xBF              // Control field for XID
      uint8_t FI      ← 0x81              // Format identifier
      uint8_t GI      ← 0xF0              // Group identifier
      uint8_t GL                          // Number of following octets
      XidParameter(1, UniqueID)
      XidParameter(3, Bitmask)
}
```

Frame format for an AISG v3.0 device scan response:

```
ALDFrame AisgV3DeviceScanResponse {
      uint8_t address      ← 0x00         // No-station address
      uint8_t Ctrl         ← 0xBF         // Control field for XID
      uint8_t FI           ← 0x81         // Format identifier
      uint8_t GI           ← 0xF0         // Group identifier
      uint8_t GL                          //   Number   of   following   octets
      XidParameter(1, UniqueID)
      XidParameter(4, ALDType)
      XidParameter(6, VendorCode)
      XidParameter(10, PortNumber)
```

```
     // List of supported AISG base standard versions
     // PL must be a multiple of 3 since the size of struct AISGVersion is 3
     XidParameter(22, AISGVersion_t[PL/3])
     XidParameter(27, ListOfSubunitTypes)
}
```

Frame format for an AISG v2 device scan response:

```
ALDFrame AisgV2DeviceScanResponse {
     uint8_t address       ← 0x00          // No-station address
     uint8_t Ctrl          ← 0xBF          // Control field for XID
     uint8_t FI            ← 0x81          // Format identifier
     uint8_t GI            ← 0xF0          // Group identifier
     uint8_t GL                            // Number of following octets
     XidParameter(1, UniqueID)
     XidParameter(4, ALDType)
     XidParameter(6, VendorCode)
}
```

Primary Specification (Normative):

An AISG primary shall perform an AISG v3.0 device scan with DeviceScanVersion ← 1. If the primary supports AISG v2 device scan it shall then perform an AISGv2 device scan.

This ensures that all AISG v3-compliant ALDs will act in AISG v3.0 mode and not in AISG v2 backward compatible mode.

During the device scan, ALD addresses shall be assigned to all detected ALDs so that they do not respond to further device scan messages. This applies to AISG v3.0 and AISG v2 device scans.

During an AISG v2 device scan, care must be taken to ensure that the communication timeout timer does not expire, causing the ALD to perform a reset.

The device scan will utilise both PI = 10 (Port number) and PI = 1 (UniqueID). The two PIs should be considered as a 21-octet pattern by the ALD.

NOTE: It might happen that a primary is connected to several AISG ports of an ALD. In this case the ALD responds to the same UniqueID pattern on all of these ports, which will end up in a collision on the AISG bus. Considering the two PIs as a 21-octet pattern solves this by ensuring a unique response from the ALD.

Primary Specification (Normative):

Perform an AISG v3.0 device scan with DeviceScanVersion ← 1

IF AISG v2 device scan is supported THEN
    Perform an AISG v2 device scan
ENDIF

EXIT

ALD Specification (Normative):

IF the LinkState ≠NoAddress THEN
    EXIT
ENDIF

```
IF the frame contains UniqueID (PI = 1)
        AND the frame contains BitmaskUniqueID (PI = 8)
        AND the frame contains PortNumber (PI = 10)
        AND the frame contains BitmaskPortNumber (PI = 11)
        AND the frame contains DeviceScanVersion (PI = 19) THEN
        AISGv3DeviceScan()
ELSEIF the frame contains UniqueID (PI = 1)
        AND the frame contains Bitmask (PI = 3) THEN
        AISGv2DeviceScan()
ENDIF

EXIT

AISGv3DeviceScan():
        uint8_t Version
        uint8_t MaskedValue[19]
        uint8_t Length
        uint8_t MaskLength
        uint16_t RxPortNumber
        uint8_t N
        Version ← DeviceScanVersion (PI = 19)

        IF Version ≠ 1 THEN
                EXIT
        ENDIF

        Length ← length (PL) of PortNumber (PI = 10)
        MaskLength ← length (PL) of BitmaskPortNumber (PI = 11)

        IF MaskLength ≠ Length OR Length > 2 THEN
                EXIT
        ENDIF

RxPortNumber ← port number on which the frame was received
        MaskedValue[0..Length − 1] ← length right-most octets of RxPortNumber
        MaskedValue ← MaskedValue bitwise AND BitmaskPortNumber (PI = 11)

        IF MaskedValue ≠ PortNumber (PI = 10) THEN
                EXIT
        ENDIF

        Length ← length (PL) of UniqueID (PI = 1)
        MaskLength ← length (PL) of BitmaskUniqueID (PI = 8)

        IF MaskLength ≠ Length OR MaskLength > 19 THEN
                EXIT
        ENDIF
```

```
IF MaskLength ⩾ 1 THEN
    // Compare the vendor code as follows:
    N ← min(L,2)
    MaskedValue [N] ← N left-most octets of the UniqueID
    B[N] ← N left-most octets of BitMaskUniqueID (PI = 8)
    MaskedValue ← bitwise AND B
    C ← N-left most octets of UniqueID (PI = 1)

    IF MaskedValue ≠ C THEN
        EXIT
    ENDIF

ENDIF

IF MaskLength ⩾ 3 THEN
    // Compare the right-most characters of the UniqueID as follows:
    N ←MaskLength − 2
    MaskedValue [N] ← the N right-most octets of the ALD UniqueID
    B[N] ← N right-most octets of BitMaskUniqueID (PI = 8)
    MaskedValue ← MaskedValue bitwise AND B
    C ← N right-most octets of UniqueID (PI = 1)

    IF MaskedValue ≠ C THEN
        EXIT
    ENDIF

ENDIF
```

Send AisgV3DeviceScanResponse with the ALD identification data in the fields PI = 1 (complete UniqueID), PI = 4 (ALD type), PI = 6 (vendor code), PI = 10 (port number), the list of base standard versions supported by the DeviceScanVersion (PI = 22) and PI = 27 (complete list of subunit types supported by this ALD)
EXIT

```
AISGv2DeviceScan():
    uint8_t variable K ← the length (PL) of UniqueID (PI = 1)
    uint8_t variable L ← the length (PL) of Bitmask (PI = 3)

    IF L ≠ K OR L > 19 THEN
        EXIT
    ENDIF

    IF L ⩾ 1 THEN
        // Compare the vendor code as follows:
        uint8_t N ← min (L,2)
        uint8_t A[N] ← N left-most octets of the ALD UniqueID
        uint8_t B[N] ← the N left-most octets of Bitmask (PI = 3)
        A ← A bitwise AND B
        uint8_t C[N] ← the N left-most octets of UniqueID (PI = 1)
```

```
        IF A ≠ C THEN
            EXIT
        ENDIF
    ENDIF
    IF L ⩾ 3 THEN
        // Compare the right-most characters of the UniqueID as follows:
        uint8_t N ← L-2
        uint8_t A[N] ← the N right-most octets of the ALD UniqueID
        uint8_t B[N] ← the N right-most octets of Bitmask (PI = 3)
        A ← A bitwise AND B
        uint8_t C[N] ← N right-most octets of UniqueID (PI = 1)
        IF A ≠ C THEN
            EXIT
        ENDIF
    ENDIF
```

Send AisgV2DeviceScanResponse with the ALD identification data in the fields PI = 1 (complete UniqueID), PI = 4 (ALD type) and PI = 6 (vendor code)
EXIT

NOTE: All ALDs which support AISG v3.0 and higher shall support device scan with PI = 8.

NOTE: All ALDs which support AISG v2 or lower shall support device scan with PI = 3.

NOTE: The DeviceScanVersion enables future updates of the device scan process. In later releases the meaning of DeviceScanVersion greater than 1 may be defined.

NOTE: Due to different drive capabilities of individual RS-485 components, one ALD may over-power the signal from the other ALDs. In order to detect any overpowered ALDs, the primary shall perform suitable extra device scan commands.

11.11.3. Address assignment

Description (Informative):

The XID command AddressAssignment is used by the primary to assign an ALDAddress to an ALD.

Command Specification (Normative):

```
PrimaryFrame AisgV3AddrAssignCommand {
     uint8_t address ← 0xFF          // All-station address
     uint8_t Ctrl    ← 0xBF          // Control field for XID
     uint8_t FI      ← 0x81          // Format identifier
     uint8_t GI      ← 0xF0          // Group identifier
     uint8_t GL                      // number of following octets

     XidParameter(2, ALDAddress)
     XidParameter(22, BaseStandardVersion)     // PL = 3
     XidParameter(26, PrimaryID)               // PrimaryID PI, PL = 4
```

```
      Optional XidParameter(1, UniqueID)        // UniqueID can be supplied
                                                // partially
      Optional XidParameter(4, ALDType)
      Optional XidParameter(6, VendorCode)
      Optional XidParameter(10, PortNumber)
}

PrimaryFrame AisgV2AddrAssignCommand {
      uint8_t address ← 0xFF          // All-station address
      uint8_t Ctrl    ← 0xBF          // Control field for XID
      uint8_t FI      ← 0x81          // Format identifier
      uint8_t GI      ← 0xF0          // Group identifier
      uint8_t GL                      // Number of following octets

      XidParameter(2, ALDAddress)


      Optional XidParameter(1, UniqueID)   // UniqueID can be supplied partially
      Optional XidParameter(4, ALDType)
      Optional XidParameter(6, VendorCode)
}

ALDFrame AisgV3AddrAssignResponse {
      uint8_t address                 // Assigned ALDAddress
      uint8_t Ctrl    ← 0xBF          // Control field for XID
      uint8_t FI      ← 0x81          // Format identifier
      uint8_t GI      ← 0xF0          // Group identifier
      uint8_t GL                      // Number of following octets

      XidParameter(1, UniqueID)       // ALD UniqueID
      XidParameter(4, ALDType)
      XidParameter(10, PortNumber)
}

ALDFrame AisgV2AddrAssignResponse {
      uint8_t address                 // Assigned ALDAddress
      uint8_t Ctrl    ← 0xBF          // Control field for XID
      uint8_t FI      ← 0x81          // Format identifier
      uint8_t GI      ← 0xF0          // Group identifier
      uint8_t GL                      // Number of following octets

      XidParameter(1, UniqueID)       // ALD UniqueID
      XidParameter(4, ALDType)
}
```

Primary Specification (Normative):

The primary broadcasts the XID command to which all matching ALD(s) will respond. The primary shall ensure that only one ALD matches the supplied parameter(s).

The UniqueID field can be supplied partially with a length of 1 to 19 octets. If the UniqueID field (PI = 1) is supplied partially, the right-most PL octets shall be supplied.

If the primary discovered an ALD during an AISG v2 device scan, it shall assign a unique ALDAddress to the ALD with an AisgV2AddrAssign command.

If the primary discovered an ALD during an AISG v3.0 device scan with DeviceScanVersion ← 1, it shall assign a unique ALDAddress using an AisgV3AddrAssign command, specifying the selected AISG base standard version.

It is not permitted to send more than one AISG base standard version.

ALD Specification (Normative):

```
IF LinkState ≠ NoAddress THEN
        EXIT
ENDIF

IF the frame contains the ALDAddress (PI = 2)
   AND the frame contains BaseStandardVersion (PI = 22) THEN
        Continue with AISGv3AddrAssign()
        EXIT
ENDIF

IF the frame contains ALDAddress (PI = 2) THEN
        Continue with AISGv2AddrAssign()
        EXIT
ELSE
        EXIT
ENDIF

AISGv3AddrAssign():
uint8_t L ← length (PL) of BaseStandardVersion (PI = 22)

IF L ≠ 3 OR BaseStandardVersion is not supported THEN
        EXIT
ENDIF

IF the frame contains UniqueID (PI = 1) THEN
        uint8_t N ← the length of the ALD UniqueID
        uint8_t L ← length (PL) of UniqueID (PI = 1)

        IF L > N THEN
                EXIT
        ELSE
                uint8_t A[L] ← the L right-octets of the ALD UniqueID
                uint8_t B[L] ← UniqueID (PI = 1)

                IF A ≠ B THEN
                        EXIT
                ENDIF

        ENDIF
ENDIF

IF the frame contains ALDType (PI = 4) THEN
        uint8_t L ← length (PL) of ALDType (PI = 4)

        IF L ≠ 1 THEN
                EXIT
        ELSE
                uint8_t A ← the ALDType
                uint8_t B ← ALDType (PI = 4)
```

```
        IF A ≠ B THEN
            EXIT
        ENDIF

    ENDIF
ENDIF

IF the frame contains the VendorCode (PI = 6) THEN
    uint8_t L ← length (PL) of VendorCode (PI = 6)

    IF L ≠ 2 THEN
        EXIT
    ELSE
        uint8_t A[L] ← the ALD Vendor code
        uint8_t B[L] ← VendorCode (PI = 6)

        IF A ≠ B THEN
            EXIT
        ENDIF

    ENDIF
ENDIF

IF frame contains PortNumber (PI = 10) THEN
    uint8_t L ← length (PL) of PortNumber (PI = 10)

    IF L ≠ 2 THEN
        EXIT
    ELSE

        uint8_t A[L] ← the port number on which the
          frame was received
        uint8_t B[L] ← PortNumber (PI = 10)

        IF A ≠ B THEN
            EXIT
        ENDIF

    ENDIF
ENDIF

Send AisgV3AddrAssignResponse with the ALD identification data in the fields PI = 1
(complete UniqueID) and PI = 4 (ALDType), and PI = 10 (PortNumber on which the request
was received)
PrimaryIDs[CurrentPort]← PV of PI = 26
EXIT

AISGv2AddrAssign():
IF frame contains the UniqueID (PI = 1) THEN
    uint8_t variable N ← the length of the ALD UniqueID
    uint8_t variable L ← the length (PL) of UniqueID (PI = 1)
```

```
        IF L > N THEN
                EXIT
        ELSE
                uint8_t A[L] ← the L right-octets of the ALD UniqueID
                uint8_t B[L] ← UniqueID (PI = 1)

                IF A ≠ B THEN
                        EXIT
                ENDIF

        ENDIF
ENDIF

IF frame contains the ALDType (PI = 4) THEN
        uint8_t variable L ← the length (PL) of ALDType (PI = 4)

        IF L ≠ 1 THEN
                EXIT
        ELSE
                uint8_t A ← the ALDType
                uint8_t B[L] ← ALDType (PI = 4)

                IF A ≠ B THEN
                        EXIT
                ENDIF

        ENDIF
ENDIF

IF frame contains the VendorCode (PI = 6) THEN
        uint8_t L ← the length (PL) of VendorCode (PI = 6)

        IF L ≠ 2 THEN
                EXIT
        ELSE
                uint8_t A[L] ← the ALD VendorCode
                uint8_t B[L] ← VendorCode (PI = 6)

                IF A ≠ B THEN
                        EXIT
                ENDIF

        ENDIF
ENDIF

Send AisgV2AddrAssignResponse with the ALD identification data in the fields PI = 1
(complete UniqueID) and PI = 4 (ALDType)
EXIT
```

11.11.4. Reset port

Description (Informative):

The ResetPort command is used to reset the layer 2 link of the ALD and to switch an AISG port into the NoAddress LinkState.

Command Specification (Normative):

```
PrimaryFrame ResetPortCommand {
      uint8_t address                 // All-station address or ALDAddress
      uint8_t Ctrl    ← 0xBF          // Control field for XID
      uint8_t FI      ← 0x81          // Format identifier
      uint8_t GI      ← 0xF0          // Group identifier
      uint8_t GL      ← 0x02

      XidParameter(7)                 // Reset port PI
}

ALDFrame ResetPortResponse {
      uint8_t address                 // ALDAddress
      uint8_t Ctrl    ← 0xBF          // Control field for XID
      uint8_t FI      ← 0x81          // Format identifier
      uint8_t GI      ← 0xF0          // Group identifier
      uint8_t GL      ← 0x02

      XidParameter(7)                 // Reset port PI
}
```

Primary Specification (Normative):

ALD Specification (Normative):

```
IF any other XID parameter than the ResetPort parameter is supplied in the frame THEN
      EXIT
ENDIF

IF the PL value of the ResetPort field ≠ 0 THEN
      EXIT
ENDIF

IF the XID command is received as addressed to the ALD THEN
      Send the ResetPort response
ENDIF

Switch to NoAddress LinkState      // without performing a reset
EXIT
```

11.11.5. Reset ALD

Description (Informative):

The ResetALD command is used to perform a reset on an ALD. This command affects the whole ALD and communication with other primaries connected to the ALD.

This command can also be broadcast to all ALDs. In such case it can be targeted to a specific ALD through the use of the UniqueID XID parameter.

Command Specification (Normative):

```
PrimaryFrame ResetALDCommand {
      uint8_t address                   // All-station address or ALDAddress
      uint8_t Ctrl     ← 0xBF           // Control field for XID
      uint8_t FI       ← 0x81           // Format identifier
      uint8_t GI       ← 0xF0           // Group identifier
      uint8_t GL                        // Number of following octets
      XidParameter(24)                  // ResetALD PI
      XidParameter(1, UniqueID)         // Entire ALD UniqueID
}
ALDFrame ResetALDResponse {
      uint8_t address                   // ALDAddress
      uint8_t Ctrl     ← 0xBF           // Control field for XID
      uint8_t FI       ← 0x81           // Format identifier
      uint8_t GI       ← 0xF0           // Group identifier
      uint8_t GL                        // Number of following octets
      XidParameter(24)                  // ResetALD PI
      XidParameter(1, UniqueID)         // Entire ALD UniqueID
}
```

Primary Specification (Normative):

ALD Specification (Normative):

```
IF the frame contains any other XID parameters the ResetALD (PI = 24) AND UniqueID
  (PI = 1) THEN
      EXIT
ENDIF

IF the PL value of the ResetALD field ≠ 0 THEN
      EXIT
ENDIF

IF the frame is addressed to the all-station address
      AND contains the XID parameter UniqueID (PI = 1) THEN
      uint8_t L
      uint8_t A[19]
      uint8_t B[19]
      L ← Length (PL) of UniqueID (PL=1)

      F L≠19 THEN
            EXIT
      ENDIF

      A[L] ← the ALD UniqueID
      B[L] ← UniqueID (PL=1)

      IF A≠B THEN
            EXIT
      ENDIF
ENDIF

IF the XID command is received as addressed to the ALD THEN
      Send the ResetALD response
ENDIF
```

Perform a reset
EXIT

11.11.6. Trigger ping

Description (Informative):

The XID command TriggerPing is used by the primary to synchronise the sending and monitoring of a Ping message.

Refer to Section 8.4. “Pinging” for details.

Command Specification (Normative):

```
PrimaryFrame TriggerPing {
     uint8_t address ← 0xFF          // All-station address
     uint8_t Ctrl    ← 0xBF          // Control field for XID
     uint8_t FI      ← 0x81          // Format identifier
     uint8_t GI      ← 0xF0          // Group identifier
     uint8_t GL                      // Number of following octets
     XidParameter(25)                // TriggerPing PI
}
```

Primary Specification (Normative):

ALD Specification (Normative):

IF the XID command is not addressed to the all-station address THEN
    EXIT
ENDIF

IF PL value ≠ 0 THEN
    EXIT
ENDIF

IF ALDState = PingerBroadcastWaitState THEN
    Switch ALDState to PingerRestrictedState
    Initiate the PingTimer at 45 ms
    Select the PingSendRFPort RF port

    IF the ALD is a MALD THEN
        Deactivate all OOK paths that do not have active layer 2 links
          to other primaries
    ELSE
        Deactivate all OOK paths
    ENDIF

ELSEIF ALDState = ListenerBroadcastWaitState THEN
    Switch ALDState to ListenerRestrictedPreparationState
    Initiate the PingTimer at 40 ms
    Select the PingMonitorRFPort

IF the ALD is a MALD THEN
Deactivate all OOK paths that do not have active layer 2 links to other primaries
ELSE
Deactivate all OOK paths
ENDIF

ENDIF
EXIT

11.11.7. Ping message

Description (Informative):

The XID response PingMessage is used to detect the connected RF path between ports and provides the pingee with the primary's ID.

NOTE: The PingMessage does not follow the concept of unbalanced data links according to [6] and is an AISG-specific exception to the HDLC standard.

Command Specification (Normative):

```
ALDFrame PingMessage {
      uint8_t address ← 0xFF          // All-station address
      uint8_t Ctrl    ← 0xBF          // Control field for XID
      uint8_t FI      ← 0x81          // Format identifier
      uint8_t GI      ← 0xF0          // Group identifier
      uint8_t GL                      // Number of following octets
      XidParameter(28)                // Ping message
      XidParameter(26, PrimaryID)     // PrimaryID PI, PL = 4
}
```

Primary Specification (Normative):

ALD Specification (Normative):

IF ALDState = ListenerRestrictedMonitorState THEN
Store PV of PI = 26 as PrimaryID
PingReceivedFlag ← 1
Select the initiating AISG port
Activate all previously deactivated OOK paths
Switch to OperatingState
ENDIF
EXIT

11.11.8. Disable OOK bypass

Description (Informative):

The primary shall use this command to enable and disable the OOK bypasses between RF ports within an ALD. This command is used during the device scan to discover the order of ALDs.

Command Specification (Normative):

```
PrimaryFrame DisableOOKBypassCommand {
      uint8_t address ← 0xFF          // All-station address
      uint8_t Ctrl    ← 0xBF          // Control field for XID
      uint8_t FI      ← 0x81          // Format identifier
      uint8_t GI      ← 0xF0          // Group identifier
      uint8_t GL                      // Number of following octets

      XidParameter(12, Flag)          // PI = Disable OOK bypass
                                      // 1: Disable OOK bypass
                                      // 0: Enable OOK bypass
}
```

Primary Specification (Normative):

ALD Specification (Normative):

IF any other XID parameter than the DisableOOKBypass parameter is supplied in the
    frame THEN
    EXIT
ENDIF

IF the PL value of the DisableOOKBypass field ≠ 1 THEN
    EXIT
ENDIF

IF the flag is 1 THEN
    Close all OOK bypasses belonging to the port at which the frame was received
ELSE
    Open all OOK bypasses belonging to the port at which the frame was received
ENDIF

EXIT

11.12. Link establishment

Description (Informative):

Once the ALD has been assigned an ALDAddress via an AISG port, the primary may initiate the link establishment by sending the SNRM command frame on this AISG port.

Command Specification (Normative):

```
PrimaryFrame SNRM {
      uint8_t address                 // ALDAddress
      uint8_t Ctrl          ← 0x93    // SNRM (Set Normal Response Mode)
}

ALDFrame UA {
      uint8_t address                 // ALDAddress
      uint8_t Ctrl          ← 0x73    // UA (Unnumbered acknowledge)
}

ALDFrame DM {
      uint8_t address                 // ALDAddress
      uint8_t Ctrl          ← 0x1F    // DM (Disconnected mode)
}
```

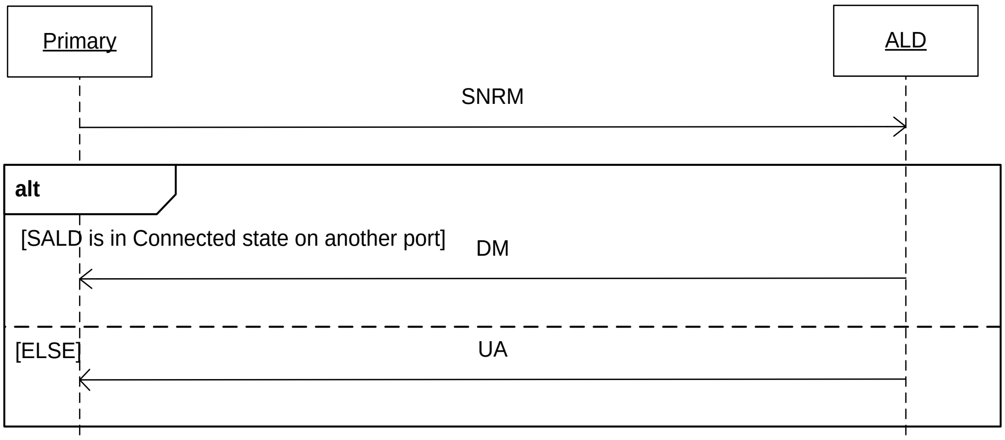


Figure 11.12-1: Sequence diagram for link establishment

Primary specification (Normative):

The primary shall use this command to establish an active layer 2 link to every ALD which it can detect on the AISG bus.

NOTE: These active links are needed so that the ALD is aware of active AISG communication on these ports and associated ports and does not deactivate its internal OOK bypasses due to an ongoing ping process started by another primary.

ALD specification (Normative):

```
IF the ALD is a SALD THEN
      IF the SALD is not in Connected LinkState OR the SALD is in Connected Linkstate on
        the same port where the frame was received THEN
            Switch LinkState to Connected
            Assign the AISG input port that received the SNRM command as the control port
            Respond with UA frame
            EXIT
      ELSE
            Respond with DM frame
            EXIT
      ENDIF
ELSEIF the ALD is a MALD THEN
      Change the HDLC link state to Connected LinkState
      Respond with UA frame
      EXIT
ENDIF
EXIT
```

11.13. Communication timeout

An ALD shall implement a communication timeout timer that is common for all AISG ports, which is started immediately after reset with a timeout of 24 hours. Whenever the ALD receives a valid HDLC frame addressed to itself, or to the all-station address, on any AISG port, it shall restart the communication timeout timer with a timeout of 3 minutes.

In either case, if the communication timeout timer expires, the ALD shall perform a reset.

11.14. HDLC description

This is an overview of the defined HDLC Class UNC1,15.1 TWA according to [6].

In the descriptions below, station A refers to the transmitting station (a primary or an ALD) and station B refers to the receiving station (an ALD or a primary).

11.14.1. Basic structure

In AISG v3.0 the primary controls the bus and a number of ALDs which are only allowed to transmit when the primary gives them permission to do so.

All frames are transmitted with the layout shown in Table 11.14.1-1: “Format of an HDLC frame”.

Flag **1 octet**	**ADDR** **1 octet**	**Control** **1 octet**	**INFO** **N octets**	**FCS** **2 octets**	**Flag** **1 octet**
0x7E	Address	Control bits	Variable length	CRC	0x7E

Table 11.14.1-1: Format of an HDLC frame

All frames begin with a starting flag (0x7E) and end with a closing flag (0x7E).

Station A calculates a Frame Check Sequence (CRC16) on all octets which follow the starting flag but not including the FCS octets. The checksum is calculated using the code found in Section A.1. in [9]. The checksum is transmitted as FCS in little endian order and is followed by the closing flag.

Station B calculates the checksum using the same procedure on all octets between the flags. When it finds the closing flag it compares the checksum to 0xF0B8. If it is a match, the frame is processed otherwise it is discarded.

The address field contains the ALDAddress of the targeted ALD. The ALD shall evaluate every frame which is sent to its ALDAddress.

If the primary sends the frame, it is called a (layer 2) command and the address field contains the ALDAddress of the ALD as destination.

If the ALD sends the frame, it is called a (layer 2) response and the address field contains the ALDAddress of the ALD as source.

NOTE: ALDs cannot communicate directly with each other.

11.14.2. All-station address

An ALD shall evaluate every frame which is sent to the all-station address (0xFF).

11.14.3. No-station address

An ALD shall send device scan responses from the no-station address (0x00).

11.14.4. Basic transparency conversion

Since the frame may contain 0x7E, basic transparency is used, which means that 0x7E is transmitted as 0x7D 0x5E and 0x7D is transmitted as 0x7D 0x5D. The receiving station converts back on reception.

Basic transparency conversion is performed after the checksum has been calculated and added to the frame, that is the basic transparency conversion also applies to the checksum.

11.14.5. Layer 2 frame types

Three different frame types are defined in the layer 2:

- I-frames contain data as well as a send and receive counter
- S frames contain a receive counter (RR/RNR frames)
- U frames are unnumbered (XID, UA, DM, SNRM and FRMR frames)

The INFO field is only present in I-frames, XID frames and FRMR frames.

Table 11.14.5-1: “Frame types:” lists the valid frame types that may be sent by the primary and the ALDs.

Primary	ALD
Frame type I	Frame type I
Frame type RR	Frame type RR
Frame type RNR	Frame type RNR
Frame type SNRM	Frame type UA
Frame type XID	Frame type DM
Frame type DISC	Frame type XID
	Frame type FRMR

Table 11.14.5-1: Frame types

11.14.5.1. SNRM frame (Set Normal Response Mode)

On reception of this command the ALD enters the Connected LinkState, re-initialise its sequence number variables and then responds with UA. If the ALD rejects the SNRM it responds with DM.

11.14.5.2. DISC frame (Disconnect)

On reception of this command while the ALD is in the Connected LinkState, it enters the AddressAssigned LinkState and then response with UA. Otherwise, it responds with DM.

11.14.5.3. UA frame (Unnumbered Acknowledge)

This response is used by the ALD to confirm that the ALD received and acted on an SNRM or DISC command.

11.14.5.4. DM frame (Disconnected Mode)

This response is used by the ALD to inform the primary that the ALD is disconnected or it cannot enter the Connected LinkState.

11.14.5.5. RR frame (Receiver Ready)

This frame is used by station A to inform station B (primary or ALD) that station A is ready to receive an I-frame, that is that it has empty buffers. This aspect is used for flow control.

The RR frame also contains the sequence number of the next frame station A expects to see. This works both as an ACK and a NAK depending on the value of the transmitted sequence number.

If a station does not receive an ACK in the defined timeout (see Section 11.8. “Frame timing”), it shall retransmit the frame with the same sequence number.

11.14.5.6. RNR frame (Receiver Not Ready)

This frame is used by station A to inform station B (primary or ALD) that station A is not ready to receive an I-frame, for instance because it has no empty buffers. Station B shall then stop transmitting I-frames. This aspect is used for flow control.

An ALD shall have at least two I-frame buffers for each primary that it can support.

The RNR frame also contains the sequence number of the next frame that station A expects to see. This works both as an ACK and a NAK depending on the value of the transmitted sequence number.

If a station does not receive an ACK in the defined timeout (see Section 11.8. “Frame timing”), it shall retransmit the frame with the same sequence number.

NOTE: At some point the primary may give up and report an alarm.

11.14.5.7. I-Frame (Information)

This frame is used to transfer a block of data together with its sequence number. The frame also includes the sequence number of the next frame station A expects to see. This way, it works as an RR. Like RR, it enables transmission of I-frames fromstation B.

If a station does not receive an ACK in the defined timeout (see Section 11.8. “Frame timing”), it shall retransmit the frame with the same sequence number.

The INFO field in an I-frame contains the layer 7 messages.

11.14.5.8. FRMR (Frame Reject)

This response is used by the ALD to indicate an error condition in the following cases:

- The ALD receives an invalid (layer 2) command. (The text of the standard defines explicitly what is required or permitted. An invalid command is anything that is not defined in this standard, for instance an SREJ or incorrectly formatted XID frame.)
- ALD receives an I-frame or XID-frame with an information field which exceeded the maximum information field length which can be accommodated by the receiving station.

- The ALD receives an invalid N(R) which identifies an I-frame which has previously been transmitted and acknowledged or an I-frame which has not been transmitted and is not the next sequential I-frame awaiting transmission.
- The ALD receives a (layer 2) command containing an information field, where no information field is permitted by the associated control field.

First bit transmitted (bit 0)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				z	y	x	w	N(R)			C/R	N(S)			0	Rejected frame control field							
3rd octet								2nd octet								1st octet							

Table 11.14.5.8-1: Format of the INFO field of an FRMR frame

11.14.6. XID frame

XID frames are used to scan the AISG bus, to negotiate AISG parameters and to broadcast commands from the primary to all ALDs or from one ALD to the primary.

XID frames use the following format in the INFO field to transfer parameters:

FI	GI	GL	PI	PL	PV	PI	PL	PV

Table 11.14.6-1: Format of the INFO field of an XID frame

The parameters are identified by a one octet Format Identifier (FI) code and a one octet Group Identifier (GI) code. The Group Length (GL) is a one octet unsigned integer giving the length in octets of the parameters following it.

A parameter consists of a sequence of PI/PL/PV values. The Parameter Identifier (PI) is a one octet code identifying the parameter. Parameter Length (PL) is a one octet unsigned integer giving the length in octets of the Parameter Value (PV).

The parameter order is defined for every AISG XID command.

11.14.7. Control field definition

Table 11.14.7-1: "Definition of control fields" defines the control field for every layer 2 frame type.

First bit transmitted

Control field format for	**Control field bits**							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
I-frame	N(R)			P/F	N(S)			0
RR frame	N(R)			P/F	0	0	0	1
RNR frame	0	0	0	P/F	1	0	0	1
UA frame	0	1	1	P/F	0	0	1	1
SNRM frame	1	0	0	P/F	0	0	1	1
FRMR frame	1	0	0	P/F	0	1	1	1
DM frame	0	0	0	P/F	1	1	1	1
XID frame	1	0	1	P/F	1	1	1	1

Table 11.14.7-1: Definition of control fields

NOTE: N(S) = transmitting send sequence number (bit 1 = low-order bit)
N(R) = transmitting receive sequence number (bit 5 = low-order bit)

11.14.8. Poll

A poll is a frame from the primary where the P/F (Poll / Final) bit in the control field is set to 1. This informs the ALD that it is allowed to transmit a response frame.

All I-frames, S-frames and U-frames from the ALD shall have the P/F bit set.

A broadcast frame shall not have the P/F bit set.

U-frames require a specific U-frame response (for instance an SNMR/UA exchange).

When the ALD receives an I-frame or S-frame, it shall transmit the oldest queued I-frame, if any; otherwise it shall transmit an S-frame.

12. LAYER 7

12.1. General

Layer 7 defines the commands and responses for direct communication between a primary and an ALD.

This section outlines and defines commands that are common and applicable to all ALDs, while subunit type-specific commands and functionality are defined in subunit type standards.

12.2. Integer representation in layer 7

Multi-octet integer values are transmitted in little endian order. Signed integers are represented as 2-complement values.

12.3. Services expected from layer 2

Layer 7 requires an assured in-sequence delivery service from layer 2. Layer 7 must be informed by layer 2 if the assured in-sequence delivery service is no longer available.

12.4. Layer 7 message timing

ALD commands shall, unless otherwise specified, provide a response message within 1 second. Commands declared as Time-Consuming Commands (TCC) have a longer maximum response time.

The response time is measured from the time the message frame was received by layer 2 to the time the response message is ready for transfer by layer 2.

12.5. Alarms

In some situations, a command may cause a change of operating conditions; for instance, a SetTilt command might cause a RET subunit to discover that an actuator is jammed or that a previously jammed actuator works again. In these cases, an AlarmIndication reporting the change of operating conditions shall be issued in addition to the response message to those primaries that have subscribed to alarms.

An alarm informs the receiver of a fault. There is no response to an alarm. Alarms are cleared when the cause of the fault has ceased.

Response code GeneralError is a special response for the ALD vendor to provide more information about the issue ALD is having. When using response code GeneralError, the ALD shall provide meaningful additional information about the reason why GeneralError was issued by using the GetDiagnosticInformation command.

12.6. General command handling

This section defines the general behaviour of ALDs.

```
uint16_t CurrentPort // The port number the command was received on
```

The primary shall support all return codes listed both in this document and in subunit type standards that it supports.

12.6.1. Alarm handling

An ALD shall have an array LinkAlarms which contains the current alarm state (both common alarms and subunit alarms) for every layer 2 link.

```
Enumeration Severity_t : uint8_t {
      Cleared          ← 0
      Warning          ← 1
      Minor            ← 2
      Major            ← 3
      Critical         ← 4
}
```

The Alarm severity state is defined in [8]. Severity Indeterminate, defined in [8], is not used in this standard.

```
struct AlarmState_t {
      AlarmCode_t      Alarm
      Severity_t       Severity
}

struct SubunitAlarmStates_t {
      AlarmState_t Alarms[NrOfSubunitAlarms]
}

struct LinkAlarmStates_t {
      AlarmState_t CommonAlarms[NrOfCommonAlarms]
      SubunitAlarmStates_t SubunitAlarms[NrOfSubunits]
}

LinkAlarmStates_t LinkAlarms[NrOfLinks]
```

The table CommonAlarms contains all alarms relevant for the whole ALD and the tables SubunitAlarms contain all subunit-specific alarms. NrOfSubunitAlarms is subunit type-specific and defined in the relevant subunit type standard. All the tables shall be initialised with Severity = Cleared for every Alarm during the startup of the ALD.

After a reset, all alarm states shall be cleared.

```
IF the ALD responds with an error message AND the ReturnCode_t is also an AlarmCode
THEN
      Set the corresponding alarm state to 1 in the corresponding tables.
ELSEIF the ALD detects a fault, which is valid for the whole ALD THEN
      Set the corresponding alarm state to 1 in CommonAlarms for every layer 2 link
ELSEIF the ALD detects that a fault no longer exists THEN
      Set the corresponding alarm state to 0 in the corresponding tables.
ENDIF
EXIT
```

12.6.2. Command message interpretation

The following rules shall be used if an ALD receives a command.

The following conditions shall be checked before the parsing of command-specific behaviour.

```
IF the command is shorter than 4 octets THEN
        EXIT
ELSEIF the command has a length inconsistent with its "Number of data octets" field value
THEN
        RETURN FormatError
ELSEIF the command is undefined for this ALD type
        OR the command is undefined for this subunit type
        OR the command is optional and not supported for this ALD type
        OR the command is optional and not supported for this subunit type THEN
        RETURN UnknownCommand
ELSEIF the command has a length inconsistent with the defined message length in the
command definition THEN
        RETURN FormatError
ENDIF
EXIT
```

12.6.2.1. Validation of subunit number and type

The following condition for subunit number and type shall be checked before parsing of command-specific behaviour for all commands specified in Subunit Type Standards:

```
IF UpperOctet(Cmd.Command) ≠ 0 THEN

        IF Cmd.Subunit is not one of 1..NrOfSubunits THEN
                RETURN InvalidSubunitNumber
                EXIT
        ENDIF
```

```
    IF Subunits[Cmd.Subunit].Type ≠ UpperOctet(Cmd.Command) THEN
        RETURN InvalidSubunitType
        EXIT
    ENDIF
ENDIF
```

12.6.3. Overview of commands (informative):

The table below shows an overview of all commands used in this base standard.

The following abbreviations are used in the Table 12.6.3-1: "Commands for ALDs":

M Mandatory

O Optional

\- Not applicable

RF Mandatory if the ALD has RF ports, otherwise not applicable

P Mandatory if the ALD supports pinger functionality, otherwise not applicable

L Mandatory if the ALD supports listener functionality, otherwise not applicable

P/L Mandatory if the ALD supports pinger or listener functionality, otherwise not applicable

	Code	Initiator	Subunit	Timeout	TCC	Mandatory for:			Changes the ConnectionState	Minimum required authority	MALD Configuration Security	MALD SW Download Security
						Primary	SALD	MALD				
Common commands												
Get Alarm Status	0x0004	Primary	any	1 s	no	O	M	M	no	RO	—	—
Get Information	0x0005	Primary	0	1 s	no	O	M	M	no	—	—	—
Clear Active Alarms	0x0006	Primary	any	1 s	no	O	M	M	no	RW	—	—
Alarm Subscribe	0x0012	Primary	0	1 s	no	O	M	M	no	—	—	—
Alarm Indication	0x0007	ALD	any	—	—	O	M	M	no	RO	—	—
Download Start	0x0040	Primary	0	21 s	yes	M	M	M	yes	—	—	yes
Download File	0x0041	Primary	0	1 s	no	M	M	M	no	—	—	yes
Download End	0x0042	Primary	0	10 s	yes	M	M	M	yes	—	—	yes
Get Subunit List	0x0008	Primary	0	1 s	no	O	M	M	no	—	—	—
Get Reset Cause	0x0009	Primary	0	1 s	no	O	M	M	no	—	—	—
Get AISG Port DC Power Information	0x001D	Primary	0	1 s	no	O	M	M	no	—	—	—
Get Diagnostic Information	0x000B	Primary	any	1 s	no	O	M	M	no	RO	—	—
Set Subunit Type Standard Versions	0x000C	Primary	0	1 s	no	M	M	M	no	—	—	—

	Code	Initiator	Subunit	Timeout	TCC	Mandatory for: Primary	Mandatory for: SALD	Mandatory for: MALD	Changes the ConnectionState	Minimum required authority	MALD Configuration Security	MALD SW Download Security
Get Subunit Type Standard Versions	`0x000D`	Primary	0	1 s	no	M	M	M	no	—	—	—
Upload Info	`0x003C`	Primary	0	1 s	no	M	M	M	no	—	—	—
Upload Start	`0x003D`	Primary	0	2 s	no	M	M	M	yes	—	—	—
Upload File	`0x003E`	ALD	0	1 s	no	M	M	M	no	—	—	—
Upload End	`0x003F`	ALD	0	2 s	no	M	M	M	yes	—	—	—
Send Layer 1 Test Pattern	`0x00B1`	Primary	0	1 s	no	O	M	M	no	—	—	—
Generate Test Alarm	`0x00B2`	Primary	0	1 s	no	O	M	M	no	—	—	—
Vendor specific command	`0x0090`	Primary	any			O	O	O	no	—	—	—
MALD Configuration Commands												
MALD Download Initiated	`0x0013`	ALD	0	1 s	no	M	—	M	yes	—	—	—
MALD Get Information	`0x0014`	Primary	0	1 s	no	O	—	M	no	—	—	—
MALD Start Conf	`0x0018`	Primary	0	2 s	no	O	—	M	yes	—	yes	—
MALD Commit Conf	`0x0019`	Primary	0	2 s	no	O	—	M	yes	—	yes	—
MALD Abort Conf	`0x001A`	Primary	0	2 s	no	O	—	M	yes	—	yes	—
MALD Reset Conf	`0x0017`	Primary	0	2 s	no	O	—	M	yes	—	yes	—
MALD Set Subunit Conf	`0x0015`	Primary	0	1 s	no	O	—	M	no	—	yes	—
MALD Get Subunit Conf	`0x0016`	Primary	0	1 s	no	O	—	M	no	—	—	—
MALD Set Security Setting	`0x001B`	Primary	0	1 s	no	O	—	M	no	—	—	—
MALD Get Security Setting	`0x001C`	Primary	0	1 s	no	O	—	M	no	—	—	—
SiteMapping Commands												
Get Number Of Ports	`0x001E`	Primary	0	1 s	no	O	M	M	no	—	—	—
Get Port Info	`0x001F`	Primary	0	1 s	no	O	M	M	no	—	—	—
Get RF Port Frequency Info	`0x0025`	Primary	0	1 s	no	O	RF	RF	no	—	—	—
Get Port Interconnections	`0x0020`	Primary	0	1 s	no	O	M	M	no	—	—	—
Set RF Path IDs	`0x0021`	Primary	0	1 s	no	O	M	M	no	—	—	—
Set RF Path ID Alias	`0x0022`	Primary	0	1 s	no	O	M	M	no	—	—	—
Get RF Path IDs	`0x0023`	Primary	0	1 s	no	O	M	M	no	—	—	—
Get RF Path ID Alias	`0x0024`	Primary	0	1 s	no	O	M	M	no	—	—	—
Get Connector Plate Marking Info	`0x0025`	Primary	0	1 s	no	O	M	M	no	—	—	—
Ping Commands												
Send Ping	`0x0026`	Primary	0	2 s	no	O	P	P	yes	—	—	—

	Code	Initiator	Subunit	Timeout	TCC	Mandatory for:			Changes the ConnectionState	Minimum required authority	MALD Configuration Security	MALD SW Download Security
						Primary	SALD	MALD				
Monitor Ping	`0x0027`	Primary	0	2 s	no	O	L	L	yes	—	—	—
Abort Ping	`0x0028`	Primary	0	2 s	no	O	P/L	P/L	yes	—	—	—

Table 12.6.3-1: Commands for ALDs

12.6.4. Layer 7 timeout definitions

If a command is rejected then the error response must be transmitted within 1 second.

If a command is accepted the maximum response time is specified in Table 12.6.3-1: “Commands for ALDs”.

If a command changes the ConnectionState, this change may require up to 1 second for other commands to finish. This time is included in the maximum response time specified in Table 12.6.3-1: “Commands for ALDs”.

12.7. Parallel command handling

Parallel execution of primary commands on the same layer 2 link is not permitted and the second command will be rejected with Busy. In the case of a MALD, parallel execution of TCCs on multiple layer 2 links is not allowed and the second command will be rejected with InUseByAnotherPrimary.

A MALD will wait for all non-TCCs on other layer 2 link(s) to be completed before executing a command which changes any ConnectionState.

Global variables

```
Mutex StateLock                       // Mutual exclusion
uint16_t CommandCount                 // The number of Commands running
Boolean ActiveTCC                     // Shows if a TCC-Command is running
Boolean PendingConnectionStateChange // Shows if the ALD waits to change state(s)
```

Variables for each command

```
Boolean IsTCC[CommandCode]            // True if this Command is a TCC
```

Variables for each layer 2 link

```
Boolean ActiveCommand[1+MaxPort]      // Shows if a command is running for a layer 2
                                      // link

struct StateAllowed_t {
     Boolean allowed
     ReturnCode_t code
}
```

```
FUNCTION IsCommandAllowed(ConnectionState_t AcceptedStates[],CommandCode_t
CurrentCommand, uint16_t CurrentPort) RETURNING StateAllowed_t result IS
        uint8_t L
        LOCK StateLock
        result.allowed ← false
        L ← length of AcceptedStates
        FOREACH N FROM 0 TO L-1 DO
                IF AcceptedStates[N] = ConnectionState[CurrentPort] THEN
                        result.allowed ← true
                ENDIF
        ENDFOR
        IF NOT result.allowed THEN
                result.code ← IncorrectState
        ELSEIF ActiveCommand[CurrentPort] THEN
                result.allowed ← false
                result.code ← Busy
        ELSEIF PendingConnectionStateChange THEN
                result.allowed ← false
                result.code ← InUseByAnotherPrimary
        ELSE
                IF IsTCC[CurrentCommand] THEN
                        IF ActiveTCC THEN
                                result.allowed ← false
                                result.code ← InUseByAnotherPrimary
                        ELSE
                                ActiveTCC ← true
                        ENDIF
                ENDIF
                IF result.allowed THEN
                        CommandCount ← CommandCount + 1
                        ActiveCommand[CurrentPort] ← true
                ENDIF
        ENDIF
        UNLOCK StateLock
END
FUNCTION IsStateChangeAllowed(ConnectionState_t AcceptedStates[],CommandCode_t
CurrentCommand, uint16_t CurrentPort) RETURNING StateAllowed_t result IS
        LOCK StateLock
        uint8_t L
        result.allowed ← false
        L ← length of AcceptedStates
```

```
    FOREACH N FROM 0 TO L-1 DO
        IF AcceptedStates[N] = ConnectionState[CurrentPort] THEN
            result.allowed ← true
        ENDIF
    ENDFOR

    IF NOT result.allowed THEN
        result.code ← IncorrectState
    ELSEIF ActiveCommand[CurrentPort] THEN
        result.allowed ← false
        result.code ← BUSY
    ELSEIF PendingConnectionStateChange THEN
        result.allowed ← false
        result.code ← InUseByAnotherPrimary
    ELSEIF ActiveTCC THEN
        result.allowed ← false
        result.code ← InUseByAnotherPrimary
    ELSE
        PendingConnectionStateChange ← true

        IF CommandCount > 0 THEN
            UNLOCK StateLock
            WAIT UNTIL CommandCount = 0
            LOCK StateLock
        ENDIF

        CommandCount ← CommandCount + 1
        ActiveCommand[CurrentPort] ← true
        ActiveTCC ←IsTCC[CurrentCommand]
    ENDIF

    UNLESS result.allowed THEN
        UNLOCK StateLock
    ENDIF

END

FUNCTION CommandExit(CommandCode_t CurrentCommand, uint16_t CurrentPort) IS
    LOCK StateLock

    IF ActiveCommand[CurrentPort] THEN
        CommandCount ← CommandCount – 1
        ActiveCommand[CurrentPort] ← false
        ActiveTCC ← false
    ENDIF

    UNLOCK StateLock
END
```

12.8. Common commands

12.8.1. Get Alarm Status

Description (Informative):

On successful completion of GetAlarmStatus command, t ALD return the alarm code and severity of all active alarms.

Specification (Normative):

```
PrimaryCommand GetAlarmStatusCommand {
      CommandCode_t          Command ← 0x0004
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit
      DataLength_t           DataLength ← 0
}

ALDResponse GetAlarmStatusResponse {
      CommandCode_t          Command ← 0x0004
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
            uint8_t          NrOfActiveAlarms
            uint8_t          ActiveAlarms[NrOfActiveAlarms]
      }
      else {
            ALDState_t             ALDState
            ConnectionState_t      ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
      NotAuthorised
}
```

Primary specification (Normative):

ALD specification (Normative):

IF Cmd.Subunit = 0 THEN
    RETURN OK and CommonAlarms for this layer 2 link
    EXIT
ELSEIF Cmd.Subunit is not one of 1..NrOfSubunits THEN
    RETURN InvalidSubunitNumber
    EXIT
ENDIF

```
IF ALDType = MALD
        AND ActiveAuth[CurrentPort].Authority[Cmd.Subunit] = NoAccess THEN
        RETURN NotAuthorised
        EXIT
ENDIF

result ← IsCommandAllowed( LIST{    OperatingConnectionState,
                                    RestrictedConnectionState},
                                    Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
        RETURN result.code
        EXIT
ENDIF

RETURN OK and SubunitAlarms for requested subunit
CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.8.2. Get Information

Description (Informative):

On successful completion of GetInformation command, the ALD returns its product number, serial number, hardware version and software version.

Specification (Normative):

```
PrimaryCommand GetInformationCommand {
      CommandCode_t         Command ← 0x0005
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit ← 0
      DataLength_t          DataLength ← 0
}

ALDResponse GetInformationResponse {
      CommandCode_t         Command ← 0x0005
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
            uint8_t         LengthOfProductNumber
            UTF8String_t    ProductNumber
            uint8_t         LengthOfSerialNumber
            UTF8String_t    SerialNumber
            uint8_t         LengthOfHwVersion
            UTF8String_t    HwVersion
            uint8_t         LengthOfSWVersion
            UTF8String_t    SWVersion
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

UNLESS Cmd.Subunit = 0 THEN
    RETURN InvalidSubunitNumber
    EXIT
ENDIF

result ← IsCommandAllowed(LIST{ OperatingConnectionState,
RestrictedConnectionState,
MALDConfigConnectionState},
Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
    RETURN result.code
    EXIT
ENDIF

RETURN OK, length(ProductNumber), ProductNumber, length(SerialNumber), SerialNumber, length(HwVersion), HwVersion, length(SWVersion), SWVersion
CommandExit(Cmd.Command, CurrentPort)
EXIT

12.8.3. Clear Active Alarms

Description (Informative):

On successful completion of the ClearActiveAlarms command, the ALD clears all stored alarm states and diagnostic information, including all test alarms.

Specification (Normative):

```
PrimaryCommand ClearActiveAlarmsCommand {
      CommandCode_t            Command ← 0x0006
      CommandSequence_t        PrimaryCommandSequence
      Subunit_t                Subunit
      DataLength_t             DataLength ← 0
}
```

```
ALDResponse ClearActiveAlarmsResponse {
      CommandCode_t              Command ← 0x0006
      CommandSequence_t          PrimaryCommandSequence
      ReturnCode_t               ReturnCode
      DataLength_t               DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t           ALDState
            ConnectionState_t    ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
      NotAuthorised
}
```

Primary specification (Normative):

ALD specification (Normative):

IF Cmd.Subunit = 0 THEN
    Clear all CommonAlarms for this layer 2 link
    RETURN OK
    EXIT
ENDIF

IF Cmd.Subunit is not one of 1..NrOfSubunits THEN
    RETURN InvalidSubunitNumber
    EXIT
ENDIF

IF ALDType = MALD
    AND ActiveAuth[CurrentPort].Authority[Cmd.Subunit] ≠ ReadWrite THEN
    RETURN NotAuthorised
    EXIT
ENDIF

result ← IsCommandAllowed(LIST{ OperatingConnectionState,
    RestrictedConnectionState},
    Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
    RETURN result.code
    EXIT
ENDIF

Clear all SubunitAlarms for the requested subunit for this layer 2 link
RETURN OK
CommandExit(Cmd.Command, CurrentPort)
EXIT

12.8.4. Alarm Subscribe

Description (Informative):

On successful completion of the AlarmSubscribe command, the ALD starts reporting alarms to the primary by sending AlarmIndication commands to the primary.

NOTE: The reason alarms are not reported before the primary subscribes to alarms is that the primary may not be ready to receive them.

Specification (Normative):

```
PrimaryCommand AlarmSubscribeCommand {
      CommandCode_t          Command ← 0x0012
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength ← 0
}

ALDResponse AlarmSubscribeResponse {
      CommandCode_t          Command ← 0x0012
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
      }
      else {
            ADLState_t             ALDState
            ConnectionState_t      ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

result ← IsCommandAllowed( LIST{    OperatingConnectionState,
                                    RestrictedConnectionState},
                                    Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF

AlarmSubscribeFlag[CurrentPort] ← true
RETURN OK
```

IF at least one Alarm in LinkAlarms for this layer 2 link is raised THEN
    Send AlarmIndication
ENDIF

CommandExit(Cmd.Command, CurrentPort)
EXIT

12.8.5. Alarm Indication

Description (Informative):

The ALD issues the AlarmIndication command to report alarm state changes to the primary.

NOTE: This command is only issued if the primary has subscribed to alarms since the ALD reset.

Specification (Normative):

```
ALDCommand AlarmIndicationCommand {
      CommandCode_t          Command ← 0x0007
      CommandSequence_t      ALDCommandSequence
      Subunit_t              Subunit
      DataLength_t           DataLength
      uint8_t                NrOfAlarms
      for (i = 0; i < NrOfAlarms; i++) {
            Result_t         AlarmCode_t
            Severity_t       Severity
      }
}
```

FOREACH layer 2 link DO

    IF AlarmSubscribeFlag for this layer 2 link is not set THEN
        CONTINUE
    ENDIF

    FOR every subunit
        IF SubunitAlarms for this subunit has changed AND
        ActiveAuth[CurrentPort].Authority[Cmd.Subunit] ≠ NoAccess THEN
            Send AlarmIndication with subunit number and all
            alarm states changes that have not been reported for this layer 2 link
        ENDIF
    ENDFOR

    IF the table CommonAlarms for this layer 2 link has changed THEN
        Send AlarmIndication with subunit number 0 and all changes in
        CommonAlarms that have not been reported for this layer 2 link
    ENDIF

ENDFOR
EXIT

12.8.6. Download Start

Description (Informative):

The DownloadStart command initiates the download process for data or firmware files.

The supported file types defined in this base standard are listed in Table 12.8.6-1: “Description of the file types”. Additional file types may be defined by subunit type standards. The filetypes not listed in any AISG standard are unknown.

Usage of the file types for download and upload is defined in Table 12.8.6-2: “Usage of the file types”. A download or upload to the ALD is indicated by specifying subunit 0 to the DownloadStart and UploadStart command.

Name	Description
Firmware File	Contains the executable binary of the ALD
Configuration File	Configuration data for the ALD and all subunits
Log File	For diagnostic use
Information File	Contains data that have no operational impact within the subunit or ALD

Table 12.8.6-1: Description of the file types

Name	Download to ALD	Download to subunit	Upload from ALD	Upload from Subunit
Firmware File	X			
Configuration File	X			
Log File			X	X
Information File	X	X	X	X

Table 12.8.6-2: Usage of the file types

Specification (Normative):

```
// Number of MALD ports waiting for a response
uint16_t MALDDownloadInitiatedResponseCounter

PrimaryCommand DownloadStartCommand {
      CommandCode_t         Command ← 0x0040
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit
      DataLength_t          DataLength ← 1
      FileType_t            FileType
}

ALDResponse DownloadStartResponse {
      CommandCode_t         Command ← 0x0040
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
      }
      else {
           ALDState_t            ALDState
           ConnectionState_t     ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      UnsupportedFileType
      IncorrectState
      NotAuthorised
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
IF Cmd.FileType is not supported or is unknown THEN
      RETURN UnsupportedFileType
      EXIT
ENDIF

IF ALDType = MALD AND Cmd.FileType = FirmwareFile
      AND ActiveAuth[CurrentPort].SWDownload = NotAllowed THEN
      RETURN NotAuthorised
      EXIT
ENDIF

result ← IsStateChangeAllowed( LIST{ OperatingConnectionState,
                                     DownloadConnectionState,
                                     DownloadFailedConnectionState},
                                     Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF

Switch ConnectionState[CurrentPort] to DownloadConnectionState
Switch ALDState to DownloadState
ReceivedFileType ← Cmd.FileType
DownloadPort ← CurrentPort
MALDDownloadInitiatedResponseCounter ← 0

FOREACH PORT FROM 1 TO MaxPort DO
      NEXT IF PORT = DownloadPort
      IF ALDType = MALD AND LinkState[PORT] = Connected THEN
            Switch ConnectionState[PORT] to DownloadNotificationConnectionState
            Queue MALDDownloadInitiatedCommand for transmission on port PORT
MALDDownloadInitiatedResponseCounter ← MALDDownloadInitiatedResponseCounter +1
      ELSE // It is a SALD's non-control port or MALD's port without Layer 2 link
            Switch ConnectionState[PORT] to OffConnectionState
            Switch LinkState[PORT] to NoAddress
            Disable serial port PORT
      ENDIF
ENDFOR
```

```
IF MALDDownloadInitiatedResponseCounter = 0 THEN
        Switch ConnectionState[DownloadPort] to DownloadConnectionState
        PendingConnectionStateChange ← false
        UNLOCK StateLock
        RETURN OK
        CommandExit(Cmd.Command, CurrentPort)
        EXIT
ELSE
        Initialise DownloadInitialDelayTimer at 10 seconds
ENDIF

EXIT

ON DownloadInitialDelayTimer OR StartDownloadEvent DO

        UNLESS MALDDownloadInitiatedResponseCounter = 0 THEN
                FOREACH PORT FROM 1 TO MaxPort DO
                        IF ConnectionState[PORT] = DownloadNotificationConnectionState THEN
                                Switch ConnectionState[PORT] to OffConnectionState
                                Switch LinkState[PORT] to NoAddress
                                Disable serial port PORT
                        ENDIF
                ENDFOR
        ENDIF

        Switch ConnectionState[CurrentPort] to DownloadConnectionState
        PendingConnectionStateChange ← false
        UNLOCK StateLock
        RETURN OK on DownloadPort
        CommandExit(Cmd.Command, DownloadPort)

DONE
```

NOTE: The list of file type codes may be extended by subunit type standards.

NOTE: To prevent simultaneous downloads the DownloadPort is stored, so a second attempt to download on a different port can be rejected.

12.8.7. Download File

Description (Informative):

This command is used once or several times to transfer data from the primary to the ALD.

All blocks except the last shall be 256 octets. The last block may not be 0 octets.

Specification (Normative):

```
PrimaryCommand DownloadFileCommand {
      CommandCode_t          Command ← 0x0041
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit
      DataLength_t           DataLength
      uint8_t                Block[DataLength]
}
```

```
ALDResponse DownloadFileResponse {
      CommandCode_t         Command ← 0x0041
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidFileContent
      InvalidSubunitNumber
      DownloadFailed
      GeneralError
}
```

Primary specification (Normative):

ALD specification (Normative):

```
result ← IsCommandAllowed( LIST{    DownloadConnectionState,
                                    DownloadFailedConnectionState},
                                    Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF

IF ConnectionState[CurrentPort] = DownloadFailedConnectionState THEN
      RETURN DownloadFailed
      CommandExit(Cmd.Command, CurrentPort)
      EXIT
ENDIF

Collect data and verify the data format and size

IF the ALD detects invalid data THEN
      RETURN InvalidFileContent
ELSE
      Store data to memory

      IF the ALD detects a hardware error during storing data THEN
            // Replace "Hardware error" with a text describing the problem
            RAISE AlarmGeneralError SEVERITY Major ON ALD, "Hardware error"
            RETURN GeneralError
      ELSE
            RETURN OK
      ENDIF
```

ENDIF

IF Response.Returncode ≠ OK
    LOCK StateLock
    Switch ConnectionState[CurrentPort] TO DownloadFailedConnectionState
    UNLOCK StateLock
ENDIF

CommandExit(Cmd.Command, CurrentPort)
EXIT

12.8.8. Download End

Description (Informative):

This command signals the end of a multi-message data transfer to the ALD.

Specification (Normative):

```
Enumeration OptionCode_t : uint8_t {
      Complete    ← 0
      Cancel      ← 1
}

PrimaryCommand DownloadEndCommand {
      CommandCode_t          Command ← 0x0042
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit
      DataLength_t           DataLength ← 1
      OptionCode_t           Option
}

ALDResponse DownloadEndResponse {
      CommandCode_t          Command ← 0x0042
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
      }
      else {
           ALDState_t             ALDState
           ConnectionState_t      ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidFileContent
      DownloadFailed
      OutOfRange
      InvalidSubunitNumber
      GeneralError
}
```

Primary specification (Normative):

ALD specification (Normative):

```
result ← IsStateChangeAllowed( LIST{  DownloadConnectionState,
                                      DownloadFailedConnectionState},
                                      Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
        RETURN result.code
        EXIT
ENDIF

UNLESS Cmd.Option = Complete THEN

        IF Cmd.Option = Cancel THEN
                RETURN OK
        ELSE
                RETURN OutOfRange
        ENDIF

        Switch ALDState to OperatingState
        Switch ConnectionState[CurrentPort] to OperatingConnectionState
        PendingConnectionStateChange ← false

        FOREACH PORT FROM 1 TO MaxPort DO
                NEXT IF PORT = CurrentPort
                Switch ConnectionState[PORT] TO NoConnectionState
                Enable serial port PORT
        ENDFOR

        UNLOCK StateLock
        CommandExit(Cmd.Command, CurrentPort)
        EXIT
ENDIF

IF ConnectionState[CurrentPort] = DownloadFailedConnectionState THEN
        RETURN DownloadFailed
        CommandExit(Cmd.Command, CurrentPort)
        EXIT
ENDIF

Verify the whole downloaded file
IF the ALD detects a hardware error THEN
        // Replace "Hardware error" with a text describing the problem
        RAISE AlarmGeneralError SEVERITY Major ON ALD, "Hardware error"
        RETURN GeneralError
        Switch ALDState to OperatingState
        Switch ConnectionState[CurrentPort] to OperatingConnectionState
        PendingConnectionStateChange ← false
```

```
        FOREACH PORT FROM 1 TO MaxPort DO
                NEXT IF PORT = CurrentPort
                Switch ConnectionState[PORT] TO NoConnectionState
                Enable serial port PORT
        ENDFOR

        UNLOCK StateLock
        CommandExit(Cmd.Command, CurrentPort)
        EXIT
ENDIF

IF the ALD detects a corrupted file THEN
        RETURN InvalidFileContent
        Switch ALDState to OperatingState
        Switch ConnectionState[CurrentPort] to OperatingConnectionState
        PendingConnectionStateChange ← false

        FOREACH PORT FROM 1 TO MaxPort DO
                NEXT IF PORT = CurrentPort
                Switch ConnectionState[PORT] TO NoConnectionState
                Enable serial port PORT
        ENDFOR

        UNLOCK StateLock
        CommandExit(Cmd.Command, CurrentPort)
        EXIT
ENDIF
RETURN OK

// If the filetype is InformationFile do nothing, since it is already stored

IF ReceivedFileType = FirmwareFile THEN
        Select the new firmware as the active firmware
        Wait for layer 2 acknowledgement (RR) from the primary
        Immediately perform a reset
ELSEIF ReceivedFileType = ConfigurationFile THEN
        Select the new configuration as the active configuration
ENDIF

CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.8.9. Get Subunit List

Description (Informative):

The ALD returns the number of subunits for which it has ReadWrite or ReadOnly authority. In the case of a SALD, every subunit is reported.

Specification (Normative):

```
PrimaryCommand GetSubunitListCommand {
      CommandCode_t         Command ← 0x0008
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit ← 0
      DataLength_t          DataLength ← 0
}
```

```
ALDResponse GetSubunitListResponse {
      CommandCode_t         Command ← 0x0008
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
            uint16_t                 NrOfVisibleSubunits
            SubunitTypeListElement_t Subunits[NrOfVisibleSubunits]
      }
      else {
            ALDState_t           ALDState
            ConnectionState_t    ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
uint8_t I

UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

result ← IsCommandAllowed( LIST{    OperatingConnectionState,
                                    RestrictedConnectionState},
                                    Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF
```

```
I ← 0
FOREACH N FROM 0 TO NrOfSubunits − 1 DO
        IF ALDType = MALD AND ActiveAuth[CurrentPort].Authority[N] = NoAccess THEN
                NEXT
        ENDIF
        Response.Subunits[I].Subunit ← N
        Response.Subunits[I].SubunitType ← Subunits[N].Type
        I ← I + 1
ENDFOR
Response.NrOfVisibleSubunits ← I
CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.8.10. Get Reset Cause

Description (Informative):

The ALD returns the cause of the last executed reset.

Specification (Normative):

```
Enumeration ResetCause_t : uint8_t {
      SWUpdate         ← 0
      Layer2Reset      ← 1
      InternalError    ← 2
      PowerUp          ← 3   // This covers both power up and power
                             // cycle
      MALDConfChanged ← 4
      Layer2Timeout    ← 5   // See Section 11.13. "Communication timeout"
}

PrimaryCommand GetResetCauseCommand {
      CommandCode_t          Command ← 0x0009
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength ← 0
}
```

```
ALDResponse GetResetCauseResponse {
      CommandCode_t          Command ← 0x0009
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
            ResetCause_t     ResetCause
            If (ResetCause = PowerUp
                  OR ResetCause = Layer2Reset
                  OR ResetCause = SWUpdate
                  OR ResetCause = MALDConfChanged) {
                  uint16_t Portnumber
            }
      }
      else {
            ALDState_t             ALDState
            ConnectionState_t      ConnectionState
      }
}
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

UNLESS Cmd.Subunit = 0 THEN
    RETURN InvalidSubunitNumber
    EXIT
ENDIF

result ← IsCommandAllowed(LIST{ OperatingConnectionState,
RestrictedConnectionState},
Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
    RETURN result.code
    EXIT
ENDIF

IF the last reset was due to a power on THEN
    RETURN OK, Reset Cause ← PowerUp and the port number at which the power on was processed
ELSEIF the last reset was due to a SW Update THEN
    RETURN OK, Reset Cause ← SWUpdate and the port number at which the download was processed
ELSEIF the last reset was due to layer 2 reset sent by a primary THEN
    RETURN OK, Reset Cause ← Layer2Reset and the port number at which the layer 2 reset was sent
ELSEIF the last reset was due to internal error THEN
    RETURN OK, Reset Cause ← InternalError

ELSEIF the last reset was due to MALD configuration changed THEN
RETURN OK, Reset Cause ← MALDConfigChanged and the port number at which MALD configuration was initiated
ELSEIF the last reset was due to communication timeout THEN
RETURN OK, Reset Cause ← Layer2Timeout
ENDIF

CommandExit(Cmd.Command, CurrentPort)
EXIT

NOTE: The list of reset causes may be extended by subunit type standards.

12.8.11. Get AISG Port DC Power Information

Description (Informative):

The ALD returns its DC power consumption information according to Section 10.4.2. "Definition of power modes".

Specification (Normative):

```
PrimaryCommand GetAISGPortDCPowerInformationCommand {
      CommandCode_t         Command ← 0x001D
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit ← 0
      DataLength_t          DataLength ← 0
}

ALDResponse GetAISGPortDCPowerInformationResponse {
      CommandCode_t         Command ← 0x001D
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
            uint16_t P1
            uint16_t P2
            uint16_t P3
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

UNLESS Cmd.Subunit = 0 THEN
    RETURN InvalidSubunitNumber
    EXIT
ENDIF

result ← IsCommandAllowed(LIST{   OperatingConnectionState,
    RestrictedConnectionState},
    Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
    RETURN result.code
    EXIT
ENDIF

RETURN OK, P1, P2 and P3 as defined in Section 10.4.2. “Definition of power modes”.
CommandExit(Cmd.Command, CurrentPort)
EXIT

12.8.12. Get Diagnostic Information

Description (Informative):

The response to this command shall provide useful additional vendor specific information about the cause of the active alarm. Providing such additional information is mandatory for the General Error alarm and optional for all other alarms.

Specification (Normative):

```
PrimaryCommand GetDiagnosticInformationCommand {
      CommandCode_t         Command ← 0x000B
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit
      DataLength_t          DataLength ← 2
      AlarmCode_t           Alarm
}

ALDResponse GetDiagnosticInformationResponse {
      CommandCode_t         Command ← 0x000B
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
            uint8_t         ProductSpecificDiagnosticCode
            uint8_t         LengthOfDiagnosticString
            UTF8String_t    DiagnosticString     // max 254 octets
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
      NotAuthorised
      OutOfRange
}
```

Primary specification (Normative):

ALD specification (Normative):

```
IF Cmd.Subunit is not one of 0..NrOfSubunits THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

IF ALDType = MALD
      AND ActiveAuth[CurrentPort].Authority[Cmd.Subunit] = NoAccess THEN
      RETURN NotAuthorised
      EXIT
ENDIF

result ← IsCommandAllowed( LIST{    OperatingConnectionState,
                                    RestrictedConnectionState,
                                    MALDConfigConnectionState,
                                    UploadConnectionState,
                                    DownloadConnectionState},
                                    Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF

IF the requested AlarmCode is not supported THEN
      RETURN OutOfRange
ELSEIF the requested AlarmCode is not raised for the requested subunit THEN
      RETURN OK, ProductSpecificDiagnosticCode = 0 and LengthOfDiagnosticString = 0
ELSE
      RETURN OK, the related ProductSpecificDiagnosticCode, the
LengthOfDiagnosticString and the corresponding DiagnosticString
ENDIF

CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.8.13. Set Subunit Type Standard Version

Description (Informative):

The ALD sets one common subunit type standard version (numbers a, b and c according to Chapter 14) for all subunits of the requested subunit type visible on a port.

Subunits do not have a default subunit type standard version, therefore the primary needs to set the subunit type standard version after every restart to get access to the subunits.

Specification (Normative):

```
PrimaryCommand SetSubunitTypeStandardVersionCommand {
      CommandCode_t          Command ← 0x000C
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength ← 4
      SubunitType_t          SubunitType
      AISGVersion_t          Version
}

ALDResponse SetSubunitTypeStandardVersionResponse {
      CommandCode_t          Command ← 0x000C
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t             ALDState
            ConnectionState_t      ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      InvalidSubunitType
      UnsupportedConfiguration
      IncorrectState
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

result ← IsCommandAllowed( LIST{    OperatingConnectionState,
                                    RestrictedConnectionState},
                                    Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF
```

FOREACH I FROM 1 TO NrOfSubunits DO

IF Subunits[I].Type = Cmd.SubunitType THEN

IF the requested subunit type standard version is not supported THEN
RETURN UnsupportedConfiguration
ELSE
ConfiguredSubunitTypeVersion[Cmd.CurrentPort] ← Cmd.Version
RETURN OK
ENDIF

CommandExit(Cmd.Command, CurrentPort)
EXIT

ENDIF

ENDFOR

RETURN InvalidSubunitType
CommandExit(Cmd.Command, CurrentPort)
EXIT

12.8.14. Get Subunit Type Standard Versions

Description (Informative):

The ALD returns currently configured subunit type standard version information (variables a, b and c according to Section 14.2. "Subunit type standard versions") for the subunit types visible on the port on which the command was received. These versions are used by all subunits of the same type.

Specification (Normative):

```
PrimaryCommand GetSubunitTypeStandardVersionsCommand {
      CommandCode_t         Command ← 0x000D
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit ← 0
      DataLength_t          DataLength ← 1}

ALDResponse GetSubunitTypeStandardVersionsResponse {
      CommandCode_t         Command ← 0x000D
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
            AISGVersion_t   ConfiguredVersion
            unit8_t         NrOfSupportedVersions
            AISGVersion_t   SupportedVersions[NrOfSupportedVersions]
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      InvalidSubunitType
      InvalidSubunitNumber
      IncorrectState
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

result ← IsCommandAllowed( LIST{      OperatingConnectionState,
                                      RestrictedConnectionState},
                          Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF

IF the requested subunitType is not supported THEN
      RETURN InvalidSubunitType
ELSE
      Response.ConfiguredVersion ← ConfiguredSubunitTypeVersion[Cmd.CurrentPort]
      Set Response.SupportedVersions TO the list of supported subunit type
      standard versions for the requested subunit type
      RETURN OK and the list of currently configured subunit standard version(s) of the
      subunit types(s) visible on the port that the command was received on
ENDIF

CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.8.15. Upload Info

Description (Informative):

The UploadInfo command requests information about the uploadable files available within the ALD.

Specification (Normative):

```
PrimaryCommand UploadInfoCommand {
      CommandCode_t         Command ← 0x003C
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit
      DataLength_t          DataLength ← 1
      FileType_t            FileType        // See Section 12.8.6. "Download Start"
}
```

```
ALDResponse UploadInfoResponse {
      CommandCode_t          Command ← 0x003C
      CommandSequece_t       PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
            uint32_t FileLength
      }
      else {
            ALDState_t             ALDState
            ConnectionState_t      ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
      UnsupportedFileType
}
```

Primary specification (Normative):

ALD specification (Normative):

IF Cmd.FileType is not supported THEN
    RETURN UnsupportedFileType
    EXIT
ENDIF

result ← IsCommandAllowed(LIST{ OperatingConnectionState},
Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
    RETURN result.code
    EXIT
ENDIF

RETURN OK and the file FileLength of the requested file type
CommandExit(Cmd.Command, CurrentPort)
EXIT

NOTE: Upload Info Command cannot be used while Upload is already started.

12.8.16. Upload Start

Description (Informative):

The UploadStart command initiates the upload process of the requested file type from the ALD.

The supported file types defined in this base standard are listed in Table 12.8.6-1: “Description of the file types”. Additional file types may be defined by subunit type standards.

How the file types may be used in download and upload is defined in Table 12.8.6-2: “Usage of the file types”.

Specification (Normative):

```
PrimaryCommand UploadStartCommand {
      CommandCode_t          Command ← 0x003D
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit
      DataLength_t           DataLength ← 1
      FileType_t             FileType        // See Section 12.8.6. "Download Start"
}

ALDResponse UploadStartResponse {
      CommandCode_t          Command ← 0x003D
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
            uint32_t FileLength
      }
      else {
            ALDState_t             ALDState
            ConnectionState_t      ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      UnsupportedFileType
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
IF the FileType is not supported THEN
      RETURN UnsupportedFileType
      EXIT
ENDIF

result ← IsStateChangeAllowed( LIST{ OperatingConnectionState},
                                     Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF

FOREACH PORT FROM 1 to MaxPort DO
      NEXT IF PORT = CurrentPort
      Switch ConnectionState[Port] to RestrictedConnectionState
ENDFOR
```

Switch ConnectionState[CurrentPort] to UploadConnectionState
PendingConnectionStateChange ← false
UNLOCK StateLock
UploadRemainingLength[CurrentPort] ← length(requested file)
UploadPosition[CurrentPort] ← 0
RETURN OK
Signal UploadFileEvent(CurrentPort)
CommandExit(Cmd.Command, CurrentPort)
EXIT

NOTE: The list of file type codes may be extended by subunit type standards.

12.8.17. Upload File

Description (Informative):

The UploadFile command transfers a block of file from the ALD to the primary.

All blocks except the last shall be 256 octets. The last block may not be 0 octets.

Specification (Normative):

```
ALDCommand UploadFileCommand {
      CommandCode_t         Command ← 0x003E
      CommandSequence_t     ALDCommandSequence
      Subunit_t             Subunit
      DataLength_t          DataLength
      uint8_t               Block[DataLength]
}
```

```
PrimaryResponse UploadFileResponse {
      CommandCode_t         Command ← 0x003E
      CommadSequence_t      ALDCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
      }
      else {
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      UploadRejected
}
```

Primary specification (Normative):

IF primary detects any problem storing the upload file block THEN
    RETURN UploadRejected
ENDIF

ALD specification (Normative):

ON UploadFileEvent(uint16_t PORT, UploadStatus_t Status)
    Cmd.result ← Event.Status
    LOCK StateLock

```
	UNLESS ConnectionState[PORT] = UploadConnectionState THEN
		// This is for the case when there is no upload in progress.
		Signal UploadEndEvent(PORT, failed)
		UNLOCK StateLock
		EXIT
	ENDIF
	UNLOCK StateLock
	IF UploadFileResponse(PORT) is out of sequence THEN
		Signal UploadEndEvent(PORT, failed)
		EXIT
	ENDIF
	IF UploadRemainingLength[PORT] = 0 THEN
		Signal UploadEndEvent(PORT, success)
		EXIT
	ENDIF
	IF UploadRemainingLength[PORT] > 256 THEN
		Store 256 octets from UploadPosition[PORT] in file in Cmd.Block
		Queue UploadFileCommand for transmission on port PORT
		Increase UploadPosition[PORT] by 256
		Decrease UploadRemainingLength[PORT] by 256
	ELSE
		Store UploadRemainingLength[PORT] octets from
			UploadPosition[PORT] in file in Cmd.Block
		Queue UploadFileCommand for transmission on port PORT
		UploadRemainingLength[PORT] ← 0
	ENDIF
DONE
```

12.8.18. Upload End

Description (Informative):

This command indicates successful completion of the upload process.

Specification (Normative):

```
ALDCommand UploadEndCommand {
	CommandCode_t		Command ← 0x003F
	CommandSequence		ALDCommandSequence
	Subunit_t		Subunit
	DataLength_t		DataLength ← 1
	Boolean			UploadSuccessful
}

PrimaryResponse UploadEndResponse {
	CommandCode_t		Command ← 0x003F
	CommandSequence		ALDCommandSequence
	ReturnCode_t		ReturnCode ← OK
	DataLength_t		DataLength ← 0
}
```

Primary specification (Normative):

RETURN OK

ALD specification (Normative):

ON UploadEndEvent(PORT) DO

IF UploadRemainingLength[PORT] = 0 THEN
UploadSuccessful ← True
ELSE
UploadSuccessful ← False
ENDIF

Queue UploadEnd for transmission on port PORT
LOCK StateLock
PendingConnectionStateChange ← true

IF CommandCount > 0 THEN
UNLOCK StateLock
WAIT UNTIL CommandCount = 0
LOCK StateLock
END

Switch ConnectionState[PORT] to OperatingConnectionState
PendingConnectionStateChange ← false
UNLOCK StateLock

DONE

12.8.19. Send Layer 1 Test Pattern

Description (Informative):

The SendLayer1TestPattern command is used to transmit test patterns for the specified time so that the signal levels, emission levels and BER on the OOK and the RS-485 ports can be measured.

While the ALD is transmitting the test pattern it shall ensure that the communication timeout timer does not expire.

During the test transmission period regular AISG communication on the AISG bus is not possible. The only way to interrupt the test transmission is to perform DC power cycle.

Concurrent execution of SendLayer1TestPattern is allowed on more than one AISG port.

Specification (Normative):

```
Enumeration PatternType_t : uint8_t {
      CW                   ← 0   // This test signal is not useful for RS-485
      Octet                ← 1
      PseudoRandomCode     ← 2
}
```

```
PrimaryCommand SendLayer1TestPatternCommand {
      CommandCode_t          Command ← 0x00B1
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength
      PatternType_t          Type
      uint32_t               Time // Up to ca 49710 days
      if (Type == CW) {
      }
      elseif (Type == Octet) then {
            uint8_t          TestPattern
      }
      elseif (Type == PseudoRandomCode) then {
            uint16_t         Seed
      }
}

ALDResponse SendLayer1TestPatternResponse {
      CommandCode_t          Command ← 0x00B1
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
}
```

Primary Specification (Normative):

ALD Specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

result ← IsCommandAllowed( LIST{ OperatingConnectionState},
                                 Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF
```

RETURN OK

IF Type = CW THEN
    Send continuous logical 0 for Time seconds
ELSEIF Type = Octet THEN
    Send a continuous stream of TestPattern octets for Time seconds
ELSE
Send the pseudo-random test pattern generated from Seed for Time seconds according to [10] Section 2.1.
ENDIF

CommandExit(Cmd.Command, CurrentPort)
EXIT

12.8.20. Generate Test Alarm

Description (Informative):

The ALD generates a test alarm when it receives this command with an alarm severity other than Cleared.

When an ALD receives this command with alarm severity Cleared, the ALD will clear the alarm indicated by the Alarm Code.

The test alarm works exactly the same way as a real alarm, except that alarms do not clear automatically. Normal alarm behaviour is defined in Section 12.6.1. “Alarm handling”

NOTE: ClearActiveAlarms command cancels all alarms.

Specification (Normative):

The Alarm severity state is defined in [8].

```
PrimaryCommand GenerateTestAlarmCommand {
      CommandCode_t         Command ← 0x00B2
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit
      DataLength_t          DataLength
      AlarmCode_t           AlarmCode
      Severity_t            Severity
      uint8_t               Length
      UTF8String_t          Text
}
```

```
ALDResponse GenerateTestAlarmResponse {
      CommandCode_t         Command ← 0x00B2
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      InvalidSubunitNumber
      IncorrectState
      NoAlarmSubscription
}
```

Primary specification (Normative):

ALD specification (Normative):

```
IF Cmd.Subunit is not one of 0..NrOfSubunits THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

RETURN OK
IF AlarmSubscribeFlag[CurrentPort] = true THEN
      IF Cmd. Severity = Cleared THEN
            CLEAR Cmd.AlarmCode ON Cmd.Subunit
      ELSE
            RAISE Cmd.AlarmCode SEVERITY Cmd.Severity ON Cmd.Subunit, Cmd.Text
      ENDIF
ELSE
      RETURN NoAlarmSubscription
ENDIF

CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.8.21. Vendor specific command

Description (Informative):

This command code is reserved to allow the addition of vendor-specific functionality (for example production testing). Only the structure of the command is defined here. Vendor-specific commands shall not be used to work around possible problems within this standard. In particular, the vendor should check the subunit and (if applicable) the configured MALD authority.

NOTE: The ALD may support vendor-specific commands defined by other vendor(s).

Specification (Normative):

```
PrimaryCommand VendorSpecificCommand {
      CommandCode_t         Command ← 0x0090
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit
      DataLength_t          DataLength
      AsciiString_t         VendorCode[2]
      // more data defined by the vendor
}
```

```
ALDResponse VendorSpecificResponse {
      CommandCode_t          Command ← 0x0090
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      AsciiString_t          VendorCode[2]
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      InvalidSubunitNumber
      OutOfRange
      //more ReturnCode_ts as defined by the vendor
}
```

Primary specification (Normative):

ALD specification (Normative):

// Behaviour is defined by the vendor

12.9. MALD commands

12.9.1. MALD Download Initiated

Description (Informative):

The MALD sends this command to notify the other connected primaries that one primary has initiated a download of a file.

Specification (Normative):

```
ALDCommand MALDDownloadInitiatedCommand {
      CommandCode_t          Command ← 0x0013
      CommandSequence_t      ALDCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength ← 0
}
```

```
PrimaryResponse MALDDownloadInitiatedResponse {
      CommandCode_t          Command ← 0x0013
      CommandSequence_t      ALDCommandSequence
      ReturnCode_t           ReturnCode ← OK
      DataLength_t           DataLength ← 0
}
```

Primary specification (Normative):

Send DownloadInitiatedResponse
Try to re-establish the layer 2 link to the MALD
EXIT

ALD specification (Normative):

ON receipt of the MALDDownloadInitiatedResponse(PORT) DO

UNLESS ConnectionState[PORT] = DownloadNotificationConnectionState THEN
EXIT
ENDIF

Decrement MALDDownloadInitiatedResponseCounter by 1
Switch ConnectionState[PORT] to OffConnectionState
Switch LinkState[PORT] to NoAddress
Disable serial port PORT

IF MALDDownloadInitiatedResponseCounter = 0 THEN
Signal StartDownloadEvent
ENDIF

DONE

12.9.2. MALD Get Information

Description (Informative):

The ALD provides information about MALD physical organisation. The ALD returns the following:

- configuration commit counter value
- number of control ports within the MALD
- list of control port numbers
- number of subunits within the MALD
- list of subunit number and type tuples

Specification (Normative):

```
PrimaryCommand MALDGetInformationCommand {
      CommandCode_t         Command ← 0x0014
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit ← 0
      DataLength_t          DataLength ← 0
}

ALDResponse MALDGetInformationResponse {
      CommandCode_t         Command ← 0x0014
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
            uint16_t        MALDConfCommitCounter
            uint16_t        NrOfMALDControlPorts
            uint16_t        MALDControlPortNumber[NrOfMALDControlPorts]
            uint16_t        NrOfSubunits
            SubunitTypeListElement_t  Subunits[NrOfSubunits]
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      UnknownCommand
      Busy
      IncorrectState
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

result ← IsCommandAllowed( LIST{   OperatingConnectionState,
                                   RestrictedConnectionState,
                                   MALDConfigConnectionState},
                                   Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF

RETURN OK, configuration commit counter value, number of control ports, list of control ports numbers, number of subunits, the list of subunits including subunit number and subunit type.
CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.9.3. MALD Start Conf

Description (Informative):

The ALD copies the active configuration to the volatile configuration copy and enters the MALD Config State. The control port used to issue the command is stored as the transaction port and the 5-minute transaction timeout timer is started.

Specification (Normative):

```
PrimaryCommand MALDStartConfCommand {
      CommandCode_t         Command ← 0x0018
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit ← 0
      DataLength_t          DataLength ← 2
      uint16_t              MALDConfCommitCounter
}
```

```
ALDResponse MALDStartConfResponse {
      CommandCode_t         Command ← 0x0018
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      UnknownCommand
      IncorrectCommitCounter
      NotAuthorised
      IncorrectState
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

IF ActiveAuth.[CurrentPort].Config = NotAllowed THEN
      RETURN NotAuthorised
      EXIT
ENDIF

UNLESS Msg.MALDConfCommitCounter = ActiveCommitCounter THEN
      RETURN IncorrectCommitCounter
      EXIT
ENDIF

result ← IsStateChangeAllowed( LIST{ OperatingConnectionState},
                                     Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF

VolatileAuth ← ActiveAuth
Switch ALDState to MALDConfigState

FOREACH PORT FROM 1 TO MaxPort DO
      NEXT IF PORT = CurrentPort
      Switch ConnectionState[Port] to RestrictedConnectionState
ENDFOR
```

Switch ConnectionState[CurrentPort] to MALDConfigConnectionState
PendingConnectionStateChange ← false
UNLOCK StateLock
RETURN OK
CommandExit(Cmd.Command, CurrentPort)
EXIT

12.9.4. MALD Commit Conf

Description (Informative):

The ALD validates the configuration in the volatile copy and if validation is successful:

- Copies the volatile MALD configuration to the active configuration
- Sets the transaction state to inactive
- Increments the MALD conf commit counter by 1 (by 2 if it was 65535)
- Activates the new configuration
- Performs a reset

Specification (Normative):

```
PrimaryCommand MALDCommitConfCommand {
      CommandCode_t          Command ← 0x0019
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength ← 0
}

ALDResponse MALDCommitConfResponse {
      CommandCode_t          Command ← 0x0019
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t             ALDState
            ConnectionState_t      ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      UnknownCommand
      UnsupportedConfiguration
      InvalidSubunitNumber
      GeneralError
}
```

Primary specification (Normative):

ALD specification (Normative):

```
uint16_t COUNT
UNLESS Cmd.Subunit = 0 THEN
        RETURN InvalidSubunitNumber
        EXIT
ENDIF

result ← IsStateChangeAllowed( LIST{ MALDConfigConnectionState},
                                     Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
        RETURN result.code
        EXIT
ENDIF

IF VolatileAuth[CurrentPort].Config = NotAllowed
        OR supplied configuration is unsupported THEN
        RETURN UnsupportedConfiguration
        Switch ALDState to OperatingState
        Switch ConnectionState[1..MaxPort] to OperatingConnectionState
        PendingConnectionStateChange ← false
        UNLOCK StateLock
        CommandExit(Cmd.Command, CurrentPort)
EXIT
ENDIF

FOREACH N FROM 1 TO NrOfSubunits DO
        COUNT ← 0

        FOREACH PORT FROM 1 TO MaxPort DO
                IF VolatileAuth[PORT].Authority[N] = ReadWrite THEN
                        COUNT ← COUNT + 1
                ENDIF
        ENDFOR

        IF COUNT > 1 THEN
                RETURN UnsupportedConfiguration
                CommandExit(Cmd.Command, CurrentPort)
                EXIT
        ENDIF
ENDFOR

IF ActiveCommitCounter = 65535 THEN
        VolatileCommitCounter ← 1
ELSE
        VolatileCommitCounter ← ActiveCommitCounter + 1
ENDIF

Store VolatileAuth in ActiveAuth
Store VolatileCommitCounter in ActiveCommitCounter
```

```
IF the ALD detects a hardware error THEN
    // Replace "Hardware error" with a text describing the problem
    RAISE AlarmGeneralError SEVERITY Major ON ALD, "Hardware error"
    RETURN GeneralError
    Switch ALDState to OperatingState
    Switch ConnectionState[1..MaxPort] to OperatingConnectionState
    PendingConnectionStateChange ← false
    UNLOCK StateLock
    CommandExit(Cmd.Command, CurrentPort)
    EXIT
ENDIF

RETURN OK
Wait for layer 2 acknowledgement (RR) from the primary
Perform a reset
EXIT
```

12.9.5. MALD Abort Conf

Description (Informative):

The ALD discards the contents of the volatile configuration, exits the MALD Config State and returns to the OperatingState.

Specification (Normative):

```
PrimaryCommand MALDAbortConfCommand {
      CommandCode_t         Command ← 0x001A
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit ← 0
      DataLength_t          DataLength ← 0
}

ALDResponse MALDAbortConfResponse {
      CommandCode_t         Command ← 0x001A
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      UnknownCommand
      IncorrectState
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
        RETURN InvalidSubunitNumber
        EXIT
ENDIF

result ← IsStateChangeAllowed( LIST{  MALDConfigConnectionState},
                                      Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
        RETURN result.code
        EXIT
ENDIF

RETURN OK
Switch ALDState to OperatingState
Switch ConnectionState[1..MaxPort] to OperatingConnectionState
PendingConnectionStateChange ← false
UNLOCK StateLock
CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.9.6. MALD Reset Conf

Description (Informative):

The ALD re-initialises all configurable authorities within the MALD to the default configuration.

Specification (Normative):

```
PrimaryCommand MALDResetConfCommand {
      CommandCode_t          Command ← 0x0017
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength ← 0
}

ALDResponse MALDResetConfResponse {
      CommandCode_t          Command ← 0x0017
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t             ALDState
            ConnectionState_t      ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      UnknownCommand
      InvalidSubunitNumber
      IncorrectState
      GeneralError
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

result ← IsStateChangeAllowed( LIST{   MALDConfigConnectionState},
                                       Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF

FOREACH PORT FROM 1 TO MaxPort DO
      VolatileAuth[PORT].Config = Allowed
      VolatileAuth[PORT].SWDownload = Allowed

      FOREACH SUBUNIT FROM 1 TO NrOfSubunits DO
            VolatileAuth[PORT].Authority[SUBUNIT] = ReadWrite
      ENDFOR

ENDFOR

VolatileCommitCounter ← 0
Store VolatileAuth in ActiveAuth
Store VolatileCommitCounter in ActiveCommitCounter

IF the ALD detects a hardware error THEN
      // Replace “Hardware error” with a text describing the problem
      RAISE AlarmGeneralError SEVERITY Major ON ALD, “Hardware error”
      RETURN GeneralError
      Switch ALDState to OperatingState
      Switch ConnectionState[1..MaxPort] to OperatingConnectionState
      PendingConnectionStateChange ← false
      UNLOCK StateLock
      CommandExit(Cmd.Command, CurrentPort)
      EXIT
ENDIF
```

RETURN OK
Wait for layer 2 acknowledgement (RR) from the primary
Perform a reset
EXIT

12.9.7. MALD Set Subunit Conf

Description (Informative):

The ALD modifies the control authority of the specified configuration target subunit for the specified configuration target control port (that is, for the primary connected to that port).

This is stored in the volatile configuration copy.

Specification (Normative):

```
PrimaryCommand MALDSetSubunitConfCommand {
      CommandCode_t           Command ← 0x0015
      CommandSequence_t       PrimaryCommandSequence
      Subunit_t               Subunit ← 0
      DataLength_t            DataLength ← 5
      Subunit_t               ConfTargetSubunit
      uint16_t                ConfTargetPortNumber
      AuthorityType_t         AuthorityValue
}

ALDResponse MALDSetSubunitConfResponse {
      CommandCode_t           Command ← 0x0015
      CommandSequence_t       PrimaryCommandSequence
      ReturnCode_t            ReturnCode
      DataLength_t            DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t              ALDState
            ConnectionState_t       ConnectionState
      }
}

Enumeration ReturnCode_t{
      OK
      FormatError
      Busy
      UnknownCommand
      IncorrectState
      InvalidSubunitNumber
      InvalidAuthorityValue
      InvalidConfTargetSubunitNumber
      InvalidConfTargetPortNumber
      NotControlCapablePort
}
```

Primary specification (Normative):

ALD specification (Normative):

IF Cmd.Subunit ≠ 0 THEN
    RETURN InvalidSubunitNumber
    EXIT
ENDIF

```
IF Cmd.ConfTargetSubunit is not one of 1..NrOfSubunits THEN
        RETURN InvalidConfTargetSubunitNumber
        EXIT
ENDIF

IF Cmd.ConfTargetPortNumber is not one of 1..MaxPort THEN
        RETURN InvalidConfTargetPortNumber
        EXIT
ENDIF

UNLESS PortProperties[Cmd.ConfTargetPortNumber] = RS485ControlPort
        OR PortProperties[Cmd.ConfTargetPortNumber] = RFControlPort THEN
        RETURN NotControlCapablePort
        EXIT
ENDIF IF Cmd.AuthorityValue is not one of AuthorityType THEN
        RETURN InvalidAuthorityValue
        EXIT
ENDIF

result ← IsCommandAllowed( LIST{      MALDConfigConnectionState},
                                      Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
        RETURN result.code
        EXIT
ENDIF

VolatileAuth[Cmd.ConfTargetPortNumber].Authority[Cmd.ConfTargetSubunit] ←
Cmd.AuthorityValue
RETURN OK
CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.9.8. MALD Get Subunit Conf

Description (Informative):

The ALD responds with the control authority off the specified configuration target subunit for the specified configuration target control port (that is, for the primary connected to that port).

Specification (Normative):

```
PrimaryCommand MALDGetSubunitConfCommand {
      CommandCode_t          Command ← 0x0016
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength ← 5
      Subunit_t              ConfTargetSubunit
      uint16_t               ConfTargetPortNumber
      SettingSourceType_t    SettingSourceType
}
```

```
ALDResponse MALDGetSubunitConfResponse {
      CommandCode_t         Command ← 0x0016
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
            AuthorityType_t AuthorityValue
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      UnknownCommand
      TransactionNotInProgress
      InvalidSubunitNumber
      InvalidSettingSource
      IncorrectState
      InvalidConfTargetSubunitNumber
      InvalidConfTargetPortNumber
      NotControlCapablePort
}
```

Primary specification (Normative):

ALD specification (Normative):

```
IF Cmd.Subunit ≠ 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

IF Cmd.ConfTargetSubunit is not one of 1..NrOfSubunits THEN
      RETURN InvalidConfTargetSubunitNumber
      EXIT
ENDIF

IF Cmd.PortNumber is not one of 1..MaxPort THEN
      RETURN InvalidPortNumber
      EXIT
ENDIFUNLESS PortProperties[Cmd.ConfTargetPortNumber] = RS485ControlPort
      OR PortProperties[Cmd.ConfTargetPortNumber] = RFControlPort THEN
      RETURN NotControlCapablePort
      EXIT
ENDIF

result ← IsCommandAllowed( LIST{    OperatingConnectionState,
                                    RestrictedConnectionState,
                                    MALDConfigConnectionState},
                                    Cmd.Command, CurrentPort)
```

```
UNLESS result.allowed THEN
        RETURN result.code
        EXIT
ENDIF
CASE Cmd.SettingSourceType IS
        WHEN Volatile:
                IF ConnectionState[CurrentPort] ≠ MALDConfigState THEN
                        RETURN TransactionNotInProgress
                ENDIF
                RETURN OK and VolatileAuth[Cmd.PortNumber].Authority[Cmd.Subunit]
        WHEN Active:
                RETURN OK and ActiveAuth[Cmd.PortNumber].Authority[Cmd.Subunit]
        OTHERWISE
                RETURN InvalidSettingSource
ENDCASE
CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.9.9. MALD Set Security Setting

Description (Informative):

The ALD modifies the specified security setting of the specified control port (that is for the connected primary) in the volatile configuration copy.

Specification (Normative):

```
PrimaryCommand MALDSetSecuritySettingCommand {
     CommandCode_t          Command ← 0x001B
     CommandSequence_t      PrimaryCommandSequence
     Subunit_t              Subunit ← 0
     DataLength_t           DataLength ← 4
     uint16_t               PortNumber
     SecurityType_t         SecurityType
     SecuritySetting_t           SecuritySetting
}
ALDResponse MALDSetSecuritySettingResponse {
     CommandCode_t          Command ← 0x001B
     CommandSequence_t      PrimaryCommandSequence
     ReturnCode_t           ReturnCode
     DataLength_t           DataLength
     if (ReturnCode == OK) {
     }
     else {
          ALDState_t             ALDState
          ConnectionState_t      ConnectionState
     }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      UnknownCommand
      OutOfRange
      IncorrectState
      InvalidSubunitNumber
      InvalidPortNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

IF Cmd.PortNumber is not one of 1..MaxPort THEN
      RETURN InvalidPortNumber
      EXIT
ENDIF

IF Cmd.SecuritySetting is not one of SecuritySetting THEN
      RETURN OutOfRange
      EXIT
ENDIF

result ← IsCommandAllowed( LIST{     MALDConfigConnectionState},
                                     Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF

CASE Cmd.SecurityType IS
      WHEN MALDConfiguration:
            VolatileAuth[Cmd.PortNumber].Config ← Cmd.SecuritySetting
      WHEN SWDownload:
            VolatileAuth[Cmd.PortNumber].SWDownload ← Cmd.SecuritySetting
      OTHERWISE
            RETURN UnsupportedSecuritySetting
ENDCASE

RETURN OK
CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.9.10. MALD Get Security Setting

Description (Informative):

The ALD responds with the specified MALD configuration security setting of the specified control port (that is for the connected primary).

Specification (Normative):

```
PrimaryCommand MALDGetSecuritySettingCommand {
      CommandCode_t          Command ← 0x001C
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength ← 34
      uint16_t               PortNumber
      SettingSource_t        SettingSourceType
}

ALDResponse MALDGetSecuritySettingResponse {
      CommandCode_t          Command ← 0x001C
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
            SecuritySetting_t    MALDConfigSecuritySetting
            SecuritySetting_t    SWDownloadSecuritySetting
      }
      else {
            ALDState_t           ALDState
            ConnectionState_t    ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      UnknownCommand
      InvalidSettingSource
      InvalidPortNumber
      InvalidSubunitNumber
      IncorrectState
}
```

Primary specification (Normative):

ALD specification (Normative):

UNLESS Cmd.Subunit = 0 THEN
    RETURN InvalidSubunitNumber
    EXIT
ENDIF

IF Cmd.PortNumber is not one of 1..MaxPort THEN
    RETURN InvalidPortNumber
    EXIT
ENDIF

```
result ← IsCommandAllowed( LIST{     OperatingConnectionState,
                                     RestrictedConnectionState,
                                     MALDConfigConnectionState},
                                     Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
    RETURN result.code
    EXIT
ENDIF

CASE SettingSourceType IS
    WHEN Volatile:
        Response.Config ← VolatileAuth[Cmd.PortNumber].Config
        Response.SWDownload ← VolatileAuth[Cmd.PortNumber].SWDownload
        RETURN OK
    WHEN Active:
        Response.Config ← ActiveAuth[Cmd.PortNumber].Config
        Response.SWDownload ← ActiveAuth[Cmd.PortNumber].SWDownload
        RETURN OK
    OTHERWISE
        RETURN InvalidSettingSource
ENDCASE

CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.10. Site mapping commands

12.10.1. Get Number Of Ports

Description (Informative):

The ALD responds with its total number of ports regardless of their functionality (RF, AISG control, power supply …).

Specification (Normative):

```
PrimaryCommand GetNumberOfPortsCommand {
     CommandCode_t          Command ← 0x001E
     CommandSequence_t      PrimaryCommandSequence
     Subunit_t              Subunit ← 0
     DataLength_t           DataLength ← 0
}
```

```
ALDResponse GetNumberOfPortsResponse {
      CommandCode_t          Command ← 0x001E
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
            uint16_t   NrOfPorts ← MaxPort + 1
      }
      else if {
            ALDState_t              ALDState
            ConnectionState_t       ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

result ← IsCommandAllowed( LIST{   OperatingConnectionState,
                                   RestrictedConnectionState,
                                   MALDConfigConnectionState},
                                   Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF

RETURN OK and number of ALD's ports
CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.10.2. Get Port Info

Description (Informative):

The ALD provides, for the specified port, its properties, direction and the subunits associated with the port regardless authority setting.

Specification (Normative):

Port Property	Description
Direction	The direction of the port is indicated by the value: 0 means the port connects towards the antenna, 1 means the port connects towards the base station.
RS-485	The port is an RS-485 port.
RF	The port is an RF port.
Control	A layer 2 link can be established to this ALD through this port.
Ping	The port is able to send or receive a Ping message.
OOK	The port is an OOK port.
DC-IN	The ALD can be supplied with DC through this port.
DC-OUT	The port is able to supply DC.

Table 12.10.2-1: Description of Port Properties

Port Properties Type	Direction Towards	RF	Ping	DC OUT	DC IN	RS 485	OOK	Control
RFAntPort	Antenna	✓						
RFAntPingPort	Antenna	✓	✓					
RFAntOOKPort	Antenna	✓		✓			✓	
RS485OutPort	Antenna			✓		✓		
RFBTSPort	Base station	✓						
RFBTSPingPort	Base station	✓	✓					
RFBTSOOKPort	Base station	✓			✓		✓	
RS485InPort	Base station				✓	✓		
RS485ControlPort	Base station				✓	✓		✓
RFControlPort	Base station	✓			✓		✓	✓

Table 12.10.2-2: List of allowed PortPropertiesType and their composition

NOTE: The RF port on an antenna has the direction value 1 (TowardsBasestation). The RF port of base station has the direction value 0 (TowardsAntenna).

```
Enumeration Provenance_t : uint8_t {
      NotSet    ← 0  // is not set
      Factory   ← 1  // is factory set
      File      ← 2  // is set by a configuration file
      Automatic ← 3  // set by the ALD automatically
      Manual    ← 4  // manually set by the user
}

CONSTANT uint8_t TowardsAntenna     ← 0
CONSTANT uint8_t TowardsBasestation ← 1

Enumeration PortPropertyMasks_t : uint8_t {
      Direction ← 00000001B    // 1
      RF        ← 00000010B    // 2
      Ping      ← 00000100B    // 4
      DC_OUT    ← 00001000B    // 8
      DC_IN     ← 00010000B    // 16
      RS_485    ← 00100000B    // 32
      OOK       ← 01000000B    // 64
      Control   ← 10000000B    // 128
}
```

```
Enumeration PortPropertiesType_t : uint8_t {
      RFAntPort        ← TowardsAntenna + RF                // 00000010B (2)
      RFAntPingPort    ← TowardsAntenna + RF + Ping    // 00000110B (6)
      RFAntOOKPort          ← TowardsAntenna + DC_OUT + RF + OOK
                                                          // 01001010B (74)
      RS485OutPort          ← TowardsAntenna + DC_OUT + RS_485   // 00101000B (40)
      RFBTSPort        ← TowardsBaseStation + RF       // 00000011B (3)
      RFBTSPingPort    ← TowardsBaseStation + RF + Ping     // 00000111B
                                                               // (7)
      RFBTSOOKPort          ← TowardsBaseStation + DC_IN + RF + OOK
                                                          // 01010011B (83)
      RS485InPort           ← TowardsBaseStation + DC_IN + RS_485
                                                          // 00110001B (49)
      RS485ControlPort      ← Control + TowardsBaseStation + DC_IN + RS_485
                                                          // 10110001B (177)
      RFControlPort    ← Control + TowardsBaseStation + DC_IN + RF + OOK
                                                     // 11010011B (211)
}
PortPropertiesType_t PortProperties[MaxPort+1]

PrimaryCommand GetPortInfoCommand {
      CommandCode_t          Command ← 0x001F
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength ← 2
      uint16_t               PortNumber
}

ALDResponse GetPortInfoResponse {
      CommandCode_t          Command ← 0x001F
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
            PortPropertiesType_t PortProperties
            Provenance_t         PortPropertiesProvenance
            uint16_t             NrOfAssociatedSubunits
            uint16_t   AssociatedSubunits[NrOfAssociatedSubunits]
            }
            Provenance_t         AssociatedSubunitProvenance
      }
      else {
            ALDState_t           ALDState
            ConnectionState_t    ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidPortNumber
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
	RETURN InvalidSubunitNumber
	EXIT
ENDIF

IF Cmd.PortNumber is not one of 1..MaxPort THEN
	RETURN InvalidPortNumber
	EXIT
ENDIF

result ← IsCommandAllowed( LIST{	OperatingConnectionState,
					RestrictedConnectionState,
					MALDConfigConnectionState},
					Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
	RETURN result.code
	EXIT
ENDIF

RETURN OK, PortProperties[Cmd.PortNumber] with the corresponding provenances, the list
of subunit numbers associated with the port with the corresponding provenances
CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.10.3. Get RF Port Frequency Info

Description (Informative):

The ALD provides the frequency information for the specified RF port.

Specification (Normative):

```
PrimaryCommand GetRFPortFrequencyInfoCommand {
	CommandCode_t		Command ← 0x0025
	CommandSequence_t	PrimaryCommandSequence
	Subunit_t		Subunit ← 0
	DataLength_t		DataLength ← 2
	uint16_t		PortNumber
}
```

```
ALDResponse GetRFPortFrequencyInfoResponse {
      CommandCode_t           Command ← 0x0025
      CommandSequence_t       PrimaryCommandSequence
      ReturnCode_t            ReturnCode
      DataLength_t            DataLength
      uint16_t                PortNumber
      if (ReturnCode == OK) {
            Frequencies_t     PortFrequencies        // See Section 13. "BINARY BASED
                                                      // FREQUENCY CODING"
            Provenance_t      PortFrequenciesProvenance
      }
      else {
            ALDState_t             ALDState
            ConnectionState_t      ConnectionState
      }
}
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidPortNumber
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

IF Cmd.PortNumber is not one of 1..MaxPort OR
      PortProperties of Cmd.PortNumber ≠ RF THEN
      RETURN InvalidPortNumber
      EXIT
ENDIF

result ← IsCommandAllowed( LIST{    OperatingConnectionState,
                                    RestrictedConnectionState,
                                    MALDConfigConnectionState},
                                    Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
RETURN result.code
      EXIT
ENDIF
```

RETURN OK, the port frequency information and the corresponding provenances
CommandExit(Cmd.Command, CurrentPort)
EXIT

12.10.4. Get Port Interconnections

Description (Informative):

The ALD provides information about interconnections between ports within an ALD.

NOTE: The interconnection must be reported symmetrically i.e. by both endpoint ports of the interconnection.

Specification (Normative):

Interconnection Type	Description
AISGCom	AISG communication is connected between two RS-485 ports or between an OOK port and an RS-485 port within an ALD.
OOKBypass	OOK signal is connected between two RF ports within an ALD, bypassing a functionality that does not support OOK transmission. If an ALD supports the ping process, the OOKBypass is deactivated during the ping process.
RF	RF signal is connected between two RF ports within an ALD. This interconnection may contain devices such as amplifiers or filters.
DC	DC is connected between two DC ports within an ALD.

Table 12.10.4-1: Description of Interconnection Types

```
Bitfield InterconnectionType_t : uint8_t {
      InterAISGCom          : Bit 0
      InterOOKBypass        : Bit 1
      InterRF               : Bit 2
      InterDC               : Bit 3
}

PrimaryCommand GetPortInterconnectionsCommand {
      CommandCode_t         Command ← 0x0020
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit ← 0
      DataLength_t          DataLength ← 2
      uint16_t              PortNumber
}

ALDResponse GetPortInterconnectionsResponse {
      CommandCode_t         Command ← 0x0020
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
            uint16_t        NrOfConnectedPorts
            uint16_t        ConnectedPortNumber[NrOfConnectedPorts]
            InterconnectionType_t PortConnections[NrOfConnectedPorts]
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
      InvalidPortNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

UNLESS Cmd.Subunit = 0 THEN
    RETURN InvalidSubunitNumber
    EXIT
ENDIF

IF Cmd.PortNumber is not one of 1..MaxPort THEN
    RETURN InvalidPortNumber
    EXIT
ENDIF

result ← IsCommandAllowed(LIST{ OperatingConnectionState,
    RestrictedConnectionState,
    MALDConfigConnectionState},
    Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
    RETURN result.code
    EXIT
ENDIF

IF Cmd.PortNumber has at least one interconnected Port THEN
    RETURN OK, the requested PortNumber, number of interconnected ports, the currently interconnected ports and the type of interconnection with corresponding interconnection type
ELSE
    RETURN OK, the number of interconnected ports
ENDIF

CommandExit(Cmd.Command, CurrentPort)
EXIT

12.10.5. Set RF Path IDs

Description (Informative):

The ALD stores the list of RF Path IDs to the specified port. The ALD has a separate RF Path IDs list on each port for each primary.

Specification (Normative):

NOTE: After any antenna line configuration change, the mapping of the RF Path ID Alias must be revalidated and possibly regenerated.

```
PrimaryCommand SetRFPathIDsCommand {
      CommandCode_t          Command ← 0x0021
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength
      uint16_t               PortNumber
      uint8_t                NrOfRFPathIDs
      uint16_t               RFPathIDs[NrOfRFPathIDs]
}
```

```
ALDResponse SetRFPathIDsResponse {
      CommandCode_t          Command ← 0x0021
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t             State
            ConnectionState_t      ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
      InvalidPortNumber
      TooManyArguments
      IncorrectPortType
      GeneralError
}
```

Primary specification (Normative):

ALD specification (Normative):

UNLESS Cmd.Subunit = 0 THEN
    RETURN InvalidSubunitNumber
    EXIT
ENDIF

IF Cmd.PortNumber is not one of 1..MaxPort THEN
    RETURN InvalidPortNumber
    EXIT
ENDIF

IF (InterconnectionType[Cmd.PortNumber] bitwise AND InterRF) ≠ InterRF THEN
    RETURN InvalidPortNumber
    EXIT
ENDIF

```
IF Cmd. NrOfRFPathIDs > 6 THEN
	RETURN TooManyArguments
	EXIT
ELSEIF (PortProperties[Cmd.PortNumber] bitwise AND RF) ≠ RF THEN
	RETURN IncorrectPortType
	EXIT
ENDIF

result ← IsCommandAllowed( LIST{	OperatingConnectionState},
					Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
	RETURN result.code
	EXIT
ENDIF

Store the RF path IDs for the supplied PortNumber and Cmd.PortNumber to non-volatile
memory
IF the ALD detects a hardware error THEN
	// Replace "Hardware error" with a text describing the problem
	RAISE AlarmGeneralError SEVERITY Major ON ALD, "Hardware error"
	RETURN GeneralError
ELSE
	RETURN OK
ENDIF

CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.10.6. Set RF Path ID Alias

Description (Informative):

The ALD stores the list of RF Path ID Alias to the specified port. The ALD has a separate RF Path ID Alias list on each port for each primary. This alias may be used to give a user-friendly description of the RF path identified by RFPathID.

Specification (Normative):

```
PrimaryCommand SetRFPathIDAliasCommand {
	CommandCode_t		Command ← 0x0022
	CommandSequence_t	PrimaryCommandSequence
	Subunit_t		Subunit ← 0
	DataLength_t		DataLength
	uint16_t		RFPathID
	uint8_t			LengthOftheRFPathAlias
	UTF8String_t		Alias			// max 32 octets
}
```

```
ALDResponse SetRFPathIDAliasResponse {
      CommandCode_t         Command ← 0x0022
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}
```

```
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
      OutOfRange
      InvalidRFPathID
      GeneralError
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

IF Cmd.RFPathID is out of range THEN
      RETURN InvalidRFPathID
      EXIT
ENDIF

IF Cmd.LengthOftheRFPathAlias > 32 THEN
      RETURN OutOfRange
      EXIT
ENDIF

result ← IsCommandAllowed( LIST{      OperatingConnectionState},
                                      Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF
```

Store the RF path ID Alias for the supplied RFPathID and Cmd.PortNumber to non-volatile memory

IF the ALD detects a hardware error THEN
    // Replace "Hardware error" with a text describing the problem
    RAISE AlarmGeneralError SEVERITY Major ON ALD, "Hardware error"
    RETURN GeneralError
ELSE
    RETURN OK
ENDIF

CommandExit(Cmd.Command, CurrentPort)
EXIT

12.10.7. Get RF Path IDs

Description (Informative):

The ALD provides the RFPathID list for the requested port number. The ALD has a separate RF Path ID list on each port for each primary.

Specification (Normative):

```
PrimaryCommand GetRFPathIDsCommand {
      CommandCode_t          Command ← 0x0023
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength ← 2
      uint16_t               PortNumber
}

ALDResponse GetRFPathIDsResponse {
      CommandCode_t          Command ← 0x0023
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
            uint16_t         NrOfRFPathIDs
            uint16_t         RFPathIDs[NrOfRFPathIDs]
      }
      else {
            ALDState_t             ALDState
            ConnectionState_t      ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
      InvalidPortNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
        RETURN InvalidSubunitNumber
        EXIT
ENDIF

IF Cmd.PortNumber is not one of 1..MaxPort THEN
        RETURN InvalidPortNumber
        EXIT
ENDIF

result ← IsCommandAllowed( LIST{    OperatingConnectionState,
                                    RestrictedConnectionState,
                                    MALDConfigConnectionState},
                                    Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
        RETURN result.code
        EXIT
ENDIF

RETURN OK, number of stored RF path IDs and the list of RF path IDs of requested port
and Cmd.PortNumber
CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.10.8. Get RF Path ID Alias

Description (Informative):

The ALD provides the RFPathID Alias for the RFPathID. The ALD has a separate RF Path ID Alias list on each port for each primary.

Specification (Normative):

```
PrimaryCommand GetRFPathIDAliasCommand {
      CommandCode_t          Command ← 0x0024
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength ← 2
      uint16_t               RFPathID
}
```

```
ALDResponse GetRFPathIDAliasResponse {
      CommandCode_t         Command ← 0x0024
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
            uint8_t         LengthOftheRFPathAlias
            UTF8String_t    Alias                   // max 32 octets
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}
Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      IncorrectState
      InvalidSubunitNumber
      InvalidRFPathID
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

IF Cmd.RFPathID is out of range THEN
      RETURN InvalidRFPathID
      EXIT
ENDIF

result ← IsCommandAllowed( LIST{   OperatingConnectionState,
                                   RestrictedConnectionState,
                                   MALDConfigConnectionState},
                                   Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
      RETURN result.code
      EXIT
ENDIF

RETURN OK, the length of the RF path ID alias and the RFPathIDAlias of the requested
RFPathID and Cmd.PortNumber
CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.10.9. Get Connector Plate Marking Info

Description (Informative):

On the receipt of this command the ALD returns connector plate marking information for the indicated port. This information contains a group of strings that describe marking of the RF ports on the connector plate(s) of the device. If no data is provided for a particular field, an empty string shall be returned.

Specification (Normative):

```
Enumeration Provenance_t : uint8_t {
      NotSet     ← 0
      Factory    ← 1
      File       ← 2
      Automatic  ← 3
      Manual     ← 4
}

PrimaryCommand GetConnectorPlateMarkingInfo {
      CommandCode_t   Command ← 0x0025
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t       Subunit
      DataLength_t          DataLength ← 2
      uint16_t              PortNumber
}

ALDResponse GetConnectorPlateMarkingInfo {
      CommandCode_t   Command ← 0x0025
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {

            uint8_t               LengthOfPortNumberString
            TextString_t          PortNumberString
            Provenance_t          PortNumberStringPorvenance
            uint8_t               LengthOfPortLabelString
            TextString_t          PortLabelString
            Provenance_t          PortLabelStringProvenance
            uint8_t               LengthOfArrayIDString
            TextString_t          ArrayIDString
            Provenance_t          ArrayIDStringProvenance
            uint8_t               LengthOfPolarizationMarkingString
            TextString_t          PolarizationMarkingString
            Provenance_t          PolarizationMarkingStringProvenance
            uint8_t               LengthOfFrequencyMarkingString
            TextString_t          FrequencyMarkingString
            Provenance_t          FrequencyMarkingStringProvenance
            uint8_t               LengthOfArrayPositionInfoString
            TextString_t          ArrayPositionInfoString
            Provenance_t          ArrayPositionInfoStringProvenance
            uint8_t               LengthOfPortColourString
            TextString_t          PortColourString
            Provenance_t          PortColourStringProvenance
            uint8_t               LengthOfAdditionalMarkingString
            TextString_t          AdditionalMarkingString
```

```
          Provenance_t          AdditionalMarkingStringProvenance
     }
     else {
          ALDState_t      ALDState
          ConnectionState_t     ConnectionState
     }
}
```

```
Enumeration ReturnCode_t {
     OK
     FormatError
     Busy
     UnknownCommand
     IncorrectState
     InvalidPortNumber
     InvalidSubunitNumber
     InvalidSubunitType
     NotAntennaPort
     NotAuthorised
}
```

Primary specification (Normative):

ALD specification (Normative):

IF Cmd.PortNumber is not one of 1…MaxPort THEN
    RETURN InvalidPortNumber
    EXIT
ENDIF

IF ALDType = MALD
    AND ActiveAuth[CurrentPort].Authority[Cmd.Subunit] = NoAccess THEN
    RETURN NotAuthorised
    EXIT
ENDIF

result ← IsCommandAllowed(LIST{ OperatingConnectionState,
RestrictedConnectionState},
Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
    RETURN result.code
    EXIT
RETURN OK, PortNumberString, PortLabelString, ArrayIDString, PolarizationMarkingString, FrequencyMarkingString, ArrayPositionInfoString, PortColourString, AdditionalMarkingString and corresponding string lengths and provenances.
CommandExit(Cmd.Command, CurrentPort)
EXIT

12.11. Ping commands

12.11.1. Send Ping

Description (Informative):

This command makes the pinger enter PingerBroadcastWaitState.

If the ALD receives the layer 2 command TriggerPing within the next 10 seconds, it sends a layer 2 Ping message on the requested port.

If the 10 seconds expires, the ALD sends a PingTimeout command.

Refer to Section 8.4. “Pinging” for details.

Specification (Normative):

```
PrimaryCommand SendPingCommand {
      CommandCode_t          Command ← 0x0026
      CommandSequence_t      PrimaryCommandSequence
      Subunit_t              Subunit ← 0
      DataLength_t           DataLength ← 6
      uint16_t               PortNumber
      uint32_t               PrimaryID
}

ALDResponse SendPingResponse {
      CommandCode_t          Command ← 0x0026
      CommandSequence_t      PrimaryCommandSequence
      ReturnCode_t           ReturnCode
      DataLength_t           DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t             ALDState
            ConnectionState_t      ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      UnknownCommand
      PortInUse
      IncorrectState
      InvalidPortNumber
      InvalidSubunitNumber
      IncorrectPortType
      IncorrectDirection
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

IF Cmd.PortNumber is not one of 1..MaxPort THEN
      RETURN InvalidPortNumber
      EXIT
ENDIF
```

```
UNLESS (PortProperties[Cmd.PortNumber] bitwise AND Direction) = TowardsBaseStation
THEN
        RETURN IncorrectDirection
        EXIT
ENDIF

UNLESS (PortProperties[Cmd.PortNumber] bitwise AND Ping) = Ping THEN
        RETURN IncorrectPortType
        EXIT
ENDIF

IF ALDType = MALD
        AND Cmd.PortNumber ≠ CurrentPort
        AND LinkState[Cmd.PortNumber] = Connected THEN
        RETURN PortInUse
        EXIT
ENDIF

result ← IsStateChangeAllowed( LIST{  OperatingConnectionState},
                                      Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
        RETURN result.code
        EXIT
ENDIF

PrimaryID ← Cmd.PrimaryID
RETURN OK

FOREACH PORT FROM 1 to MaxPort DO
        NEXT IF PORT = CurrentPort
        Switch ConnectionState[Port] to RestrictedConnectionState
ENDFOR

Switch ConnectionState[CurrentPort] to PingerConnectionState
Switch ALDState to PingerBroadcastWaitState
PendingConnectionStateChange ← false
UNLOCK StateLock
Initiate PingTimer at 10 seconds
CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.11.2. Monitor Ping

Description (Informative):

This command makes the listener enter the ListenerBroadcastWaitState.

If the ALD receives the layer 2 command TriggerPing within 10 seconds, it monitors the requested port for a Ping message.

If the 10 seconds expires, the ALD sends a PingTimeout command.

Refer to Section 8.4. “Pinging” for details.

Specification (Normative):

```
Enumeration MonitorPhase_t : uint8_t {
      Start       ← 0
      Middle      ← 1
      End         ← 2
}

PrimaryCommand MonitorPingCommand {
      CommandCode_t           Command ← 0x0027
      CommandSequence_t       PrimaryCommandSequence
      Subunit_t               Subunit ← 0
      DataLength_t            DataLength ← 3
      MonitorPhase_t          Phase
      uint16_t                PortNumber
}

ALDResponse MonitorPingResponse {
      CommandCode_t           Command ← 0x0027
      CommandSequence_t       PrimaryCommandSequence
      ReturnCode_t            ReturnCode
      DataLength_t            DataLength
      if (ReturnCode == OK) {
            uint32_t          PrimaryID
      }
      else {
            ALDState_t              ALDState
            ConnectionState_t       ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      UnknownCommand
      PortInUse
      IncorrectState
      IncorrectPortType
      InvalidPortNumber
      InvalidSubunitNumber
      InvalidMonitorPhase
      IncorrectDirection
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
      RETURN InvalidSubunitNumber
      EXIT
ENDIF

IF Cmd.PortNumber is not one of 1..MaxPort THEN
      RETURN InvalidPortNumber
      EXIT
ENDIF
```

```
IF Cmd.Phase is not one of (Start, Middle, End) THEN
	RETURN InvalidMonitorPhase
	EXIT
ENDIF

IF (PortProperties[Cmd.PortNumber] bitwise AND Direction) ≠ TowardsAntenna THEN
	RETURN IncorrectDirection
	EXIT
ENDIF

UNLESS (PortProperties[Cmd.PortNumber] bitwise AND Ping) = Ping THEN
	RETURN IncorrectPortType
	EXIT
ENDIF

IF ALDType = MALD AND Cmd.PortNumber port has an enabled OOK bypass to a port with
an active connection to another primary THEN
	RETURN PortInUse
	EXIT
ENDIF

IF Cmd.Phase is one of (Start, Middle) THEN
	result ← IsStateChangeAllowed( LIST{	OperatingConnectionState},
						Cmd.Command, CurrentPort)
	UNLESS result.allowed THEN
		RETURN result.code
		EXIT
	ENDIF

	FOREACH PORT FROM 1 to MaxPort DO
		NEXT IF PORT = CurrentPort
		Switch ConnectionState[Port] to RestrictedConnectionState
	ENDFOR

	Switch ConnectionState[CurrentPort] to ListenerConnectionState
	Switch ALDState to ListenerBroadcastWaitState
	PendingConnectionStateChange ← false
	UNLOCK StateLock
	PingMonitorRFPort ← Cmd.PortNumber
	Initiate PingTimer at 10 seconds
ELSE
	result ← IsCommandAllowed( LIST{	OperatingConnectionState},
						Cmd.Command, CurrentPort)

	UNLESS result.allowed THEN
		RETURN result.code
		CommandExit(Cmd.Command, CurrentPort)
		EXIT
	ENDIF
```

ENDIF

RETURN OK, PingReceivedFlag and PrimaryID
CommandExit(Cmd.Command, CurrentPort)
EXIT

NOTE: If a Ping message was received, the returned PingReceivedFlag will be 1 and the PrimaryID is the ID received in the Ping message.

NOTE: If the monitoring timed out (Ping message not received), the PingReceivedFlag will be 0 and the returned PrimaryID is whatever was previously set.

NOTE: If the Monitoring phase is Start, the PingReceivedFlag will be 0 and the returned PrimaryID is whatever was previously set.

12.11.3. Abort Ping

Description (Informative):

If the ALD is in the ListenerBroadcastWaitState or in the PingerBroadcastWaitState, it shall return to the OperatingState.

Refer to Section 8.4. “Pinging” for details.

Specification (Normative):

```
PrimaryCommand AbortPingCommand {
      CommandCode_t         Command ← 0x0028
      CommandSequence_t     PrimaryCommandSequence
      Subunit_t             Subunit ← 0
      DataLength_t          DataLength ← 0
}

ALDResponse AbortPingResponse {
      CommandCode_t         Command ← 0x0028
      CommandSequence_t     PrimaryCommandSequence
      ReturnCode_t          ReturnCode
      DataLength_t          DataLength
      if (ReturnCode == OK) {
      }
      else {
            ALDState_t            ALDState
            ConnectionState_t     ConnectionState
      }
}

Enumeration ReturnCode_t {
      OK
      FormatError
      Busy
      UnknownCommand
      IncorrectState
      InvalidSubunitNumber
}
```

Primary specification (Normative):

ALD specification (Normative):

```
UNLESS Cmd.Subunit = 0 THEN
        RETURN InvalidSubunitNumber
        EXIT
ENDIF

result ← IsStateChangeAllowed( LIST{  ListenerConnectionState,
                                      PingerConnectionState},
                                      Cmd.Command, CurrentPort)

UNLESS result.allowed THEN
        RETURN result.code
        EXIT
ENDIF

Switch ConnectionState[1..MaxPort] to OperatingConnectionState
Switch ALDState to OperatingState
PendingConnectionStateChange ← false
UNLOCK StateLock
RETURN OK
CommandExit(Cmd.Command, CurrentPort)
EXIT
```

12.12. Timers

12.12.1. Ping Timer

Description (Informative):

This timer is used to synchronise the ping process.

Refer to 8.4. “Pinging” for details.

Specification (Normative)

Upon PingTimer expiration:

```
IF ALDState = ListenerBroadcastWaitState THEN
        RAISE AlarmPingTimeoutExpired SEVERITY Warning ON ALL, “”
        Switch ALDState to OperatingState
        Switch ConnectionState[1..MaxPort] to OperatingConnectionState
ELSEIF ALDState = PingerBroadcastWaitState THEN
        Send PingTimeout command
        Switch ALDState to OperatingState
        Switch ConnectionState[1..MaxPort] to OperatingConnectionState
ELSEIF ALDState = ListenerRestrictedPreparationState THEN
        PingReceivedFlag ← 0
        Clear the receive buffer
        Switch ALDState to ListenerRestrictedMonitorState
        Initiate PingTimer at 40 ms
```

```
ELSEIF ALDState = PingerRestrictedState THEN
        Queue Ping message for transmission
        Switch ALDState to PingerRestrictedTransmitState
ELSEIF ALDState = ListenerRestrictedMonitorState THEN
        Select the initiating AISG port
        Switch ALDState to OperatingState
        Switch ConnectionState[1..MaxPort] to OperatingConnectionState
ENDIF
EXIT
```

13. BINARY BASED FREQUENCY CODING

In this scheme, the frequency information is coded using the structure Frequencies below.

This scheme shall be used on all frequency related information.

For each defined frequency range, the first position in the table contains the number of frequency ranges to be defined. This is followed by one or more frequency ranges containing a link descriptor and tuples of lower and upper frequency limits. These are listed in ascending order of lower, secondarily upper, frequency limit. The frequencies are always stated in kHz (rounded to the nearest integer).

If a device covers non-contiguous frequency bands they shall be separately specified.

If the characteristics of a device are not frequency-dependent, the number of frequency ranges shall be 0.

The Link Descriptor Bidirectional shall be used for devices that cover a frequency range which is for Uplink and Downlink. The descriptors Uplink and Downlink shall be used in all cases where a frequency range supports only one of these functions.

```
Enumeration LinkDescriptor_t: uint8_t {
      Uplink          ← 1
      Downlink        ← 2
      Bidirectional   ← 3
}

struct FrequencyRange_t {
LinkDescriptor_t     Link
      uint32_t        MinFrequency    // in kHz
      uint32_t        MaxFrequency    // in kHz
}
```

Number of frequency ranges	Link Descriptor, range 1	Fmin, range 1	Fmax, range 1		Link Descriptor, range n	Fmin, range n	Fmax, range n

Table 13-1: Frequency information

14. VERSION MANAGEMENT

The version numbering of this base standard and the subunit type standards uses the following scheme.

14.1. Base standard versions

Base standard version va.b.c.d

- a is used to identify AISG release. This document is release 3
- b is used for feature introductions. Incremented every time a new feature is introduced into the standard.
- c is for technical updates. Incremented every time a technical change is introduced into the standard. Once under change control, such changes shall only occur when AISG approves one or more change requests. Set to zero every time b is incremented. Major changes require an update of b.
- d is for editorial updates. Clarifications of missing or ambiguous definitions shall be considered as editorial updates. Incremented every time a purely editorial change is introduced into the standard. Set to zero every time c is incremented or set to zero. d shall not be used in version negotiation.
- Document version v3.b.c.d defines base standard version v3.b.c, which is used for version negotiation (PI=22).

14.2. Subunit type standard versions

Subunit type standard version vXXXa.b.c.d (XXX is the subunit type acronym)

- a is used to identify AISG release. This document is release 3.
- b is used for feature introductions. Incremented every time a new feature is introduced into the standard.
- c is for technical updates. Incremented every time a technical change is introduced into the standard. Once under change control, such changes shall only occur when AISG approves one or more change requests. Set to zero every time b is incremented. Major changes require an update of b.
- d is for editorial updates. Clarifications of missing or ambiguous definitions shall be considered as editorial updates. Incremented every time a purely editorial change is introduced into the standard. Set to zero every time c is incremented or set to zero. d shall not be used in version negotiation.
- Document version vXXXa.b.c.d defines subunit type standard version vXXXa.b.c. Numbers a, b and c are used version negotiation on layer 7.

Each subunit type standard lists the base standard versions with which it is compatible.

Annex A: Examples of binary based frequency coding (Informative):

This annex shows the binary frequency coding of various devices as examples.

Dec	0
Hex	0x00

Table A-1: Device (for example a sensor) with no frequency range, coded as having no frequency ranges

Dec	1	3	790000	960000
Hex	0x01	0x03	0x000C0DF0	0x000EA600

Table A-2: Antenna array with 790-960 MHz frequency range

Dec	2	1	832000	862000	2	791000	821000
Hex	0x02	0x01	0x000CB200	0x000D2730	0x02	0x000C11D8	0x000C8708

Table A-3: Dual TMA with 832-862 MHz RX and 791-821 MHz TX frequency ranges

Coded as two frequency ranges, defining transmit and receive direction of signals

Dec	4	1	1710000	1785000	1	1920000	1980000	•••
Hex	0x04	0x01	0x001A17B0	0x001B3CA8	0x01	0x001D4C00	0x001E3660	

•••	2	1805000	1880000	2	2110000	2170000
	0x02	0x001B8AC8	0x001CAFC0	0x02	0x00203230	0x00211C90

Table A-4: Twin TMA with 1710-1785 MHz and 1920-1980 MHz RX frequency ranges and 1805-1880 MHz and 2110-2170 MHz TX frequency ranges

NOTE: Coded as four frequency ranges, defining transmit and receive direction of signals

Annex B: Version management example (Informative):

The green digits are used for version negotiation at layer 2, the red digits are configured for each subunit type standard at layer 7.

Release history	Base document version	Subunit type standard ST-TMA document version	Subunit type standard ST-RET document version
Next AISG release	v3.0.0.0	-	-
Technical update of base doc	v3.0.1.0	-	-
Editorial update of base doc	v3.0.1.1	-	-
First version of ST-TMA subunit type standard	v3.0.1.1	vTMA3.0.0.0	-
Change of substance of subunit type standard ST-TMA	v3.0.1.1	vTMA3.0.1.0	-
Editorial update of subunit type standard ST-TMA	v3.0.1.1	vTMA3.0.1.1	-
Major change to base document that does not affect ST-TMA	v3.1.0.0	vTMA3.0.1.2	-
Major change to base document that affects ST-TMA	v3.2.0.0	vTMA3.1.0.0	-
First version of ST-RET subunit type standard	v3.2.0.0	vTMA3.1.0.0	vRET3.0.0.0

Table B-1: Version management example

Annex C: Ping process timing (Informative):

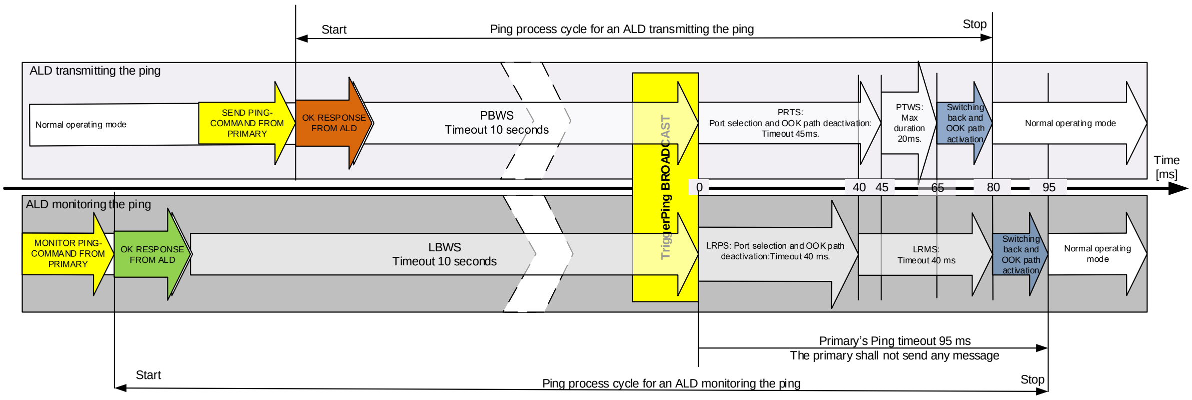


Figure C-1: Ping process with maximum state timeouts and durations

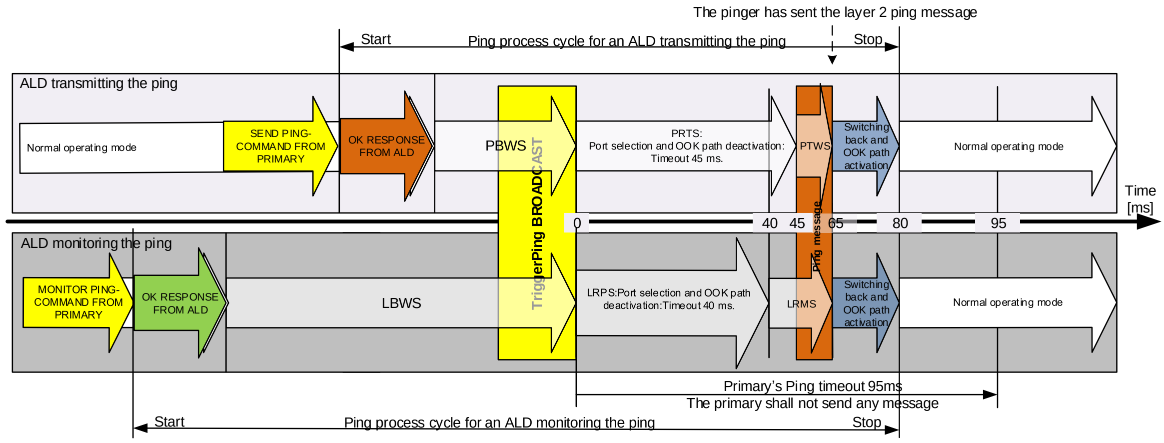


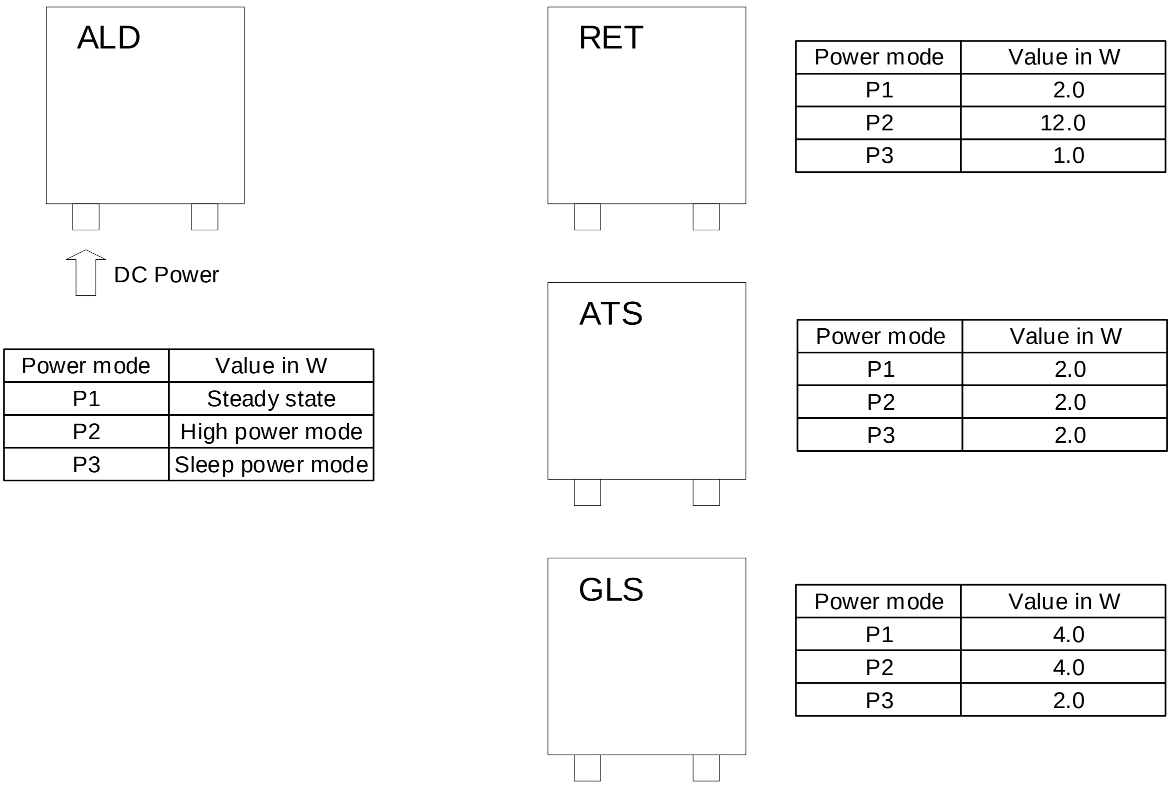
Figure C-2: Ping process timings in typical case

Ping State Abbreviations	Ping State Names
LBWS	ListenerBroadcastWaitState
LRMS	ListenerRestrictedMonitorState
LRPS	ListenerRestrictedPreparationState
PBWS	PingerBroadcastWaitState
PRTS	PingerRestrictedTransmitState
PTWS	PingerTransmitWaitState

Table C-1: Ping state abbreviations

Annex D: Examples of ALDs with different power mode values (Informative):

This annex shows various devices as examples. ATS is a temperature sensor and GLS is a geographic location sensor.



ALD:

Power mode	Value in W
P1	Steady state
P2	High power mode
P3	Sleep power mode

RET:

Power mode	Value in W
P1	2.0
P2	12.0
P3	1.0

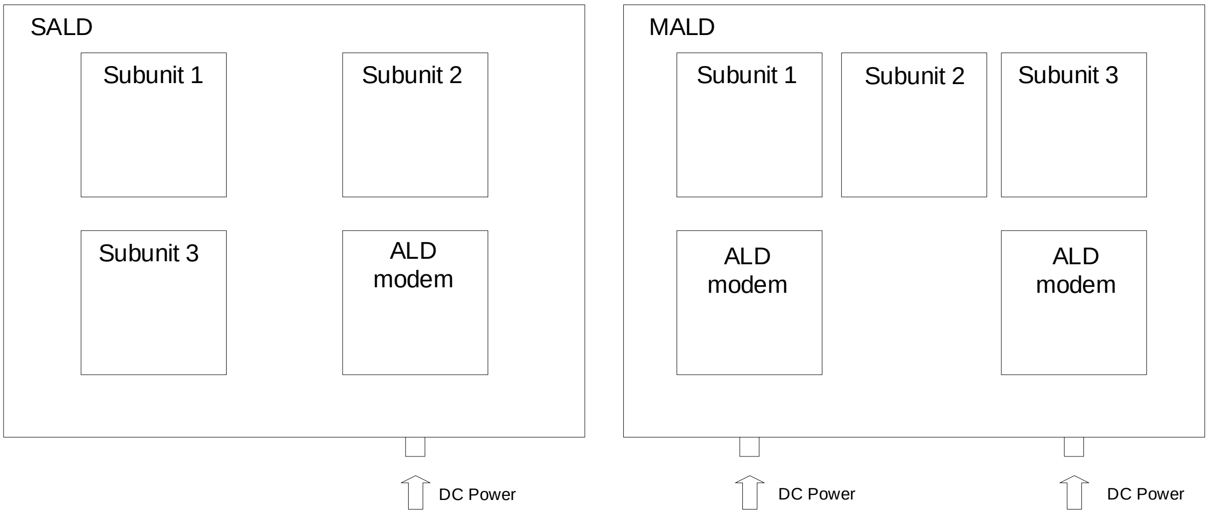
ATS:

Power mode	Value in W
P1	2.0
P2	2.0
P3	2.0

GLS:

Power mode	Value in W
P1	4.0
P2	4.0
P3	2.0

Figure D-1: An example for stand alone ALDs each with single subunit



Power mode	Value in W
P1	10.0
P2	20.0
P3	2.0

Power mode	Value in W
P1	11.0
P2	21.0
P3	3.0

Figure D-2: An example for SALD and MALD