Tower Mounted Amplifier

AISG-ST-TMA

vTMA3.0.2.0

Revision History

<table>
<thead>
<tr>
<th>DATE</th>
<th>ISSUE</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>11th June 2019</td>
<td>vTMA3.0.2.0</td>
<td>Third public release</td>
</tr>
<tr>
<td>6th February 2019</td>
<td>vTMA3.0.1.0</td>
<td>Second public release</td>
</tr>
<tr>
<td>5th November 2018</td>
<td>vTMA3.0.0.6</td>
<td>First public release</td>
</tr>
</tbody>
</table>
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. AISG v3.0 base document describes the common behaviour of antenna line devices with AISG interfaces and type-specific functionality is defined in subunit type standards. This subunit type standard covers tower mounted amplifiers (sometimes referred to as masthead or ground mounted amplifiers).

This standard is independent of previous 3GPP standards.
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 antennas.

The standard is divided into the base document and several subunit type standards. This subunit type standard document describes the specific behaviour of the tower mounted amplifier (TMA) subunit type.

This standard defines the functional behaviour of the TMA subunits.

2.1 Interpretation (Normative)

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

All statements in the current document are normative, unless indicated as informative or example.

For purposes of compliance and AISG interoperability, users should note that the implementation of this subunit type standard is optional. However, once it is selected for inclusion in a product, the entire standard becomes mandatory.
3. 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. AISG v3.0: “Base Standard AISG v3.0”
2. AISG v3.0 STCM: “Subunit Type Compliance Matrix”
4. VERSION COMPLIANCE (Informative)

The compliance of this standard with different version of AISG v3 baseline standard is defined in [2].
5. ABBREVIATIONS (Informative)

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

- **ALD**: Antenna Line Device
- **DC**: Direct Current
- **LNA**: Low Noise Amplifier
- **MALD**: Multi-primary ALD
- **RF**: Radio Frequency
- **SALD**: Single-primary ALD
- **TCC**: Time-Consuming Command
- **TMA**: Tower Mounted Amplifier
- **3GPP**: 3rd Generation Partnership Project
6. TERMINOLOGY

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

- **Gain**: A factor by which signal power is increased in an amplifier, usually expressed in dB.
- **LNA path**: A path including LNA and possible bypass switches.
- **Low noise amplifier**: An amplifier that amplifies a very low power signal without significantly degrading its signal-to-noise ratio.
- **TMA bypass**: An RF path around an LNA to provide alternative route.
- **TMA bypass path**: An RF path around an LNA including bypass switches.
- **TMA bypass switch**: A functionality that switches between normal and bypass mode.
- **Tower mounted amplifier**: A unit typically providing uplink and downlink path and containing an LNA and associated filters.
7. DEFINITIONS

GainTable shall be initialised on reset to contain the possible gain settings of the TMA.

```c
uint8_t NrOfGainRanges
GainRange_t GainTable[NrOfGainRanges]
```

TMAMode_t is used to identify the normal or bypass mode of TMA.

```c
Enumeration TMAMode_t: uint8_t {
    BypassMode ← 0
    NormalMode ← 1
}
```

PERSISTENT TMAMode_t TMARequestedMode

```c
Enumeration TMAAlarmCode_t : uint16_t {
    TMAAlarmLNAFailed ← 0x0200
    TMAAlarmBypassFailed ← 0x0201
    TMAAlarmNoRXConnection ← 0x0202
}
```

```c
Enumeration ActivePath_t : uint8_t {
    LNAPath ← 0
    BypassPath ← 1
    None ← 2
}
```

PERSISTENT dB_t TMARequestedGain

```c
CONST uint16_t NrOfSubunitAlarms ← 3 // Number of subunit alarm types for this subunit type
```
8. GENERAL ASPECTS

8.1. Subunit association

A TMA subunit may be associated with two or more RF ports with interconnections between some or all of these ports.

8.2. State models

8.2.1. TMA state model for layer 7

TMA state model has 11 states, which are listed with ID numbers in Annex A. The inputs for the state model are shown in figure 8.2.1-1 “Input to TMAState state model”, which shows the relationship between different states of the TMA path, Bypass path and TMA mode. Transitions to certain states may require diagnostic capabilities.

In LNAPathOKState the LNA path is fully operational and amplifies the signal.

In LNAPathImpairedState the LNA path is operational but with reduced amplification performance.

In LNAPathBrokenState the LNA path is unable to amplify the signal and may be heavily attenuated.

In BypassPathOKState the bypass path is fully operational and transfers the signal.

In BypassPathBrokenState the bypass path may be heavily attenuated.

```cpp
Enumeration LNAPathState_t : uint8_t {
    LNAPathOKState ← 0
    LNAPathBrokenState ← 1
    LNAPathImpairedState ← 2
}

LNAPathState_t LNAPathState

Enumeration BypassPathState_t : uint8_t {
    BypassPathOKState ← 0
    BypassPathBrokenState ← 1
}

BypassPathState_t BypassPathState
```
### 8.3. TMA capabilities

The TMACapabilities bitfield shall be initialised during reset.

```c
Bitfield TMACapabilities_t : uint8_t {
    LNA.Diagnostics : Bit 0   // Able to diagnose LNA failure
    Bypass : Bit 1
    RedundantLNA : Bit 2   // Multiple redundant amplifiers
    Bypass.Diagnostics : Bit 3   // Able to diagnose bypass
    AdjustableGain : Bit 4
}
```

TMACapabilities_t TMACapabilities

### 8.4. Return codes

This subunit type standard contains the following subunit type specific return codes.

---

**Figure 8.2.1-1 Input to TMAState state model**
```c
enum ReturnCode_t : uint16_t {
    TMAUnsupportedGainValue  = 0x0200,
    TMAMajorFault           = 0x0201,
    TMAMinorFault           = 0x0202,
    TMANotPredictableGain   = 0x0203
};
```

8.5. Resumption of operation

The following data shall be retained after reset:

- TMARequestedMode
- TMARequestedGain

After reset, TMA shall perform TMAEvaluateState function.
9. LAYER 1

All definitions and standards for ALDs in [1] regarding layer 1 shall be valid for ALDs which contain TMA subunits.

9.1. DC power consumption

This subunit type standard does not define the power consumption of a TMA. See [1] for details on how power consumption is reported to the primary.

This subunit type standard does not contain any commands that switch the ALD from SteadyStatePowerMode to HighPowerMode or to SleepPowerMode.
10. LAYER 2

All definitions and standards for ALDs in [1] regarding layer 2 shall be valid for ALDs which contain TMA subunits.
11. LAYER 7
An ALD which contains TMA subunit(s) shall support the command set which is defined in [1] in addition to those commands specified in this standard.

11.1. Subunit type

<table>
<thead>
<tr>
<th>Subunit type</th>
<th>1-octet unsigned integer code</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMA</td>
<td>0x02</td>
</tr>
</tbody>
</table>

Table 11.1-1: Subunit type code

11.2. Overview of commands for TMA subunits
The table below shows an overview of all commands used in this TMA subunit type standard. The following abbreviations are used in the Table 11.2-1 "Commands for TMA subunits":
M Mandatory
O Optional
- Not applicable

<table>
<thead>
<tr>
<th>TMA Command</th>
<th>Code</th>
<th>Initiator</th>
<th>Subunit</th>
<th>Timeout</th>
<th>TCC</th>
<th>Mandatory for:</th>
<th>Changes the</th>
<th>Changes the</th>
<th>Minimum required</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMA Get Capabilities</td>
<td>0x0200</td>
<td>Primary</td>
<td>&gt;0</td>
<td>1 s</td>
<td>no</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>no</td>
</tr>
<tr>
<td>TMA Set Mode</td>
<td>0x0201</td>
<td>Primary</td>
<td>&gt;0</td>
<td>1 s</td>
<td>no</td>
<td>M</td>
<td>O</td>
<td>O</td>
<td>no</td>
</tr>
<tr>
<td>TMA Get Mode</td>
<td>0x0202</td>
<td>Primary</td>
<td>&gt;0</td>
<td>1 s</td>
<td>no</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>no</td>
</tr>
<tr>
<td>TMA Set Gain</td>
<td>0x0203</td>
<td>Primary</td>
<td>&gt;0</td>
<td>1 s</td>
<td>no</td>
<td>M</td>
<td>O</td>
<td>O</td>
<td>no</td>
</tr>
<tr>
<td>TMA Get Gain</td>
<td>0x0204</td>
<td>Primary</td>
<td>&gt;0</td>
<td>1 s</td>
<td>no</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 11.2-1: Commands for TMA subunits

11.3. Updating of TMA state
Function EvaluateTMAState() is called:
- on reset – to determine initial state of TMA
- when user requests a change of TMA mode
- when a state of LNA or bypass path changes
The function is executed in the context of a TMA subunit. Re-raising an alarm with same severity shall result only in updating the diagnostic string associated with the alarm. TMA state IDs are described in Annex A.

```c
struct AlarmDetails_t {
    Severity_t Severity
    UTF8String_t DiagnosticString
};

FUNCTION EvaluateTMAState(uint16_t Subunit) IS
    // Outputs of the evaluation
    AlarmDetails_t LNAAlarm
    AlarmDetails_t BypassAlarm
    AlarmDetails_t RxConnectionAlarm
    LNAAlarm.Severity ← Cleared
    LNAAlarm.DiagnosticString ← ""
    BypassAlarm.Severity ← Cleared
    BypassAlarm.DiagnosticString ← ""
    RxConnectionAlarm.Severity ← Cleared
    RxConnectionAlarm.DiagnosticString ← ""
    CASE LNAPathStatus IS
        WHEN LNAPathOK:
            IF TMARequestedMode = BypassMode THEN
                IF BypassPathState = BypassPathBrokenState THEN // TMA state ID 4
                    TMAActivePath ← LNAPath
                    BypassAlarm.Severity ← Minor
                    BypassAlarm.DiagnosticString ← "Bypass path failed"
                ELSEIF // TMA state ID 3
                    TMAActivePath ← BypassPath
                ENDIF
            ELSEIF // TMA state ID 1, 2
                TMAActivePath ← LNAPath
            ENDIF
        ELSE // TMA state ID 1,2
            TMAActivePath ← LNAPath
        ENDIF
```

WHEN LNAPathImpaired:
   LNAAlarm.Severity ← Minor
   LNAAlarm.DiagnosticString ← “LNA path impaired”
   IF TMARequestedMode = BypassMode THEN
      IF BypassPathState = BypassPathBrokenState THEN // TMA state ID 8
         TMAActivePath ← LNAPath
         BypassAlarm.Severity ← Minor
         BypassAlarm.DiagnosticString ← “Bypass path failed”
      ELSE
         TMAActivePath ← BypassPath // TMA state ID 7
      ENDIF
   ELSE
      TMAActivePath = LNAPath // TMA state ID 5,6
   ENDIF
   ELSEIF BypassPathState = BypassPathOKState // TMA state ID 10
      TMAActivePath ← BypassPath
      LNAAlarm.Severity ← Major
      LNAAlarm.DiagnosticString ← “LNA path broken”
   ELSE // BypassPathState = BypassPathBrokenState, TMA state ID 11
      TMAActivePath ← None
      RxConnectionAlarm.Severity ← Major
      RxConnectionAlarm.DiagnosticString ← “LNA and bypass paths broken”
   ENDIF
ENDIF
ENDCASE
IF TMAActivePath = LNAPath
   «Switch signal path to LNA path»
ELSEIF TMAActivePath = BypassPath
   «Switch signal path to TMA bypass path»
ENDIF
CASE LNAAlarm.Severity IS
   WHEN Major:
      RAISE TMAAlarmLNAFailed SEVERITY Major ON Cmd.Subunit,
         LNAAlarm.DiagnosticString
   WHEN Minor:
      RAISE TMAAlarmLNAFailed SEVERITY Minor ON Cmd.Subunit,
         LNAAlarm.DiagnosticString
   OTHERWISE:
      CLEAR TMAAlarmLNAFailed ON Cmd.Subunit
ENDIF
ENDCASE
IF RxConnectionAlarm.Severity = Cleared THEN
   CLEAR TMAAlarmNoRXConnection ON Cmd.Subunit
ELSE
   RAISE TMAAlarmNoRXConnection SEVERITY Major ON Cmd.Subunit,
   LNAAlarm.DiagnosticString
ENDIF
IF BypassAlarm.Severity = Cleared THEN
   CLEAR TMAAlarmBypassFailed ON Cmd.Subunit
ELSE
   RAISE TMAAlarmBypassFailed SEVERITY Minor ON Cmd.Subunit,
   LNAAlarm.DiagnosticString
ENDIF
END

11.3.1 Events
ON «TMA bypass path’s condition change» DO
   IF «bypass path is broken» THEN
      SWITCH BypassPathState TO BypassPathBrokenState
   ELSE
      SWITCH BypassPathState TO BypassPathOKState
   ENDIF
   EvaluateTMAState(Cmd.Subunit)
DONE
ON «LNA path’s condition change» DO
   IF «LNA path is broken» THEN
      SWITCH LNAPathState TO LNAPathBrokenState
   ELSE IF «at least one, but not all, redundant amplifiers have failed» THEN
      SWITCH LNAPathState TO LNAPathImpairedState
   ELSE
      SWITCH LNAPathState TO LNAPathOKState
   ENDIF
   EvaluateTMAState(Cmd.Subunit)
DONE

11.4. Reset
ON Reset DO
   EvaluateTMAState()
DONE
11.5. TMA commands

11.5.1. TMA Get Capabilities

Description (Informative):

On receipt of the command, the ALD shall respond with the capabilities and parameters indicating the functionalities of the addressed TMA subunit.

The supported gain ranges shall follow the definitions in [1]. Gain value in bypass state shall not be reported as part of the gain ranges.

The frequency ranges in the response reflect the capabilities of the TMA; not the current usage by the primary.

Bit value 0 represents function is not supported. Bit value 1 represents function is supported.

Message format:

PrimaryCommand TMAGetCapabilitiesCommand {
    CommandCode_t Command ← 0x0200
    CommandSequence_t PrimaryCommandSequence
    Subunit_t Subunit
    DataLength_t DataLength ← 0
}

ALDResponse TMAGetCapabilitiesResponse {
    CommandCode_t Command ← 0x0200
    CommandSequence_t PrimaryCommandSequence
    ReturnCode_t ReturnCode
    DataLength_t DataLength
    if (ReturnCode == OK) {
        TMACapabilities_t TMACapabilities
        uint8_t NrOfGainRanges
        GainRange_t Ranges[1..NrOfGainRanges]
        uint8_t NrOfFrequencyRanges
        FrequencyRange_t Ranges[1..NrOfFrequencyRanges]
        }
    else {
        ALDState_t ALDState
        ConnectionState_t ConnectionState
    }
}

Enumeration ReturnCode_t {
    OK
    FormatError
    Busy
    InvalidSubunitNumber
    InvalidSubunitType
    IncorrectState
    UnknownCommand
    NotAuthorised
}

Primary pseudocode:

(This section is intentionally left blank)
ALD pseudocode:

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, TMACapabilities, NrOfGainRanges, LIST for every gain range {Min, Max, StepSize}, NrOfFrequencyRanges, LIST for every frequency range {LinkDescriptor, Min, Max}
CommandExit(Cmd.Command, CurrentPort)
EXIT

11.5.2. TMA Set Mode

Description (Informative):

The TMA Set Mode procedure shall only be supported if the TMA subunit can be set in bypass mode. On receipt of the command, the ALD shall first initialise the TMA subunit in the requested mode, and then return a response message. If a TMA subunit in bypass mode receives the command TMASetMode to BypassMode, the command shall not be performed but the response OK shall be returned. Similarly, if a TMA subunit in normal mode receives the command TMASetMode to NormalMode, the command shall not be performed but the response OK shall be returned.

Message format:

PrimaryCommand TMASetModeCommand {
  CommandCode_t Command ← 0x0201
  CommandSequence_t PrimaryCommandSequence
  Subunit_t Subunit
  DataLength_t DataLength ← 1
  TMAMode_t TMAReturnedMode
}
ALDResponse TMASetModeResponse {
    CommandCode_t   Command ← 0x0201
    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
    InvalidSubunitType
    UnknownCommand
    NotAuthorised
    UnsupportedCapability
    UnsupportedValue
}

Primary pseudocode:
(This section is intentionally left blank)

ALD pseudocode:

UNLESS TMACapabilities.Bypass THEN
    RETURN UnsupportedCapability
    EXIT
ELSEIF Cmd.TMAMode ≠ NormalMode AND Cmd.TMAMode ≠ BypassMode THEN
    RETURN UnsupportedValue
    EXIT
ENDIF
IF ALDType = MALD THEN
    UNLESS ActiveAuth[CurrentPort].Authority[Cmd.Subunit] = ReadWrite THEN
        RETURN NotAuthorised
    EXIT
ENDIF
ENDIF
result ← IsCommandAllowed( LIST{ OperatingConnectionState, RestrictedConnectionState}, 
                           Cmd.Command, CurrentPort)
UNLESS result.allowed THEN
    RETURN result.code
    EXIT
ENDIF
IF TMARequestedMode ≠ TMAMode THEN
    TMAMode ← TMARequestedMode
    EvaluateTMAState(Cmd.Subunit)
ENDIF
RETURN OK
CommandExit(Cmd.Command, CurrentPort)
EXIT

11.5.3. TMA Get Mode

Description (Informative):

On receipt of the command, the ALD shall respond with the mode flag indicating whether the TMA subunit is in NormalMode or in BypassMode. TMA subunits which do not support BypassMode shall return NormalMode.

Message format:

PrimaryCommand TMAGetModeCommand {
    CommandCode_t Command ← 0x0202
    CommandSequence_t PrimaryCommandSequence
    Subunit_t Subunit
    DataLength_t DataLength ← 0
}

ALDResponse TMAGetModeResponse {
    CommandCode_t Command ← 0x0202
    CommandSequence_t PrimaryCommandSequence
    ReturnCode_t ReturnCode
    DataLength_t DataLength
    if (ReturnCode == OK) {
        TMAMode_t RequestedMode
        ActivePath_t ActivePath
    } else {
        ALDState_t ALDState
        ConnectionState_t ConnectionState
    }
}

Enumeration ReturnCode_t {
    OK
    FormatError
    Busy
    InvalidSubunitNumber
    InvalidSubunitType
    IncorrectState
    UnknownCommand
    NotAuthorised
}

Primary pseudocode:

(This section is intentionally left blank)
ALD pseudocode:

IF ALDType = MALD
    AND ActiveAuth[CurrentPort].Authority[Cmmand.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, TMARequestedMode, TMAActivePath
CommandExit(Cmd.Command, CurrentPort)
EXIT

11.5.4. TMA Set Gain

Description (Informative):

On receipt of the command (if supported), the ALD shall first set the addressed TMA subunit to the gain determined by the TMA gain figure parameter, and then return the response message. The TMA gain figure parameter is calculated as 10 times the required gain expressed in dB. This encoding allows the gain to be set with a resolution of 0.1 dB while using an integer parameter.

If the TMA subunit is set in bypass mode by TMASetMode, and TMASetGain is received, then the procedure shall be performed and bypass mode shall be retained.

Gain shall be accepted if the requested gain value is supported.

For unsupported values the TMA subunit shall respond TMAUnsupportedGainValue.

Message format:

PrimaryCommand TMASetGainCommand {
    CommandCode_t Command ← 0x0203
    CommandSequence_t PrimaryCommandSequence
    Subunit_t Subunit
    DataLength_t DataLength ← 2
    ddB_t Gain
}
ALDResponse TMASetGainResponse {
    CommandCode_t Command ← 0x0203
    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
    InvalidSubunitType
    TMAUnsupportedGainValue
    UnknownCommand
    NotAuthorised
    TMAMajorFault
    TMAMinorFault
}

Primary pseudocode:

(This section is intentionally left blank)

ALD pseudocode:

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

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

UNLESS result.allowed THEN
    RETURN result.code
    EXIT
ENDIF

IF «TMA is in Bypass mode due to major TMA fault» THEN
    RETURN TMAMajorFault
    CommandExit(Cmd.Command, CurrentPort)
    EXIT
ENDIF
IF «TMA is in NormalMode AND gain setting cannot be achieved» THEN // The possible gain settings are defined by the vendor
RETURN TMAMinorFault
CommandExit(Cmd.Command, CurrentPort)
EXIT
ENDIF
FOREACH I FROM 1 TO NrOfGainRanges DO
NEXT IF Cmd.Gain > GainTable[I].Max
IF Cmd.Gain < GainTable[I].Min THEN
RETURN TMAUnsupportedGainValue
ELSEIF Cmd.Gain = GainTable[I].Min // Special case covering
// GainTable[I].Stepsize = 0
OR (Cmd.Gain − GainTable[I].Min) MOD GainTable[I].StepSize = 0 THEN
TMARequestedGain ← Cmd.Gain
RETURN OK
ELSE
RETURN TMAUnsupportedGainValue
ENDIF
CommandExit(Cmd.Command, CurrentPort)
EXIT
DONE
RETURN TMAUnsupportedGainValue
CommandExit(Cmd.Command, CurrentPort)
EXIT

11.5.5. TMA Get Gain

Description (Informative):

On receipt of the command, the ALD shall return the set gain of the TMA subunit. EstimatedGain is used to represent the actual gain. In NormalMode without a TMA fault, the EstimatedGain shall be equal to the SetGain. In BypassMode and in case of a TMA fault in NormalMode, the EstimatedGain shall be a prediction of the actual LNA gain.

Message format:

PrimaryCommand TMAGetGainCommand {
CommandCode_t Command ← 0x0204
CommandSequence_t PrimaryCommandSequence
Subunit_t Subunit
DataLength_t DataLength ← 0
}
ALDResponse TMAGetGainResponse {
  CommandCode_t  Command ← 0x0204
  CommandSequence_t PrimaryCommandSequence
  ReturnCode_t  ReturnCode
  DataLength_t  DataLength
  if (ReturnCode == OK) {
    dB_t   SetGain
    dB_t   EstimatedGain
    TMAMode_t  TMAMode
  } else {
    ALDState_t   ALDState
    ConnectionState_t ConnectionState
  }
}

Enumeration ReturnCode_t {
  OK
  FormatError
  Busy
  IncorrectState
  InvalidSubunitNumber
  InvalidSubunitType
  UnknownCommand
  NotAuthorised
  TMANotPredictableGain
}

Primary pseudocode:
(This section is intentionally left blank)

ALD pseudocode:

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
IF «gain can be determined» THEN // This is defined by the
  RETURN OK, EstimatedGain, TMAMode
ELSE
  RETURN TMANotPredictableGain
ENDIF
CommandExit(Cmd.Command, CurrentPort)
EXIT
Annex A: TMA State ID (Informative):

TMA state IDs shown in the pseudocode are listed in Table A-1.

<table>
<thead>
<tr>
<th>TMA State ID</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Expected gain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LNA path status</td>
<td>Mode</td>
<td>Bypass path status</td>
</tr>
<tr>
<td>1</td>
<td>OK</td>
<td>Normal</td>
<td>OK</td>
</tr>
<tr>
<td>2</td>
<td>OK</td>
<td>Normal</td>
<td>Broken</td>
</tr>
<tr>
<td>3</td>
<td>OK</td>
<td>Bypass</td>
<td>OK</td>
</tr>
<tr>
<td>4</td>
<td>OK</td>
<td>Bypass</td>
<td>Broken</td>
</tr>
<tr>
<td>5</td>
<td>Impaired</td>
<td>Normal</td>
<td>OK</td>
</tr>
<tr>
<td>6</td>
<td>Impaired</td>
<td>Normal</td>
<td>Broken</td>
</tr>
<tr>
<td>7</td>
<td>Impaired</td>
<td>Bypass</td>
<td>OK</td>
</tr>
<tr>
<td>8</td>
<td>Impaired</td>
<td>Bypass</td>
<td>Broken</td>
</tr>
<tr>
<td>9</td>
<td>Broken</td>
<td>Normal (Mode selection not applicable)</td>
<td>Bypass not supported</td>
</tr>
<tr>
<td>10</td>
<td>Broken</td>
<td>Any</td>
<td>OK</td>
</tr>
<tr>
<td>11</td>
<td>Broken</td>
<td>Any</td>
<td>Broken</td>
</tr>
</tbody>
</table>

Table A-1: TMA State IDs