XPDR/DME
TCAS/ADS-B/TIS/UAT
TEST SET
IFR 6000

Remote Commands Manual

REMOTE COMMANDS MANUAL

XPDR/DME/TCAS/ADS-B/TIS/UAT

TEST SET

IFR 6000

PUBLISHED BY
VIAVI Solutions, Inc.

Original Printing	Mar 2014
Issue 1	Sep 2016
Issue 2	Nov 2019

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

INTRODUCTION

About this Manual

This manual contains the following:

- Describes format of Values/Ranges/Default Listings, Quick Reference Guides and Detailed Remote Commands.
- Identifies conventions used in the manual;
- Describes common remote commands;
- Lists remote commands for the IFR 6000 Test Set.

Nomenclature Statement

In this manual Test Set or Unit refers to the IFR 6000 XPDR/DME/TCAS/ADS-B/TIS Test Set.

Intended Audience

This manual is intended for personnel familiar with the use of remote command language. Review the IFR 6000 Operation Manual prior to using the Test Set.

Test Set Requirements

Refer to the IFR 6000 Operation Manual for information on the following:

- Safety Precautions
- Power Requirements
- Platform Performance Data Specifications
- Repacking / Shipping Test Set

THIS PAGE INTENTIONALLY LEFT BLANK.

DEFINITIONS

The parameters that are passed to and from the instrument are mostly those types defined by IEEE488.2 but there are some that allow some of the extra functionality as defined in SCPI. The following tables define which parameters are used by the instrument. The first table defines parameters that are passed to the instrument and the second table those that are returned by the instrument.

Parameters accepted by the instrument:

<NRf>	Also known as DECIMAL NUMERIC PROGRAM DATA. This is a flexible numeric format allowing integer, real and exponent formats. Any value greater that +/-2147483647 will give a [-222, ‘Data out of range’] execution error. No suffixes (e.g. kHz) are allowed.
<NUMERIC VALUE>	This is an extension of the <NRf> format allowing suffixes as follows: <NRf> | <NRf>+<SUFFIX PROGRAM DATA> An error will be generated if the parameter is meaningless for the particular command.
<BOOLEAN PROGRAM DATA>	ON | OFF | <NRf> ON is the same as 1 (TRUE) and OFF is the same as 0 (FALSE). If the <NRf> is used, the value is rounded to an integer and any number other than 0 is treated as 1.
<CPD>	Also known as Character Program Data. This allows a number of fixed items to be specified and is typically used as a one of n selector.
<STRING PROGRAM DATA>	This allows a string to be passed to the instrument.
<ARBITRARY BLOCK PROGRAM DATA>	This allows any 8-bit binary data to be sent to the instrument.

Parameters sent by IFR 6000:

<NR1>	This format is used to return integers. A negative integer will be preceded with a - sign, a positive integer may or may not have the + sign.
<NR2>	This format is used to return real numbers. An exponential form is not used. The decimal point will always be output and at least one digit before the decimal point and one digit after the decimal point will be output. A negative number will be preceded with a - sign, a positive number may or may not have the + sign. Suffixes will never be output, all values will be in the fundamental units, e.g. for frequency the value will always be in Hz not kHz or MHz etc.
<CRD>	Also known as Character Response Data. Used to return one of many selections.
<ARBITRARY ASCII RESPONSE DATA>	Used to return any number of ASCII characters (except newline). Terminated by newline (with EOI - on interfaces that support it).
<BOOLEAN RESPONSE DATA>	Used to return a true/false indication. Can only return 0 or 1, where 0 is FALSE and 1 is TRUE.
<STRING RESPONSE DATA>	Used to return a string.
<DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA>	Used to return a block of 8-bit binary data. The data is preceded with its length.
<HEXADECIMAL NUMERIC RESPONSE DATA>	Used to return an integer number in hexadecimal notation.
<OCTAL NUMERIC RESPONSE DATA>	Used to return an integer number in octal notation.

Command List notation:

\ (back slash) following list command
Command example:
XPDR
ADDRess
STATe\?
	Indicates there is a corresponding action command and a query command for the list command.
	Action Command example:
	`XPDR:ADDR:STAT AUTO`
	Query Command example:
	`XPDR:ADDR:STAT?`

[ ] (square brackets) Example: XPDR DIVersity [ENABle]\?	Used to enclose a key word that is optional when programming the command. The instrument will process the command and have the same effect whether the option node is omitted by the programmer or not.
Upper and lower case letters Example: DIAGnostic	Upper and lower case letters are combined into one command to differentiate between the accepted short form (the uppercase characters) and the long form (the complete keyword).

COMMON COMMANDS

*CLS

Parameters: none

Description: Clear Status clears the Standard Event Status Register, the Error Queue, the Operation Status Event Register and the Questionable Status Event Register.

Example: `*CLS`

Clear the status reporting structure.

*ESE

Parameters: <NRf>

mask

Valid values: mask: integer. Valid values are 0 to 255. Values outside range are rejected and an error generated.

Description: The Standard Event Status Enable command sets the Standard Event Status Enable Register. This is an eight bit register.

Example: `*ESE 1`

*Set the Standard Event Status Enable Register to 1 (00000001 in binary). This will allow OPC (Operation Complete) messages generated by the instrument to be reported in the Event Summary Bit. (OPC is issued by the instrument when an overlapped command completes and a *OPC command has been received).*

*ESE?

Parameters: none

Response: <NR1>

mask

Returned values: mask: integer. Values are in the range 0 to 255.

Description: Read the Standard Event Status Enable Register. This is an eight bit register.

Example: `*ESE?`

*ESR?

Parameters: none

Response: <NR1>

register contents

Returned values: register contents: integer. Values are in the range 0 to 255.

Description: Read the value of the Standard Event Status Register. This is an eight bit register.

Example: `*ESR?`

*IDN?

Parameters: none

Response: <ARBITRARY ASCII RESPONSE DATA>

Instrument Identification

Returned values: Instrument Identification: string

Description: The Identification Query command allows information about the instrument to be read.

The Instrument Identification is split into four fields:
- Manufacturer
- Model
- Serial number
- Software Issue No.

Manufacturer returns 'VIAVI'

Model returns the instrument model number - 6000.

Serial number is in the form sssssssss (9 digits) where s is an ASCII digit in the range 0 to 9.

Software Issue No. is in the form nn.nn.nn where n are ASCII digits in the range 0 to 9.

Example: `*IDN?`

Read information on the instrument.

Example response: VIAVI, 6000, 104000013, 02.05.00

*OPC

Parameters: none

Description: The Operation Complete command sets the Operation Complete bit in the Standard Event Status Register when execution of all overlapped commands have completed.

This command is really only useful after an overlapped command when it will indicate when that command has been completed. Other (non-overlapped) commands can be executed whilst the overlapped command is still being executed. If there is more than one overlapped command being executed, the Operation Complete bit will only be set once all of the overlapped commands complete.

*OPC should be the final <PROGRAM MESSAGE UNIT> of the <PROGRAM MESSAGE>.

Example: `*OPC`

Since there are no overlapped commands in the instrument, the Operation Complete bit will be set in the Standard Event Status Register immediately.

*OPC?

Parameters: none

Response: <NR1>

operation complete

Returned values: operation complete: integer. Value is 1

Description: The Operation Complete Query returns a '1' when all overlapped commands have completed.

This command is really only useful after an overlapped command when it will indicate when that command has been completed.

*OPC? should be the final <QUERY MESSAGE UNIT> of the <PROGRAM MESSAGE>.

Example: `*OPC?`

Since there are no overlapped commands in the instrument, the value '1' will be placed in the output queue immediately.

*OPT?

Parameters: none

Response: <ARBITRARY ASCII RESPONSE DATA>

options

Returned values: options: string

Description: Read options present. If no options are present a single “0” is returned otherwise the response is a number of strings separated by commas. If the option is present it will return the string shown below, if it is not present nothing is returned for that option.

String	Meaning
MS	Mode S option is present
TCAS	TCAS option is present
ADSB	ADS-B option is present

Note that :SYST:OPT? can be used instead if a bit-field is easier to work with than this string format.

Example: `*OPT?`

Example response: MS,TCAS

*RST

Parameters: none

Description: Reset the instrument. This command places the instrument in its default state. This is the same state as when the instrument is first powered on.

If the Remote control system fails to respond to *RST, it may be cleared by sending a DEVICE CLEAR command. It is good practice to precede *RST with DEVICE CLEAR.

Appendix D lists reset settings and which parameters are not reset.

Example: `*RST`

Reset instrument to known state.

*SRE

Parameters: <NRf>

mask

Valid values: mask: integer. Valid values are 0 to 255. Values outside range are rejected and an error generated.

Description: Set the Service Request Enable Register. This is an eight bit register.

Example: `*SRE 32`

Set the Service Request Enable Register to 32 (0010 0000 in binary) to enable service requests when the Standard Event Status Register Summary Bit is set.

*SRE?

Parameters: none

Response: <NR1>

mask

Returned values: mask: integer. Values are in the range 0 to 255.

Description: Read the Service Request Enable Register. This is an eight bit register.

Example: `*SRE?`

*STB?

Parameters: none

Response: <NR1>

status byte

Returned values: status byte: integer. Values are in the range 0 to 255.

Description: Read the Status Byte Register. This is an eight bit register. Bit 6 of the register contains the Master Summary Status.

See Appendix A for details.

Example: `*STB?`

*TST?

Parameters: none

Response: <NR1>

self test completed

Returned values: self test completed: integer. Values are in the range 0 to 1.

Description: Self Test Query. This performs a self test and returns a value of 0 if the test passes and 1 if it fails.

If the test fails, use the :SYST:TEST:... commands to read the test results.

The self test takes a significant amount of time to execute, any timeout on waiting for a response must take this into account.

Example: `*TST?`

Perform a self test.

*WAI

Parameters: none

Description: The Wait to Continue command inhibits execution of a command until the execution of all overlapped commands has been completed.

This command is really only useful after an overlapped command when it will hold off further commands until that command has been completed. If there is more than one overlapped command being executed, the next command will be held off until all of the overlapped commands complete.

Example: `*WAI`

*Since there are no overlapped commands in the instrument, *WAI will complete immediately.*

THIS PAGE INTENTIONALLY LEFT BLANK.

XPDR COMMANDS

XPDR Mode provides flight line test capability for ATCRBS and Mode S transponders using an Auto Test, a series of tests displayed over several screens. All data normally required to verify transponder operation in accordance with FAR 91.413, Part 43, Appendix F, is displayed on one main Auto Test Screen.

XPDR SUBSYSTEM

```
XPDR
    ADDRess
        STATe\?
        [VALue]\?
    ANTenna
        BOTTom\?
        GAIN\?
        SELect\?
        TOP\?
    CCAPability\?
    R47\?
    CLOSs
        ANTenna
            MODE\?
            [VALue]\?
        [CURRent]\?
        DIRect\?
    CONFig
        CURRent?
        LIST?
        NUMBer?
        [SELect]
    DIAGnostic
        ADDRess\?
        COUNt?
        DATA?
        DTESt\?
        GENerate\?
        PRF\?
        RATTenuation\?
        SELect\?
        SLS
            ATCRbs\?
            MS\?
        STARt
        STOP
        TLEVel\?
```

```
XPDR
    DIVersity
        [ENABle]\?
    MEASure
        ATCRbs
            ACALl
                [DATA]?
                ENABled?
                STARt
            ADURation
                [DATA]?
            CDURation
                [DATA]?
            AAMPlitude
                [DATA]?
            CAMPlitude
                [DATA]?
            DECoder
                [DATA]?
                ENABled?
                STARt
            POWer
                [DATA]?
                ENABled?
                STARt
            PTIMing
                [DATA]?
                ENABled?
                STARt
            RDELay
                [DATA]?
                ENABled?
                STARt
            RDRoop
                [DATA]?
                ENABled?
                STARt
```

XPDR SUBSYSTEM

- XPDR
  - MEASure
    - ATCRbs
      - REPLy
        - [DATA]?
        - ENABled?
        - STARt
      - RJITter
        - [DATA]?
        - ENABled?
        - STARt
      - RRATio
        - [DATA]
          - PERCent?
          - [STATe]?
        - ENABled?
        - STARt
      - SLS
        - [DATA]?
        - ENABled?
        - STARt
    - [AUTO]?
    - CAPabilities?
    - COUNt?
    - FREQuency
      - [DATA]?
      - ENABled?
      - STARt
    - MS
      - BD10
        - [DATA]?
        - ENABled?
        - STARt
      - BD17
        - [DATA]?
        - ENABled?
        - STARt

- XPDR
  - MEASure
    - MS
      - BD18
        - [DATA]?
        - ENABled?
        - STARt
      - BD19
        - [DATA]?
        - ENABled?
        - STARt
      - BD1A
        - [DATA]?
        - ENABled?
        - STARt
      - BD1B
        - [DATA]?
      - BD1B
        - ENABled?
        - STARt
      - BD1C
        - [DATA]?
        - ENABled?
        - STARt
      - BD20
        - [DATA]?
        - ENABled?
        - STARt
      - BD30
        - [DATA]?
        - ENABled?
        - STARt
      - BD40
        - [DATA]?
        - ENABled?
        - STARt
      - BD50
        - [DATA]?
        - ENABled?
        - STARt

- XPDR
  - MEASure
    - MS
      - BD60
        - [DATA]?
        - ENABled?
        - STARt
      - DIVersity
        - [DATA]?
        - ENABled?
        - STARt
      - ADDress
        - [DATA]?
        - ENABled?
        - STARt
          - [AUTO]
          - MANual
      - POWer
        - [DATA]?
        - ENABled?
        - STARt
      - PTIMing
        - [DATA]
          - SPACing?
          - WIDTh?
        - ENABled?
        - STARt
      - RDELay
        - [DATA]?
        - ENABled?
        - STARt
      - RDRoop
        - [DATA]?
        - ENABled?
        - STARt
      - RJITter
        - [DATA]?
        - ENABled?
        - STARt
      - RRATio
        - [DATA]
          - PERCent?
          - [STATe]?
        - ENABled?
        - STARt

- XPDR
  - MEASure
    - MS
      - SLS
        - [DATA]?
        - ENABled?
        - STARt
      - SQUitter
        - [DATA]?
        - ENABled?
        - STARt
      - UF0
        - [DATA]?
        - ENABled?
        - STARt
      - UF11
        - [DATA]?
        - ENABled?
        - STARt
      - UF16
        - [DATA]?
        - ENABled?
        - STARt
      - UF20
        - [DATA]?
        - ENABled?
        - STARt
      - UF21
        - [DATA]?
        - ENABled?
        - STARt
      - UF24
        - [DATA]?
        - ENABled?
        - STARt
      - UF4
        - [DATA]?
        - ENABled?
        - STARt
      - UF5
        - [DATA]?
        - ENABled?
        - STARt

XPDR SUBSYSTEM

```
XPDR
    MEASure
        MS
            BD60
                [DATA]?
                ENABled?
                STARt
            DIVersity
                [DATA]?
                ENABled?
                STARt
            ADDress
                [DATA]?
                ENABled?
                STARt
                    [AUTO]
                    MANual
            POWer
                [DATA]?
                ENABled?
                STARt
            PTIMing
                [DATA]
                    SPACing?
                    WIDTh?
                ENABled?
                STARt
            RDELay
                [DATA]?
                ENABled?
                STARt
            RDRoop
                [DATA]?
                ENABled?
                STARt
            RJITter
                [DATA]?
                ENABled?
                STARt
```

```
XPDR
    MEASure
        MS
            RRATio
                [DATA]
                    PERCent?
                    [STATe]?
                ENABled?
                STARt
            SLS
                [DATA]?
                ENABled?
                STARt
            SQUitter
                [DATA]?
                ENABled?
                STARt
            UF0
                [DATA]?
                ENABled?
                STARt
            UF11
                [DATA]?
                ENABled?
                STARt
            UF16
                [DATA]?
                ENABled?
                STARt
            UF20
                [DATA]?
                ENABled?
                STARt
            UF21
                [DATA]?
                ENABled?
                STARt
            UF24
                [DATA]?
                ENABled?
                STARt
```

XPDR SUBSYSTEM

- XPDR
  - MEASure
    - MS
      - UF4
        - [DATA]?
        - ENABled?
        - STARt
      - UF5
        - [DATA]?
        - ENABled?
        - STARt
    - MSACall
      - ACALl
        - [DATA]?
        - ENABled?
        - STARt
      - IRADdress
        - [DATA]?
        - ENABled?
        - STARt
      - IRDelay
        - [DATA]?
        - ENABled?
        - STARt
      - IRJitter
        - [DATA]?
        - ENABled?
        - STARt
      - IRRatio
        - [DATA]
          - PERCent?
          - [STATe]?
        - ENABled?
        - STARt
        - STOP
  - PLIMits\?

:XPDR

:ADDRess

:STATe

Parameters: <CPD>

address state

Valid values: Address state: [AUTO | MANual]. Values outside range are rejected and an error generated.

Description: Select whether the mode S transponder address is determined automatically or not (outside of transponder diagnostics mode).

Example: `XPDR:ADDR:STAT AUTO`

Determine transponder address for mode S interrogations automatically.

:XPDR

:ADDRess

:STATe?

Parameters: None

Response: <CRD>

address state

Returned values: address: [AUTO | MAN]

Description: Determine whether the transponder address is user entered or automatically detected.

Example: `XPDR:ADDR:STAT?`

:XPDR

:ADDRess

[:VALue]

Parameters: <NRf>

address

Valid values: address: integer. Valid values are 0 to 16777215. Values outside range are rejected and an error generated.

Description: Set the mode S transponder address (outside of transponder diagnostics mode) for use when not automatically determining the transponder address.

The address can also be entered in hexadecimal using #Hxxxxxx, or in octal using #Qxxxxxxxx, or in binary using #Bxxxxxxxxxxxxxxxxxxxxxxxx.

Example: `XPDR:ADDR: 238467`

Select an address for mode S interrogations.

:XPDR

:ADDRess

[:VALue]?

Parameters: None

Response: <NR1>

address

Returned values: address: integer. Values are in the range 0 to 16777215.

Description: Determine the transponder address. Always returns a decimal number.

Example: `XPDR:ADDR:VAL?`

:XPDR

:ANTenna

:BOTTom

Parameters: <NRf>, <NRf>

range, height

Valid values: range: real

height: real

Description: Set the horizontal distance (range) and vertical distance (height) from the test set antenna to the aircraft bottom antenna.

The values can be set at any time but will only be used if the instrument is set to use over the air (via test set antenna) and the bottom aircraft antenna is selected for measurements.

The entered values are interpreted as being in the current distance units (as set by SYST:UNIT:DIST).

Entered values will be clipped to the appropriate max/min limits for the current distance units and will be rounded to the nearest foot or 0.5 meter.

Example: `XPDR:ANT:BOTT 100, 20`

Set the bottom aircraft antenna to be 100 ft away and 20 ft higher than the test set antenna – assuming distance units are set to feet.

:XPDR

:ANTenna

:BOTTom?

Parameters: none

Response: <NR2>, <NR2>

range, height

Returned values: range: real

height: real

Description: Read back the range and height from the test set antenna to the bottom aircraft antenna.

Values are returned in the currently selected distance units (as set by SYST:UNIT:DIST).

Example: `XPDR:ANT:BOTT?`

:XPDR

:ANTenna

:GAIN

Parameters: <NRf>, <NRf>

gain 960, gain1030, gain1090

Valid values: gain 960: real. Valid values are 0.0 to 20.9. Values outside range are rejected and an error generated.
gain1030: real. Valid values are 0.0 to 20.9. Values outside range are rejected and an error generated.
gain1090: real. Valid values are 0.0 to 20.9. Values outside range are rejected and an error generated.

Description: Set the gain of the 6000 antenna at the transponder transmit and receive frequencies.

Example: `XPDR:ANT:GAIN 9.6, 9.5, 9.7`

Set gain of the 6000 antenna.

:XPDR

:ANTenna

:GAIN?

Parameters: None

Response: <NR2>, <NR2>

gain 960, gain1030, gain1090

Returned values: Gain 960: real. Values are in the range 0.0 to 20.9.
gain1030: real. Values are in the range 0.0 to 20.9.
gain1090: real. Values are in the range 0.0 to 20.9.

Description: Determine the gain of the 6000 antenna at the transponder transmit and receive frequencies.

Example: `XPDR:ANT:GAIN?`

:XPDR

:ANTenna

:SELect

Parameters: <CPD>

aircraft antenna selection

Valid values: aircraft antenna selection: [TOP | BOTTom]. Values other than those stated are rejected and an error generated.

Description: Set whether we are measuring the top or bottom antenna on the aircraft.

This command will generate an error if the currently selected config is an ATCRBS type and the command is selecting the top antenna – for atcrbs transponders, only the bottom antenna can be tested.

Example: `XPDR:ANT:SEL TOP`

Make measurements using the top aircraft antenna.

:XPDR

:ANTenna

:SELect?

Parameters: none

Response: <CRD>

aircraft antenna selection

Returned values: aircraft antenna selection: [TOP | BOTT]

Description: Determine whether measurements are being performed on the top or bottom aircraft antenna.

Example: `XPDR:ANT:SEL?`

:XPDR

:ANTenna

:TOP

Parameters: <NRf>, <NRf>

range, height

Valid values: range: real

height: real

Description: Set the horizontal distance (range) and vertical distance (height) from the test set antenna to the aircraft top antenna.

The values can be set at any time but will only be used if the instrument is set to use over the air (via test set antenna) and the top aircraft antenna is selected for measurements.

The entered values are interpreted as being in the current distance units (as set by SYST:UNIT:DIST).

Example: `XPDR:ANT:TOP 100, 35`

Set the top aircraft antenna to be 100 ft away and 35 ft higher than the test set antenna – assuming distance units are set to feet.

:XPDR

:ANTenna

:TOP?

Parameters: none

Response: <NR2>, <NR2>

range, height

Returned values: range: real

height: real

Description: Read back the distance and height from the test set antenna to the top aircraft antenna.

Values are returned in the currently selected distance units (as set by SYST:UNIT:DIST).

Example: `XPDR:ANT:TOP?`

:XPDR

:CCAPability

Parameters: <BOOLEAN PROGRAM DATA>

check capability state

Description: This sets whether the transponder is interrogated to determine which BDS tests can be performed. If set ON then tests that cannot be performed will not be attempted. If OFF then all tests are attempted.

Example: `XPDR:CCAP ON`

Enable checking the transponder capabilities.

:XPDR

:CCAPability?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

check capability state

Description: Determine whether the transponder is interrogated to determine which BDS registers are available.

Example: `XPDR:CCAP?`

:XPDR

:R47

Parameters: <BOOLEAN PROGRAM DATA>

check capability state

Description: This sets whether the transponder is interrogated to determine which BDS tests can be performed. If set ON then tests that cannot be performed will not be attempted. If OFF then all tests are attempted.

Example: `XPDR:R47 ON`

Enable checking the transponder capabilities.

:XPDR

:R47?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

check capability state

Description: Determine whether the transponder is interrogated to determine which BDS registers are available.

Example: `XPDR:R47?`

:XPDR

:CLOSs

:ANTenna

:MODE

Parameters: <CPD>

ant cable loss mode

Valid values: ant cable loss mode: [UDEFined | L25 | L50 | L75]. Values other than those stated are rejected and an error generated.

Description: Select whether one of the VIAVI supplied cables is used to connect to the antenna (the 6000 has the cable loss programmed into it) or a user cable is used and its cable loss must be entered using XPDR:CLOS:ANT.

Example: `XPDR:CLOS:ANT:MODE L50`

The user is using a VIAVI supplied 50 ft cable and the 6000 automatically handles its cable loss..

:XPDR

:CLOSs

:ANTenna

:MODE?

Parameters: none

Response: <CRD>

ant cable loss mode

Returned values: ant cable loss mode: [UDEF | L25 | L50 | L75]

Description: Determine whether we are using a VIAVI supplied cable or a user defined cable to connect to the antenna.

Example: `XPDR:CLOS:ANT:MODE?`

:XPDR

:CLOSs

:ANTenna

[:VALue]

Parameters: <NRf>

cable loss

Valid values: cable loss: real

Description: This command sets the cable loss (in dB) of the cable used to connect to the test set antenna. This command will always set the cable loss of the antenna cable even if the current selection is direct connect. This value is only used if XPDR:CLOS:ANT:MODE is UDEF.

The DME cable loss value is kept separate from the transponder cable loss value.

The value is in units of dB.

Example: `XPDR:CLOS:ANT 1.7`

Inform the instrument of the loss of the cable connected to the 6000 antenna.

:XPDR

:CLOSs

:ANTenna

[:VALue]?

Parameters: None

Response: <NR2>

cable loss

Returned values: cable loss: real

Description: Determine the loss of the antenna cable. This is the value used when XPDR:CLOS:ANT:MODE is set to UDEF.

Example: `XPDR:CLOS:ANT?`

:XPDR

:CLOSs

[:CURRent]

Parameters: <NRf>

cable loss

Valid values: Cable loss: real

Description: This command sets the cable loss (in dB) of the cable used to direct connect to the aircraft antenna or the cable used to connect to the test set antenna if performing over the air measurements.

The DME cable loss value is kept separate from the transponder cable loss value.

The value is in units of dB.

Example: `XPDR:CLOS 1.7`

Inform the instrument of the loss of the cable currently being used.

:XPDR

:CLOSs

[:CURRent]?

Parameters: None

Response: <NR2>

cable loss

Returned values: cable loss: real

Description: Determine the loss of the cable currently being used.

Example: `XPDR:CLOS?`

:XPDR

:CLOSs

:DIRect

Parameters: <NRf>

cable loss

Valid values: cable loss: real

Description: This command sets the cable loss (in dB) of the cable used to directly connect from the test set to the transponder under test. This command will always set the cable loss of the direct connect cable even if the current selection is antenna (over the air).

The DME cable loss value is kept separate from the transponder cable loss value.

The value is in units of dB.

Example: `XPDR:CLOS:DIR 1.7`

Inform the instrument of the loss of the direct connect cable.

:XPDR

:CLOSs

:DIRect?

Parameters: None

Response: <NR2>

cable loss

Returned values: cable loss: real

Description: Determine the loss of the direct connect cable.

Example: `XPDR:CLOS:DIR?`

:XPDR

:CONFig

:CURRent?

Parameters: none

Response: <STRING RESPONSE DATA>

config in use

Returned values: config in use: string

Description: Determine which config is in use. This string will be the same as that used to select the device using :XPDR:CONF[:SEL].

Example: `XPDR:CONF:CURR?`

:XPDR

:CONFig

:LIST?

Parameters: None

Response: <STRING RESPONSE DATA>, ..., <STRING RESPONSE DATA>

first config, ..., last config

Returned values: first config: string. Maximum length of 20 characters excluding quotes.
...
last config: string. Maximum length of 20 characters excluding quotes.

Description: List all configs available in the instrument. A config may then be selected using :XPDR:CONF with one of the strings returned by this command.

Example: `XPDR:CONF:LIST?`

List all the configs available.

:XPDR

:CONFig

:NUMBer?

Parameters: none

Response: <NR1>

number of configs

Returned values: number of configs: integer

Description: This command is to be used in conjunction with :XPDR:CONF:LIST. This command determines how many configs will be returned by the :XPDR:CONF:LIST? command. This can be useful in reserving space for the :XPDR:CONF:LIST? response.

Example: `XPDR:CONF:NUMB?`

Determine how many configs are available in the instrument.

:XPDR

:CONFig

[:SELect]

Parameters: <STRING PROGRAM DATA>

config

Valid values: config: string. Maximum length of 20 characters excluding quotes. Excess characters will be ignored.

Description: Select a config. The config string must be one of the strings read using :XPDR:CONF:LIST? or an error will be generated. Selecting a config will set which tests are to be performed and the limits for parameters.

Example: `XPDR:CONF “ATCRBS A”`

Select ATCRBS class A config.

:XPDR

:DIAGnostic

:ADDRess

Parameters: <NRf>

address

Valid values: address: integer. Valid values are 0 to 16777215. Values outside range are rejected and an error generated.

Description: Set the address to use in diagnostics mode

The address can also be entered in hexadecimal using #Hxxxxxx, or in octal using #Qxxxxxxxx, or in binary using #Bxxxxxxxxxxxxxxxxxxxxxxxx.

Example: `XPDR:DIAG:ADDR 238467`

Select an address for mode S interrogations in diagnostics mode.

:XPDR

:DIAGnostic

:ADDRess?

Parameters: None

Response: <NR1>

address

Returned values: address: integer. Values are in the range 0 to 16777215.

Description: Determine the selected manual transponder address.

Example: `XPDR:DIAG:ADDR?`

:XPDR

:DIAGnostic

:COUNt?

Parameters: none

Response: <NR1>, <NR1>

sent count, received count

Returned values: sent count: integer. Values are in the range 0 to 99999.

received count: integer. Values are in the range 0 to 99999.

Description: Determine the number of interrogations and replies. The value wraps back to 0 after 99999.

Example: `XPDR:DIAG:COUN?`

:XPDR

:DIAGnostic

:DATA?

Parameters: none

Response: < NR1>, < NR1>, < NR1>, < NR1>, < NR1>, < NR1>, < NR1>, < NR1>, < NR1>, < NR1>

valid, value1, value2, value3, value4, value5, value6, value7, value8, value9

Returned values: valid: integer

value1: integer

value2: integer

value3: integer

value4: integer

value5: integer

value6: integer

value7: integer

value8: integer

value9: integer

:XPDR

:DIAGnostic

:DATA? (cont)

Description: Return the diagnostics data in a decoded form. The returned data is dependent on the test being performed. If there is no valid data, valid is set to zero and all nine values will be returned as zero. Otherwise valid is set to 1 and the data is returned as defined in the following table.

Test	value1	value2	value3	value4	value5	value6	value7	value8	value9
UF0	DF	VS	CC	SL	RI	AC	AA	0	0
UF4	DF	FS	DR	UM	AC	AA	0	0	0
UF5	DF	FS	DR	UM	ID	AA	0	0	0
UF11	DF	CA	AA	0	0	0	0	0	0
UF16	DF	VS	SL	RI	AC	AA	MVu	MVl	0
UF20	DF	FS	DR	UM	AC	AA	MBu	MBl	0
UF21	DF	FS	DR	UM	ID	AA	MBu	MBl	0
Squitter	DF	CA	AA	PI	0	0	0	0	0
Mode A	ID	SPI	0	0	0	0	0	0	octal id
Mode C	AC	0	0	0	0	0	0	0	altitude
ITM A(A rply)	ID	SPI	0	0	0	0	1	0	octal id
ITM A(S rply)	DF	CA	AA	0	0	0	0	0	0
ITM C(C rply)	AC	0	0	0	0	0	1	0	altitude
ITM C(S rply)	DF	CA	AA	0	0	0	0	0	0
CW	0	0	0	0	0	0	0	0	0
DSP	meas	0	0	0	0	0	0	0	0

MVu and MBu are the upper 24 bits of the 56-bit value. MVl and MBl are the bottom 32 bits.

Example: `XPDR:DIAG:DATA?`

:XPDR

:DIAGnostic

:DTESt

Parameters: <CPD>

decoder test state

Valid values: decoder test state: [OFF | ILOW | IHIGh | OLOW | OHIGh]. Values other than those stated are rejected and an error generated.

Description: Sets whether the interrogation pulse timing should be changed from the normal to one of those used for the decoder test.

This is only used when Mode A or Mode C test is selected.

Example: `XPDR:DIAG:DTES OLOW`

Change the interrogation pulse to that used for the outer low test. The transponder should not reply since the timing of the pulses is out of spec.

:XPDR

:DIAGnostic

:DTESt?

Parameters: none

Response: <CRD>

decoder test state

Returned values: decoder test state: [OFF | ILOW | IHIG | OLOW | OHIG]

Description: Determine whether the interrogation pulse timing has been changed to one of the decoder test pulse timings.

Example: `XPDR:DIAG:DTES?`

:XPDR

:DIAGnostic

:GENerate

Parameters: <BOOLEAN PROGRAM DATA>

GEN IF during dsp measure state

Description: This sets whether the 6000 generates an output while performing transponder diagnostic dsp measure. If set then a signal is output at the IF stage of the generate chain.

Example: `XPDR:DIAG:GEN ON`

Enable a signal at IF generate stage.

:XPDR

:DIAGnostic

:GENerate?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

GEN IF during dsp measure state

Description: Determine whether a signal is generated at the IF stage during transponder diagnostics dsp measure.

Example: `XPDR:DIAG:GEN?`

:XPDR

:DIAGnostic

:PRF

Parameters: <NRf>

PRF

Valid values: PRF: integer. Valid values are 1 to 2500. Values outside range are rejected and an error generated.

Description: Set the pulse repetition frequency (PRF) of interrogations to the transponder.

Example: `XPDR:DIAG:PRF 78`

Select the PRF for interrogations in diagnostics mode.

:XPDR

:DIAGnostic

:PRF?

Parameters: none

Response: <NR1>

PRF

Returned values: PRF: integer. Values are in the range 1 to 2500.

Description: Determine the current PRF for diagnostics.

Example: `XPDR:DIAG:PRF?`

:XPDR

:DIAGnostic

:RATTenuation

Parameters: <NRf>

attenuation

Valid values: attenuation: integer. Valid values are 0 to 55. Values outside range are rejected and an error generated.

Description: Set the receiver attenuator to specified setting (units of dB). Since the attenuation is settable in discrete steps, the entered value will be rounded to the nearest valid value: 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55.

Example: `XPDR:DIAG:RATT 20`

Set the attenuator for the receiver.

:XPDR

:DIAGnostic

:RATTenuation?

Parameters: None

Response: <NR1>

attenuation

Returned values: attenuation: integer. Values are in the range 0 to 55.

Description: Determine the selected receiver attenuation.

Example: `XPDR:DIAG:RATT?`

:XPDR

:DIAGnostic

:SELect

Parameters: <CPD>

selected test

Valid values: selected test: [UF0 | UF4 | UF5 | UF11 | UF16 | UF20 | UF21 | SQUitter | A | C | IA | IC | IAS | ICS | CW | DSP]. Values other than those stated are rejected and an error generated.

Description: Selects which test to perform in diagnostics mode.

Example: `XPDR:DIAG:SEL UF20`

Select the UF20 test for diagnostics.

:XPDR

:DIAGnostic

:SELect?

Parameters: None

Response: <CRD>

selected test

Returned values: selected test: [UF0 | UF4 | UF5 | UF11 | UF16 | UF20 | UF21 | SQU | A | C | IA | IC | IAS | ICS | CW | DSP]

Description: Determine which test is being performed for transponder diagnostics.

Example: `XPDR:DIAG:SEL?`

:XPDR

:DIAGnostic

:SLS

:ATCRbs

Parameters: <CPD>

sls

Valid values: sls: [OFF | ZERO | MNINe]. Values other than those stated are rejected and an error generated.

Description: Selects whether the P2 SLS pulse is present and at what level when sending ATCRBS interrogations during diagnostics.

Example: `XPDR:DIAG:SLS:ATCR MNIN`

Select the P2 SLS pulse on, and at –9 dB below P1 pulse.

:XPDR

:DIAGnostic

:SLS

:ATCRbs?

Parameters: none

Response: <CRD>

sls

Returned values: sls: [OFF | ZERO | MNIN]

Description: Determine SLS state and level.

Example: `XPDR:DIAG:SLS:ATCR?`

:XPDR

:DIAGnostic

:SLS

MS

Parameters: <CPD>

sls state

Valid values: Sls state: [OFF | THRee | MTWelve]. Values other than those stated are rejected and an error generated.

Description: This sets SLS on (at +3 dB or –12 dB) or off (SLS on means P5 pulse present) for mode S interrogations during diagnostics mode.

Example: `XPDR:DIAG:SLS:MS THR`

Set SLS on (+3 dB) for mode S interrogations during diagnostics mode.

:XPDR

:DIAGnostic

:SLS

:MS?

Parameters: None

Response: <CRD>

sls state

Returned values: sls state: [OFF | THR | MTW]

Description: Determine whether SLS is on or off for diagnostics mode.

Example: `XPDR:DIAG:SLS:MS?`

:XPDR

:DIAGnostic

:STARt

Parameters: None

Description: This starts the currently selected diagnostic test. Interrogrations will be sent and any replies received from the transponder stored. It is possible to read back the last response received.

The interrogations are sent out at a fixed rate. To determine if a new interrogation has been sent use XPDR:DIAG:COUN?. Then get the response data using the XPDR:DIAG:DATA:… set of commands.

Example: `XPDR:DIAG:STAR`

Start diagnostics test.

:XPDR

:DIAGnostic

:STOP

Parameters: None

Description: This stops the current diagnostic test.

Example: `XPDR:DIAG:STOP`

Stop diagnostics test.

:XPDR

:DIAGnostic

:TLEVel

Parameters: <NRf>

level

Valid values: level: integer

Description: Set the level to transmit on. Always in units of dBm.

For the Antenna port, valid range is –2 dBm to –67 dBm
For the Direct Connect port, valid range is –47 dBm to –115 dBm.

Example: `XPDR:DIAG:TLEV –26`

Select the level for mode S interrogations in diagnostics mode.

:XPDR

:DIAGnostic

:TLEVel?

Parameters: none

Response: <NR1>

level

Returned values: level: integer. Values are in the range -115 to -2.

Description: Determine the selected transmit level.

Example: `XPDR:DIAG:TLEV?`

:XPDR

:DIVersity

[:ENABle]

Parameters: <BOOLEAN PROGRAM DATA>

diversity state

Description: This sets whether the diversity test will be performed. Diversity test will only be performed if it is enabled as part of selecting the config and this command is set to ON.

Example: `XPDR:DIV ON`

Enable diversity test as long as the test is not disabled in the config.

:XPDR

:DIVersity

[:ENABle]?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

diversity state

Description: Determine whether diversity test is enabled. Even if it is enabled the test will still not run if the selected config does not have the test enabled.

Example: `XPDR:DIV?`

:XPDR

:MEASure

:ATCRbs

:ACALl

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <CRD>
all-call test state, Mode A result, Mode C result

Returned values: all-call test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

Mode A result: [PASS | FAIL | INV | NDAT]

Mode C result: [PASS | FAIL | INV | NDAT]

Description: Return the data from the ATCRBS only all-call test.

First param is the overall all-call test state – passed, failed, or no data (no reply or test not run). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the all-call test – each one can be either pass or fail, or can be invalid (measurement could not be performed).

Example: `XPDR:MEAS:ATCR:ACAL?`

:XPDR

:MEASure

:ATCRbs

:ACALl

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

all-call test enabled

Description: Determine whether the ATCRBS only all-call test is enabled or not. This is set when the config is selected. If the all-call test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:ATCR:ACAL:ENAB?`

:XPDR

:MEASure

:ATCRbs

:ACALl

:STARt

Parameters: none

Description: This starts the ATCRBS only all-call test. If the all-call test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:ATCR:ACAL:STAR`

Start all-call test.

:XPDR

:MEASure

:ATCRbs

:DECoder

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>
decoder test state, inner A low, inner A high, outer A low, outer A high, inner C low, inner C high, outer C low, outer C high

Returned values: decoder test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

inner A low: [PASS | FAIL | INV | NDAT]

inner A high: [PASS | FAIL | INV | NDAT]

outer A low: [PASS | FAIL | INV | NDAT]

outer A high: [PASS | FAIL | INV | NDAT]

inner C low: [PASS | FAIL | INV | NDAT]

inner C high: [PASS | FAIL | INV | NDAT]

outer C low: [PASS | FAIL | INV | NDAT]

outer C high: [PASS | FAIL | INV | NDAT]

Description: Return the data from the decoder test.

The decoder test sends out Mode A interrogations with the P1 to P3 spacing altered so that it is:
a) just less than the minimum spec (outer low) – transponder should not reply
b) just greater than the minimum spec (inner low) – transponder should reply
c) just less than the maximum spec (inner high) – transponder should reply
d) just greater than the maximum spec (outer high) – transponder should not reply
It then repeats for mode C interrogations.

First param is the overall Mode A Duration test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the Mode A Duration test – each one can be either pass or fail, or can be invalid (measurement could not be performed).

Example: `XPDR:MEAS:ATCR:DEC?`

:XPDR

:MEASure

:ATCRbs

:DECoder

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

decoder test enabled

Description: Determine whether the decoder test is enabled or not. This is set when the config is selected. If the decoder test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:ATCR:DEC:ENAB?`

:XPDR

:MEASure

:ATCRbs

:DECoder

:STARt

Parameters: none

Description: This starts the decoder test. If the decoder test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:ATCR:DEC:STAR`

Start decoder test.

:XPDR

:MEASure

:ATCRbs

:POWer

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>

overall power test state, top ERP test state, top ERP value, bottom ERP test state, bottom ERP value, instantaneous ERP test state, instantaneous ERP value, top MTL test state, top MTL value, bottom MTL test state, bottom MTL value, instantaneous MTL test state, instantaneous MTL value, top MTL difference test state, top MTL difference value, bottom MTL difference test state, bottom MTL difference value, instantaneous MTL difference test state, instantaneous MTL difference value, top allcall MTL test state, top allcall MTL value, bottom allcall MTL test state, bottom allcall MTL value, instantaneous allcall MTL test state, instantaneous allcall MTL value

Returned values: overall power test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

top ERP test state: [PASS | FAIL | INV | NDAT]

top ERP value: real

bottom ERP test state: [PASS | FAIL | INV | NDAT]

bottom ERP value: real

instantaneous ERP test state: [PASS | FAIL | INV | NDAT]

instantaneous ERP value: real

top MTL test state: [PASS | FAIL | INV | NDAT]

top MTL value: real

bottom MTL test state: [PASS | FAIL | INV | NDAT]

bottom MTL value: real

instantaneous MTL test state: [PASS | FAIL | INV | NDAT]

instantaneous MTL value: real

top MTL difference test state: [PASS | FAIL | INV | NDAT]

top MTL difference value: real

:XPDR

:MEASure

:ATCRbs

:POWer

[:DATA]? (cont)

Returned values: bottom MTL difference test state: [PASS | FAIL | INV | NDAT]

bottom MTL difference value: real

instantaneous MTL difference test state: [PASS | FAIL | INV | NDAT]

instantaneous MTL difference value: real

top allcall MTL test state: [PASS | FAIL | INV | NDAT]

top allcall MTL value: real

bottom allcall MTL test state: [PASS | FAIL | INV | NDAT]

bottom allcall MTL value: real

instantaneous allcall MTL test state: [PASS | FAIL | INV | NDAT]

instantaneous allcall MTL value: real

Description: Return the data from the power test. The ERP and MTL measurements are performed on atcrbs replies. Data from both antennas is returned along with instantaneous data from the currently selected antenna.

First param is the overall power test state – passed, failed, or no data (no reply). Both the ERP and MTL tests must pass for this to return pass.

All remaining parameters are paired - a state and a value. The value is only meaningful if the state is PASS or FAIL.

Example: `XPDR:MEAS:ATCR:POW?`

:XPDR

:MEASure

:ATCRbs

:POWer

:ENABled?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

power test enabled

Description: Determine whether the power test is enabled or not. This is set when the config is selected. If the power test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:ATCR:POW:ENAB?`

:XPDR

:MEASure

:ATCRbs

:POWer

:STARt

Parameters: none

Description: This starts the power test. If the power test is disabled then an error will be given and the test will not start. Mode A interrogations are used to determine MTL and ERP.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:ATCR:POW:STAR`

Start power (MTL and ERP) test.

:XPDR

:MEASure

:ATCRbs

:PTIMing

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>
pulse timing test state, A F1 test state, A F1, A F2 test state, A F2, A F1F2 test state, A F1F2, C F1 test state, C F1, C F2 test state, C F2, C F1F2 test state, C F1F2

Returned values: Pulse timing test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

A F1 test state: [PASS | FAIL | INV | NDAT]

A F1: real

A F2 test state: [PASS | FAIL | INV | NDAT]

A F2: real

A F1F2 test state: [PASS | FAIL | INV | NDAT]

A F1F2: real

C F1 test state: [PASS | FAIL | INV | NDAT]

C F1: real

C F2 test state: [PASS | FAIL | INV | NDAT]

C F2: real

C F1F2 test state: [PASS | FAIL | INV | NDAT]

C F1F2: real

Description: Return the data from the pulse timing test.

First param is the overall pulse timing test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the pulse timing test – each one can be either pass or fail, or can be invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

All timing values are in micro seconds.

Example: `XPDR:MEAS:ATCR:PTIM?`

:XPDR

:MEASure

:ATCRbs

:PTIMing

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

pulse timing test enabled

Description: Determine whether the pulse timing test is enabled or not. This is set when the config is selected. If the pulse timing test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:ATCR:PTIM:ENAB?`

:XPDR

:MEASure

:ATCRbs

:PTIMing

:STARt

Parameters: none

Description: This starts the pulse timing (width and spacing) test. If the pulse timing test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:ATCR:PTIM:STAR`

Start pulse timing (width and spacing) test.

:XPDR

:MEASure

:ATCRbs

:RDELay

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR2>, <CRD>, <NR2>
reply delay test state, Mode A test state, Mode A, Mode C test state, Mode C

Returned values: reply delay test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

Mode A test state: [PASS | FAIL | INV | NDAT]

Mode A: real

Mode C test state: [PASS | FAIL | INV | NDAT]

Mode C: real

Description: Return the data from the reply delay test. Measurements are performed for mode A and mode C.

First param is the overall reply delay test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the reply delay test – each one can be either pass or fail, or can be invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Example: `XPDR:MEAS:ATCR:RDEL?`

:XPDR

:MEASure

:ATCRbs

:RDELay

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

reply delay test enabled

Description: Determine whether the reply delay test is enabled or not. This is set when the config is selected. If the reply delay test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:ATCR:RDEL:ENAB?`

:XPDR

:MEASure

:ATCRbs

:RDELay

:STARt

Parameters: none

Description: This starts the reply delay test. If the reply delay test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:ATCR:RDEL:STAR`

Start reply delay test.

:XPDR

:MEASure

:ATCRbs

:RDRoop

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR2>, <CRD>, <NR2>
reply droop test state, mode A state, mode A value, mode C state, mode C value

Returned values: Reply droop test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

mode A state: [PASS | FAIL | INV | NDAT]

mode A value: real

mode C state: [PASS | FAIL | INV | NDAT]

mode C value: real

Description: Return the data from the reply droop test.

First param is the overall reply droop test state – passed, failed, no data (no reply), or invalid (could not perform measurement).

Remaining params are the reply droop values (in dB) for mode A and mode C replies.

Example: `XPDR:MEAS:ATCR:RDR?`

:XPDR

:MEASure

:ATCRbs

:RDRoop

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

reply droop test enabled

Description: Determine whether the reply droop test is enabled or not. This is set when the config is selected. If the reply droop test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:ATCR:RDR:ENAB?`

:XPDR

:MEASure

:ATCRbs

:RDRoop

:STARt

Parameters: none

Description: This starts the reply droop test. If the reply droop test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:ATCR:RDR:STAR`

Start reply droop test.

:XPDR

:MEASure

:ATCRbs

:REPLy

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR2>, <CRD>, <NR2>
reply test state, mode A state, mode A value, mode C state, mode C value

Returned values: reply test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

mode A code state: [PASS | FAIL | INV | NDAT]

mode A code value: integer

mode A spi state: [PASS | FAIL | INV | NDAT]

mode A spi value: [YES | NO]

mode C raw state: [PASS | FAIL | INV | NDAT]

mode C raw value: integer

mode C altitude state: [PASS | FAIL | INV | NDAT]

mode C altitude value: integer

Description: Return the data from the reply test.

First param is the overall reply droop test state – passed, failed, no data (no reply), or invalid (could not perform measurement).

Remaining params are mode A code and spi and mode C altitude (raw and decoded).

Example: `XPDR:MEAS:ATCR:REPL?`

:XPDR

:MEASure

:ATCRbs

:REPLy

:ENABled?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

reply test enabled

Description: Determine whether the reply test is enabled or not. This is set when the config is selected. If the reply test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:ATCR:REPL:ENAB?`

:XPDR

:MEASure

:ATCRbs

:REPLy

:STARt

Parameters: None

Description: This starts the reply test. If the reply test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:ATCR:REPL:STAR`

Start reply test.

:XPDR

:MEASure

:ATCRbs

:RJITter

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR2>, <CRD>, <NR2>
reply jitter test state, Mode A test state, Mode A, Mode C test state, Mode C

Returned values: Reply jitter test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

Mode A test state: [PASS | FAIL | INV | NDAT]

Mode A: real

Mode C test state: [PASS | FAIL | INV | NDAT]

Mode C: real

Description: Return the data from the reply jitter test.

First param is the overall reply jitter test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the reply jitter test – each one can be either pass or fail, or can be invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Example: `XPDR:MEAS:ATCR:RJIT?`

:XPDR

:MEASure

:ATCRbs

:RJITter

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

reply jitter test enabled

Description: Determine whether the reply jitter test is enabled or not. This is set when the config is selected. If the reply jitter test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:ATCR:RJIT:ENAB?`

:XPDR

:MEASure

:ATCRbs

:RJITter

:STARt

Parameters: none

Description: This starts the reply jitter test. If the reply jitter test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:ATCR:RJIT:STAR`

Start reply jitter test.

:XPDR

:MEASure

:ATCRbs

:RRATio

[:DATA]

:PERCent?

Parameters: none

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>
reply ratio test state, Mode A test state, Mode A reply ratio, Mode C test state, Mode C reply ratio, Mode A low power test state, Mode A low power reply ratio, Mode C low power test state, Mode C low power reply ratio

Returned values: Reply ratio test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

Mode A test state: [PASS | FAIL | INV | NDAT]

Mode A reply ratio: integer

Mode C test state: [PASS | FAIL | INV | NDAT]

Mode C reply ratio: integer

Mode A low power test state: [PASS | FAIL | INV | NDAT]

Mode A low power reply ratio: integer

Mode C low power test state: [PASS | FAIL | INV | NDAT]

Mode C low power reply ratio: integer

Description: Return the data from the reply ratio test. The reply ratios are a percentage (0 to 100).

First param is the overall reply ratio test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the reply ratio test – each one can be either pass or fail, or can be invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Example: `XPDR:MEAS:ATCR:RRAT:PERC?`

:XPDR

:MEASure

:ATCRbs

:RRATio

[:DATA]

[:STATe]?

Parameters: none

Response: <CRD>, <CRD>, <CRD>, <CRD>, <CRD>
reply ratio test state, Mode A test state, Mode A reply ratio, Mode C test state, Mode C reply ratio

Returned values: Reply ratio test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

Mode A test state: [PASS | FAIL | INV | NDAT]

Mode A reply ratio: [PASS | FAIL]

Mode C test state: [PASS | FAIL | INV | NDAT]

Mode C reply ratio: [PASS | FAIL]

Description: Return the data from the reply ratio test.

This only provides a summary and does not return the results from the low power test. It is recommended that XPDR:MEAS:ATCR:RRAT:PERC? is used instead.

First param is the overall reply ratio test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the reply ratio test – each one can be either pass or fail, or can be invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Example: `XPDR:MEAS:ATCR:RRAT?`

:XPDR

:MEASure

:ATCRbs

:RRATio

:ENABled?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

reply ratio test enabled

Description: Determine whether the reply ratio test is enabled or not. This is set when the config is selected. If the reply ratio test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:ATCR:RRAT:ENAB?`

:XPDR

:MEASure

:ATCRbs

:RRATio

:STARt

Parameters: None

Description: This starts the reply ratio test. If the reply ratio test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:ATCR:RRAT:STAR`

Start reply ratio test.

:XPDR

:MEASure

:ATCRbs

:SLS

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <CRD>, <CRD>, <CRD>
sls test state, A –9 dB, A 0 dB, C –9 dB, C 0 dB

Returned values: sls test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

A –9 dB: [PASS | FAIL | INV | NDAT]

A 0 dB: [PASS | FAIL | INV | NDAT]

C –9 dB: [PASS | FAIL | INV | NDAT]

C 0 dB: [PASS | FAIL | INV | NDAT]

Description: Return the data from the sls test.

First param is the overall sls test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the sls test – each one can be either pass or fail, or can be invalid (measurement could not be performed).

Example: `XPDR:MEAS:ATCR:SLS?`

:XPDR

:MEASure

:ATCRbs

:SLS

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

sls test enabled

Description: Determine whether the sls test is enabled or not. This is set when the config is selected. If the sls test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:ATCR:SLS:ENAB?`

:XPDR

:MEASure

:ATCRbs

:SLS

:STARt

Parameters: none

Description: This starts the sls test. If the sls test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:ATCR:SLS:STAR`

Start sls test.

:XPDR

:MEASure

[:AUTO]?

Parameters: none

Response: <CRD>

auto test state

Returned values: auto test state: [PASS | FAIL | NDAT]

Description: Perform a transponder autotest and return status – runs all tests that are currently selected (using the config).

The command takes about a minute to perform the test before returning results.

The returned param is the overall auto test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests making up the auto test pass.

All other data can be read by sending the appropriate commands for the tests that were run as part of the auto test and XPDR:MEAS:CAP?.

Example: `XPDR:MEAS?`

:XPDR

:MEASure

:CAPabilities?

Parameters: none

Response: <CRD>, <CRD>, <CRD>, <NR1>
replies state, replies, xpdr level state, xpdr level

Returned values: replies state: [PASS | FAIL | INV | NDAT]

replies: [NONE | A | C | AC | S | AS | CS | ACS]

xpdr level state: [PASS | FAIL | INV | NDAT]

xpdr level: integer

Description: Indicates what replies were received from the transponder during the autotest, and the level of the (mode S) transponder.

First param indicates if the second param (replies) is valid or not.

Second param indicates which types of replies the transponder sent during the autotest.

Third param indicates if the fourth param (transponder level) is valid or not.

Fourth param indicates the Mode S transponder level.

Example: `XPDR:MEAS:CAP?`

:XPDR

:MEASure

:COUNt?

Parameters: none

Response: <NR1>

test count

Returned values: test count: integer. Values are in the range 0 to 99999.

Description: Determine the number of times round the test loop. The value wraps back to 0 after 99999. Incremented by one whenever new data is available from the test. Set to zero when a new test is started.

Example: `XPDR:MEAS:COUN?`

:XPDR

:MEASure

:FREQuency

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR1>
frequency test state, freq state, freq value

Returned values: Frequency test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

freq state: [PASS | FAIL | INV | NDAT]

freq value: integer

Description: Return the data from the frequency test.

First param is the overall frequency test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Second param is the frequency value (in Hz).

Example: `XPDR:MEAS:FREQ?`

:XPDR

:MEASure

:FREQuency

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

frequency test enabled

Description: Determine whether the frequency test is enabled or not. This is set when the config is selected. If the frequency test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:FREQ:ENAB?`

:XPDR

:MEASure

:FREQuency

:STARt

Parameters: none

Description: This starts the frequency test. If the frequency test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:FREQ:STAR`

Start frequency test.

:XPDR

:MEASure

:MS

:BD10

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>
bds10 test state, df state, df, sub network state, sub network, enh protocol indicator state, enh protocol indicator, spec serv cap state, spec serv cap, uelm cap state, uelm cap, delm cap state, delm cap, aircraft id cap state, aircraft id cap, surv ident cap state, surv ident cap

Returned values: bds10 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

df state: [PASS | FAIL | INV | NDAT]

df: integer

sub network state: [PASS | FAIL | INV | NDAT]

sub network: integer

enh protocol indicator state: [PASS | FAIL | INV | NDAT]

enh protocol indicator: integer

spec serv cap state: [PASS | FAIL | INV | NDAT]

spec serv cap: integer

uelm cap state: [PASS | FAIL | INV | NDAT]

uelm cap: integer

delm cap state: [PASS | FAIL | INV | NDAT]

delm cap: integer

aircraft id cap state: [PASS | FAIL | INV | NDAT]

aircraft id cap: integer

surv ident cap state: [PASS | FAIL | INV | NDAT]

surv ident cap: integer

:XPDR

:MEASure

:MS

:BD10

[:DATA]? (cont)

Description: Return the data from the mode S bds 1,0 test.

First param is the overall bds 1,0 test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

The remaining params are paired – a status and a value. The value is only meaningful if the status is PASS or FAIL.

Example: `XPDR:MEAS:MS:BD10?`

:XPDR

:MEASure

:MS

:BD10

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

bds10 test enabled

Description: Determine whether the mode S bds 1,0 test is enabled or not. This is set when the config is selected. If the bds 1,0 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:BD10:ENAB?`

:XPDR

:MEASure

:MS

:BD10

:STARt

Parameters: none

Description: This starts the mode S bds 1,0 test. If the bds1,0 test is disabled in the config then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:BD10:STAR`

Start mode S bds 1,0 test.

:XPDR

:MEASure

:MS

:BD17

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>

bds17 test state, df state, df, bds0_5 enabled, bds0_6 enabled, bds0_7 enabled, bds0_8 enabled, bds0_9 enabled, bds0_A enabled, bds2_0 enabled, bds2_1 enabled, bds4_0 enabled, bds4_1 enabled, bds4_2 enabled, bds4_3 enabled, bds4_4 enabled, bds4_5 enabled, bds4_8 enabled, bds5_0 enabled, bds5_1 enabled, bds5_2 enabled, bds5_3 enabled, bds5_4 enabled, bds5_5 enabled, bds5_6 enabled, bds5_F enabled, bds6_0 enabled

Returned values: bds17 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

df state: [PASS | FAIL | INV | NDAT]

df: integer

bds0_5 enabled: integer

bds0_6 enabled: integer

bds0_7 enabled: integer

bds0_8 enabled: integer

bds0_9 enabled: integer

bds0_A enabled: integer

bds2_0 enabled: integer

bds2_1 enabled: integer

bds4_0 enabled: integer

bds4_1 enabled: integer

bds4_2 enabled: integer

bds4_3 enabled: integer

bds4_4 enabled: integer

bds4_5 enabled: integer

:XPDR

:MEASure

:MS

:BD17

[:DATA]? (cont)

Returned values: bds4_8 enabled: integer

bds5_0 enabled: integer

bds5_1 enabled: integer

bds5_2 enabled: integer

bds5_3 enabled: integer

bds5_4 enabled: integer

bds5_5 enabled: integer

bds5_6 enabled: integer

bds5_F enabled: integer

bds6_0 enabled: integer

Description: Return the data from the mode S bds 1,7 test.

First param is the overall bds 1,7 test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Second and third param are linked, a status and a value. The value is only meaningful if the status is PASS or FAIL.

Remaining params indicate if the specified bds register is available to be read from the transponder or not.

Example: `XPDR:MEAS:MS:BD17?`

:XPDR

:MEASure

:MS

:BD17

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

bds17 test enabled

Description: Determine whether the mode S bds 1,7 test is enabled or not. This is set when the config is selected. If the bds 1,7 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:BD17:ENAB?`

:XPDR

:MEASure

:MS

:BD17

:STARt

Parameters: None

Description: This starts the mode S bds 1,7 test. If the bds1,7 test is disabled in the config then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:BD17:STAR`

Start mode S bds 1,7 test.

:XPDR

:MEASure

:MS

:BD18

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <ARBITRARY ASCII RESPONSE DATA>

bds18 test state, df state, df, bds18 data

Returned values: bds18 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

df state: [PASS | FAIL | INV | NDAT]

df: integer

bds18 data: string

Description: Return the data from the mode S bds 1,8 test.

First param is the overall bds 1,8 test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Second param is the status of the third param. If PASS or FAIL then the third param indicates the downlink format used to return the response.

Fourth param holds the 56-bit value returned. Output as a 14 digit hex number.

Example: `XPDR:MEAS:MS:BD18?`

:XPDR

:MEASure

:MS

:BD18

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

bds18 test enabled

Description: Determine whether the mode S bds 1,8 test is enabled or not. This is set when the config is selected. If the bds 1,8 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:BD18:ENAB?`

:XPDR

:MEASure

:MS

:BD18

:STARt

Parameters: None

Description: This starts the mode S bds 1,8 test. If the bds1,8 test is disabled in the config then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:BD18:STAR`

Start mode S bds 1,8 test.

:XPDR

:MEASure

:MS

:BD19

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <ARBITRARY ASCII RESPONSE DATA>

bds19 test state, df state, df, bds19 data

Returned values: bds19 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

df state: [PASS | FAIL | INV | NDAT]

df: integer

bds19 data: string

Description: Return the data from the mode S bds 1,9 test.

First param is the overall bds 1,9 test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Second param is the status of the third param. If PASS or FAIL then the third param indicates the downlink format used to return the response.

Fourth param holds the 56-bit value returned. Output as a 14 digit hex number.

Example: `XPDR:MEAS:MS:BD19?`

:XPDR

:MEASure

:MS

:BD19

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

bds19 test enabled

Description: Determine whether the mode S bds 1,9 test is enabled or not. This is set when the config is selected. If the bds 1,9 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:BD19:ENAB?`

:XPDR

:MEASure

:MS

:BD19

:STARt

Parameters: None

Description: This starts the mode S bds 1,9 test. If the bds1,9 test is disabled in the config then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:BD19:STAR`

Start mode S bds 1,9 test.

:XPDR

:MEASure

:MS

:BD1A

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <ARBITRARY ASCII RESPONSE DATA>

bds1A test state, df state, df, bds1A data

Returned values: bds1A test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

df state: [PASS | FAIL | INV | NDAT]

df: integer

bds1A data: string

Description: Return the data from the mode S bds 1,A test.

First param is the overall bds 1,A test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Second param is the status of the third param. If PASS or FAIL then the third param indicates the downlink format used to return the response.

Fourth param holds the 56-bit value returned. Output as a 14 digit hex number.

Example: `XPDR:MEAS:MS:BD1A?`

:XPDR

:MEASure

:MS

:BD1A

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

bds1A test enabled

Description: Determine whether the mode S bds 1,A test is enabled or not. This is set when the config is selected. If the bds 1,A test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:BD1A:ENAB?`

:XPDR

:MEASure

:MS

:BD1A

:STARt

Parameters: None

Description: This starts the mode S bds 1,A test. If the bds1,A test is disabled in the config then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:BD1A:STAR`

Start mode S bds 1,A test.

:XPDR

:MEASure

:MS

:BD1B

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <ARBITRARY ASCII RESPONSE DATA>

bds1B test state, df state, df, bds1B data

Returned values: bds1B test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

df state: [PASS | FAIL | INV | NDAT]

df: integer

bds1B data: string

Description: Return the data from the mode S bds 1,B test.

First param is the overall bds 1,B test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Second param is the status of the third param. If PASS or FAIL then the third param indicates the downlink format used to return the response.

Fourth param holds the 56-bit value returned. Output as a 14 digit hex number.

Example: `XPDR:MEAS:MS:BD1B?`

:XPDR

:MEASure

:MS

:BD1B

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

bds1B test enabled

Description: Determine whether the mode S bds 1,B test is enabled or not. This is set when the config is selected. If the bds 1,B test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:BD1B:ENAB?`

:XPDR

:MEASure

:MS

:BD1B

:STARt

Parameters: None

Description: This starts the mode S bds 1,B test. If the bds1,B test is disabled in the config then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:BD1B:STAR`

Start mode S bds 1,B test.

:XPDR

:MEASure

:MS

:BD1C

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <ARBITRARY ASCII RESPONSE DATA>

bds1C test state, df state, df, bds1C data

Returned values: bds1C test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

df state: [PASS | FAIL | INV | NDAT]

df: integer

bds1C data: string

Description: Return the data from the mode S bds 1,C test.

First param is the overall bds 1,C test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Second param is the status of the third param. If PASS or FAIL then the third param indicates the downlink format used to return the response.

Fourth param holds the 56-bit value returned. Output as a 14 digit hex number.

Example: `XPDR:MEAS:MS:BD1C?`

:XPDR

:MEASure

:MS

:BD1C

:ENABled?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

bds1C test enabled

Description: Determine whether the mode S bds 1,C test is enabled or not. This is set when the config is selected. If the bds 1,C test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:BD1C:ENAB?`

:XPDR

:MEASure

:MS

:BD1C

:STARt

Parameters: None

Description: This starts the mode S bds 1,C test. If the bds1,C test is disabled in the config then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:BD1C:STAR`

Start mode S bds 1,C test.

:XPDR

:MEASure

:MS

:BD20

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

bds20 test state, df state, df, flight id state, flight id

Returned values: Bds20 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

df state: [PASS | FAIL | INV | NDAT]

df: integer

flight id state: [PASS | FAIL | INV | NDAT]

flight id: string

Description: Return the data from the mode S bds 2,0 test.

First param is the overall bds 2,0 test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

The second and third parameter are the downlink format used for the bds2,0 response. The value is only valid if the status is PASS or FAIL.

The remaining params are the flight id. This is an 8 character string.

Example: `XPDR:MEAS:MS:BD20?`

:XPDR

:MEASure

:MS

:BD20

:ENABled?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

bds20 test enabled

Description: Determine whether the mode S bds 2,0 test is enabled or not. This is set when the config is selected. If the bds 2,0 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:BD20:ENAB?`

:XPDR

:MEASure

:MS

:BD20

:STARt

Parameters: none

Description: This starts the mode S bds 2,0 test. If the bds 2,0 test is disabled in the config then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:BD20:STAR`

Start mode S bds 2,0 test.

:XPDR

:MEASure

:MS

:BD30

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>

bds30 test state, df state, df, ara state, ara, rac state, rac

Returned values: bds30 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

df state: [PASS | FAIL | INV | NDAT]

df: integer

ara state: [PASS | FAIL | INV | NDAT]

ara: integer

rac state: [PASS | FAIL | INV | NDAT]

rac: integer

Description: Return the data from the mode S bds 3,0 test.

First param is the overall bds 3,0 test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

The second and third parameter are the downlink format used for the bds3,0 response. The value is only valid if the status is PASS or FAIL.

The fourth and fifth params are the ara. The value is only valid if the status is PASS or FAIL.

The six and seventh params are the rac. The value is only valid if the status is PASS or FAIL.

Example: `XPDR:MEAS:MS:BD30?`

:XPDR

:MEASure

:MS

:BD30

:ENABled?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

bds30 test enabled

Description: Determine whether the mode S bds 3,0 test is enabled or not. This is set when the config is selected. If the bds 3,0 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:BD30:ENAB?`

:XPDR

:MEASure

:MS

:BD30

:STARt

Parameters: None

Description: This starts the mode S bds 3,0 test. If the bds3,0 test is disabled in the config then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:BD30:STAR`

Start mode S bds 3,0 test.

:XPDR

:MEASure

:MS

:BD40

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>
bds40 test state, df state, df, selected altitude state, selected altitude, barometric pressure setting status, barometric pressure setting

Returned values: Bds40 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

df state: [PASS | FAIL | INV | NDAT]

df: integer

selected altitude state: [PASS | FAIL | INV | NDAT]

selected altitude: integer

barometric pressure setting state: [PASS | FAIL | INV | NDAT]

barometric pressure setting: real

Description: Return the data from the mode S bds 4,0 test.

First param is the overall bds 4,0 test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Second and third param are the downlink format used to return the bds 4,0 data. If df state is not PASS or FAIL the the df value is meaningless.

Fourth and fifth parameter are the selected altitude.

Sixth and seventh parameter are the barometric pressure setting. Valid values for this are between 800.0 and 1209.5. But this value can also return 0 if the measurement could not be performed.

Example: `XPDR:MEAS:MS:BD40?`

:XPDR

:MEASure

:MS

:BD40

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

bds40 test enabled

Description: Determine whether the mode S bds 4,0 test is enabled or not. This is set when the config is selected. If the bds 4,0 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:BD40:ENAB?`

:XPDR

:MEASure

:MS

:BD40

:STARt

Parameters: none

Description: This starts the mode S bds 4,0 test. If the bds 4,0 test is disabled in the config then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:BD40:STAR`

Start mode S bds 4,0 test.

:XPDR

:MEASure

:MS

:BD50

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR1>
bds50 test state, df state, df, roll angle state, roll angle, true track angle state, true track angle, ground speed state, ground speed, track angle rate state, track angle rate, true air speed state, true air speed

Returned values: Bds50 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

df state: [PASS | FAIL | INV | NDAT]

df: integer

roll angle state: [PASS | FAIL | INV | NDAT]

roll angle: real

true track angle state: [PASS | FAIL | INV | NDAT]

true track angle: real

ground speed state: [PASS | FAIL | INV | NDAT]

ground speed: integer

track angle rate state: [PASS | FAIL | INV | NDAT]

track angle rate: real

true air speed state: [PASS | FAIL | INV | NDAT]

true air speed: integer

Description: Return the data from the mode S bds 5,0 test.

First param is the overall bds 5,0 test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Remaining params are paired – a status and a value. The value is only meaningful if the status is PASS or FAIL.

Example: `XPDR:MEAS:MS:BD50?`

:XPDR

:MEASure

:MS

:BD50

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

bds50 test enabled

Description: Determine whether the mode S bds 5,0 test is enabled or not. This is set when the config is selected. If the bds 5,0 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:BD50:ENAB?`

:XPDR

:MEASure

:MS

:BD50

:STARt

Parameters: none

Description: This starts the mode S bds 5,0 test. If the bds 5,0 test is disabled in the config then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:BD50:STAR`

Start mode S bds 5,0 test.

:XPDR

:MEASure

:MS

:BD60

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR1>, <CRD>, <NR1>
bds60 test state, df state, df, mag heading state, mag heading, ind air speed state, ind air speed, mach no state, mach no, invert vert vel state, invert vert vel, barometric alt state, barometric alt

Description: Return the data from the mode S bds 6,0 test.

First param is the overall bds 6,0 test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Remaining params are paired status/value items. The values are only valid if the status param is PASS or FAIL.

Example: `XPDR:MEAS:MS:BD60?`

:XPDR

:MEASure

:MS

:BD60

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

bds60 test enabled

Description: Determine whether the mode S bds 6,0 test is enabled or not. This is set when the config is selected. If the bds 6,0 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:BD60:ENAB?`

:XPDR

:MEASure

:MS

:BD60

:STARt

Parameters: none

Description: This starts the mode S bds 6,0 test. If the bds 6,0 test is disabled in the config then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:BD60:STAR`

Start mode S bds 6,0 test.

:XPDR

:MEASure

:MS

:DIVersity

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR2>
diversity test state, diversity isolation state , diversity isolation

Returned values: diversity test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

diversity isolation state: [PASS | FAIL | INV | NDAT]

diversity isolation: real

Description: Return the data from the mode S diversity test.

First param is the overall diversity test state – passed, failed, no data (no reply), or error (measurement could not be performed).

Second and third param is the diversity isolation value in dB. Only valid if diversity isolation status param is PASS or FAIL.

Example: `XPDR:MEAS:MS:DIV?`

:XPDR

:MEASure

:MS

:DIVersity

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

diversity test enabled

Description: Determine whether the mode S diversity test is enabled or not. This is set when the config is selected. If the diversity test is not enabled, then the test cannot be run.

Note also that the diversity test must also be enabled using XPDR:MS:DIV ON.

Example: `XPDR:MEAS:MS:DIV:ENAB?`

:XPDR

:MEASure

:MS

:DIVersity

:STARt

Parameters: none

Description: This starts the mode S diversity test. If the diversity test is disabled (either in the config or using XPDR:MS:DIV OFF) then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:DIV:STAR`

Start mode S diversity test.

:XPDR

:MEASure

:MS

:IADDress

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <CRD>
invalid address test state, invalid address state, test result

Returned values: invalid address test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

invalid address state: [PASS | FAIL | INV | NDAT]

test result: [PASS | FAIL]

Description: Return the data from the invalid address test.

First param is the overall invalid address test state – passed, failed, or no data (no reply).

The second param indicates if the third param is valid or not.

The third param gives the result of the test:
a) PASS if the transponder does not reply to either of the two tested address
b) FAIL if the transponder replies to at least one of the addresses

Example: `XPDR:MEAS:MS:IADD?`

:XPDR

:MEASure

:MS

:IADDress

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

spr test enabled

Description: Determine whether the mode S invalid address test is enabled or not. This is set when the config is selected. If the invalid address test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:IADD:ENAB?`

:XPDR

:MEASure

:MS

:IADDress

:STARt

[:AUTO]

Parameters: none

Description: This starts the mode S invalid address test, automatically determining the two addresses to perform the test at. If the invalid address test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:IADD:STAR`

Start invalid address test – using internally generated addresses.

:XPDR

:MEASure

:MS

:IADDress

:STARt

:MANual

Parameters: <NRf>, <NRf>

invalid address 1, invalid address 2

Valid values: Invalid address 1: integer. Valid values are 0 to 16777215. Values outside range are rejected and an error generated.

Invalid address 2: integer. Valid values are 0 to 16777215. Values outside range are rejected and an error generated.

Description: This starts the mode S invalid address test, using the two user specified addresses to perform the test at. If the invalid address test is disabled then an error will be given and the test will not start. If the two addresses are identical then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

The addresses can also be entered in hex, octal, or binary using the #H, #Q, and #B formats.

Example: `XPDR:MEAS:MS:IADD:STAR:MAN 4827, 77296`

Start invalid address test – using specific addresses.

:XPDR

:MEASure

:MS

:POWer

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>

MTL test state, top ant MTL state, top ant MTL value, bottom ant MTL state, bottom ant MTL value, instantaneous MTL state, instantaneous MTL value

Returned values: MTL test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

top ant MTL state: [PASS | FAIL | NDAT | INV]

top ant MTL value: real

bottom ant MTL state: [PASS | FAIL | NDAT | INV]

bottom ant MTL value: real

instantaneous MTL state: [PASS | FAIL | NDAT | INV]

instantaneous MTL value: real

Description: Return the data from the mode S power test. The MTL measurements are performed on Mode S replies. Data from the both antennas is returned.

An error will be generated if an ATCRBS config has been selected.

First param indicates the state of the MTL measurement. If this is INV then the MTL values returned are meaningless.

Second and third params are the top antenna MTL value (in dB) and status.

Fourth and fifth params are the bottom antenna MTL value (in dB) and status.

Sixth and seventh params are the instantaneous MTL value (in dB) from the currently selected antenna and status.

Example: `XPDR:MEAS:MS:POW?`

:XPDR

:MEASure

:MS

:POWer

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

power test enabled

Description: Determine whether the mode S power test is enabled or not. This is set when the config is selected. If the power test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:POW:ENAB?`

:XPDR

:MEASure

:MS

:POWer

:STARt

Parameters: none

Description: This starts the power test. If the power test is disabled then an error will be given and the test will not start. Mode S interrogations are used to determine MTL.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:POW:STAR`

Start mode S power (MTL) test.

:XPDR

:MEASure

:MS

:PTIMing

[:DATA]

:SPACing?

Parameters: none

Response: <CRD>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>
pulse timing test state, spacing12 state, spacing12 value, spacing13 state, spacing13 value, spacing14 state, spacing14 value, spacing1d state, spacing1d value

Returned values: pulse timing test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

spacing12 state: [PASS | FAIL | INV | NDAT]

spacing12 value: real

spacing13 state: [PASS | FAIL | INV | NDAT]

spacing13 value: real

spacing14 state: [PASS | FAIL | INV | NDAT]

spacing14 value: real

spacing1d state: [PASS | FAIL | INV | NDAT]

spacing1d value: real

Description: Return the data from the mode S pulse timing test.

First param is the overall pulse timing test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the pulse timing test – each one can be either pass or fail, or can be invalid (measurement could not be performed).

Example: `XPDR:MEAS:MS:PTIM:SPAC?`

:XPDR

:MEASure

:MS

:PTIMing

[:DATA]

:WIDTh?

Parameters: None

Response: <CRD>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>
pulse timing test state, width1 state, width1 value, width2 state, width2 value, width3 state, width3 value, width4 state, width4 value

Returned values: pulse timing test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

width1 state: [PASS | FAIL | INV | NDAT]

width1 value: real

width2 state: [PASS | FAIL | INV | NDAT]

width2 value: real

width3 state: [PASS | FAIL | INV | NDAT]

width3 value: real

width4 state: [PASS | FAIL | INV | NDAT]

width4 value: real

Description: Return the data from the mode S pulse timing test.

First param is the overall pulse timing test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the pulse timing test – each one can be either pass or fail, or can be invalid (measurement could not be performed).

Example: `XPDR:MEAS:MS:PTIM:WIDT?`

:XPDR

:MEASure

:MS

:PTIMing

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

pulse timing test enabled

Description: Determine whether the mode S pulse timing test is enabled or not. This is set when the config is selected. If the pulse timing test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:PTIM:ENAB?`

:XPDR

:MEASure

:MS

:PTIMing

:STARt

Parameters: none

Description: This starts the mode S pulse timing (width and spacing) test. If the pulse timing test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:PTIM:STAR`

Start mode S pulse timing (width and spacing) test.

:XPDR

:MEASure

:MS

:RDELay

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR2>
reply delay test state, Mode S reply delay state, Mode S reply delay

Returned values: reply delay test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

Mode S reply delay state: [PASS | FAIL | INV | NDAT]

Mode S reply delay: real

Description: Return the data from the mode S reply delay test.

First param is the overall reply delay test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Second param is the reply delay value. Only valid if first param is PASS or FAIL.

Example: `XPDR:MEAS:MS:RDEL?`

:XPDR

:MEASure

:MS

:RDELay

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

reply delay test enabled

Description: Determine whether the mode S reply delay test is enabled or not. This is set when the config is selected. If the reply delay test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:RDEL:ENAB?`

:XPDR

:MEASure

:MS

:RDELay

:STARt

Parameters: none

Description: This starts the mode S reply delay test. If the reply delay test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:RDEL:STAR`

Start mode S reply delay test.

:XPDR

:MEASure

:MS

:RDRoop

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR2>, <CRD>, <NR2>
reply droop test state, short test state, short, long test state, long

Returned values: reply droop test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

short test state: [PASS | FAIL | INV | NDAT]

short: real

long test state: [PASS | FAIL | INV | NDAT]

long: real

Description: Return the data from the mode S reply droop test. The test is performed on both short (56 bit) replies and long (112 bit) replies.

First param is the overall reply droop test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the reply droop test – each one can be either pass or fail, or can be invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Example: `XPDR:MEAS:MS:RDR?`

:XPDR

:MEASure

:MS

:RDRoop

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

reply droop test enabled

Description: Determine whether the mode S reply droop test is enabled or not. This is set when the config is selected. If the reply droop test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:RDR:ENAB?`

:XPDR

:MEASure

:MS

:RDRoop

:STARt

Parameters: none

Description: This starts the mode S reply droop test. If the reply droop test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:RDR:STAR`

Start mode S reply droop test.

:XPDR

:MEASure

:MS

:RJITter

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR2>
reply jitter test state, Mode S reply jitter state, Mode S reply jitter

Returned values: reply jitter test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

Mode S reply jitter state: [PASS | FAIL | INV | NDAT]

Mode S reply jitter: real

Description: Return the data from the mode S reply jitter test.

First param is the overall reply jitter test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Second param is the reply jitter value. Only valid if first param is PASS or FAIL.

Example: `XPDR:MEAS:MS:RJIT?`

:XPDR

:MEASure

:MS

:RJITter

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

reply jitter test enabled

Description: Determine whether the mode S reply jitter test is enabled or not. This is set when the config is selected. If the reply jitter test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:RJIT:ENAB?`

:XPDR

:MEASure

:MS

:RJITter

:STARt

Parameters: none

Description: This starts the mode S reply jitter test. If the reply jitter test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:RJIT:STAR`

Start mode S reply jitter test.

:XPDR

:MEASure

:MS

:RRATio

[:DATA]

:PERCent?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>
reply ratio test state, Mode S reply ratio state, Mode S reply ratio value, Mode S low power reply ratio state, Mode S low power reply ratio value

Returned values: reply ratio test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

Mode S reply ratio state: [PASS | FAIL | INV | NDAT]

Mode S reply ratio value: integer

Mode S low power reply ratio state: [PASS | FAIL | INV | NDAT]

Mode S low power reply ratio value: integer

Description: Return the data from the mode S reply ratio test. The reply ratios are returned as percentages (0 to 100)

This only provides a summary and does not return the results from the low power test. It is recommended that XPDR:MEAS:MS:RRAT:PERC? is used instead.

First param is the overall reply ratio test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Second param is the reply ratio state. This indicates if the reply ratio value is valid or not. If the state is PASS or FAIL then the value is valid and indicates if the reply ratio test passed or failed.

Example: `XPDR:MEAS:MS:RRAT:PERC?`

:XPDR

:MEASure

:MS

:RRATio

[:DATA]

[:STATe]?

Parameters: None

Response: <CRD>, <CRD>, <CRD>
reply ratio test state, Mode S reply ratio state, Mode S reply ratio value

Returned values: reply ratio test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

Mode S reply ratio state: [PASS | FAIL | INV | NDAT]

Mode S reply ratio value: [PASS | FAIL]

Description: Return the data from the mode S reply ratio test.

This only provides a summary and does not return the results from the low power test. It is recommended that XPDR:MEAS:MS:RRAT:PERC? is used instead.

First param is the overall reply ratio test state – passed, failed, no data (no reply), or invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Second param is the reply ratio state. This indicates if the reply ratio value is valid or not. If the state is PASS or FAIL then the value is valid and indicates if the reply ratio test passed or failed.

Example: `XPDR:MEAS:MS:RRAT?`

:XPDR

:MEASure

:MS

:RRATio

:ENABled?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

reply ratio test enabled

Description: Determine whether the mode S reply ratio test is enabled or not. This is set when the config is selected. If the reply ratio test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:RRAT:ENAB?`

:XPDR

:MEASure

:MS

:RRATio

:STARt

Parameters: None

Description: This starts the mode S reply ratio test. If the reply ratio test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:RRAT:STAR`

Start mode S reply ratio test.

:XPDR

:MEASure

:MS

:SLS

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <CRD>, <CRD>, <CRD>
sls test state, on state, on, off state, off

Returned values: sls test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

on state: [PASS | FAIL | INV | NDAT]

on: [REPL | NREP]

off state: [PASS | FAIL | INV | NDAT]

off: [REPL | NREP]

Description: Return the data from the sls test.

First param is the overall sls test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the spr test – each one can be either reply or no-reply, or can be invalid (measurement could not be performed).

The transponder should reply when SLS is off (no P5 pulse) and should not reply when SLS is on (P5 pulse overlaying sync phase reversal).

Example: `XPDR:MEAS:MS:SLS?`

:XPDR

:MEASure

:MS

:SLS

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

sls test enabled

Description: Determine whether the mode S sls test is enabled or not. This is set when the config is selected. If the sls test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:SLS:ENAB?`

:XPDR

:MEASure

:MS

:SLS

:STARt

Parameters: none

Description: This starts the mode S sls test. If the sls test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:SLS:STAR`

Start sls test.

:XPDR

:MEASure

:MS

:SQUitter

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR2>, <CRD>, <CRD>
squitter test state, period state, period, DF17 state, DF17s

Returned values: squitter test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

period state: [PASS | FAIL | INV | NDAT]

period: real

DF17 state: [PASS | INV | NDAT]

DF17s: [YES | NO]

Description: Return the data from the squitter test.

First param is the overall squitter test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

The second param indicates if the acquisition squitter period measurement passed, failed or is invalid.

The third parameter is the acquisition squitter period value – unless the second param is invalid, in which case the value is meaningless.

The fourth/fifth param indicates if extended DF17 squitters were detected while the squitter test was running. This has no effect on the state of the squitter test.

Example: `XPDR:MEAS:MS:SQU?`

:XPDR

:MEASure

:MS

:SQUitter

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

squitter test enabled

Description: Determine whether the mode S squitter test is enabled or not. This is set when the config is selected. If the squitter test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:SQU:ENAB?`

:XPDR

:MEASure

:MS

:SQUitter

:STARt

Parameters: none

Description: This starts the mode S squitter test. If the squitter test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:SQU:STAR`

Start squitter test.

:XPDR

:MEASure

:MS

:UF0

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>
UF0 test state, DF state, DF, VS state, VS, CC state, CC, SL state, SL, RI state, RI, AC state, AC, AA state, AA, alt state, altitude, altitude units, alt compare state, alt compare, address compare state, address compare

Returned values: UF0 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

DF state: [PASS | FAIL | INV | NDAT]

DF: integer

VS state: [PASS | FAIL | INV | NDAT]

VS: integer

CC state: [PASS | FAIL | INV | NDAT]

CC: integer

SL state: [PASS | FAIL | INV | NDAT]

SL: integer

RI state: [PASS | FAIL | INV | NDAT]

RI: integer

AC state: [PASS | FAIL | INV | NDAT]

AC: integer

AA state: [PASS | FAIL | INV | NDAT]

AA: integer

alt state: [PASS | FAIL | INV | NDAT]

altitude: integer

altitude units: [FEET | MET]

:XPDR

:MEASure

:MS

:UF0

[:DATA]? (cont)

Returned values: alt compare state: [PASS | FAIL | INV | NDAT]

alt compare: [MATC | DIFF]

address compare state: [PASS | FAIL | INV | NDAT]

address compare: [MATC | DIFF]

Description: Return the data from the UF0 test.

First param is the overall UF0 test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the UF0 test – each one can be either reply or no-reply, or can be invalid (measurement could not be performed).

Example: `XPDR:MEAS:MS:UF0?`

:XPDR

:MEASure

:MS

:UF0

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

UF0 test enabled

Description: Determine whether the mode S UF0 test is enabled or not. This is set when the config is selected. If the UF0 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:UF0:ENAB?`

:XPDR

:MEASure

:MS

:UF0

:STARt

Parameters: none

Description: This starts the mode S UF0 test. If the UF0 test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:UF0:STAR`

Start UF0 test.

:XPDR

:MEASure

:MS

:UF11

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <NR1>, <CRD>, <CRD>, <NR1>, <CRD>, <HEXADECIMAL NUMERIC RESPONSE DATA>, <CRD>, <HEXADECIMAL NUMERIC RESPONSE DATA>, <HEXADECIMAL NUMERIC RESPONSE DATA>

UF11 test state, DF state, DF, CA state, CA, AA state, AA, PI state, PI, II lockout test state, II lockout state, II lockout timer, SI lockout test state, SI lockout state, SI lockout timer, II state, II, SI state, SI upper, SI lower

Returned values: UF11 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

DF state: [PASS | FAIL | INV | NDAT]

DF: integer

CA state: [PASS | FAIL | INV | NDAT]

CA: integer

AA state: [PASS | FAIL | INV | NDAT]

AA: integer

PI state: [PASS | FAIL | INV | NDAT]

PI: integer

II lockout test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

II lockout state: [PASS | FAIL | INV | NDAT]

II lockout timer: integer

SI lockout test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

SI lockout state: [PASS | FAIL | INV | NDAT]

SI lockout timer: integer

II state: [PASS | FAIL | INV | NDAT]

II: integer

:XPDR

:MEASure

:MS

:UF11

[:DATA]? (cont)

Returned values: SI state: [PASS | FAIL | INV | NDAT]

SI upper: integer

SI lower: integer

Description: Return data from the UF11 test. The UF11 test is performed for all valid combinations of CL and IC – this equates to II codes of 0 to 15, and SI codes of 1 to 63.

First param is the overall UF11 test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the UF11 test – each one can be either reply or no-reply, or can be invalid (measurement could not be performed).

Example: `XPDR:MEAS:MS:UF11?`

:XPDR

:MEASure

:MS

:UF11

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

UF11 test enabled

Description: Determine whether the mode S UF11 test is enabled or not. This is set when the config is selected. If the UF11 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:UF11:ENAB?`

:XPDR

:MEASure

:MS

:UF11

:STARt

Parameters: none

Description: This starts the mode S UF11 test. If the UF11 test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP). All 79 permutations of II and SI code will be tested. It is not possible to run each permutation individually. After all permutations have been tested, the results for each can be read back individually.

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:UF11:STAR`

Start UF11 test.

:XPDR

:MEASure

:MS

:UF16

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>
UF16 test state, DF state, DF, VS state, VS, SL state, SL, RI state, RI, AC state, AC, AA state, AA, alt state, altitude, altitude units, AC compare state, AC compare, address compare state, address compare, MV state, MV

Returned values: UF16 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

DF state: [PASS | FAIL | INV | NDAT]

DF: integer

VS state: [PASS | FAIL | INV | NDAT]

VS: integer

SL state: [PASS | FAIL | INV | NDAT]

SL: integer

RI state: [PASS | FAIL | INV | NDAT]

RI: integer

AC state: [PASS | FAIL | INV | NDAT]

AC: integer

AA state: [PASS | FAIL | INV | NDAT]

AA: integer

alt state: [PASS | FAIL | INV | NDAT]

altitude: integer

altitude units: [FEET | MET]

AC compare state: [PASS | FAIL | INV | NDAT]

AC compare: [MATC | DIFF]

:XPDR

:MEASure

:MS

:UF16

[:DATA]? (cont)

Returned values: address compare state: [PASS | FAIL | INV | NDAT]

address compare: [MATC | DIFF]

MV state: [PASS | FAIL | INV | NDAT]

Description: Return the data from the UF16 test.

First param is the overall UF16 test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the UF16 test – each one can be either reply or no-reply, or can be invalid (measurement could not be performed). The MV is output as a 56-bit hexadecimal number.

Example: `XPDR:MEAS:MS:UF16?`

:XPDR

:MEASure

:MS

:UF16

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

UF16 test enabled

Description: Determine whether the mode S UF16 test is enabled or not. This is set when the config is selected. If the UF16 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:UF16:ENAB?`

:XPDR

:MEASure

:MS

:UF16

:STARt

Parameters: none

Description: This starts the mode S UF16 test. If the UF16 test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:UF16:STAR`

Start UF16 test.

:XPDR

:MEASure

:MS

:UF20

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>
UF20 test state, DF state, DF, FS state, FS, DR state, DR, UM state, UM, AC state, AC, AA state, AA, alt state, altitude, altitude units, alt compare state, alt compare, address compare state, address compare, MB state, MB

Returned values: UF20 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

DF state: [PASS | FAIL | INV | NDAT]

DF: integer

FS state: [PASS | FAIL | INV | NDAT]

FS: integer

DR state: [PASS | FAIL | INV | NDAT]

DR: integer

UM state: [PASS | FAIL | INV | NDAT]

UM: integer

AC state: [PASS | FAIL | INV | NDAT]

AC: integer

AA state: [PASS | FAIL | INV | NDAT]

AA: integer

alt state: [PASS | FAIL | INV | NDAT]

altitude: integer

altitude units: [FEET | MET]

alt compare state: [PASS | FAIL | INV | NDAT]

alt compare: [MATC | DIFF]

:XPDR

:MEASure

:MS

:UF20

[:DATA]? (cont)

Returned values: address compare state: [PASS | FAIL | INV | NDAT]

address compare: [MATC | DIFF]

MB state: [PASS | FAIL | INV | NDAT]

Description: Return the data from the UF20 test.

First param is the overall UF20 test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the UF20 test – each one can be either reply or no-reply, or can be invalid (measurement could not be performed). The MB is output as a 56-bit hexadecimal number.

Example: `XPDR:MEAS:MS:UF20?`

:XPDR

:MEASure

:MS

:UF20

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

UF20 test enabled

Description: Determine whether the mode S UF20 test is enabled or not. This is set when the config is selected. If the UF20 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:UF20:ENAB?`

:XPDR

:MEASure

:MS

:UF20

:STARt

Parameters: none

Description: This starts the mode S UF20 test. If the UF20 test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:UF20:STAR`

Start UF20 test.

:XPDR

:MEASure

:MS

:UF21

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <OCTAL NUMERIC RESPONSE DATA>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

UF21 test state, DF state, DF, FS state, FS, DR state, DR, UM state, UM, ID state, ID, ID octal state, ID octal, AA state, AA, ID compare state, ID compare, address compare state, address compare, MB state, MB

Returned values: UF21 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

DF state: [PASS | FAIL | INV | NDAT]

DF: integer

FS state: [PASS | FAIL | INV | NDAT]

FS: integer

DR state: [PASS | FAIL | INV | NDAT]

DR: integer

UM state: [PASS | FAIL | INV | NDAT]

UM: integer

ID state: [PASS | FAIL | INV | NDAT]

ID: integer

ID octal state: [PASS | FAIL | INV | NDAT]

ID octal: integer

AA state: [PASS | FAIL | INV | NDAT]

AA: integer

ID compare state: [PASS | FAIL | INV | NDAT]

ID compare: [MATC | DIFF]

:XPDR

:MEASure

:MS

:UF21

[:DATA]? (cont)

Returned values: address compare state: [PASS | FAIL | INV | NDAT]

address compare: [MATC | DIFF]

MB state: [PASS | FAIL | INV | NDAT]

Description: Return the data from the UF21 test.

First param is the overall UF21 test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the UF21 test – each one can be either reply or no-reply, or can be invalid (measurement could not be performed). The MB is output as a 56-bit hexadecimal number.

Example: `XPDR:MEAS:MS:UF21?`

:XPDR

:MEASure

:MS

:UF21

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

UF21 test enabled

Description: Determine whether the mode S UF21 test is enabled or not. This is set when the config is selected. If the UF21 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:UF21:ENAB?`

:XPDR

:MEASure

:MS

:UF21

:STARt

Parameters: none

Description: This starts the mode S UF21 test. If the UF21 test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:UF21:STAR`

Start UF21 test.

:XPDR

:MEASure

:MS

:UF24

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>

UF24 test state, res DF state, res DF, res IIS state, res IIS, res IDS state, res IDS, res AA state, res AA, ack DF state, ack DF, ack KE state, ack KE, ack ND state, ack ND, ack TAS state, ack TAS, ack AA state, ack AA, clo DF state, clo DF, clo IIS state, clo IIS, clo IDS state, clo IDS, clo AA state, clo AA

Returned values: UF24 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

res DF state: [PASS | FAIL | INV | NDAT]

res DF: integer

res IIS state: [PASS | FAIL | INV | NDAT]

res IIS: integer

res IDS state: [PASS | FAIL | INV | NDAT]

res IDS: integer

res AA state: [PASS | FAIL | INV | NDAT]

res AA: integer

ack DF state: [PASS | FAIL | INV | NDAT]

ack DF: integer

ack KE state: [PASS | FAIL | INV | NDAT]

ack KE: integer

ack ND state: [PASS | FAIL | INV | NDAT]

ack ND: integer

ack TAS state: [PASS | FAIL | INV | NDAT]

ack TAS: integer

ack AA state: [PASS | FAIL | INV | NDAT]

:XPDR

:MEASure

:MS

:UF24

[:DATA]? (cont)

Returned values: ack AA: integer

clo DF state: [PASS | FAIL | INV | NDAT]

clo DF: integer

clo IIS state: [PASS | FAIL | INV | NDAT]

clo IIS: integer

clo IDS state: [PASS | FAIL | INV | NDAT]

clo IDS: integer

clo AA state: [PASS | FAIL | INV | NDAT]

clo AA: integer

Description: Return the data from the UF24 test.

First param is the overall UF24 test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the UF24 test – each one can be either reply or no-reply, or can be invalid (measurement could not be performed).

Example: `XPDR:MEAS:MS:UF24?`

:XPDR

:MEASure

:MS

:UF24

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

UF24 test enabled

Description: Determine whether the mode S UF24 test is enabled or not. This is set when the config is selected. If the UF24 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:UF24:ENAB?`

:XPDR

:MEASure

:MS

:UF24

:STARt

Parameters: none

Description: This starts the mode S UF24 test. If the UF24 test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:UF24:STAR`

Start UF24 test.

:XPDR

:MEASure

:MS

:UF4

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>
UF4 test state, DF state, DF, FS state, FS, DR state, DR, UM state, UM, AC state, AC, AA state, AA, alt state, altitude, altitude units, alt compare state, alt compare, address compare state, address compare

Returned values: UF4 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

DF state: [PASS | FAIL | INV | NDAT]

DF: integer

FS state: [PASS | FAIL | INV | NDAT]

FS: integer

DR state: [PASS | FAIL | INV | NDAT]

DR: integer

UM state: [PASS | FAIL | INV | NDAT]

UM: integer

AC state: [PASS | FAIL | INV | NDAT]

AC: integer

AA state: [PASS | FAIL | INV | NDAT]

AA: integer

alt state: [PASS | FAIL | INV | NDAT]

altitude: integer

altitude units: [FEET | MET]

alt compare state: [PASS | FAIL | INV | NDAT]

alt compare: [MATC | DIFF]

:XPDR

:MEASure

:MS

:UF4

[:DATA]?

Returned values: address compare state: [PASS | FAIL | INV | NDAT]

address compare: [MATC | DIFF]

Description: Return the data from the UF4 test.

First param is the overall UF4 test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the UF4 test – each one can be either reply or no-reply, or can be invalid (measurement could not be performed).

Example: `XPDR:MEAS:MS:UF4?`

:XPDR

:MEASure

:MS

:UF4

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

UF4 test enabled

Description: Determine whether the mode S UF4 test is enabled or not. This is set when the config is selected. If the UF4 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:UF4:ENAB?`

:XPDR

:MEASure

:MS

:UF4

:STARt

Parameters: none

Description: This starts the mode S UF4 test. If the UF4 test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:UF4:STAR`

Start UF4 test.

:XPDR

:MEASure

:MS

:UF5

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <OCTAL NUMERIC RESPONSE DATA>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <CRD>
UF5 test state, DF state, DF, FS state, FS, DR state, DR, UM state, UM, ID state, ID, ID octal state, ID octal, AA state, AA, ID compare state, ID compare, address compare state, address compare

Returned values: UF5 test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

DF state: [PASS | FAIL | INV | NDAT]

DF: integer

FS state: [PASS | FAIL | INV | NDAT]

FS: integer

DR state: [PASS | FAIL | INV | NDAT]

DR: integer

UM state: [PASS | FAIL | INV | NDAT]

UM: integer

ID state: [PASS | FAIL | INV | NDAT]

ID: integer

ID octal state: [PASS | FAIL | INV | NDAT]

ID octal: integer

AA state: [PASS | FAIL | INV | NDAT]

AA: integer

ID compare state: [PASS | FAIL | INV | NDAT]

ID compare: [MATC | DIFF]

address compare state: [PASS | FAIL | INV | NDAT]

:XPDR

:MEASure

:MS

:UF5

[:DATA]? (cont)

Returned values: address compare: [MATC | DIFF]

Description: Return the data from the UF5 test.

First param is the overall UF5 test state – passed, failed, or no data (no reply). This can only be pass if all the individual tests pass.

Remaining params are the results for the individual tests making up the UF5 test – each one can be either reply or no-reply, or can be invalid (measurement could not be performed).

Example: `XPDR:MEAS:MS:UF5?`

:XPDR

:MEASure

:MS

:UF5

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

UF5 test enabled

Description: Determine whether the mode S UF5 test is enabled or not. This is set when the config is selected. If the UF5 test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MS:UF5:ENAB?`

:XPDR

:MEASure

:MS

:UF5

:STARt

Parameters: none

Description: This starts the mode S UF5 test. If the UF5 test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MS:UF5:STAR`

Start UF5 test.

:XPDR

:MEASure

:MSACall

:ACALl

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <STRING RESPONSE DATA>
overall allcall test state, allcall state, allcall, allcall address state, allcall address, tail number state, tail number, country state, country

Returned values: overall allcall test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

allcall state: [PASS | FAIL | INV]

allcall: [PASS | FAIL]

allcall address state: [PASS | FAIL | INV]

allcall address: integer. Values are in the range 0 to 16777215.

tail number state: [PASS | FAIL | INV]

tail number: string. Maximum length of 6 characters excluding quotes.

country state: [PASS | FAIL | INV]

country: string. Maximum length of 17 characters excluding quotes.

Description: Return the data from the mode S only all call test.

First param is the overall all call test state – passed, failed, or no data (no reply)

Remaining param pairs are the results for the individual tests making up the reply address test – a status and a value for each.

Example: `XPDR:MEAS:MSAC:ACAL?`

:XPDR

:MEASure

:MSACall

:ACALl

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

all call test enabled

Description: Determine whether the Mode S only all call test is enabled or not. This is set when the config is selected. If the all call test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MSAC:ACAL:ENAB?`

:XPDR

:MEASure

:MSACall

:ACALl

:STARt

Parameters: none

Description: This starts the Mode S only all call test. If the all call test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MSAC:ACAL:STAR`

Start Mode S only all call test.

:XPDR

:MEASure

:MSACall

:IRADdress

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>
reply address test state, ITM A reply address state, ITM A reply address, ITM C reply address state, ITM C reply address

Returned values: reply address test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

ITM A reply address state: [PASS | FAIL | INV | NDAT]

ITM A reply address: integer. Values are in the range 0 to 16777215.

ITM C reply address state: [PASS | FAIL | INV | NDAT]

ITM C reply address: integer. Values are in the range 0 to 16777215.

Description: Return the data from the ITM reply address test.

First param is the overall ITM reply address test state – passed, failed, or no data (no reply)

Remaining params are the results for the individual tests making up the reply address test – each one can be either pass or fail, or can be invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Example: `XPDR:MEAS:MSAC:IRAD?`

:XPDR

:MEASure

:MSACall

:IRADdress

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

reply address test enabled

Description: Determine whether the ITM reply address test is enabled or not. This is set when the config is selected. If the reply address test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MSAC:IRAD:ENAB?`

:XPDR

:MEASure

:MSACall

:IRADdress

:STARt

Parameters: none

Description: This starts the ITM reply address test. If the reply address test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MSAC:IRAD:STAR`

Start ITM reply address test.

:XPDR

:MEASure

:MSACall

:IRDelay

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR2>, <CRD>, <NR2>
reply delay test state, ITM A reply delay state, ITM A reply delay, ITM C reply delay state, ITM C reply delay

Returned values: reply delay test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

ITM A reply delay state: [PASS | FAIL | INV | NDAT]

ITM A reply delay: real

ITM C reply delay state: [PASS | FAIL | INV | NDAT]

ITM C reply delay: real

Description: Return the data from the ITM reply delay test.

First param is the overall ITM reply delay test state – passed, failed, or no data (no reply)

Remaining params are the results for the individual tests making up the reply delay test – each one can be either pass or fail, or can be invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Example: `XPDR:MEAS:MSAC:IRD?`

:XPDR

:MEASure

:MSACall

:IRDelay

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

reply delay test enabled

Description: Determine whether the ITM reply delay test is enabled or not. This is set when the config is selected. If the reply delay test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MSAC:IRD:ENAB?`

:XPDR

:MEASure

:MSACall

:IRDelay

:STARt

Parameters: none

Description: This starts the ITM reply delay test. If the reply delay test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MSAC:IRD:STAR`

Start ITM reply delay test.

:XPDR

:MEASure

:MSACall

:IRJitter

[:DATA]?

Parameters: none

Response: <CRD>, <CRD>, <NR2>, <CRD>, <NR2>
reply jitter test state, ITM A reply jitter state, ITM A reply jitter, ITM C reply jitter state, ITM C reply jitter

Returned values: reply jitter test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

ITM A reply jitter state: [PASS | FAIL | INV | NDAT]

ITM A reply jitter: real

ITM C reply jitter state: [PASS | FAIL | INV | NDAT]

ITM C reply jitter: real

Description: Return the data from the ITM reply jitter test.

First param is the overall ITM reply jitter test state – passed, failed, or no data (no reply)

Remaining params are the results for the individual tests making up the reply jitter test – each one can be either pass or fail, or can be invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Example: `XPDR:MEAS:MSAC:IRJ?`

:XPDR

:MEASure

:MSACall

:IRJitter

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

reply jitter test enabled

Description: Determine whether the ITM reply jitter test is enabled or not. This is set when the config is selected. If the reply jitter test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MSAC:IRJ:ENAB?`

:XPDR

:MEASure

:MSACall

:IRJitter

:STARt

Parameters: none

Description: This starts the ITM reply jitter test. If the reply jitter test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MSAC:IRJ:STAR`

Start ITM reply jitter test.

:XPDR

:MEASure

:MSACall

:IRRatio

[:DATA]

:PERCent?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>
reply ratio test state, ITM A reply ratio state, ITM A reply ratio, ITM C reply ratio state, ITM C reply ratio, ITM A low power reply ratio state, ITM A low power reply ratio, ITM C low power reply ratio state, ITM C low power reply ratio

Returned values: reply ratio test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

ITM A reply ratio state: [PASS | FAIL | INV | NDAT]

ITM A reply ratio: integer

ITM C reply ratio state: [PASS | FAIL | INV | NDAT]

ITM C reply ratio: integer

ITM A low power reply ratio state: [PASS | FAIL | INV | NDAT]

ITM A low power reply ratio: integer

ITM C low power reply ratio state: [PASS | FAIL | INV | NDAT]

ITM C low power reply ratio: integer

Description: Return the data from the ITM reply ratio test. The reply ratios are returnd as percentages (0 to 100).

First param is the overall ITM reply ratio test state – passed, failed, or no data (no reply)

Remaining params are the results for the individual tests making up the reply ratio test – each one can be either pass or fail, or can be invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Example: `XPDR:MEAS:MSAC:IRR:PERC?`

:XPDR

:MEASure

:MSACall

:IRRatio

[:DATA]

[:STATe]?

Parameters: None

Response: <CRD>, <CRD>, <CRD>, <CRD>, <CRD>
reply ratio test state, ITM A reply ratio state, ITM A reply ratio, ITM C reply ratio state, ITM C reply ratio

Returned values: reply ratio test state: [NRUN | NREP | PASS | WARN | FAIL | NAV | ERR]

ITM A reply ratio state: [PASS | FAIL | INV | NDAT]

ITM A reply ratio: [PASS | FAIL]

ITM C reply ratio state: [PASS | FAIL | INV | NDAT]

ITM C reply ratio: [PASS | FAIL]

Description: Return the data from the ITM reply ratio test.

This only provides a summary and does not return the results from the low power test. It is recommended that XPDR:MEAS:MSAC:IRR:PERC? is used instead.

First param is the overall ITM reply ratio test state – passed, failed, or no data (no reply)

Remaining params are the results for the individual tests making up the reply ratio test – each one can be either pass or fail, or can be invalid (measurement could not be performed). If invalid then the associated value will be meaningless.

Example: `XPDR:MEAS:MSAC:IRR?`

:XPDR

:MEASure

:MSACall

:IRRatio

:ENABled?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

reply ratio test enabled

Description: Determine whether the ITM reply ratio test is enabled or not. This is set when the config is selected. If the reply ratio test is not enabled, then the test cannot be run.

Example: `XPDR:MEAS:MSAC:IRR:ENAB?`

:XPDR

:MEASure

:MSACall

:IRRatio

:STARt

Parameters: none

Description: This starts the ITM reply ratio test. If the reply ratio test is disabled then an error will be given and the test will not start.

The test will run continuously until stopped (using XPDR:MEAS:STOP).

The latest data can be read back from the instrument at any time, and by using the status reporting structure it will be possible to determine if a test is running and whenever new data is available to be read back.

Example: `XPDR:MEAS:MSAC:IRR:STAR`

Start ITM reply ratio test.

:XPDR

:MEASure

:STOP

Parameters: none

Description: This stops the current test that is running. If no test is running this will just be ignored – no error will be given.

Example: `XPDR:MEAS:STOP`

Stop test.

:XPDR

:PLIMits

Parameters: <CPD>

power limits

Valid values: power limits: [FAR | MODified]. Values outside range are rejected and an error generated.

Description: Select whether strict FAR limits are applied to power measurements, or whether relaxed measurments (no upper limit on ERP and no lower limit on MTL) are performed.

Example: `XPDR:PLIM FAR`

Test to FAR specifications.

:XPDR

:PLIMits?

Parameters: None

Response: <CRD>

power limits

Returned values: power limits: [FAR | MOD]

Description: Determine whether we are testing to strict FAR limits, or a relaxed set.

Example: `XPDR:PLIM?`

ADSB COMMANDS

The IFR 6000 provides flight line test capability for receiving (ADS-B MON mode), decoding and displaying full ADS-B DO-260/A DF17/DF18 extended squitter transmissions from Mode S transponders or DF18 extended squitters from 1090 MHz emitters.

Capability to generate (ADS-B GEN mode) full DO-260/A DF17/DF18 extended squitter transmissions for testing ADS-B receivers is provided. A GICB mode fully decodes and displays all Enhanced Surveillance BDS register contents. The ADS-B/GICB is a Sub-Mode of XPDR Mode.

ADSB SUBSYSTEM

```
ADSB
    CPR
        LATitude\?
        LONGitude\?
        TYPE\?
    GENerate
        ADDRess\?
        ALL
        BD05
            ALTitude
                GNSS\?
                PRESsure\?
            [DATA]?
            ENABle\?
            LATitude\?
            LONGitude\?
            PERiod\?
            SAF\?
            STARt
            SURVeillance\?
            TIME\?
            TYPE\?
        BD06
            [DATA]?
            ENABle\?
            HEADing\?
            LATitude\?
            LONGitude\?
            MOVement\?
            PERiod\?
            STARt
            TIME\?
            TYPE\?

ADSB
    GENerate
        BD08
            [DATA]?
            ECATegory\?
            ENABle\?
            FID\?
            PERiod\?
            STARt
            TYPE\?
        BD09
            ASPeed\?
            ATYPe\?
            [DATA]?
            ENABle\?
            EWDirection\?
            EWVelocity\?
            GHDifference\?
            HEADing\?
            ICAPability\?
            ICHange\?
            NACV\?
            NSDirection\?
            NSVelocity\?
            PERiod\?
            SOURce\?
            STARt
            STYPe\?
            VRATe\?
```

ADSB SUBSYSTEM

```
ADSB
    GENerate
        BD61
            ST1
                [DATA]?
                ENABle\?
                EPSTatus\?
                PERiod\?
                REServed\?
                STARt
                STYPe\?
                MODEadata\?
            ST2
                ARA\?
                [DATA]?
                ENABle\?
                MTE\?
                PERiod\?
                RAC\?
                RAT\?
                STARt
                STYPe\?
                TTD\?
                TIDA\?
                TIDB\?
                TIDR\?
                TTI\?
        BD62
            ST0
                [DATA]?
                ENABle\?
                EPSTatus\?
                HDAVailable\?
                HMINdicator\?
                NAC\?
                NICBaro\?
                PERiod\?
                RAACtive\?
                SIL\?
                STARt
                TACapability\?
                TALTitude\?
                TATYpe\?
                THEading\?
```

```
    GENerate
        BD62
            ST0
                TOPerational\?
                VDAVailable\?
                VMINdicator\?
            ST1
                ADSR
                AHME
                ALTType
                APILot
                APMOde
                [DATA]?
                ENABle\?
                LNAV
                MCP
                NAC\?
                NICBaro\?
                PERiod\?
                PRESsure
                SALTitude
                SHEading
                SHDStatus
                SIL\?
                SILSupplement
                STARt
                TOPerational\?
                VNAV
```

ADSB SUBSYSTEM

- ADSB
  - GENerate
    - BD65
      - ACCCodes
        - ARV\?
        - NTCas\?
        - TC\?
        - TS\?
        - ADSR\?
      - BAQ\?
      - CCLass?
      - CDTI\?
      - [DATA]?
        - UAT\?
      - ENABle
        - AIR\?
        - SUR\?
      - HRDirection\?
      - LWIDth\?
      - NAC\?
      - NBARo\?
      - NSUPplement\?
        - PERiod\?
      - OMCodes
        - FORMat\?
        - IDENt\?
        - RATC\?
        - TRA\?
        - SOA\?
        - SA\?
      - SCCCodes
        - B2Low\?
        - NICC\?
        - NACV\?
      - SIL\?
      - STARt
        - AIR
        - SUR
        - STYPe\?
      - TAHeading\?
      - VNUMber\?
        - POA\?
    - STOP

- ADSB
  - GICB
    - ALL?
    - BD05
      - [DATA]?
      - STARt
    - BD06
      - [DATA]?
      - STARt
    - BD07
      - [DATA]?
      - STARt
    - BD08
      - [DATA]?
      - STARt
    - BD09
      - [DATA]?
      - STARt
    - BD10
      - [DATA]?
      - STARt
    - BD17
      - [DATA]?
      - STARt
    - BD18
      - [DATA]?
      - STARt
    - BD19
      - [DATA]?
      - STARt
    - BD1A
      - [DATA]?
      - STARt

ADSB SUBSYSTEM

- ADSB
  - GICB
    - BD1B
      - [DATA]?
      - STARt
    - BD1C
      - [DATA]?
      - STARt
    - BD1D
      - [DATA]?
      - STARt
    - BD1E
      - [DATA]?
      - STARt
    - BD1F
      - [DATA]?
      - STARt
    - BD20
      - [DATA]?
      - STARt
    - BD21
      - [DATA]?
      - STARt
    - BD30
      - [DATA]?
      - STARt
    - BD40
      - [DATA]?
      - STARt
    - BD41
      - [DATA]?
      - STARt
    - BD42
      - [DATA]?
      - STARt
    - BD43
      - [DATA]?
      - STARt
    - BD50
      - [DATA]?
      - STARt

- ADSB
  - GICB
    - BD60
      - [DATA]?
      - STARt
    - BD61
      - ST1
        - [DATA]?
        - STARt
      - ST2
        - [DATA]?
        - STARt
    - BD62
      - ST0
      - ST1
        - [DATA]?
        - STARt
    - BD65
      - AIR
        - [DATA]?
        - STARt
      - SUR
        - [DATA]?
        - STARt
    - STOP
  - MONitor
    - ALL
    - BD05
      - [DATA]?
      - STARt
    - BD06
      - [DATA]?
      - STARt
    - BD08
      - [DATA]?
      - STARt
    - BD09
      - [DATA]?
      - STARt
    - BD61
      - [DATA]?
      - STARt

ADSB SUBSYSTEM

ADSB
- MONitor
  - BD62
    - ST0
      - [DATA]?
      - STARt
    - ST1
      - [DATA]?
      - STARt
  - BD65
    - AIR
      - [DATA]?
      - STARt
    - SUR
      - [DATA]?
      - STARt
    - STOP
- SETup
  - GENerate\?
  - GICB\?
  - MONitor\?

:ADSB

:CPR

:LATitude

Parameters: <NRf>, <NRf>, <NRf>, <CPD>

degrees, minutes, seconds, direction

Valid values: degrees: integer. Valid values are 0 to 90. Values outside range are rejected and an error generated.

minutes: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

seconds: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

direction: [NORTh | SOUTh]. Values other than those stated are rejected and an error generated.

Description: Set the latitude to be used for decoding local airborne and surface decodes and global surface decode.

Example: `ADSB:CPR:LAT 75, 12, 55, nort`

Set latitude 75 degrees, 12 minutes and 55 seconds north.

:ADSB

:CPR

:LATitude?

Parameters: None

Response: <NR1>, <NR1>, <NR1>, <CRD>

degrees, minutes, seconds, direction

Returned values: degrees: integer. Values are in the range 0 to 90.

minutes: integer. Values are in the range 0 to 59.

seconds: integer. Values are in the range 0 to 59.

direction: [NORT | SOUT]

Description: Determine the latitude used for compact position reporting decode.

Example: `ADSB:CPR:LAT?`

:ADSB

:CPR

:LONGitude

Parameters: <NRf>, <NRf>, <NRf>, <CPD>

degrees, minutes, seconds, direction

Valid values: degrees: integer. Valid values are 0 to 180. Values outside range are rejected and an error generated.

minutes: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

seconds: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

direction: [EAST | WEST]. Values other than those stated are rejected and an error generated.

Description: Set the longitude to be used for decoding local airborne and surface decodes and global surface decode.

Example: `ADSB:CPR:LONG -135, 32, 5, west`

Set longitude 135 degrees, 32 minutes and 5 seconds west.

:ADSB

:CPR

:LONGitude?

Parameters: None

Response: <NR1>, <NR1>, <NR1>, <CRD>

degrees, minutes, seconds, direction

Returned values: degrees: integer. Values are in the range 0 to 180.

minutes: integer. Values are in the range 0 to 59.

seconds: integer. Values are in the range 0 to 59.

direction: [EAST | WEST]

Description: Determine the longitude used for compact position reporting decode.

Example: `ADSB:CPR:LONG?`

:ADSB
:CPR
:TYPE

Parameters: <CPD>

cpr type

Valid values: cpr type: [GLOBal | LOCal]. Values other than those stated are rejected and an error generated.

Description: Select global or local compact reporting position decode. For global airborne decode the lat and long values are not used.

Example: `ADSB:CPR:TYPE GLOB`

Select global decode.

:ADSB
:CPR
:TYPE?

Parameters: None

Response: <CRD>

cpr type

Returned values: cpr type: [GLOB | LOC]

Description: Determine if the cpr decode is local decode or global decode.

Example: `ADSB:CPR:TYPE?`

:ADSB
:GENerate
:ADDRess

Parameters: <NRf>

address

Valid values: address: integer. Valid values are 0 to 16777215. Values outside range are rejected and an error generated.

Description: Set the mode S transponder address for use when transmitting adsb extended squitter messages.

The address can also be entered in hexadecimal using #Hxxxxxx, or in octal using #Qxxxxxxxx, or in binary using #Bxxxxxxxxxxxxxxxxxxxxxxxx.

Example: `ADSB:GEN:ADDR 238467`

Select the address for mode S extended squitters.

:ADSB
:GENerate
:ADDRess?

Parameters: None

Response: <NR1>

address

Returned values: address: integer. Values are in the range 0 to 16777215.

Description: Determine the mode-S address used for adsb generate commands. Always returns a decimal number.

Example: `ADSB:GEN:ADDR?`

:ADSB
: GENerate
:ALL

Parameters: None

Description: This runs all the adsb generate tests that are enabled. The tests are looped, use ADSB:GEN:STOP to stop the testing.

If all tests are disabled, tests will not be run and an error given.

Example: `ADSB:GEN:ALL`

Run all the adsb generate tests.

:ADSB
:GENerate
:BD05
:ALTitude
:GNSS

Parameters: <CPD>, <NRf>

valid data, gnss altitude

Valid values: valid data: [VALid | NAVailable]. Values other than those stated are rejected and an error generated.

gnss altitude: integer. Valid values are -1000 to 126700. Values outside range are rejected and an error generated.

Description: Set the altitude that will sent out as part of the bds 0,5 extended squitter if TYPE is within the range 20 to 22 inclusive. The altitude value is set to the nearest valid value (25 ft steps).

Example: `ADSB:GEN:BD05:ALT:GNSS VAL, 56000`

Set GNSS altitude to 56,000 ft.

:ADSB
:GENerate
:BD05
:ALTitude
:GNSS?

Parameters: None

Response: <CRD>, <NR1>

valid data, gnss altitude

Returned values: valid data: [VAL | NAV]

gnss altitude: integer. Values are in the range -1000 to 126700.

Description: Determine the gnss altitude that will be sent out as part of a bds 0,5 extended squitter if type is set to 20..22. Value in feet. If valid data is NAV then the gnss altitude is invalid.

Example: `ADSB:GEN:BD05:ALT:GNSS?`

:ADSB
:GENerate
:BD05
:ALTitude
:PRESsure

Parameters: <CPD>, <NRf>

valid data, baro altitude

Valid values: valid data: [VALid | NAVailable]. Values other than those stated are rejected and an error generated.

baro altitude: integer. Valid values are -1000 to 126700. Values outside range are rejected and an error generated.

Description: Set the altitude that will sent out as part of the bds 0,5 extended squitter if TYPE is within the range 9 to 18 inclusive. The altitude value is set to the nearest valid value (25 ft steps).

Example: `ADSB:GEN:BD05:ALT:PRES VAL, 56000`

Set barometric pressure altitude to 56,000 ft.

:ADSB
:GENerate
:BD05
:ALTitude
:PRESsure?

Parameters: None

Response: <CRD>, <NR1>

valid data, baro altitude

Returned values: valid data: [VAL | NAV]

baro altitude: integer. Values are in the range -1000 to 126700.

Description: Determine the barometric pressure altitude that will be sent out as part of a bds 0,5 extended squitter if type is set to 9..18. Value in feet. If valid data is NAV then the baro altitude is invalid.

Example: `ADSB:GEN:BD05:ALT:PRES?`

:ADSB
:GENerate
:BD05
[:DATA]?

Parameters: None

Response: <NR1>, <ARBITRARY ASCII RESPONSE DATA>

count, ME

Returned values: count: integer

Description: Read back the number of bds 0,5 extended squitters that have been sent and the value of the ME field from the outgoing extended squitters. The ME field is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GEN:BD05?`

:ADSB
:GENerate
:BD05
:ENABle

Parameters: <BOOLEAN PROGRAM DATA>

test enabled

Description: This sets whether the bds 0,5 test will be performed or not. The test must be enabled before sending ADSB:GEN:BD05:STAR or an error will be reported. For the ADSB:GEN:ALL command the test must be enabled for bds 0,5 extended squitters to be sent.

Example: `ADSB:GEN:BD05:ENAB ON`

Enable bds 0,5 test.

:ADSB
:GENerate
:BD05
:ENABle?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

test enabled

Description: Determine whether bds 0,5 test is enabled or not.

Example: `ADSB:GEN:BD05:ENAB?`

:ADSB

:GENerate

:BD05

:LATitude

Parameters: <NRf>, <NRf>, <NRf>, <CPD>

degrees, minutes, seconds, direction

Valid values: degrees: integer. Valid values are 0 to 90. Values outside range are rejected and an error generated.

minutes: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

seconds: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

direction: [NORTh | SOUTh]. Values other than those stated are rejected and an error generated.

Description: Set the latitude to be sent as part of the bds 0,5 extended squitter.

Example: `ADSB:GEN:BD05:LAT 75, 12, 55, nort`

Set latitude 75 degrees, 12 minutes and 55 seconds north.

:ADSB
:GENerate
:BD05
:LATitude?

Parameters: None

Response: <NR1>, <NR1>, <NR1>, <CRD>

degrees, minutes, seconds, direction

Returned values: degrees: integer. Values are in the range 0 to 90.

minutes: integer. Values are in the range 0 to 59.

seconds: integer. Values are in the range 0 to 59.

direction: [NORT | SOUT]

Description: Determine the latitude used for bds 0,5 extended squitters.

Example: `ADSB:GEN_BD05:LAT?`

:ADSB

:GENerate

:BD05

:LONGitude

Parameters: <NRf>, <NRf>, <NRf>, <CPD>

degrees, minutes, seconds, direction

Valid values: degrees: integer. Valid values are 0 to 180. Values outside range are rejected and an error generated.

minutes: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

seconds: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

direction: [EAST | WEST]. Values other than those stated are rejected and an error generated.

Description: Set the longitude to be sent as part of the bds 0,5 extended squitter.

Example: `ADSB:GEN:BD05:LONG -135, 32, 5, west`

Set longitude 135 degrees, 32 minutes and 5 seconds west.

:ADSB

:GENerate

:BD05

:LONGitude?

Parameters: None

Response: <NR1>, <NR1>, <NR1>, <CRD>

degrees, minutes, seconds, direction

Returned values: degrees: integer. Values are in the range 0 to 180.

minutes: integer. Values are in the range 0 to 59.

seconds: integer. Values are in the range 0 to 59.

direction: [EAST | WEST]

Description: Determine the longitude used for bds 0,5 extended squitters.

Example: `ADSB:GEN:BD05:LONG?`

:ADSB

:GENerate

:BD05

:PERiod

Parameters: <NRf>

transmit period

Valid values: transmit period: real

Description: Set the period (ie rate) of the bds 0,5 extended squitters. Value can be set to 2dp within the range 0.50 seconds and 20.00 seconds.

Example: `ADSB:GEN:BD05:PER 15.5`

Set bds 0,5 extended squitter to be sent out every 15.5 seconds.

:ADSB
:GENerate
:BD05
:PERiod?

Parameters: None

Response: <NR2>

transmit period

Returned values: transmit period: real

Description: Determine the period of the bds 0,5 extended squitter.

Example: `ADSB:GEN:BD05:PER?`

:ADSB
:GENerate
:BD05
:SAF

Parameters: <NRf>

single antenna flag

Valid values: single antenna flag: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the SAF bit of the bds 0,5 extended squitters.

Example: `ADSB:GEN:BD05:SAF 1`

Set SAF bit in bds 0,5 extended squitter to be 1.

:ADSB
:GENerate
:BD05
:SAF?

Parameters: None

Response: <NR1>

single antenna flag

Returned values: single antenna flag: integer

Description: Determine the SAF bit of the bds 0,5 extended squitter.

Example: `ADSB:GEN:BD05:SAF?`

:ADSB
:GENerate
:BD05
:STARt

Parameters: None

Description: This starts the adsb gen bds0,5 (airborne position message) test. The test will run continuously until stopped. If the test is not enabled then the test will not start and an error will be reported.

Example: `ADSB:GEN:BD05:STAR`

Start adsb gen bds0,5 test.

:ADSB
:GENerate
:BD05
:SURVeillance

Parameters: <CPD>

surveillance status

Valid values: surveillance status: [NINFo | PALert | TALert | SPI]. Values other than those stated are rejected and an error generated.

Description: Set the two bit field surveillance status in the bds 0,5 extended squitter.

NINFo	No condition information
PALert	Permanent alert (emergency condition)
TALert	Temporary alert (change in Mode A code other than emergency condition
SPI	SPI condition

Example: `ADSB:GEN:BD05:SURV TAL`

Select temporary alert.

:ADSB
:GENerate
:BD05
:SURVeillance?

Parameters: None

Response: <CRD>

surveillance status

Returned values: surveillance status: [NINF | PAL | TAL | SPI]

Description: Determine the surveillance status.

Example: `ADSB:GEN:BD05:SURV?`

:ADSB
:GENerate
:BD05
:TIME

Parameters: <CPD>

time syncronization

Valid values: time syncronization: [UTC | NUTC]. Values other than those stated are rejected and an error generated.

Description: Set whether the Time of Applicability of the message is syncronized with UTC time. Setting UTC indicates it is syncronized and setting NUTC indicates that it is not.

Example: `ADSB:GEN:BD05:TIME NUTC`

Not syncronized with UTC.

:ADSB
:GENerate
:BD05
:TIME?

Parameters: None

Response: <CRD>

time syncronization

Returned values: time syncronization: [UTC | NUTC]

Description: Determine if syncronized to UTC or not.

Example: `ADSB:GEN:BD05:TIME?`

:ADSB

:GENerate

:BD05

:TYPE

Parameters: <NRf>

format type code

Valid values: format type code: integer

Description: Set the type of the bds 0,5 extended squitters. Valid ranges are 9 to 18 and 20 to 22. All other values will generate an error.

Setting type defines the values of Horizontal Containment Radius Limit, Navigation Integrity Category, and Altitude Type.

Example: `ADSB:GEN:BD05:TYPE 9`

Set TYPE field in bds 0,5 extended squitter to be 9.

:ADSB

:GENerate

:BD05

:TYPE?

Parameters: None

Response: <NR1>

format type code

Returned values: format type code: integer

Description: Determine the TYPE field of the bds 0,5 extended squitter.

Example: `ADSB:GEN:BD05:TYPE?`

:ADSB

:GENerate

:BD06

[:DATA]?

Parameters: None

Response: <NR1>, <ARBITRARY ASCII RESPONSE DATA>

count, ME

Returned values: count: integer

Description: Read back the number of bds 0,6 extended squitters that have been sent and the value of the ME field from the outgoing extended squitters. The ME field is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GEN:BD06?`

:ADSB
:GENerate
:BD06
:ENABle

Parameters: <BOOLEAN PROGRAM DATA>

test enabled

Description: This sets whether the bds 0,6 test will be performed or not. The test must be enabled before sending ADSB:GEN:BD06:STAR or an error will be reported. For the ADSB:GEN:ALL command the test must be enabled for bds 0,6 extended squitters to be sent.

Example: `ADSB:GEN:BD06:ENAB ON`

Enable bds 0,6 test.

:ADSB
:GENerate
:BD06
:ENABle?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

test enabled

Description: Determine whether bds 0,6 test is enabled or not.

Example: `ADSB:GEN:BD06:ENAB?`

:ADSB

:GENerate

:BD06

:HEADing

Parameters: <CPD>, <NRf>

valid data, heading

Valid values: valid data: [VALid | NAVailable]. Values other than those stated are rejected and an error generated.

heading: integer. Valid values are 0 to 357. Values outside range are rejected and an error generated.

Description: Set the heading that will sent out as part of the bds 0,6 extended squitter. The heading value is set to the nearest valid value.

Example: `ADSB:GEN:BD06:HEAD VAL, 56`

Set heading to 56 degrees.

:ADSB

:GENerate

:BD06

:HEADing?

Parameters: None

Response: <CRD>, <NR1>

valid data, heading

Returned values: valid data: [VAL | NAV]

heading: integer. Values are in the range 0 to 357.

Description: Determine the heading that will be sent out as part of a bds 0,6 extended squitter. Value in degrees. If valid data is NAV then the heading is invalid.

Example: `ADSB:GEN:BD06:HEAD?`

:ADSB

:GENerate

:BD06

:LATitude

Parameters: <NRf>, <NRf>, <NRf>, <CPD>

degrees, minutes, seconds, direction

Valid values: degrees: integer. Valid values are 0 to 90. Values outside range are rejected and an error generated.

minutes: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

seconds: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

direction: [NORTh | SOUTh]. Values other than those stated are rejected and an error generated.

Description: Set the latitude to be sent as part of the bds 0,6 extended squitter.

Example: `ADSB:GEN:BD06:LAT 75, 12, 55, nort`

Set latitude 75 degrees, 12 minutes and 55 seconds north.

:ADSB

:GENerate

:BD06

:LATitude?

Parameters: None

Response: <NR1>, <NR1>, <NR1>, <CRD>

degrees, minutes, seconds, direction

Returned values: degrees: integer. Values are in the range 0 to 90.

minutes: integer. Values are in the range 0 to 59.

seconds: integer. Values are in the range 0 to 59.

direction: [NORT | SOUT]

Description: Determine the latitude used for bds 0,6 extended squitters.

Example: `ADSB:GEN_BD06:LAT?`

:ADSB

:GENerate

:BD06

:LONGitude

Parameters: <NRf>, <NRf>, <NRf>, <CPD>

degrees, minutes, seconds, direction

Valid values: degrees: integer. Valid values are 0 to 180. Values outside range are rejected and an error generated.

minutes: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

seconds: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

direction: [EAST | WEST]. Values other than those stated are rejected and an error generated.

Description: Set the longitude to be sent as part of the bds 0,6 extended squitter.

Example: `ADSB:GEN:BD06:LONG -135, 32, 5, west`

Set longitude 135 degrees, 32 minutes and 5 seconds west.

:ADSB

:GENerate

:BD06

:LONGitude?

Parameters: None

Response: <NR1>, <NR1>, <NR1>, <CRD>

degrees, minutes, seconds, direction

Returned values: degrees: integer. Values are in the range 0 to 180.

minutes: integer. Values are in the range 0 to 59.

seconds: integer. Values are in the range 0 to 59.

direction: [EAST | WEST]

Description: Determine the longitude used for bds 0,6 extended squitters.

Example: `ADSB:GEN:BD06:LONG?`

:ADSB
:GENerate
:BD06
:MOVement

Parameters: <CPD>, <NRf>

state, ground speed

Valid values: state: [NINFo | STOPped | MOVing | DECelerating | ACCelerating | BUP]. Values other than those stated are rejected and an error generated.

Ground speed: real. Valid values are 0.000 to 175.000. Values outside range are rejected and an error generated.

Description: Set the movement field that will sent out as part of the bds 0,6 extended squitter. The ground speed is ignored unless the state is MOV. The value is encoded into a 7-bit field. Ground speed value is in knots.

Example: `ADSB:GEN:BD06:MOV VAL, 156`

Set movement field to indicate ground speed is 155 knots (nearest valid value).

:ADSB
:GENerate
:BD06
:MOVement?

Parameters: None

Response: <CRD>, <NR2>

state, ground speed

Returned values: state: [NINF | STOP | MOV | DEC | ACC | BUP]

ground speed: real. Values are in the range 0.000 to 175.000.

Description: Determine the movement field that will be sent out as part of a bds 0,6 extended squitter. Ground speed value is in knots, and is only valid if state is MOV.

Example: `ADSB:GEN:BD06:MOV?`

:ADSB
:GENerate
:BD06
:PERiod

Parameters: <NRf>

transmit period

Valid values: transmit period: real

Description: Set the period (ie rate) of the bds 0,6 extended squitters. Value can be set to 2dp within the range 0.50 seconds and 20.00 seconds.

Example: `ADSB:GEN:BD06:PER 5.5`

Set bds 0,6 extended squitter to be sent out every 5.5 seconds.

:ADSB
:GENerate
:BD06
:PERiod?

Parameters: None

Response: <NR2>

transmit period

Returned values: transmit period: real

Description: Determine the period of the bds 0,6 extended squitter.

Example: `ADSB:GEN:BD06:PER?`

:ADSB

:GENerate

:BD06

:STARt

Parameters: None

Description: This starts the adsb gen bds0,6 (surface position message) test. The test will run continuously until stopped. If the test is not enabled then the test will not start and an error will be reported.

Example: `ADSB:GEN:BD06:STAR`

Start adsb gen bds0,6 test.

:ADSB
:GENerate
:BD06
:TIME

Parameters: <CPD>

time syncronization

Valid values: time syncronization: [UTC | NUTC]. Values other than those stated are rejected and an error generated.

Description: Set whether the Time of Applicability of the message is syncronized with UTC time. Setting UTC indicates it is syncronized and setting NUTC indicates that it is not.

Example: `ADSB:GEN:BD06:TIME NUTC`

Not syncronized with UTC.

:ADSB
:GENerate
:BD06
:TIME?

Parameters: None

Response: <CRD>

time syncronization

Returned values: time syncronization: [UTC | NUTC]

Description: Determine if syncronized to UTC or not.

Example: `ADSB:GEN:BD06:TIME?`

:ADSB
:GENerate
:BD06
:TYPE

Parameters: <NRf>

format type code

Valid values: format type code: integer

Description: Set the type of the bds 0,6 extended squitters. Valid range is 5 to 8. All other values will generate an error.

Setting type defines the values of Horizontal Containment Radius Limit, and Navigation Integrity Category.

Example: `ADSB:GEN:BD06:TYPE 6`

Set TYPE field in bds 0,6 extended squitter to be 6.

:ADSB
:GENerate
:BD06
:TYPE?

Parameters: None

Response: <NR1>

format type code

Returned values: format type code: integer

Description: Determine the TYPE field of the bds 0,6 extended squitter.

Example: `ADSB:GEN:BD06:TYPE?`

:ADSB

:GENerate

:BD08

[:DATA]?

Parameters: None

Response: <NR1>, <ARBITRARY ASCII RESPONSE DATA>

count, ME

Returned values: count: integer

Description: Read back the number of bds 0,8 extended squitters that have been sent and the value of the ME field from the outgoing extended squitters. The ME field is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GEN:BD08?`

:ADSB
:GENerate
:BD08
:ECATegory

Parameters: <NRf>

emitter category code

Valid values: emitter category code: integer. Valid values are 0 to 7. Values outside range are rejected and an error generated.

Description: Set the emitter category field in the bds 0,8 extended squitter. The meaning of the field depends on the category set, which is set by the type (use ADSB:GEN:BD08:TYPE command).

Example: `ADSB:GEN:BD08:TYPE 3; ECAT 1`

Set emitter category field in bds 0,8 extended squitter to be 1, meaning Glider/Sailplane since type of 3 selects emitter category set B.

:ADSB
:GENerate
:BD08
:ECATegory?

Parameters: None

Response: <NR1>, <STRING RESPONSE DATA>

emitter category code, decoded category

Returned values: emitter category code: integer

decoded category: string. Maximum length of 26 characters excluding quotes.

Description: Determine the emitter category field in the bds 0,8 extended squitter. Also returns a string indicating the emitter category set and a human readable emitter category.

Example: `ADSB:GEN:BD08:ECAT?`

:ADSB
:GENerate
:BD08
:ENABle

Parameters: <BOOLEAN PROGRAM DATA>

test enabled

Description: This sets whether the bds 0,8 test will be performed or not. The test must be enabled before sending ADSB:GEN:BD08:STAR or an error will be reported. For the ADSB:GEN:ALL command the test must be enabled for bds 0,8 extended squitters to be sent.

Example: `ADSB:GEN:BD08:ENAB ON`

Enable bds 0,8 test.

:ADSB
:GENerate
:BD08
:ENABle?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

test enabled

Description: Determine whether bds 0,8 test is enabled or not.

Example: `ADSB:GEN:BD08:ENAB?`

:ADSB
:GENerate
:BD08
:FID

Parameters: <STRING PROGRAM DATA>

flight id

Valid values: flight id: string. Maximum length of 8 characters excluding quotes. Excess characters will be ignored.

Description: This sets the flight id that is output as part of the bds 0,8 extended squitter.

Example: `ADSB:GEN:BD08:FID “N12345”`

Set flight id characters.

:ADSB
:GENerate
:BD08
:FID?

Parameters: none

Response: <STRING RESPONSE DATA>

flight id

Returned values: flight id: string. Maximum length of 8 characters excluding quotes.

Description: Determine flight id being transmitted in bds 0,8 extended squitter.

Example: `ADSB:GEN:BD08:FID?`

:ADSB
:GENerate
:BD08
:PERiod

Parameters: <NRf>

transmit period

Valid values: transmit period: real

Description: Set the period (ie rate) of the bds 0,8 extended squitters. Value can be set to 2dp within the range 0.50 seconds and 20.00 seconds.

Example: `ADSB:GEN:BD08:PER 0.5`

Set bds 0,8 extended squitter to be sent out every 0.5 seconds.

:ADSB
:GENerate
:BD08
:PERiod?

Parameters: None

Response: <NR2>

transmit period

Returned values: transmit period: real

Description: Determine the period of the bds 0,8 extended squitter.

Example: `ADSB:GEN:BD08:PER?`

:ADSB

:GENerate

:BD08

:STARt

Parameters: None

Description: This starts the adsb gen bds0,8 (identification and category message) test. The test will run continuously until stopped. If the test is not enabled then the test will not start and an error will be reported.

Example: `ADSB:GEN:BD08:STAR`

Start adsb gen bds0,8 test.

:ADSB
:GENerate
:BD08
:TYPE

Parameters: <NRf>

format type code

Valid values: format type code: integer

Description: Set the type of the bds 0,8 extended squitters. Valid range is 1 to 4. All other values will generate an error.

Setting type defines the values of emitter category set (A .. D).

Type	Emitter Category Set
1	D
2	C
3	B
4	A

Example: `ADSB:GEN:BD08:TYPE 4`

Set TYPE field in bds 0,8 extended squitter to be 4 – emitter category set A.

:ADSB
:GENerate
:BD08
:TYPE?

Parameters: None

Response: <NR1>

format type code

Returned values: format type code: integer

Description: Determine the TYPE field of the bds 0,8 extended squitter.

Example: `ADSB:GEN:BD08:TYPE?`

:ADSB
:GENerate
:BD09
:ASPeed

Parameters: <CPD>, <NRf>

valid data, speed

Valid values: valid data: [VALid | NAVailable]. Values other than those stated are rejected and an error generated.

speed: integer

Description: Set the airspeed that will sent out as part of the bds 0,9 extended squitter. This command is only valid for subtype 3 and 4. An error will be generated for subtypes 1 and 2.

For subtype 3 (normal), the airspeed value can be between 0 and 1022 knots. The 1022 value is a special case meaning speed > 1021.5 knots.

For subtype 4 (supersonic), the airspeed value can be between 0 and 4088 knots, in steps of 4 knots. The 4088 value is a special case meaning speed > 4086 knots.

Example: `ADSB:GEN:BD09:ASP VAL, 246`

Set airspeed to 246 knots.

:ADSB
:GENerate
:BD09
:ASPeed?

Parameters: None

Response: <CRD>, <NR1>

valid data, speed

Returned values: valid data: [VAL | NAV | INV]

speed: integer

Description: Determine the airspeed that will be sent out as part of a bds 0,9 extended squitter. Value in knots. If valid data is NAV or INV then the speed is invalid. INV is returned if the subtype is not 3 or 4.

Example: `ADSB:GEN:BD09:ASP?`

:ADSB

:GENerate

:BD09

:ATYPe

Parameters: <CPD>

airspeed type

Valid values: airspeed type: [IAS | TAS]. Values other than those stated are rejected and an error generated.

Description: Set the airspeed type bit in the bds 0,9 extended squitter.

This value can be set at any time but is only used for subtypes 3 and 4.

Example: `ADSB:GEN:BD09:ATYP IAS`

Set airspeed type to IAS.

:ADSB

:GENerate

:BD09

:ATYPe?

Parameters: None

Response: <CRD>

airspeed type

Returned values: airspeed type: [IAS | TAS]

Description: Determine airspeed type.

Example: `ADSB:GEN:BD09:ATYP?`

:ADSB

:GENerate

:BD09

[:DATA]?

Parameters: None

Response: <NR1>, <ARBITRARY ASCII RESPONSE DATA>

count, ME

Returned values: count: integer

Description: Read back the number of bds 0,9 extended squitters that have been sent and the value of the ME field from the outgoing extended squitters. The ME field is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GEN:BD09?`

:ADSB
:GENerate
:BD09
:ENABle

Parameters: <BOOLEAN PROGRAM DATA>

test enabled

Description: This sets whether the bds 0,9 test will be performed or not. The test must be enabled before sending ADSB:GEN:BD09:STAR or an error will be reported. For the ADSB:GEN:ALL command the test must be enabled for bds 0,9 extended squitters to be sent.

Example: `ADSB:GEN:BD09:ENAB ON`

Enable bds 0,9 test.

:ADSB
:GENerate
:BD09
:ENABle?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

test enabled

Description: Determine whether bds 0,9 test is enabled or not.

Example: `ADSB:GEN:BD09:ENAB?`

:ADSB
:GENerate
:BD09
:EWDirection

Parameters: <CPD>

EW velocity direction

Valid values: EW velocity direction: [EAST | WEST]. Values other than those stated are rejected and an error generated.

Description: Set the direction for E-W velocity bit in the bds 0,9 extended squitter.

This value can be set at any time but is only used for subtypes 1 and 2.

Example: `ADSB:GEN:BD09:EWD EAST`

Set direction bit to EAST.

:ADSB
:GENerate
:BD09
:EWDirection?

Parameters: None

Response: <CRD>

EW velocity direction

Returned values: EW velocity direction: [EAST | WEST]

Description: Determine EW direction bit.

Example: `ADSB:GEN:BD09:EWD?`

:ADSB

:GENerate

:BD09

:EWVelocity

Parameters: <CPD>, <NRf>

valid data, speed

Valid values: valid data: [VALid | NAVailable]. Values other than those stated are rejected and an error generated.

speed: integer

Description: Set the east-west velocity that will sent out as part of the bds 0,9 extended squitter. This command is only valid for subtype 1 and 2. An error will be generated for subtypes 3 and 4.

For subtype 1 (normal), the velocity value can be between 0 and 1022 knots. The 1022 value is a special case meaning speed > 1021.5 knots.

For subtype 2 (supersonic), the velocity value can be between 0 and 4088 knots, in steps of 4 knots. The 4088 value is a special case meaning speed > 4086 knots.

Example: `ADSB:GEN:BD09:EWV VAL, 746`

Set east-west velocity to 746 knots.

:ADSB

:GENerate

:BD09

:EWVelocity?

Parameters: None

Response: <CRD>, <NR1>

valid data, speed

Returned values: valid data: [VAL | NAV | INV]

speed: integer

Description: Determine the east-west velocity that will be sent out as part of a bds 0,9 extended squitter. Value in knots. If valid data is NAV or INV then the speed is invalid. INV is returned if the subtype is not 1 or 2.

Example: `ADSB:GEN:BD09:EWV?`

:ADSB

:GENerate

:BD09

:GHDifference

Parameters: <CPD>, <NRf>

valid data, alt difference

Valid values: valid data: [VALid | NAVailable]. Values other than those stated are rejected and an error generated.

alt difference: integer

Description: Set the geometric height difference from baro alt field that will sent out as part of the bds 0,9 extended squitter.

The alt difference is ignored if valid data is NAV.

The valid range of alt difference is –3150 to 3150 in steps of 25 feet. The –3150 value is a special case meaning <-3137.5 ft, and the 3150 value is a special case meaning >3137.5 ft.

Example: `ADSB:GEN:BD09:GHD VAL, 125`

Set geomatric height difference from barometric alt to be 125 ft.

:ADSB

:GENerate

:BD09

:GHDifference?

Parameters: None

Response: <CRD>, <NR1>

valid data, alt difference

Returned values: valid data: [VAL | NAV]

alt difference: integer

Description: Determine the geometric height difference that will be sent out as part of a bds 0,9 extended squitter. Value in feet. If valid data is NAV then the alt difference is invalid.

Example: `ADSB:GEN:BD09:GHD?`

:ADSB
:GENerate
:BD09
:HEADing

Parameters: <CPD>, <NRf>

valid data, heading

Valid values: valid data: [VALid | NAVailable]. Values other than those stated are rejected and an error generated.

heading: real. Valid values are 0.0 to 359.6. Values outside range are rejected and an error generated.

Description: Set the heading that will sent out as part of the bds 0,9 extended squitter. The heading value is set to the nearest valid value.

The value can be set at any time but is only used if subtype is 3 or 4.

Example: `ADSB:GEN:BD09:HEAD VAL, 356.1`

Set heading to 356.1 degrees.

:ADSB
:GENerate
:BD09
:HEADing?

Parameters: None

Response: <CRD>, <NR2>

valid data, heading

Returned values: valid data: [VAL | NAV]

heading: real. Values are in the range 0.0 to 359.6.

Description: Determine the heading that will be sent out as part of a bds 0,9 extended squitter. Value in degrees. If valid data is NAV then the heading is invalid.

Example: `ADSB:GEN:BD09:HEAD?`

:ADSB
:GENerate
:BD09
:ICAPability

Parameters: <BOOLEAN PROGRAM DATA>

ifr capability

Description: This sets the IFR Capability Flag bit that is sent out in the bds 0,9 extended squitter.

Example: `ADSB:GEN:BD09:ICAP ON`

Set IFR Capability Flag bit on.

:ADSB
:GENerate
:BD09
:ICAPability?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

ifr capability

Description: Determine whether IFR Capability Flag bit in bds 0,9 test is enabled or not.

Example: `ADSB:GEN:BD09:ICAP?`

:ADSB
:GENerate
:BD09
:ICHange

Parameters: <BOOLEAN PROGRAM DATA>

intent change

Description: This sets the Intent Change Flag bit that is sent out in the bds 0,9 extended squitter.

Example: `ADSB:GEN:BD09:ICH OFF`

Set Intent Change Flag bit off.

:ADSB
:GENerate
:BD09
:ICHange?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

intent change

Description: Determine whether Intent Change Flag bit in bds 0,9 test is enabled or not.

Example: `ADSB:GEN:BD09:ICH?`

:ADSB

:GENerate

:BD09

:NACV

Parameters: <NRf>

NACv

Valid values: NACv: integer. Valid values are 0 to 4. Values outside range are rejected and an error generated.

Description: Set the Navigation Accuracy Category for Velocity field of the bds 0,9 extended squitters.

Example: `ADSB:GEN:BD09:NACV 4`

Set NACv field in bds 0,9 extended squitter to be 4.

:ADSB

:GENerate

:BD09

:NACV?

Parameters: None

Response: <NR1>

NACv

Returned values: NACv: integer

Description: Determine the NACv field of the bds 0,9 extended squitter.

Example: `ADSB:GEN:BD09:NACV?`

:ADSB
:GENerate
:BD09
:NSDirection

Parameters: <CPD>

NS velocity direction

Valid values: NS velocity direction: [NORTh | SOUTh]. Values other than those stated are rejected and an error generated.

Description: Set the direction for N-S velocity bit in the bds 0,9 extended squitter.

This value can be set at any time but is only used for subtypes 1 and 2.

Example: `ADSB:GEN:BD09:NSD NORT`

Set direction bit to North.

:ADSB
:GENerate
:BD09
:NSDirection?

Parameters: None

Response: <CRD>

NS velocity direction

Returned values: NS velocity direction: [NORT | SOUT]

Description: Determine NS direction bit.

Example: `ADSB:GEN:BD09:NSD?`

:ADSB

:GENerate

:BD09

:NSVelocity

Parameters: <CPD>, <NRf>

valid data, speed

Valid values: valid data: [VALid | NAVailable]. Values other than those stated are rejected and an error generated.

speed: integer

Description: Set the north-south velocity that will sent out as part of the bds 0,9 extended squitter. This command is only valid for subtype 1 and 2. An error will be generated for subtypes 3 and 4.

For subtype 1 (normal), the velocity value can be between 0 and 1022 knots. The 1022 value is a special case meaning speed > 1021.5 knots.

For subtype 2 (supersonic), the velocity value can be between 0 and 4088 knots, in steps of 4 knots. The 4088 value is a special case meaning speed > 4086 knots.

Example: `ADSB:GEN:BD09:NSV VAL, 746`

Set north-south velocity to 746 knots.

:ADSB

:GENerate

:BD09

:NSVelocity?

Parameters: None

Response: <CRD>, <NR1>

valid data, speed

Returned values: valid data: [VAL | NAV | INV]

speed: integer

Description: Determine the north-south velocity that will be sent out as part of a bds 0,9 extended squitter. Value in knots. If valid data is NAV or INV then the speed is invalid. INV is returned if the subtype is not 1 or 2.

Example: `ADSB:GEN:BD09:NSV?`

:ADSB

:GENerate

:BD09

:PERiod

Parameters: <NRf>

transmit period

Valid values: transmit period: real

Description: Set the period (ie rate) of the bds 0,9 extended squitters. Value can be set to 2dp within the range 0.50 seconds and 20.00 seconds.

Example: `ADSB:GEN:BD09:PER 0.5`

Set bds 0,9 extended squitter to be sent out every 0.5 seconds.

:ADSB

:GENerate

:BD09

:PERiod?

Parameters: None

Response: <NR2>

transmit period

Returned values: transmit period: real

Description: Determine the period of the bds 0,9 extended squitter.

Example: `ADSB:GEN:BD09:PER?`

:ADSB

:GENerate

:BD09

:SOURce

Parameters: <CPD>

source bit

Valid values: source bit: [GEO | BARO]. Values other than those stated are rejected and an error generated.

Description: Set the source for vertical rate bit in the bds 0,9 extended squitter.

Example: `ADSB:GEN:BD09:SOUR BARO`

Set source for vertical rate bit to Barometric.

:ADSB

:GENerate

:BD09

:SOURce?

Parameters: None

Response: <CRD>

source bit

Returned values: source bit: [GEO | BARO]

Description: Determine source for vertical rate bit.

Example: `ADSB:GEN:BD09:SOUR?`

:ADSB
:GENerate
:BD09
:STARt

Parameters: None

Description: This starts the adsb gen bds0,9 (airborne velocity message) test. The test will run continuously until stopped. If the test is not enabled then the test will not start and an error will be reported.

Example: `ADSB:GEN:BD09:STAR`

Start adsb gen bds0,9 test.

:ADSB

:GENerate

:BD09

:STYPe

Parameters: <NRf>

subtype code

Valid values: subtype code: integer

Description: Set the sub-type of the bds 0,9 extended squitters. Valid range is 0 to 7. All other values will generate an error.

Whilst the sub-type can be set to the full range of the 3-bit field, it is recommended that only subtypes 1 to 4 are used.

Subtype	Description
1	Velocity over ground – normal
2	Velocity over ground – supersonic!
3	Airspeed and Heading – normal
4	Airspeed and Heading – supersonic

Example: `ADSB:GEN:BD09:STYP 3`

Set subtype field in bds 0,9 extended squitter to be 3.

:ADSB

:GENerate

:BD09

:STYPe?

Parameters: None

Response: <NR1>

subtype code

Returned values: subtype code: integer

Description: Determine the subtype field of the bds 0,9 extended squitter.

Example: `ADSB:GEN:BD09:STYP?`

:ADSB

:GENerate

:BD09

:VRATe

Parameters: <CPD>, <NRf>

valid data, rate

Valid values: valid data: [VALid | NAVailable]. Values other than those stated are rejected and an error generated.

rate: integer

Description: Set the vertical rate that will sent out as part of the bds 0,9 extended squitter. The vertical rate value can be between –32640 and 32640 ft/min in steps of 64. The 32640 value is a special case meaning speed > 32608 ft/min.

Example: `ADSB:GEN:BD09:VRAT VAL, -6400`

Set vertical rate to 6400 ft/minute down.

:ADSB

:GENerate

:BD09

:VRATe

Parameters: None

Response: <CRD>, <NR1>

valid data, rate

Returned values: valid data: [VAL | NAV]

rate: integer

Description: Determine the vertical rate that will be sent out as part of a bds 0,9 extended squitter. Value in ft/min. If valid data is NAV then the speed is invalid.

Example: `ADSB:GEN:BD09:VRAT?`

:ADSB

:GENerate

:BD61

:ST1

[:DATA]?

Parameters: None

Response: <NR1>, <ARBITRARY ASCII RESPONSE DATA>

count, ME

Returned values: count: integer

Description: Read back the number of bds 6,1 ST1 extended squitters that have been sent and the value of the ME field from the outgoing extended squitters. The ME field is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GEN:BD61:ST1:[DATA]?`

:ADSB

:GENerate

:BD61

:ST2

[:DATA]?

Parameters: None

Response: <NR1>, <ARBITRARY ASCII RESPONSE DATA>

count, ME

Returned values: count: integer

Description: Read back the number of bds 6,1 ST2 extended squitters that have been sent and the value of the ME field from the outgoing extended squitters. The ME field is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GEN:BD61:ST2:[DATA]??`

:ADSB
:GENerate
:BD61
:ST1
:ENABle

Parameters: <BOOLEAN PROGRAM DATA>

test enabled

Description: This sets whether the bds 6,1 ST1 test will be performed or not. The test must be enabled before sending ADSB:GEN:BD61:ST1:STAR or an error will be reported. For the ADSB:GEN:ALL command the test must be enabled for bds 6,1 extended squitters to be sent.

Example:

```
ADSB:GEN:BD61:ST1:ENAB ON
```

Enable bds 6,1 test.

:ADSB
:GENerate
:BD61
:ST2
:ENABle

Parameters: <BOOLEAN PROGRAM DATA>

test enabled

Description: This sets whether the bds 6,1 ST2 test will be performed or not. The test must be enabled before sending ADSB:GEN:BD61:ST2:STAR or an error will be reported. For the ADSB:GEN:ALL command the test must be enabled for bds 6,1 extended squitters to be sent.

Example:

```
ADSB:GEN:BD61:ST2:ENAB ON
```

Enable bds 6,1 test.

:ADSB
:GENerate
:BD61
:ST1
:ENABle?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

test enabled

Description: Determine whether bds 6,1 ST1 test is enabled or not.

Example: `ADSB:GEN:BD61:ST1:ENAB?`

:ADSB
:GENerate
:BD61
:ST2
:ENABle?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

test enabled

Description: Determine whether bds 6,1 ST2 test is enabled or not.

Example: `ADSB:GEN:BD61:ST2:ENAB?`

:ADSB
:GENerate
:BD61
:ST1
:EPSTatus

Parameters: <CPD>
emergency priority status

Valid values: emergency priority status: [NEMergency | GEMergency | LMEMergency | MFUel | NCOMms | UINTerfere | DAIRcraft | REServed]. Values other than those stated are rejected and an error generated.

Description: Set the emergency/priority status field in the bds 6,1 extended squitter.

Code	Value	Description
NEM	0	No Emergency
GEM	1	General Emergency
LMEM	2	Lifeguard/Medical Emergency
MFU	3	Minimum Fuel
NCOM	4	No Communications
UINT	5	Unlawful Interference
DAIR	6	Downed Aircraft
RES	7	Reserved

Example: `ADSB:GEN:BD61:ST1:EPST NEM`

Set No Emergency condition.

:ADSB
:GENerate
:BD61
:ST1
:EPSTatus?

Parameters: None

Response: <CRD>

emergency priority status

Returned values: emergency priority status: [NEM | GEM | LMEM | MFU | NCOM | UINT | DAIR | RES]

Description: Determine emergency/priority status field.

Example: `ADSB:GEN:BD61:ST1:EPST?`

:ADSB
:GENerate
:BD61
:ST1
:PERiod

Parameters: <NRf>

transmit period

Valid values: transmit period: real

Description: Set the period (ie rate) of the bds 6,1 ST1 extended squitters. Value can be set to 2dp within the range 0.50 seconds and 20.00 seconds.

Example: `ADSB:GEN:BD61:ST1:PER 0.5`

Set bds 6,1 ST1 extended squitter to be sent out every 0.5 seconds.

:ADSB
:GENerate
:BD61
:ST2
:PERiod

Parameters: <NRf>

transmit period

Valid values: transmit period: real

Description: Set the period (ie rate) of the bds 6,1 ST2 extended squitters. Value can be set to 2dp within the range 0.50 seconds and 20.00 seconds.

Example: `ADSB:GEN:BD61:ST2:PER 0.5`

Set bds 6,1 ST2 extended squitter to be sent out every 0.5 seconds.

:ADSB

:GENerate

:BD61

:ST1

:PERiod?

Parameters: <NRf>

transmit period

Valid values: transmit period: real

Description: Set the period (ie rate) of the bds 6,1 ST1 extended squitters. Value can be set to 2dp within the range 0.50 seconds and 20.00 seconds.

Example: `ADSB:GEN:BD61:ST1:PER? 0.5`

Set bds 6,1 extended squitter to be sent out every 0.5 seconds.

:ADSB

:GENerate

:BD61

:ST2

:PERiod?

Parameters: None

Response: <NR2>

transmit period

Returned values: transmit period: real

Description: Determine the period of the bds 6,1ST2 extended squitter.

Example: `ADSB:GEN:BD61:ST2:PER?`

:ADSB
:GENerate
:BD61
:ST1
:REServed

Parameters: <STRING PROGRAM DATA>

reserved bits

Valid values: reserved bits: string

Description: This sets the reserved bits in the bds 6,1 ST1 extended squitter.

Must be exactly 12 characters long – 12 hexadecimal numbers. Valid range is "000000000000" to "1FFFFFFFFFFF".

Example: `ADSB:GEN:BD61:ST1:RES "1234567890AB"`

Set reserved bits to a known pattern.

:ADSB
:GENerate
:BD61
:ST1
:REServed?

Parameters: none

Response: <STRING RESPONSE DATA>

reserved bits

Returned values: reserved bits: string. Maximum length of 12 characters excluding quotes.

Description: Determine reserved bits in bds 6,1 ST1 extended squitter.

Example: `ADSB:GEN:BD61:ST1:RES?`

:ADSB
:GENerate
:BD61
:ST1
:STARt

Parameters: None

Description: This starts the adsb gen bds6,1 ST1 (Aircraft Status message) test. The test will run continuously until stopped. If the test is not enabled then the test will not start and an error will be reported.

Example: `ADSB:GEN:BD61:ST1:STAR`

Start adsb gen bds 6,1 ST1 test.

:ADSB
:GENerate
:BD61
:ST2
:STARt

Parameters: None

Description: This starts the adsb gen bds6,1 ST2 (Aircraft Status message) test. The test will run continuously until stopped. If the test is not enabled then the test will not start and an error will be reported.

Example: `ADSB:GEN:BD61:ST2:STAR`

Start adsb gen bds 6,1 ST2 test.

:ADSB
:GENerate
:BD61
:ST1
:STYPe

Parameters: <NRf>

subtype code

Valid values: subtype code: integer

Description: Set the sub-type of the bds 6,1 ST1 extended squitters. Valid range is 0 to 7. All other values will generate an error.

Subtype 0 indicates No Information.
Subtype 1 indicates Emergency/Priority Status.
Remaining subtypes are reserved.

Example: `ADSB:GEN:BD61:ST1:STYP 1`

Set subtype field in bds 6,1 extended squitter to be 1.

:ADSB
:GENerate
:BD61
:ST2
:STYPe

Parameters: <NRf>

subtype code

Valid values: subtype code: integer

Description: Set the sub-type of the bds 6,1 ST2 extended squitters. Valid range is 0 to 7. All other values will generate an error.

Subtype 0 indicates No Information.
Subtype 1 indicates Emergency/Priority Status.
Remaining subtypes are reserved.

Example: `ADSB:GEN:BD61:ST2:STYP 1`

Set subtype field in bds 6,1 extended squitter to be 1.

:ADSB

:GENerate

:BD61

:ST1

:STYPe?

Parameters: None

Response: <NR1>

subtype code

Returned values: subtype code: integer

Description: Determine the subtype field of the bds 6,1 ST1 extended squitter.

Example: `ADSB:GEN:BD61:ST1:STYP?`

:ADSB

:GENerate

:BD61

:ST2

:STYPe?

Parameters: None

Response: <NR1>

subtype code

Returned values: subtype code: integer

Description: Determine the subtype field of the bds 6,1 ST2 extended squitter.

Example: `ADSB:GEN:BD61:ST2:STYP?`

:ADSB
:GENerate
:BD62
:ST1
:ADSR

Parameters: <NRf>

ADSR

Valid values: ADSR: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level Supplement field of the bds 6,2 ST1 extended squitters.

Example: `ADSB:GEN:BD62:ST1:ADSR 1`

Set ADSR field in bds 6,2 ST1 extended squitter to be 1.

:ADSB
:GENerate
:BD62
:ST1
:AHME

Parameters: <NRf>

AHME

Valid values: AHME: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level Supplement field of the bds 6,2 ST1 extended squitters.

Example: `ADSB:GEN:BD62:ST1:AHME 1`

Set AHME field in bds 6,2 ST1 extended squitter to be 1.

:ADSB

:GENerate

:BD62

:ST1

:ALTType

Parameters: <NRf>

ALTT

Valid values: ALTT: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level Supplement field of the bds 6,2 ST1 extended squitters.

Example: `ADSB:GEN:BD62:ST1:ALTT 1`

Set ALTT field in bds 6,2 ST1 extended squitter to be 1.

:ADSB

:GENerate

:BD62

:ST1

:APILot

Parameters: <NRf>

APIL

Valid values: APIL: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level Supplement field of the bds 6,2 ST1 extended squitters.

Example: `ADSB:GEN:BD62:ST1:APIL 1`

Set APIL field in bds 6,2 ST1 extended squitter to be 1.

:ADSB
:GENerate
:BD62
:ST1
:APMOde

Parameters: <NRf>

APMO

Valid values: APMO: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level Supplement field of the bds 6,2 ST1 extended squitters.

Example: `ADSB:GEN:BD62:ST1:APMO 1`

Set APMO field in bds 6,2 ST1 extended squitter to be 1.

:ADSB

:GENerate

:BD62

:ST0

[:DATA]?

Parameters: None

Response: <NR1>, <ARBITRARY ASCII RESPONSE DATA>

count, ME

Returned values: count: integer

Description: Read back the number of bds 6,2 extended squitters that have been sent and the value of the ME field from the outgoing extended squitters. The ME field is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GEN:BD62:ST0:DATA?`

:ADSB

:GENerate

:BD62

:ST1

[:DATA]?

Parameters: None

Response: <NR1>, <ARBITRARY ASCII RESPONSE DATA>

count, ME

Returned values: count: integer

Description: Read back the number of bds 6,2 extended squitters that have been sent and the value of the ME field from the outgoing extended squitters. The ME field is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GEN:BD62:ST1:DATA?`

:ADSB

:GENerate

:BD62

:ST0

:ENABle

Parameters: <BOOLEAN PROGRAM DATA>

test enabled

Description: This sets whether the bds 6,2 ST0 test will be performed or not. The test must be enabled before sending ADSB:GEN:BD62:ST0:STAR or an error will be reported. For the ADSB:GEN:ALL command the test must be enabled for bds 6,2 ST0 extended squitters to be sent.

Example: `ADSB:GEN:BD62:ST0:ENAB ON`

Enable bds 6,2 ST0 test.

:ADSB

:GENerate

:BD62

:ST1

:ENABle

Parameters: <BOOLEAN PROGRAM DATA>

test enabled

Description: This sets whether the bds 6,2 ST1 test will be performed or not. The test must be enabled before sending ADSB:GEN:BD62:ST1:STAR or an error will be reported. For the ADSB:GEN:ALL command the test must be enabled for bds 6,2 ST1 extended squitters to be sent.

Example: `ADSB:GEN:BD62:ST1:ENAB ON`

Enable bds 6,2 ST1 test.

:ADSB

:GENerate

:BD62

:ST0

:ENABle?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

test enabled

Description: Determine whether bds 6,2 ST0 test is enabled or not.

Example: `ADSB:GEN:BD62:ST0:ENAB?`

:ADSB

:GENerate

:BD62

:ST1

:ENABle?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

test enabled

Description: Determine whether bds 6,2 ST1 test is enabled or not.

Example: `ADSB:GEN:BD62:ST1:ENAB?`

:ADSB
:GENerate
:BD62
:ST0
:EPSTatus

Parameters: <CPD>

emergency priority status

Valid values: emergency priority status: [NEMergency | GEMergency | LMEMergency | MFUel | NCOMms | UINTerfere | DAIRcraft | REServed]. Values other than those stated are rejected and an error generated.

Description: Set the emergency/priority status field in the bds 6,2 ST0 extended squitter.

Code	Value	Description
NEM	0	No Emergency
GEM	1	General Emergency
LMEM	2	Lifeguard/Medical Emergency
MFU	3	Minimum Fuel
NCOM	4	No Communications
UINT	5	Unlawful Interference
DAIR	6	Downed Aircraft
RES	7	Reserved

Example: `ADSB:GEN:BD62:ST0:EPST NEM`

Set No Emergency condition.

:ADSB
:GENerate
:BD62
:ST0
:EPSTatus?

Parameters: None

Response: <CRD>

emergency priority status

Returned values: emergency priority status: [NEM | GEM | LMEM | MFU | NCOM | UINT | DAIR | RES]

Description: Determine emergency/priority status field.

Example: `ADSB:GEN:BD62:ST0:EPST?`

:ADSB
:GENerate
:BD62
:ST0
:HDAVailable

Parameters: <CPD>

horizontal data available

Valid values: horizontal data available: [NVALid | MFCU | MAINtain | FRNav]. Values other than those stated are rejected and an error generated.

Description: Set the Horizontal Data Available/Source Indicator field in the bds 6,2 ST0 extended squitter.

Code	Value	Description
NVAL	0	No valid horizontal target state data is available
MFCU	1	Autopilot control panel selected value such as MCP or FCU
MAIN	2	Maintaining current heading or track angle
FRN	3	FMS/RNAV system (track angle specified by leg type)

Example: `ADSB:GEN:BD62:ST0:HDAV MAIN`

Indicate maintaining current heading.

:ADSB
:GENerate
:BD62
:ST0
:HDAVailable?

Parameters: None

Response: <CRD>

horizontal data available

Returned values: horizontal data available: [NVAL | MFCU | MAIN | FRN]

Description: Determine Horizontal Data Available/Source Indicator field.

Example: `ADSB:GEN:BD62:ST0:HDAV?`

:ADSB
:GENerate
:BD62
:ST0
:HMINdicator

Parameters: <CPD>

horizontal mode indicator

Valid values: horizontal mode indicator: [UNKNown | ACQuiring | MAINtaining | REServed]. Values other than those stated are rejected and an error generated.

Description: Set the Horizontal Mode Indicator field in the bds 6,2 ST0 extended squitter.

Code	Value	Description
UNKN	0	Unknown mode or information unavailable
ACQ	1	Acquiring mode
MAIN	2	Capturing or maintaining mode
RES	3	Reserved

Example: `ADSB:GEN:BD62:ST0:HMIN ACQ`

Set horizontal mode indicator field to acquiring mode.

:ADSB
:GENerate
:BD62
:ST0
:HMINdicator?

Parameters: None

Response: <CRD>

horizontal mode indicator

Returned values: horizontal mode indicator: [UNKN | ACQ | MAIN | RES]

Description: Determine Horizontal Mode Indicator field.

Example: `ADSB:GEN:BD62:ST0:HMIN?`

:ADSB
:GENerate
:BD62
:ST1
:LNAV

Parameters: <NRf>

LNAV

Valid values: LNAV: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level Supplement field of the bds 6,2 ST1 extended squitters.

Example: `ADSB:GEN:BD62:ST1:LNAV 1`

Set LNAV field in bds 6,2 ST1 extended squitter to be 1.

:ADSB
:GENerate
:BD62
:ST1
:MCP

Parameters: <NRf>

MCP

Valid values: MCP: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level Supplement field of the bds 6,2 ST1 extended squitters.

Example: `ADSB:GEN:BD62:ST1:MCP 1`

Set MCP field in bds 6,2 ST1 extended squitter to be 1.

:ADSB
:GENerate
:BD62
:ST0
:NAC

Parameters: <NRf>

NACp

Valid values: NACp: integer. Valid values are 0 to 15. Values outside range are rejected and an error generated.

Description: Set the Navigation Accuracy Category for Position field of the bds 6,2 ST0 extended squitters.

Example: `ADSB:GEN:BD62:ST0:NAC 6`

Set NACp field in bds 6,2 ST0 extended squitter to be 6.

:ADSB
:GENerate
:BD62
:ST1
:NAC

Parameters: <NRf>

NACp

Valid values: NACp: integer. Valid values are 0 to 15. Values outside range are rejected and an error generated.

Description: Set the Navigation Accuracy Category for Position field of the bds 6,2 ST1 extended squitters.

Example: `ADSB:GEN:BD62:ST1:NAC 6`

Set NACp field in bds 6,2 ST1 extended squitter to be 6.

:ADSB

:GENerate

:BD62

:ST0

:NAC?

Parameters: None

Response: <NR1>

NACp

Returned values: NACp: integer

Description: Determine the NACp field of the bds 6,2 ST0 extended squitter.

Example: ADSB:GEN:BD62:ST0:NAC?

:ADSB

:GENerate

:BD62

:ST1

:NAC?

Parameters: None

Response: <NR1>

NACp

Returned values: NACp: integer

Description: Determine the NACp field of the bds 6,2 ST1 extended squitter.

Example: `ADSB:GEN:BD62:ST1:NAC?`

:ADSB
:GENerate
:BD62
:ST0
:NICBaro

Parameters: <NRf>

nic for baro

Valid values: nic for baro: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Navigation Integrity Category for Baro bit of the bds 6,2 ST0 extended squitters.

Example: `ADSB:GEN:BD62:NICB 0`

Set NIC_{BARO} field in ST0:bds 6,2 ST0 extended squitter to be 0.

:ADSB
:GENerate
:BD62
:ST1
:NICBaro

Parameters: <NRf>

nic for baro

Valid values: nic for baro: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Navigation Integrity Category for Baro bit of the bds 6,2 ST1 extended squitters.

Example: `ADSB:GEN:BD62:ST1:NICB 0`

Set NIC_{BARO} field in bds 6,2 ST1 extended squitter to be 0.

:ADSB

:GENerate

:BD62

:ST0

:NICBaro?

Parameters: None

Response: <NR1>

nic for baro

Returned values: nic for baro: integer

Description: Determine the NIC_{BARO} field of the bds 6,2 ST0 extended squitter.

Example: `ADSB:GEN:BD62:ST0:NICB?`

:ADSB

:GENerate

:BD62

:ST1

:NICBaro?

Parameters: None

Response: <NR1>

nic for baro

Returned values: nic for baro: integer

Description: Determine the NIC_{BARO} field of the bds 6,2 ST1 extended squitter.

Example: `ADSB:GEN:BD62:ST1:NICB?`

:ADSB

:GENerate

:BD62

:ST0

:PERiod

Parameters: <NRf>

transmit period

Valid values: transmit period: real

Description: Set the period (ie rate) of the bds 6,2 extended squitters. Value can be set to 2dp within the range 0.50 seconds and 20.00 seconds.

Example: `ADSB:GEN:BD62:ST0:PER 0.5`

Set bds 6,2 ST0 extended squitter to be sent out every 0.5 seconds.

:ADSB

:GENerate

:BD62

:ST1

:PERiod

Parameters: <NRf>

transmit period

Valid values: transmit period: real

Description: Set the period (ie rate) of the bds 6,2 extended squitters. Value can be set to 2dp within the range 0.50 seconds and 20.00 seconds.

Example: `ADSB:GEN:BD62:ST1:PER 0.5`

Set bds 6,2 ST1 extended squitter to be sent out every 0.5 seconds.

:ADSB

:GENerate

:BD62

:ST0

:PERiod?

Parameters: None

Response: <NR2>

transmit period

Returned values: transmit period: real

Description: Determine the period of the bds 6,2 ST0 extended squitter.

Example: `ADSB:GEN:BD62:ST0:PER?`

:ADSB

:GENerate

:BD62

:ST1

:PERiod?

Parameters: None

Response: <NR2>

transmit period

Returned values: transmit period: real

Description: Determine the period of the bds 6,2ST1 extended squitter.

Example: `ADSB:GEN:BD62:ST1:PER?`

:ADSB

:GENerate

:BD62

:ST1

:PRESsure

Parameters: <CPD>, <NRf>

valid data, baro altitude

Valid values: valid data: [VALid | NAVailable]. Values other than those stated are rejected and an error generated.

baro altitude: integer. Valid values are 0 to 408.0 mb. Values outside range are rejected and an error generated.

Description: Set the altitude that will sent out as part of the bds 6,2 extended squitter if TYPE is within the range 9 to 18 inclusive. The altitude value is set to the nearest valid value (25 ft steps).

Example: `ADSB:GEN:BD62:ST1:PRES, 0/408.0`

Set barometric pressure altitude to 56,000 ft.

:ADSB

:GENerate

:BD62

:RAACtive

Parameters: <BOOLEAN PROGRAM DATA>

RA active

Description: Sets the bit that is part of Capacity/Mode Codes that indicates whether a TCAS/ACAS resolution advisory is active or not.

Example: `ADSB:GEN:BD62:RAAC ON`

Indicates that a resolution advisory is active.

:ADSB

:GENerate

:BD62

:RAACtive?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

RA active

Description: Determine whether the bit indicating a resolution advisory is active is set or not.

Example: `ADSB:GEN:BD62:RAAC?`

:ADSB
:GENerate
:BD62
:ST1
:SALTitude

Parameters: <CPD>, <NRf>

valid data, baro altitude

Valid values: valid data: [VALid | NAVailable]. Values other than those stated are rejected and an error generated.

baro altitude: integer. Valid values are 0 to 65,472. Values outside range are rejected and an error generated.

Description: Set the altitude that will sent out as part of the bds 6,2 extended squitter if TYPE is within the range 9 to 18 inclusive. The altitude value is set to the nearest valid value (25 ft steps).

Example: `ADSB:GEN:BD62:ST1:SALT`

Set barometric pressure altitude to 65,472 ft.

:ADSB
:GENerate
:BD62
:ST1
:SHDStatus

Parameters: <NRf>

SHDS

Valid values: SHDS: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level Supplement field of the bds 6,2 ST1 extended squitters.

Example: `ADSB:GEN:BD62:ST1:SHDS 1`

Set SHDS field in bds 6,2 ST1 extended squitter to be 1.

:ADSB
:GENerate
:BD62
:ST1
:SHEading

Parameters: <CPD>, <NRf>

valid data, heading

Valid values: valid data: [VALid | NAVailable]. Values other than those stated are rejected and an error generated.

heading: real. Valid values are -179.2 to 180.0. Values outside range are rejected and an error generated.

Description: Set the heading that will sent out as part of the bds 6,2 extended squitter. The heading value is set to the nearest valid value.

The value can be set at any time but is only used if subtype is 3 or 4.

Example: `ADSB:GEN:BD62:ST1:SHE, -179.2/180.0`

Set heading to -179.2/180.0 degrees.

:ADSB

:GENerate

:BD62

:ST0

:SIL

Parameters: <NRf>

SIL

Valid values: SIL: integer. Valid values are 0 to 3. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level field of the bds 6,2 ST0 extended squitters.

Example: `ADSB:GEN:BD62:ST0:SIL 3`

Set SIL field in bds 6,2 ST0 extended squitter to be 3.

:ADSB

:GENerate

:BD62

:ST1

:SIL

Parameters: None

Response: <NR1>

SIL

Returned values: SIL: integer

Description: Determine the SIL field of the bds 6,2 ST1 extended squitter.

Example: `ADSB:GEN:BD62:ST1:SIL?`

:ADSB

:GENerate

:BD62

:ST0

:SIL?

Parameters: <NRf>

SIL

Valid values: SIL: integer. Valid values are 0 to 3. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level field of the bds 6,2 ST0 extended squitters.

Example: `ADSB:GEN:BD62:ST0:SIL 3`

Set SIL field in bds 6,2 ST0 extended squitter to be 3.

:ADSB

:GENerate

:BD62

:ST1

:SIL?

Parameters: None

Response: <NR1>

SIL

Returned values: SIL: integer

Description: Determine the SIL field of the bds 6,2 ST1 extended squitter.

Example: `ADSB:GEN:BD62:ST1:SIL?`

:ADSB

:GENerate

:BD62

:ST1

:SILSupplement

Parameters: <NRf>

SILS

Valid values: SILS: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level Supplement field of the bds 6,2 ST1 extended squitters.

Example: `ADSB:GEN:BD62:ST1:SILS 1`

Set SILS field in bds 6,2 ST1 extended squitter to be 1.

:ADSB

:GENerate

:BD62

ST0:

:STARt

Parameters: None

Description: This starts the adsb gen bds6,2 (Target State and Status Information message) test. The test will run continuously until stopped. If the test is not enabled then the test will not start and an error will be reported.

Example: `ADSB:GEN:BD62:ST0:STAR`

Start adsb gen bds 6,2 ST0 test.

:ADSB

:BD62

:ST1

:STARt

Parameters: None

Description: This starts the adsb gen bds6,2 (Target State and Status Information message) test. The test will run continuously until stopped. If the test is not enabled then the test will not start and an error will be reported.

Example: `ADSB:GEN:BD62:ST1:STAR`

Start adsb gen bds 6,2 ST1 test.

:ADSB
:GENerate
:BD62
:TACapability

Parameters: <CPD>

target altitude capability

Valid values: target altitude capability: [HALTitude | HAALtitude | HAFRnav | REServed]. Values other than those stated are rejected and an error generated.

Description: Set the Target Altitude Capability field in the bds 6,2 extended squitter.

Code	Value	Description
HALT	0	Capability for reporting holding altitude only
HAAL	1	Can report holding alt or autopilot control panel selected alt
HAFR	2	Holding alt, Autopilot selected alt, or FMS/RNAV level-off alt
RES	3	Reserved

Example: `ADSB:GEN:BD62:TAC HALT`

Set target altitude capability field to indicate can only report holding altitude.

:ADSB
:GENerate
:BD62
:TACapability?

Parameters: None

Response: <CRD>

target altitude capability

Returned values: target altitude capability: [HALT | HAAL | HAFR | RES]

Description: Determine Target Altitude Capability field.

Example: `ADSB:GEN:BD62:TAC?`

:ADSB
:GENerate
:BD62
:ST0
:TALTitude

Parameters: <CPD>, <NRf>

valid data, target altitude

Valid values: valid data: [VALid | INValid]. Values other than those stated are rejected and an error generated.

target altitude: integer

Description: Set the target altitude field that will sent out as part of the bds 6,2 ST0 extended squitter.

The target altitude value can be between –1000 ft and 100000 ft in steps of 100.

Example: `ADSB:GEN:BD62:ST0:TALT VAL, -54000`

Set target altitude to 54000 ft.

:ADSB
:GENerate
:BD62
:ST0
:TALTitude?

Parameters: None

Response: <CPD>, <NRf>

valid data, target altitude

Returned values: valid data: [VAL | INV]

target altitude: integer

Description: Determine the target altitude field that will be sent out as part of a bds 6,2 ST0 extended squitter. Value in ft. If valid data is INV thent he altitude is invalid.

Example: `ADSB:GEN:BD62:ST0:TALT?`

:ADSB
:GENerate
:BD62
:TATYpe

Parameters: <CPD>

target altitude type

Valid values: target altitude type: [FL | MSL]. Values other than those stated are rejected and an error generated.

Description: Set the Target Altitude Type bit in the bds 6,2 extended squitter.

Code	Value	Description
FL	0	The target altitude is referenced to a flight level
MSL	1	The target altitude is referenced to mean sea level

Example: `ADSB:GEN:BD62:TATY MSL`

Set target altitude type field to indicate target altitude is referenced to mean sea level.

:ADSB
:GENerate
:BD62
:TATYpe?

Parameters: None

Response: <CRD>

target altitude type

Returned values: target altitude type: [FL | MSL]

Description: Determine Target Altitude Type field.

Example: `ADSB:GEN:BD62:TATY?`

:ADSB

:GENerate

:BD62

:ST0

:THEading

Parameters: <CPD>, <NRf>

valid data, target heading

Valid values: valid data: [VALid | INValid]. Values other than those stated are rejected and an error generated.

target heading: integer

Description: Set the target heading/track angle field that will sent out as part of the bds 6,2 ST0 extended squitter.

The target heading value can be between 0 and 359 degrees.

Example: `ADSB:GEN:BD62:ST0:THE VAL, 54`

Set target heading to 54 degrees.

:ADSB

:GENerate

:BD62

:ST0

:THEading?

Parameters: None

Response: <CRD>, <NR1>

valid data, target heading

Returned values: valid data: [VAL | INV]

target heading: integer

Description: Determine the target heading field that will be sent out as part of a bds 6,2 ST0 extended squitter. Value in degrees. If valid data is INV then the heading is invalid.

Example: `ADSB:GEN:BD62:ST0:THE?`

:ADSB
:GENerate
:BD62
:ST0
:TOPerational

Parameters: <BOOLEAN PROGRAM DATA>

operational

Description: Sets the bit that is part of Capacity/Mode Codes that indicates whether a TCAS/ACAS is operational or not.

Code	Bit Value	Description
ON	0	TCAS/ACAS operational or unknown
OFF	1	TCAS/ACAS not operational

Example:
```
ADSB:GEN:BD62:ST0:TOP ON
```

Indicates tcas/acas is operational or unknown.

:ADSB
:GENerate
:BD62
:ST1
:TOPerational

Parameters: <BOOLEAN PROGRAM DATA>

operational

Description: Sets the bit that is part of Capacity/Mode Codes that indicates whether a TCAS/ACAS is operational or not.

Code	Bit Value	Description
ON	0	TCAS/ACAS operational or unknown
OFF	1	TCAS/ACAS not operational

Example:
```
ADSB:GEN:BD62:ST1:TOP ON
```

Indicates tcas/acas is operational or unknown.

:ADSB
:GENerate
:BD62
:ST0
:TOPerational?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

operational

Description: Determine whether the bit indicating tcas/acas operational is set or not.

Returned value of 1 indicates operational/unknown (ME bit is 0), value of 0 indicates not operational (ME bit is 1).

Example: `ADSB:GEN:BD62:ST0:TOP?`

:ADSB
:GENerate
:BD62
:ST1
:TOPerational?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

operational

Description: Determine whether the bit indicating tcas/acas operational is set or not.

Returned value of 1 indicates operational/unknown (ME bit is 0), value of 0 indicates not operational (ME bit is 1).

Example: `ADSB:GEN:BD62:ST1:TOP?`

:ADSB
:GENerate
:BD62
:ST0
:VDAVailable

Parameters: <CPD>

vertical data available

Valid values: vertical data available: [NAVailable | APILot | HALTitude | FRNav]. Values other than those stated are rejected and an error generated.

Description: Set the Vertical Data Available/Source Indicator field in the bds 6,2 extended squitter.

Code	Value	Description
NAV	0	No valid vertical target state data is available
APIL	1	Autopilot control panel selected value such as MCP or FCU
HALT	2	Holding altitude
FRN	3	FMS/RNAV system

Example: `ADSB:GEN:BD62:ST0:VDAV HALT`

Indicate maintaining current altitude.

:ADSB
:GENerate
:BD62
:ST0
:VDAVailable?

Parameters: None

Response: <CRD>

vertical data available

Returned values: vertical data available: [NVAL | MFCU | MAIN | FRN]

Description: Determine Vertical Data Available/Source Indicator field.

Example: `ADSB:GEN:BD62:ST0:VDAV?`

:ADSB
:GENerate
:BD62
:ST0
:VMINdicator

Parameters: <CPD>

vertical mode indicator

Valid values: vertical mode indicator: [UNKNown | ACQuiring | MAINtaining | REServed]. Values other than those stated are rejected and an error generated.

Description: Set the Vertical Mode Indicator field in the bds 6,2 extended squitter.

Code	Value	Description
UNKN	0	Unknown mode or information unavailable
ACQ	1	Acquiring mode
MAIN	2	Capturing or maintaining mode
RES	3	Reserved

Example: `ADSB:GEN:BD62:ST0:VMIN ACQ`

Set vertical mode indicator field to acquiring mode.

:ADSB
:GENerate
:BD62
:ST0
:VMINdicator?

Parameters: None

Response: <CRD>

vertical mode indicator

Returned values: vertical mode indicator: [UNKN | ACQ | MAIN | RES]

Description: Determine Vertical Mode Indicator field.

Example: `ADSB:GEN:BD62:ST0:VMIN?`

:ADSB
:GENerate
:BD62
:ST1
:VNAV

Parameters: <NRf>

VNAV

Valid values: VNAV: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level Supplement field of the bds 6,2 ST1 extended squitters.

Example: `ADSB:GEN:BD62:ST1:VNAV 1`

Set VNAV field in bds 6,2 ST1 extended squitter to be 1.

:ADSB
:GENerate
:BD65
:ACCCodes
:ADSR

Parameters: <NRf>

ADSR

Valid values: ADSR: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Target State Report Capability bit (part of the airborne capability class codes field) of the bds 6,5 extended squitters.

This bit is only valid if subtype is zero. No checking is performed so only set TS to 1 if it is valid to do so.

Example: `ADSB:GEN:BD65:ACCC:ADSR`

Set TS bit in bds 6,5 extended squitter to indicate not capable of sending messages to support target state reports.

ADSB
:GENerate
:BD65
:ACCCodes
:ADSR?

Parameters: None

Response: <NR1>

TS

Returned values: TS: integer

Description: Determine the TS bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:ACCC:ADSR?`

:ADSB
:GENerate
:BD65
:ACCCodes
:ARV

Parameters: <NRf>

ARV

Valid values: ARV: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Air-Referenced Velocity Report Capability bit (part of the airborne capability class codes field) of the bds 6,5 extended squitters.

This bit is only valid if subtype is zero. No checking is performed so only set ARV to 1 if it is valid to do so.

Example: `ADSB:GEN:BD65:ACCC:ARV 0`

Set ARV bit in bds 6,5 extended squitter to be 0.

:ADSB
:GENerate
:BD65
:ACCCodes
:ARV?

Parameters: None

Response: <NR1>

ARV

Returned values: ARV: integer

Description: Determine the ARV bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:ACCC:ARV?`

:ADSB
:GENerate
:BD65
:ACCCodes
:NTCas

Parameters: <NRf>

not tcas

Valid values: not tcas: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: DO-260A: Set the Not Traffic Alert and Collision Avoidance System Status bit (part of the airborne capability class codes field) of the bds 6,5 extended squitters.

DO-260B: Traffic Alert and Collision Avoidance System operational. This bit is only valid if subtype is zero.

Example: `ADSB:GEN:BD63:ACCC:NTC 0`

Set not-tcas bit in bds 6,5 extended squitter to be 0.

:ADSB
:GENerate
:BD65
:ACCCodes
:NTCas?

Parameters: None

Response: <NR1>

not tcas

Returned values: not tcas: integer

Description: DO-260A:
Determine the not-tcas bit of the bds 6,5 extended squitter.

DO-260B:
TCAS operational.

Example: `ADSB:GEN:BD65:ACCC:NTC?`

:ADSB
:GENerate
:BD65
:ACCCodes
:TC

Parameters: <NRf>

TC

Valid values: TC: integer. Valid values are 0 to 3. Values outside range are rejected and an error generated.

Description: Set the Target Change Report Capability field (part of the airborne capability class codes field) of the bds 6,5 extended squitters.

This bit is only valid if subtype is zero. No checking is performed so only set TC non-zero if it is valid to do so.

Example: `ADSB:GEN:BD65:ACCC:TC 2`

Set TC field in bds 6,5 extended squitter to indicate capable of sending information for multiple TC reports.

:ADSB
:GENerate
:BD65
:ACCCodes
:TC?

Parameters: None

Response: <NR1>

TC

Returned values: TC: integer

Description: Determine the TC field of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:ACCC:TC?`

:ADSB
:GENerate
:BD65
:ACCCodes
:TS

Parameters: <NRf>

TS

Valid values: TS: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Target State Report Capability bit (part of the airborne capability class codes field) of the bds 6,5 extended squitters.

This bit is only valid if subtype is zero. No checking is performed so only set TS to 1 if it is valid to do so.

Example: `ADSB:GEN:BD65:ACCC:TS 0`

Set TS bit in bds 6,5 extended squitter to indicate not capable of sending messages to support target state reports.

:ADSB
:GENerate
:BD65
:ACCCodes
:TS?

Parameters: None

Response: <NR1>

TS

Returned values: TS: integer

Description: Determine the TS bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:ACCC:TS?`

:ADSB
:GENerate
:BD65
:BAQ

Parameters: <NRf>

baq

Valid values: baq: integer. Valid values are 0 to 3. Values outside range are rejected and an error generated.

Description: DO-260A: Set the Barometric Altitude Quality field of the bds 6,5 extended squitters. DO-260A defines this field to always be zero.

DO-260B: Set the Geometric Vertical Accuracy.

This bit is only valid if subtype is zero No checking is performed so only set baq non-zero if it is valid to do so.

Example: `ADSB:GEN:BD65:BAQ 0`

Set BAQ field in bds 6,5 extended squitter to 0.

:ADSB
:GENerate
:BD65
:BAQ?

Parameters: None

Response: <NR1>

baq

Returned values: baq: integer

Description: DO-260A: Determine the BAQ field of the bds 6,5 extended squitter.

DO-260B: Determine the GVA field of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:BAQ?`

:ADSB
:GENerate
:BD65
:CCLass?

Parameters: None

Response: <NR1>

capability class codes

Returned values: capability class codes: integer

Description: Determine the capability class codes field of the bds 6,5 extended squitter.

For subtype 0, the returned value is between 0 and 65535. For subtype 1, the returned value is between 0 and 4095.

Example: `ADSB:GEN:BD65:CCL?`

:ADSB
:GENerate
:BD65
:CDTI

Parameters: <NRf>

CDTI

Valid values: CDTI: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: DO-206A: Set the Cockpit Display of Traffic Information Status bit (part of the capability class codes field) of the bds 6,5 extended squitters.

DO-206B: Set the 1090 bit (part of the capability class codes field) of the bds 6,5 extended squitters.

Example: `ADSB:GEN:BD65:CDTI 0`

Set CDTI bit in bds 6,5 extended squitter to indicate traffic display not operational.

:ADSB
:GENerate
:BD65
:CDTI?

Parameters: None

Response: <NR1>

CDTI

Returned values: CDTI: integer

Description: DO-260A: Determine the CDTI bit of the bds 6,5 extended squitter.

DO-260B: Determine the 1090 bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:CDTI?`

:ADSB

:GENerate

:BD65

[:DATA]?

Parameters: None

Response: <NR1>, <ARBITRARY ASCII RESPONSE DATA>

count, ME

Returned values: count: integer

Description: Read back the number of bds 6,5 extended squitters that have been sent and the value of the ME field from the outgoing extended squitters. The ME field is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GEN:BD65?`

:ADSB

:GENerate

:BD65

:ENABle

:AIR

Parameters: <BOOLEAN PROGRAM DATA>

test enabled

Description: This sets whether the bds 6,5 Air Subtype test will be performed or not. The test must be enabled before sending ADSB:GEN:BD65:STAR or an error will be reported. For the ADSB:GEN:ALL command the test must be enabled for bds 6,5 extended squitters to be sent.

Example: `ADSB:GEN:BD65:ENAB:AIR ON`

Enable bds 6,5 test.

:ADSB

:GENerate

:BD65

:ENABle

:SUR

Parameters: <BOOLEAN PROGRAM DATA>

test enabled

Description: This sets whether the bds 6,5 Surface Subtype test will be performed or not. The test must be enabled before sending ADSB:GEN:BD65:STAR or an error will be reported. For the ADSB:GEN:ALL command the test must be enabled for bds 6,5 extended squitters to be sent.

Example: `ADSB:GEN:BD65:ENAB:SUR ON`

Enable bds 6,5 test.

:ADSB

:GENerate

:BD65

:ENABle

:AIR?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

test enabled

Description: Determine whether bds 6,5 test is enabled or not.

Example: `ADSB:GEN:BD65:ENAB:AIR?`

:ADSB

:GENerate

:BD65

:ENABle

:SUR?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

test enabled

Description: Determine whether bds 6,5 test is enabled or not.

Example: `ADSB:GEN:BD65:ENAB:SUR?`

:ADSB

:GENerate

:BD65

:HRDirection

Parameters: <CPD>

horiz ref direction

Valid values: horiz ref direction: [MNORth | TNORth]. Values other than those stated are rejected and an error generated.

Description: Set the Horizontal Reference Direction bit in the bds 6,5 extended squitter.

Code	Value	Description
TNOR	0	True North
MNOR	1	Magnetic North

Example: `ADSB:GEN:BD65:HRD MNOR`

Set reference direction to be magnetic north.

:ADSB
:GENerate
:BD65
:HRDirection?

Parameters: None

Response: <CRD>

horiz ref direction

Returned values: horiz ref direction: [MNOR | TNOR]

Description: Determine Horizontal Reference Direction bit.

Example: `ADSB:GEN:BD65:HRD?`

:ADSB
:GENerate
:BD65
:LWIDth

Parameters: <NRf>

length and width code

Valid values: length and width code: integer. Valid values are 0 to 15. Values outside range are rejected and an error generated.

Description: Set the Aircraft Length and Width Codes field of the bds 6,5 extended squitters.

This bit is only valid if subtype is one. No checking is performed so only set length and width field non-zero if it is valid to do so.

Example: `ADSB:GEN:BD65:LWID 0`

Set aircraft length and width codes field in bds 6,3 extended squitter to 0.

:ADSB
:GENerate
:BD65
:LWIDth?

Parameters: None

Response: <NR1>

length and width code

Returned values: length and width code: integer

Description: Determine the aircraft length and width codes field of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:LWID?`

:ADSB
:GENerate
:BD65
:NAC

Parameters: <NRf>

NACp

Valid values: NACp: integer. Valid values are 0 to 15. Values outside range are rejected and an error generated.

Description: Set the Navigational Accuracy Category for Position field of the bds 6,5 extended squitters.

Example: `ADSB:GEN:BD65:NAC 0`

Set NACp field in bds 6,5 extended squitter to 0.

:ADSB

:GENerate

:BD65

:NAC?

Parameters: None

Response: <NR1>

NACp

Returned values: NACp: integer

Description: Determine the NACp field of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:NAC?`

:ADSB

:GENerate

:BD65

:NBARo

Parameters: <NRf>

nic

Valid values: nic: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Barometric Altitude Integrity Code bit of the bds 6,5 extended squitters.

This bit is only valid if subtype is zero. No checking is performed so only set nic to 1 if it is valid to do so.

Example: `ADSB:GEN:BD65:NBAR 0`

Set nic bit in bds 6,5 extended squitter to 0.

:ADSB

:GENerate

:BD65

:NBARo?

Parameters: None

Response: <NR1>

nic

Returned values: nic: integer

Description: Determine the Barometric Altitude Integrity Code bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:NBAR?`

:ADSB

:GENerate

:BD65

:NSUPplement

Parameters: <NRf>

nic supplement A

Valid values: nic supplement A: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the NIC Supplement bit of the bds 6,5 extended squitters.

Example: `ADSB:GEN:BD65:NSUP 0`

Set nic supplement bit in bds 6,5 extended squitter to 0.

:ADSB
:GENerate
:BD65
:NSUPplement?

Parameters: None

Response: <NR1>

nic supplement

Returned values: nic supplement: integer

Description: Determine the NIC Supplement bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:NSUP?`

:ADSB
:GENerate
:BD65
:OMCodes
:FORMat

Parameters: <NRf>

format

Valid values: format: integer. Valid values are 0 to 3. Values outside range are rejected and an error generated.

Description: Set the Operational Mode Subfield Format field of the bds 6,5 extended squitters.

Currently only format 0 is defined. Formats 1 to 3 are reserved.

Example: `ADSB:GEN:BD65:OMC:FORM 0`

Set operational mode format field in bds 6,5 extended squitter to 0.

:ADSB
:GENerate
:BD65
:OMCodes
:FORMat?

Parameters: None

Response: <NR1>

format

Returned values: format: integer

Description: Determine the Operational Mode Subfield Format field of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:OMC:FORM?`

:ADSB
:GENerate
:BD65
:OMCodes
:IDENt

Parameters: <NRf>

ident switch

Valid values: ident switch: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Ident Switch Active bit of the bds 6,5 extended squitters.

Only valid if operational mode format is 0. No checking is performed so only set ident switch to 1 if it is valid to do so

Example: `ADSB:GEN:BD65:OMC:IDEN 1`

Indicate ident switch active.

:ADSB
:GENerate
:BD65
:OMCodes
:IDENt?

Parameters: None

Response: <NR1>

ident switch

Returned values: ident switch: integer

Description: Determine the Ident Switch Active bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:OMC:IDEN?`

:ADSB
:GENerate
:BD65
:OMCodes
:RATC

Parameters: <NRf>

receiving atc

Valid values: receiving atc: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Receiving ATC Services bit of the bds 6,5 extended squitters.

Only valid if operational mode format is 0. No checking is performed so only set ident switch to 1 if it is valid to do so

Example: `ADSB:GEN:BD65:OMC:RATC 1`

Indicate receiving atc services.

:ADSB
:GENerate
:BD65
:OMCodes
:RATC?

Parameters: None

Response: <NR1>

receiving atc

Returned values: receiving atc: integer

Description: Determine the Receiving ATC Services bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:OMC:RATC?`

:ADSB
:GENerate
:BD65
:OMCodes
:SA

Parameters: <NRf>

Single Antenna Flag

Valid values: Single Antenna Flag: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: DO-260B only.
Set the Single Antenna Flag bit of the bds 6,5 extended squitters.

Example: `ADSB:GEN:BD65:OMC:SA 0`

Indicate single or dual antenna.

:ADSB
:GENerate
:BD65
:OMCodes
:SA?

Parameters: None

Response: <NR1>

Single Antenna Flag

Returned values: Single Antenna Flag: integer

Description: DO-260B only.
Determine the Single Antenna Flag bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:OMC:SA?`

:ADSB
:GENerate
:BD65
:OMCodes
:SDA

Parameters: <NRF>

Valid values: SDA (System Design Assurance): integer. Valid values are 0 to 3. Values outside range are rejected and an error generated.

Description: DO-260B only.

Set the System Design Assurance bit of the bds 6,5 extended squitters.

Example: `ADSB:GEN:BD65:OMC:SDA 0`

Indicate failure or caution.

:ADSB
:GENerate
:BD65
:OMCodes
:SDA?

Parameters: None

Response: <NR1>

System Design Assurance

Returned values: System Design Assurance: integer

Description: DO-260B only.

Determine the System Design Assurance bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:OMC:SDA?`

:ADSB
:GENerate
:BD65
:OMCodes
:TRA

Parameters: <NRf>

tcas RA active

Valid values: tcas RA active: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the TCAS Resolution Advisory Active bit of the bds 6,5 extended squitters.

Only valid if operational mode format is 0. No checking is performed so only set ident switch to 1 if it is valid to do so

Example: `ADSB:GEN:BD65:OMC:TRA 0`

Indicate TCAS II or ACAS RA not active.

:ADSB
:GENerate
:BD65
:OMCodes
:TRA?

Parameters: None

Response: <NR1>

tcas RA active

Returned values: tcas RA active: integer

Description: Determine the TCAS Resolution Advisory Active bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:OMC:TRA?`

:ADSB
:GENerate
:BD65
:PERiod

Parameters: <NRf>

transmit period

Valid values: transmit period: real

Description: Set the period (ie rate) of the bds 6,5 extended squitters. Value can be set to 2dp within the range 0.50 seconds and 20.00 seconds.

Example: `ADSB:GEN:BD65:PER 0.5`

Set bds 6,5 extended squitter to be sent out every 0.5 seconds.

:ADSB

:GENerate

:BD65

:PERiod?

Parameters: None

Response: <NR2>

transmit period

Returned values: transmit period: real

Description: Determine the period of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:PER?`

:ADSB

:GENerate

:BD65

:SCCCodes

:B2Low

Parameters: <NRf>

b2low

Valid values: b2low: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Class B2 Transmit Power Less Than 70 Watts bit (part of the surface capability class codes field) of the bds 6,5 extended squitters.

This bit is only valid if subtype is one. No checking is performed so only set b2low to 1 if it is valid to do so.

Example: `ADSB:GEN:BD65:SCCC:B2L 0`

Indicate greater than or equal to 70 Watts transmit power.

:ADSB
:GENerate
:BD65
:SCCCodes
:B2Low?

Parameters: None

Response: <NR1>

b2low

Returned values: b2low: integer

Description: Determine the b2low bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:SCCC:B2L?`

:ADSB
:GENerate
:BD65
:SCCCodes
:NACV

Parameters: <NRf>

NACV

Valid values: NACV: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Indicates Horizontal Velocity Error is unknown or $\geq$10 m/s

This bit is only valid if subtype is one. No checking is performed so only set NACV to 1 if it is valid to do so.

Example: `ADSB:GEN:BD65:SCCC:NACV 0`

Indicate greater than or equal to 70 Watts transmit power.

:ADSB

:GENerate

:BD65

:SCCCodes

: NACV?

Parameters: None

Response: <NR1>

NACV

Returned values: NACV: integer

Description: Determine the NACV bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:SCCC:NACV?`

:ADSB
:GENerate
:BD65
:SCCCodes
:NICC

Parameters: <NRf>

NICC

Valid values: NICC: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: DO-260B only.

Subtype 1 (Surface)

Used to encode the Radius of Containment (part of the surface capability class codes field) of the bds 6,5 extended squitters.

This bit is only valid if subtype is one. No checking is performed so only set NICC to 1 if it is valid to do so.

Example: `ADSB:GEN:BD65:SCCC:NICC 0`

Indicate NICC set to 0 for Radius of Containment encoding.

:ADSB
:GENerate
:BD65
:SCCCodes
: NICC?

Parameters: None

Response: <NR1>

NICC

Returned values: NICC: integer

Description: Determine the NICC bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:SCCC:NICC?`

:ADSB

:GENerate

:BD65

:SILSupplement

Parameters: <NRf>

SILS

Valid values: SILS: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level Supplement field of the bds 6,5 extended squitters.

Example: `ADSB:GEN:BD65:SILS 1`

Set SILS field in bds 6,5 extended squitter to be 1.

:ADSB

:GENerate

:BD65

:SILSupplement?

Parameters: None

Response: <NR1>

SISL

Returned values: SILS: integer

Description: Determine the SILS field of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:SILS?`

:ADSB
:GENerate
:BD65
:POA

Parameters: <NRf>

poa

Valid values: poa: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Position Offset Applied bit (part of the surface capability class codes field) of the bds 6,5 extended squitters.

This bit is only valid if subtype is one. No checking is performed so only set poa to 1 if it is valid to do so.

Example: `ADSB:GEN:BD65:SCCC:POA 1`

Indicate position transmitted is the ADSB position reference point.

:ADSB
:GENerate
:BD65
:POA?

Parameters: None

Response: <NR1>

poa

Returned values: poa: integer

Description: Determine the poa bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:SCCC:POA?`

:ADSB
:GENerate
:BD65
:SIL

Parameters: <NRf>

SIL

Valid values: SIL: integer. Valid values are 0 to 3. Values outside range are rejected and an error generated.

Description: Set the Surveillance Integrity Level field of the bds 6,5 extended squitters.

Example: `ADSB:GEN:BD65:SIL 3`

Set SIL field in bds 6,5 extended squitter to be 3.

:ADSB
:GENerate
:BD65
:SIL?

Parameters: None

Response: <NR1>

SIL

Returned values: SIL: integer

Description: Determine the SIL field of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:SIL?`

:ADSB
:GENerate
:BD65
:STARt

Parameters: None

Description: This starts the adsb gen bds6,5 (Aircraft Operational Status message) test. The test will run continuously until stopped. If the test is not enabled then the test will not start and an error will be reported.

Example: `ADSB:GEN:BD65:STAR`

Start adsb gen bds6,5 test.

:ADSB
:GENerate
:BD65
:STARt:AIR

Parameters: None

Description: This starts the adsb gen bds6,5 (Aircraft Operational Status message) Airborne test. The test will run continuously until stopped. If the test is not enabled then the test will not start and an error will be reported.

Example: `ADSB:GEN:BD65:STAR:AIR`

Start adsb gen bds6,5 test.

:ADSB
:GENerate
:BD65
:STARt:SUR

Parameters: None

Description: This starts the adsb gen bds6,5 (Aircraft Operational Status message) Surface test. The test will run continuously until stopped. If the test is not enabled then the test will not start and an error will be reported.

Example: `ADSB:GEN:BD65:STAR:SUR`

Start adsb gen bds6,5 test.

:ADSB
:GENerate
:BD65
:STYPe

Parameters: <NRf>

subtype code

Valid values: subtype code: integer

Description: Set the sub-type of the bds 6,5 extended squitters. Valid range is 0 to 1. All other values will generate an error.

Subtype 0 indicates Airborne Status Message.
Subtype 1 indicates Surface Status Message.

Example: `ADSB:GEN:BD65:STYP 1`

Set subtype field in bds 6,5 extended squitter to be 1.

:ADSB
:GENerate
:BD65
:STYPe?

Parameters: None

Response: <NR1>

subtype code

Returned values: subtype code: integer

Description: Determine the subtype field of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:STYP?`

:ADSB

:GENerate

:BD65

:TAHeading

Parameters: <NRf>

heading

Valid values: heading: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the Track Angle/Heading bit of the bds 6,5 extended squitters.

This bit is only valid if subtype is one. No checking is performed so only set heading to 1 if it is valid to do so.

Example: `ADSB:GEN:BD65:TAH 0`

Set track angle/heading bit to 0.

:ADSB

:GENerate

:BD65

:TAHeading?

Parameters: None

Response: <NR1>

heading

Returned values: heading: integer

Description: Determine the track angle/heading bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:TAH?`

:ADSB

:GENerate

:BD65

:UAT

Parameters: <NRf>

UAT

Valid values: UAT: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: DO-206B: Set the UAT bit (part of the capability class codes field) of the bds 6,5 extended squitters.

Example: `ADSB:GEN:BD65:UAT 0`

Set UATbit in bds 6,5 extended squitter to indicate traffic display not operational.

:ADSB

:GENerate

:BD65

:UAT?

Parameters: None

Response: <NR1>

UAT

Returned values: UAT: integer

Description: DO-260B: Determine the UAT bit of the bds 6,5 extended squitter.

Example: `ADSB:GEN:BD65:UAT?`

:ADSB
:GENerate
:BD65
:VNUMber

Parameters: <CPD>

version no

Valid values: version no: [ORIGinal | REVA | REServed]. Values other than those stated are rejected and an error generated.

Description: Set the Version Number field in the bds 6,5 extended squitter. Selects whether conforming to DO-260, DO-260A or DO-260B.

Example: `ADSB:GEN:BD65:VNUM REVA`

We conform to DO-260A and DO-260B.

:ADSB
:GENerate
:BD65
: VNUMber?

Parameters: None

Response: <CRD>

version no

Returned values: version no: [ORIG | REVA | RES]

Description: Determine Version Number field.

Example: `ADSB:GEN:BD65:VNUM?`

:ADSB

:GENerate

:STOP

Parameters: None

Description: This stops the currently running adsb generate test.

Example: `ADSB:GEN:STOP`

Stop adsb gen test.

:ADSB

:GICB

:ALL?

Parameters: None

Response: <CRD>

test status

Returned values: test status: [PASS | ERR]

Description: This runs all the adsb monitor tests. The tests will run once and then a status will be returned.

Results can be read back using the individual data query commands for each bds test.

Example: `ADSB:GICB:ALL?`

Run all the adsb gicb tests.

:ADSB

:GICB

:BD05

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>,<NR1>, <NR1>, <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type, addr state, addr, latitude state, latitude degrees, latitude minutes, latitude seconds, latitude direction, longitude state, longitude degrees, longitude minutes, longitude seconds, longitude direction, saf state, saf, t state, t, surv state, surv, baro pressure alt state, baro pressure alt, gnss alt state, gnss alt, NIC, Rc, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

latitude state: [PASS | FAIL | INV | NDAT | NAV]

latitude degrees: integer. Values are in the range 0 to 90.

latitude minutes: integer. Values are in the range 0 to 59.

latitude seconds: integer. Values are in the range 0 to 59.

latitude direction: [NORT | SOUT]

longitude state: [PASS | FAIL | INV | NDAT | NAV]

longitude degrees: integer. Values are in the range 0 to 180.

longitude minutes: integer. Values are in the range 0 to 59.

longitude seconds: integer. Values are in the range 0 to 59.

longitude direction: [EAST | WEST]

[:DATA]? (cont)

Returned values:

saf state: [PASS | FAIL | INV | NDAT | NAV]

saf: integer

t state: [PASS | FAIL | INV | NDAT | NAV]

t: integer

surv state: [PASS | FAIL | INV | NDAT | NAV]

surv: integer

baro pressure alt state: [PASS | FAIL | INV | NDAT | NAV]

baro pressure alt: integer

gnss alt state: [PASS | FAIL | INV | NDAT | NAV]

gnss alt: integer

NIC, Rc: Refer to tables in Operation Manual

MB state: [PASS | FAIL | INV | NDAT | NAV]

Description:

Read back all the measured adsb gicb bds0,5 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example:

```
ADSB:GICB:BD05?
```

:ADSB

:GICB

:BD05

:STARt

Parameters: None

Description: This starts the adsb gicb bds0,5 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD05:STAR`

Start adsb gicb bds0,5 test.

:ADSB

:GICB

:BD06

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>,<NR1>, <NR1>, <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <CRD>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type, addr state, addr, latitude state, latitude degrees, latitude minutes, latitude seconds, latitude direction, longitude state, longitude degrees, longitude minutes, longitude seconds, longitude direction, movement state, movement, t state, t, hdg state, hdg, NIC, Rc, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

latitude state: [PASS | FAIL | INV | NDAT | NAV]

latitude degrees: integer. Values are in the range 0 to 90.

latitude minutes: integer. Values are in the range 0 to 59.

latitude seconds: integer. Values are in the range 0 to 59.

latitude direction: [NORT | SOUT]

longitude state: [PASS | FAIL | INV | NDAT | NAV]

longitude degrees: integer. Values are in the range 0 to 180.

longitude minutes: integer. Values are in the range 0 to 59.

longitude seconds: integer. Values are in the range 0 to 59.

longitude direction: [EAST | WEST]

movement state: [PASS | FAIL | INV | NDAT | NAV]

movement: string. Maximum length of 17 characters excluding quotes.

t state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB

:GICB

:BD06

[:DATA]? (cont)

Returned values: t: integer

hdg state: [PASS | FAIL | INV | NDAT | NAV]

hdg: integer

NIC, Rc: Refer to tables in Operation Manual

MB state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb gicb bds0,6 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field) and the movement field which is returned as a string.

Valid movement fields are:
"NO INFO"
"STOPPED"
"0.125 kt to <1 kt"
"1 kt to <2 kt"
"2 kt to <15 kt"
"15 kt to <70 kt"
"70 kt to <100 kt"
"100 kt to <175 kt"
"175 kt"
"DECELERATING"
"ACCELERATING"
"BACKING UP"

Example: `ADSB:GICB:BD06?`

:ADSB

:GICB

:BD06

:STARt

Parameters: None

Description: This starts the adsb gicb bds0,6 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD06:STAR`

Start adsb gicb bds0,6 test.

:ADSB
:GICB
:BD07
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <STRING RESPONSE DATA>

overall state, addr state, addr, MB state, MB, surf sqtr trans rt state, surf sqtr trans rt, alt type state, alt type, ESS state, ESS

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

MB: string. Maximum length of 14 characters excluding quotes.

surf sqtr trans rt state: [PASS | FAIL | INV | NDAT | NAV]

surf sqtr trans rt: [UNKN | HIGH | LOW | RES]

alt type state: [PASS | FAIL | INV | NDAT | NAV]

alt type: [BARO | GNSS]

ESS state: [PASS | FAIL | INV | NDAT | NAV]

ESS: string. Maximum length of 14 characters excluding quotes.

Description: Read back all the measured adsb gicb bds0,7 items.

All values are returned in decimal except for the MB and ESS fields which are returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD07?`

:ADSB

:GICB

:BD07

:STARt

Parameters: None

Description: This starts the adsb gicb bds0,7 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD07:STAR`

Start adsb gicb bds0,7 test.

:ADSB
:GICB
:BD08
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>

overall state, type state, type, addr state, addr, MB state, MB, AIS state, AIS, flight ID state, flight ID, emitcatset state, emitcatset, emitcat state, emitcat

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

MB: string. Maximum length of 14 characters excluding quotes.

AIS state: [PASS | FAIL | INV | NDAT | NAV]

AIS: string. Maximum length of 12 characters excluding quotes.

flight ID state: [PASS | FAIL | INV | NDAT | NAV]

flight ID: string. Maximum length of 8 characters excluding quotes.

emitcatset state: [PASS | FAIL | INV | NDAT | NAV]

emitcatset: integer

emitcat state: [PASS | FAIL | INV | NDAT | NAV]

emitcat: string. Maximum length of 25 characters excluding quotes.

:ADSB

:GICB

:BD08

[:DATA]? (cont)

Description: Read back all the measured adsb gicb bds0,8 items.

All values are returned in decimal except for the MB field which is returned as a 14 hexadecimal digits (56-bit field) string, the AIS field which is returned as a 12 hexadecimal digits string, and the flight ID and emitcat fields which are returned as strings.

Valid emitcat fields are:

"NO ADS-B EMITTER INFO"
"LIGHT"
"SMALL"
"LARGE"
"HIGH VORTEX"
"HEAVY"
"HIGH PERFORMANCE"
"ROTOCRAFT"
"GLIDER/SAILPLANE"
"LIGHTER-THAN-AIR"
"PARACHUTIST/SKYDIVER"
"ULTRALIGHT/HANG-GLIDER"
"UNMANNED AERIAL VEHICLE"
"SPACE VEHICLE"
"SURFACE EMERGENCY VEHICLE"
"SURFACE SERVICE VEHICLE"
"FIXED GND/TETHERED OBSTR"
"CLUSTER OBSTR"
"LINE OBSTR"
"RESERVED"

Example: `ADSB:GICB:BD08?`

:ADSB

:GICB

:BD08

:STARt

Parameters: None

Description: This starts the adsb gicb bds0,8 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD08:STAR`

Start adsb gicb bds0,8 test.

:ADSB

:GICB

:BD09

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <NR1>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type, addr state, addr, ew vel state, ew velocity, ew velocity direction, ns vel state, ns velocity, ns velocity direction, nac state, nac, subtype state, subtype, vrate state, vrate, hdg state, hdg, geodiffbaro state, geodiffbaro, source state, source, intent state, intent, airspeed state, airspeed, airspeedtype state, airspeedtype, ifrcapflag state, ifrcapflag, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

ew vel state: [PASS | FAIL | INV | NDAT | NAV]

ew vel: integer

ew vel direction: [EAST | WEST]

ns vel state: [PASS | FAIL | INV | NDAT | NAV]

ns vel: integer

ns vel direction: [NORT | SOUT]

nac state: [PASS | FAIL | INV | NDAT | NAV]

nac: integer

subtype state: [PASS | FAIL | INV | NDAT | NAV]

subtype: integer

vrate state: [PASS | FAIL | INV | NDAT | NAV]

[:DATA]? (cont)

Returned values: vrate: integer

hdg state: [PASS | FAIL | INV | NDAT | NAV]

hdg: real

geodiffbaro state: [PASS | FAIL | INV | NDAT | NAV]

geodiffbaro: integer

source state: [PASS | FAIL | INV | NDAT | NAV]

source: [GEO | BARO]

intent state: [PASS | FAIL | INV | NDAT | NAV]

intent: integer

airspeed state: [PASS | FAIL | INV | NDAT | NAV]

airspeed: integer

airspeed type state: [PASS | FAIL | INV | NDAT | NAV]

airspeed type: [IAS | TAS]

ifrcap flag state: [PASS | FAIL | INV | NDAT | NAV]

ifrcap flag: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb gicb bds0,9 items.

All values are returned in decimal except for source and airspeed type fields which are character response data, and the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD09?`

:ADSB

:GICB

:BD09

:STARt

Parameters: None

Description: This starts the adsb gicb bds0,9 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD09:STAR`

Start adsb gicb bds0,9 test.

:ADSB

:GICB

:BD10

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>,<CRD>, <NR1>,<CRD>, <NR1>,<CRD>, <NR1>,<CRD>, <NR1>,<CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <STRING RESPONSE DATA>

overall state, addr state, addr, MB state, MB, comm use gicb rep state, comm use gicb rep, subnet ver nbr state, subnet ver nbr, xpdr lvl state, xpdr lvl, aircraft id cap state, aircraft id cap, si cap state, si cap, cont flag state, cont flag, sqtr cap state, sqtr cap, spec serv cap rep state, spec serv cap rep, dte state, dte, uelm state, uelm, delm state, delm

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

MB: string. Maximum length of 14 characters excluding quotes.

comm use gicb rep state: [PASS | FAIL | INV | NDAT | NAV]

comm use gicb rep: integer

subnet ver nbr state: [PASS | FAIL | INV | NDAT | NAV]

subnet ver nbr: integer

xpdr lvl state: [PASS | FAIL | INV | NDAT | NAV]

xpdr lvl: [L24 | L5]

aircraft id cap state: [PASS | FAIL | INV | NDAT | NAV]

aircraft id cap: integer

si cap state: [PASS | FAIL | INV | NDAT | NAV]

si cap: integer

cont flag state: [PASS | FAIL | INV | NDAT | NAV]

cont flag: integer

[:DATA]? (cont)

Returned values: sqtr cap state: [PASS | FAIL | INV | NDAT | NAV]

sqtr cap: integer

spec serv cap rep state: [PASS | FAIL | INV | NDAT | NAV]

spec serv cap rep: integer

dte state: [PASS | FAIL | INV | NDAT | NAV]

dte: integer

uelm state: [PASS | FAIL | INV | NDAT | NAV]

uelm: string. Maximum length of 10 characters excluding quotes.

delm state: [PASS | FAIL | INV | NDAT | NAV]

delm: string. Maximum length of 10 characters excluding quotes.

Description: Read back all the measured adsb gicb bds1,0 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field), and the uelm/delm strings which are defined below:

Uelm string will return one of the following strings:

"NO UELM"
"16/1 S"
"16/500 mS"
"16/250 mS"
"16/125 mS"
"16/60 mS"
"16/30 mS"
"UNASSIGNED"

Delm string will return one of the following strings:

"NO DELM"
"4/1 S"
"8/1 S"
"16/1 S"
"16/500 mS"
"16/250 mS"
"16/125 mS"
"UNASSIGNED"

Example: `ADSB:GICB:BD10?`

:ADSB
:GICB
:BD10
:STARt

Parameters: None

Description: This starts the adsb gicb bds1,0 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD10:STAR`

Start adsb gicb bds1,0 test.

:ADSB

:GICB

:BD17

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, cap05, cap06, cap07, cap08, cap09, cap0a, cap20, cap21, cap40, cap41, cap42, cap43, cap44, cap45, cap48, cap50, cap51, cap52, cap53, cap54, cap55, cap56, cap5f, cap60, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

cap05: integer

cap06: integer

cap07: integer

cap08: integer

cap09: integer

cap0a: integer

cap20: integer

cap21: integer

cap40: integer

cap41: integer

cap42: integer

cap43: integer

cap44: integer

cap45: integer

cap48: integer

:ADSB

:GICB

:BD17

[:DATA]? (cont)

Returned values: cap50: integer

cap51: integer

cap52: integer

cap53: integer

cap54: integer

cap55: integer

cap56: integer

cap5f: integer

cap60: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb gicb bds1,7 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD17?`

:ADSB
:GICB
:BD17
:STARt

Parameters: None

Description: This starts the adsb gicb bds1,7 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD17:STAR`

Start adsb gicb bds1,7 test.

:ADSB

:GICB

:BD18

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, cap01, cap02, cap03, cap04, cap05, cap06, cap07, cap08, cap09, cap0a, cap0b, cap0c, cap0d, cap0e, cap0f, cap10, cap11, cap12, cap13, cap14, cap15, cap16, cap17, cap18, cap19, cap1a, cap1b, cap1c, cap1d, cap1e, cap1f, cap20, cap21, cap22, cap23, cap24, cap25, cap26, cap27, cap28, cap29, cap2a, cap2b, cap2c, cap2d, cap2e, cap2f, cap30, cap31, cap32, cap33, cap34, cap35, cap36, cap37, cap38, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

cap01: integer

cap02: integer

cap03: integer

cap04: integer

cap05: integer

cap06: integer

cap07: integer

cap08: integer

cap09: integer

cap0a: integer

cap0b: integer

cap0c: integer

:ADSB

:GICB

:BD18

[:DATA]? (cont)

Returned values: cap0d: integer

cap0e: integer

cap0f: integer

cap10: integer

cap11: integer

cap12: integer

cap13: integer

cap14: integer

cap15: integer

cap16: integer

cap17: integer

cap18: integer

cap19: integer

cap1a: integer

cap1b: integer

cap1c: integer

cap1d: integer

cap1e: integer

cap1f: integer

cap20: integer

cap21: integer

cap22: integer

cap23: integer

:ADSB

:GICB

:BD18

[:DATA]? (cont)

Returned values:

cap24: integer

cap25: integer

cap26: integer

cap27: integer

cap28: integer

cap29: integer

cap2a: integer

cap2b: integer

cap2c: integer

cap2d: integer

cap2e: integer

cap2f: integer

cap30: integer

cap31: integer

cap32: integer

cap33: integer

cap34: integer

cap35: integer

cap36: integer

cap37: integer

cap38: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB
:GICB
:BD18
[:DATA]? (cont)

Description: Read back all the measured adsb gicb bds1,8 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD18?`

:ADSB
:GICB
:BD18
:STARt

Parameters: None

Description: This starts the adsb gicb bds1,8 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD18:STAR`

Start adsb gicb bds1,8 test.

:ADSB
:GICB
:BD19
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, cap39, cap3a, cap3b, cap3c, cap3d, cap3e, cap3f, cap40, cap41, cap42, cap43, cap44, cap45, cap46, cap47, cap48, cap49, cap4a, cap4b, cap4c, cap4d, cap4e, cap4f, cap50, cap51, cap52, cap53, cap54, cap55, cap56, cap57, cap58, cap59, cap5a, cap5b, cap5c, cap5d, cap5e, cap5f, cap60, cap61, cap62, cap63, cap64, cap65, cap66, cap67, cap68, cap69, cap6a, cap6b, cap6c, cap6d, cap6e, cap6f, cap70, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

cap39: integer

cap3a: integer

cap3b: integer

cap3c: integer

cap3d: integer

cap3e: integer

cap3f: integer

cap40: integer

cap41: integer

cap42: integer

cap43: integer

cap44: integer

:ADSB

:GICB

:BD19

[:DATA]? (cont)

Returned values: cap45: integer

cap46: integer

cap47: integer

cap48: integer

cap49: integer

cap4a: integer

cap4b: integer

cap4c: integer

cap4d: integer

cap4e: integer

cap4f: integer

cap50: integer

cap51: integer

cap52: integer

cap53: integer

cap54: integer

cap55: integer

cap56: integer

cap57: integer

cap58: integer

cap59: integer

cap5a: integer

cap5b: integer

:ADSB

:GICB

:BD19

[:DATA]? (cont)

Returned values: cap5c: integer

cap5d: integer

cap5e: integer

cap5f: integer

cap60: integer

cap61: integer

cap62: integer

cap63: integer

cap64: integer

cap65: integer

cap66: integer

cap67: integer

cap68: integer

cap69: integer

cap6a: integer

cap6b: integer

cap6c: integer

cap6d: integer

cap6e: integer

cap6f: integer

cap70: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB
:GICB
:BD19
[:DATA]? (cont)

Description: Read back all the measured adsb gicb bds1,9 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD19?`

:ADSB
:GICB
:BD19
:STARt

Parameters: None

Description: This starts the adsb gicb bds1,9 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD19:STAR`

Start adsb gicb bds1,9 test.

:ADSB

:GICB

:BD1A

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, cap71, cap72, cap73, cap74, cap75, cap76, cap77, cap78, cap79, cap7a, cap7b, cap7c, cap7d, cap7e, cap7f, cap80, cap81, cap82, cap83, cap84, cap85, cap86, cap87, cap88, cap89, cap8a, cap8b, cap8c, cap8d, cap8e, cap8f, cap90, cap91, cap92, cap93, cap94, cap95, cap96, cap97, cap98, cap99, cap9a, cap9b, cap9c, cap9d, cap9e, cap9f, capa0, capa1, capa2, capa3, capa4, capa5, capa6, capa7, capa8, MB state, MB

Returned values: Overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

cap71: integer

cap72: integer

cap73: integer

cap74: integer

cap75: integer

cap76: integer

cap77: integer

cap78: integer

cap79: integer

cap7a: integer

cap7b: integer

cap7c: integer

:ADSB
:GICB
:BD1A
[:DATA]? (cont)

Returned values: cap7d: integer

cap7e: integer

cap7f: integer

cap80: integer

cap81: integer

cap82: integer

cap83: integer

cap84: integer

cap85: integer

cap86: integer

cap87: integer

cap88: integer

cap89: integer

cap8a: integer

cap8b: integer

cap8c: integer

cap8d: integer

cap8e: integer

cap8f: integer

cap90: integer

cap91: integer

cap92: integer

cap93: integer

:GICB

:BD1A

[:DATA]? (cont)

Returned values: cap94: integer

cap95: integer

cap96: integer

cap97: integer

cap98: integer

cap99: integer

cap9a: integer

cap9b: integer

cap9c: integer

cap9d: integer

cap9e: integer

cap9f: integer

capa0: integer

capa1: integer

capa2: integer

capa3: integer

capa4: integer

capa5: integer

capa6: integer

capa7: integer

capa8: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB
:GICB
:BD1A
[:DATA]? (cont)

Description: Read back all the measured adsb gicb bds1,A items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD1A?`

:ADSB
:GICB
:BD1A
:STARt

Parameters: None

Description: This starts the adsb gicb bds1,A test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD1A:STAR`

Start adsb gicb bds1,A test.

:ADSB

:GICB

:BD1B

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, capa9, capaa, capab, capac, capad, capae, capaf, capb0, capb1, capb2, capb3, capb4, capb5, capb6, capb7, capb8, capb9, capba, capbb, capbc, capbd, capbe, capbf, capc0, capc1, capc2, capc3, capc4, capc5, capc6, capc7, capc8, capc9, capca, capcb, capcc, capcd, capce, capcf, capd0, capd1, capd2, capd3, capd4, capd5, capd6, capd7, capd8, capd9, capda, capdb, capdc, capdd, capde, capdf, cape0, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

capa9: integer

capaa: integer

capab: integer

capac: integer

capad: integer

capae: integer

capaf: integer

capb0: integer

capb1: integer

capb2: integer

capb3: integer

capb4: integer

:ADSB

:GICB

:BD1B

[:DATA]? (cont)

Returned values:

capb5: integer

capb6: integer

capb7: integer

capb8: integer

capb9: integer

capba: integer

capbb: integer

capbc: integer

capbd: integer

capbe: integer

capbf: integer

capc0: integer

capc1: integer

capc2: integer

capc3: integer

capc4: integer

capc5: integer

capc6: integer

capc7: integer

capc8: integer

capc9: integer

capca: integer

capcb: integer

[:DATA]? (cont)

Returned values:

capcc: integer

capcd: integer

capce: integer

capcf: integer

capd0: integer

capd1: integer

capd2: integer

capd3: integer

capd4: integer

capd5: integer

capd6: integer

capd7: integer

capd8: integer

capd9: integer

capda: integer

capdb: integer

capdc: integer

capdd: integer

capde: integer

capdf: integer

cape0: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB
:GICB
:BD1B
[:DATA]? (cont)

Description: Read back all the measured adsb gicb bds1,b items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD1B?`

:ADSB
:GICB
:BD1B
:STARt

Parameters: None

Description: This starts the adsb gicb bds1,b test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD1B:STAR`

Start adsb gicb bds1,b test.

:ADSB

:GICB

:BD1C

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, cape1, cape2, cape3, cape4, cape5, cape6, cape7, cape8, cape9, capea, capeb, capec, caped, capee, capef, capf0, capf1, capf2, capf3, capf4, capf5, capf6, capf7, capf8, capf9, capfa, capfb, capfc, capfd, capfe, capff, MB state, MB

Returned values: Overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

cape1: integer

cape2: integer

cape3: integer

cape4: integer

cape5: integer

cape6: integer

cape7: integer

cape8: integer

cape9: integer

capea: integer

capeb: integer

capec: integer

caped: integer

:ADSB

:GICB

:BD1C

[:DATA]? (cont)

Returned values: capee: integer

capef: integer

capf0: integer

capf1: integer

capf2: integer

capf3: integer

capf4: integer

capf5: integer

capf6: integer

capf7: integer

capf8: integer

capf9: integer

capfa: integer

capfb: integer

capfc: integer

capfd: integer

capfe: integer

capff: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb gicb bds1,C items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD1C?`

:ADSB

:GICB

:BD1C

:STARt

Parameters: None

Description: This starts the adsb gicb bds1,C test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD1C:STAR`

Start adsb gicb bds1,C test.

:ADSB

:GICB

:BD1D

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, up1, up2, up3, up4, up5, up6, up7, up8, up9, up10, up11, up12, up13, up14, up15, up16, up17, up18, up19, up20, up21, up22, up23, up24, up25, up26, up27, up28, dn1, dn2, dn3, dn4, dn5, dn6, dn7, dn8, dn9, dn10, dn11, dn12, dn13, dn14, dn15, dn16, dn17, dn18, dn19, dn20, dn21, dn22, dn23, dn24, dn25, dn26, dn27, dn28, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

up1: integer

up2: integer

up3: integer

up4: integer

up5: integer

up6: integer

up7: integer

up8: integer

up9: integer

up10: integer

up11: integer

up12: integer

:ADSB

:GICB

:BD1D

[:DATA]? (cont)

Returned values: up13: integer

up14: integer

up15: integer

up16: integer

up17: integer

up18: integer

up19: integer

up20: integer

up21: integer

up22: integer

up23: integer

up24: integer

up25: integer

up26: integer

up27: integer

up28: integer

dn1: integer

dn2: integer

dn3: integer

dn4: integer

dn5: integer

dn6: integer

dn7: integer

:ADSB

:GICB

:BD1D

[:DATA]? (cont)

Returned values:

dn8: integer

dn9: integer

dn10: integer

dn11: integer

dn12: integer

dn13: integer

dn14: integer

dn15: integer

dn16: integer

dn17: integer

dn18: integer

dn19: integer

dn20: integer

dn21: integer

dn22: integer

dn23: integer

dn24: integer

dn25: integer

dn26: integer

dn27: integer

dn28: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB
:GICB
:BD1D
[:DATA]? (cont)

Description: Read back all the measured adsb gicb bds1,d items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD1D?`

:ADSB
:GICB
:BD1D
:STARt

Parameters: None

Description: This starts the adsb gicb bds1,d test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD1D:STAR`

Start adsb gicb bds1,d test.

:ADSB
:GICB
:BD1E
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, up29, up30, up31, up32, up33, up34, up35, up36, up37, up38, up39, up40, up41, up42, up43, up44, up45, up46, up47, up48, up49, up50, up51, up52, up53, up54, up55, up56, dn29, dn30, dn31, dn32, dn33, dn34, dn35, dn36, dn37, dn38, dn39, dn40, dn41, dn42, dn43, dn44, dn45, dn46, dn47, dn48, dn49, dn50, dn51, dn52, dn53, dn54, dn55, dn56, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

up29: integer

up30: integer

up31: integer

up32: integer

up33: integer

up34: integer

up35: integer

up36: integer

up37: integer

up38: integer

up39: integer

up40: integer

[:DATA]? (cont)

Returned values: up41: integer

up42: integer

up43: integer

up44: integer

up45: integer

up46: integer

up47: integer

up48: integer

up49: integer

up50: integer

up51: integer

up52: integer

up53: integer

up54: integer

up55: integer

up56: integer

dn29: integer

dn30: integer

dn31: integer

dn32: integer

dn33: integer

dn34: integer

dn35: integer

[:DATA]? (cont)

Returned values: dn36: integer

dn37: integer

dn38: integer

dn39: integer

dn40: integer

dn41: integer

dn42: integer

dn43: integer

dn44: integer

dn45: integer

dn46: integer

dn47: integer

dn48: integer

dn49: integer

dn50: integer

dn51: integer

dn52: integer

dn53: integer

dn54: integer

dn55: integer

dn56: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB
:GICB
:BD1E
[:DATA]? (cont)

Description: Read back all the measured adsb gicb bds1,e items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD1E?`

:ADSB
:GICB
:BD1E
:STARt

Parameters: None

Description: This starts the adsb gicb bds1,e test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD1E:STAR`

Start adsb gicb bds1,e test.

:ADSB
:GICB
:BD1F
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, up57, up58, up59, up60, up61, up62, up63, dn57, dn58, dn59, dn60, dn61, dn62, dn63, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

up57: integer

up58: integer

up59: integer

up60: integer

up61: integer

up62: integer

up63: integer

dn57: integer

dn58: integer

dn59: integer

dn60: integer

dn61: integer

dn62: integer

dn63: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB
:GICB
:BD1F
[:DATA]?

Description: Read back all the measured adsb gicb bds1,f items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD1F?`

:ADSB
:GICB
:BD1F
:STARt

Parameters: None

Description: This starts the adsb gicb bds1,f test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD1F:STAR`

Start adsb gicb bds1,f test.

:ADSB
:GICB
:BD20
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <STRING RESPONSE DATA>

overall state, addr state, addr, MB state, MB, flight ID state, flight ID

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

MB: string. Maximum length of 14 characters excluding quotes.

flight ID state: [PASS | FAIL | INV | NDAT | NAV]

flight ID: string. Maximum length of 8 characters excluding quotes.

Description: Read back all the measured adsb gicb bds2,0 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD20?`

:ADSB
:GICB
:BD20
:STARt

Parameters: None

Description: This starts the adsb gicb bds2,0 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD20:STAR`

Start adsb gicb bds2,0 test.

:ADSB
:GICB
:BD21
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <STRING RESPONSE DATA>

overall state, addr state, addr, MB state, MB, ARN ID state, ARN ID, AR ID state, AR ID

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

MB: string. Maximum length of 14 characters excluding quotes.

ARN ID state: [PASS | FAIL | INV | NDAT | NAV]

ARN ID: string. Maximum length of 7 characters excluding quotes.

AR ID state: [PASS | FAIL | INV | NDAT | NAV]

AR ID: string. Maximum length of 2 characters excluding quotes.

Description: Read back all the measured adsb gicb bds2,1 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD21?`

:ADSB

:GICB

:BD21

:STARt

Parameters: None

Description: This starts the adsb gicb bds2,1 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD21:STAR`

Start adsb gicb bds2,1 test.

:ADSB

:GICB

:BD30

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>,<CRD>, <NR1>,<CRD>, <NR2>,<CRD>, <NR1>,<CRD>, <NR1>,<CRD>, <NR1>,<CRD>, <NR1>,<CRD>, <NR1>,<CRD>, <CRD>,<CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, tidb state, tidb, tida state, tida, tidr state, tidr, ara state, ara, tid state, tid, rac state, rac, rat state, rat, mte state, mte, tti state, tti, threat addr state, threat addr, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

tidb state: [PASS | FAIL | INV | NDAT | NAV]

tidb: integer

tida state: [PASS | FAIL | INV | NDAT | NAV]

tida: integer

tidr state: [PASS | FAIL | INV | NDAT | NAV]

tidr: real

ara state: [PASS | FAIL | INV | NDAT | NAV]

ara: integer

tid state: [PASS | FAIL | INV | NDAT | NAV]

tid: integer

rac state: [PASS | FAIL | INV | NDAT | NAV]

rac: integer

rat state: [PASS | FAIL | INV | NDAT | NAV]

rat: integer

mte state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB
:GICB
:BD30
[:DATA]? (cont)

Returned values: mte: integer

tti state: [PASS | FAIL | INV | NDAT | NAV]

tti: [NTD | TADD | ARBD | NASS]

threat addr state: [PASS | FAIL | INV | NDAT | NAV]

threat addr: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb gicb bds3,0 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD30?`

:ADSB
:GICB
:BD30
:STARt

Parameters: None

Description: This starts the adsb gicb bds3,0 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD30:STAR`

Start adsb gicb bds3,0 test.

:ADSB

:GICB

:BD40

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, target alt state, target alt, mcpfcuselalt state, mcpfcuselalt, fms sel alt state, fms sel alt, vnav hold mode state, vnav hold mode, alt hold mode state, alt hold mode, source state, source, mode state, mode, baro pres set state, baro pres set, app mode active state, app mode active, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

target alt state: [PASS | FAIL | INV | NDAT | NAV]

target alt: [UNKN | AALT | FMSA | FSAL]

mcpfcuselalt state: [PASS | FAIL | INV | NDAT | NAV]

mcpfcuselalt: integer

fms sel alt state: [PASS | FAIL | INV | NDAT | NAV]

fms sel alt: real

vnav hold mode state: [PASS | FAIL | INV | NDAT | NAV]

vnav hold mode: integer

alt hold mode state: [PASS | FAIL | INV | NDAT | NAV]

alt hold mode: integer

source state: [PASS | FAIL | INV | NDAT | NAV]

source: integer

mode state: [PASS | FAIL | INV | NDAT | NAV]

mode: integer

baro pres set state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB
:GICB
:BD40
[:DATA]? (cont)

Returned values: baro pres set state: [PASS | FAIL | INV | NDAT | NAV]

baro pres set: integer

app mode active state: [PASS | FAIL | INV | NDAT | NAV]

app mode active: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb gicb bds4,0 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD40?`

:ADSB
:GICB
:BD40
:STARt

Parameters: None

Description: This starts the adsb gicb bds4,0 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD40:STAR`

Start adsb gicb bds4,0 test.

:ADSB
:GICB
:BD41
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, waypoint name state, waypoint name, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

waypoint name state: [PASS | FAIL | INV | NDAT | NAV]

waypoint name: string. Maximum length of 9 characters excluding quotes.

MB state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb gicb bds4,1 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD41?`

:ADSB
:GICB
:BD41
:STARt

Parameters: None

Description: This starts the adsb gicb bds4,1 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD41:STAR`

Start adsb gicb bds4,1 test.

:ADSB

:GICB

:BD42

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <NR1>, <NR1>, <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <CRD>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, lat state, lat deg, lat min, lat sec, lat direction, lon state, lon deg, lon min, lon sec, lon direction, crossing alt state, crossing alt, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

lat state: [PASS | FAIL | INV | NDAT | NAV]

lat deg: integer

lat min: integer

lat sec: integer

lat direction: [NORT | SOUT]

lon state: [PASS | FAIL | INV | NDAT | NAV]

lon deg: integer

lon min: integer

lon sec: integer

lon direction: [EAST | WEST]

crossing alt state: [PASS | FAIL | INV | NDAT | NAV]

crossing alt: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB
:GICB
:BD42
[:DATA]? (cont)

Description: Read back all the measured adsb gicb bds4,2 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD42?`

:ADSB
:GICB
:BD42
:STARt

Parameters: None

Description: This starts the adsb gicb bds4,2 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD42:STAR`

Start adsb gicb bds4,2 test.

:ADSB
:GICB
:BD43
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR2>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, bearing state, bearing, time state, time, dist state, dist, reserved, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

bearing state: [PASS | FAIL | INV | NDAT | NAV]

bearing: integer

time state: [PASS | FAIL | INV | NDAT | NAV]

time: real

dist state: [PASS | FAIL | INV | NDAT | NAV]

dist: real

reserved: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb gicb bds4,3 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD43?`

:ADSB

:GICB

:BD43

:STARt

Parameters: None

Description: This starts the adsb gicb bds4,3 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD43:STAR`

Start adsb gicb bds4,3 test.

:ADSB
:GICB
:BD50
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, roll angle state, roll angle, rate state, rate, ground speed state, ground speed, truen air speed state, true air speed, true track angle state, true track angle, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

roll angle state: [PASS | FAIL | INV | NDAT | NAV]

roll angle: real

rate state: [PASS | FAIL | INV | NDAT | NAV]

rate: real

ground speed state: [PASS | FAIL | INV | NDAT | NAV]

ground speed: integer

true air speed state: [PASS | FAIL | INV | NDAT | NAV]

true air speed: integer

true track angle state: [PASS | FAIL | INV | NDAT | NAV]

true track angle: real

MB state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb gicb bds5,0 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD50?`

:ADSB

:GICB

:BD50

:STARt

Parameters: None

Description: This starts the adsb gicb bds5,0 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD50:STAR`

Start adsb gicb bds5,0 test.

:ADSB

:GICB

:BD60

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, addr state, addr, ind air speed state, ind air speed, mach state, mach, baro alt state, baro alt, mag hdg state, mag hdg, ivvel addr state, ivvel addr, MB state, MB

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

ind air speed state: [PASS | FAIL | INV | NDAT | NAV]

ind air speed: real

mach state: [PASS | FAIL | INV | NDAT | NAV]

mach: real

baro alt state: [PASS | FAIL | INV | NDAT | NAV]

baro alt: integer

mag hdg state: [PASS | FAIL | INV | NDAT | NAV]

mag hdg: integer

ivvel addr state: [PASS | FAIL | INV | NDAT | NAV]

ivvel addr: real

MB state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb gicb bds6,0 items.

All values are returned in decimal except for the MB field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:GICB:BD60?`

:ADSB

:GICB

:BD60

:STARt

Parameters: None

Description: This starts the adsb gicb bds6,0 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD60:STAR`

Start adsb gicb bds6,0 test.

:ADSB
:GICB
:BD61
:ST1
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <CRD>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>

overall state, type state, type, addr state, addr, MB state, MB, empricode state, empricode, MODEacode, res state, res, subtype state, subtype

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

MB: string. Maximum length of 14 characters excluding quotes.

MODEacode:

empricode state: [PASS | FAIL | INV | NDAT | NAV]

empricode: [NEM | GEM | LMEM | MFU | NCOM | UINT | DAIR | RES]

res state: [PASS | FAIL | INV | NDAT | NAV]

res: string. Maximum length of 12 characters excluding quotes.

subtype state: [PASS | FAIL | INV | NDAT | NAV]

subtype: integer

Description: Read back all the measured adsb gicb bds 6,1 ST1 items.

All values are returned in decimal except for the MB field which is returned as a 14 hexadecimal digits (56-bit field) string, and the res field which is returned as a 12 hexadecimal digits string.

Example: `ADSB:GICB:BD61:ST1:[DATA]?`

:ADSB
:GICB
:BD61
:ST2
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <CRD>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>

ARA, overall state, type state, type, addr state, addr, MB state, MB, MTE, RAC, RAT, subtype state, subtype, TID, TIDA, TIDB, TIDR, TTI

Returned values: ARA:

overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

MB: string. Maximum length of 14 characters excluding quotes.

MTE:

RAC:

RAT:

TID:

TIDA:

TIDB:

TIDR:

TTI:

subtype state: [PASS | FAIL | INV | NDAT | NAV]

subtype: integer

:ADSB

:GICB

:BD61

:ST2

[:DATA]? (cont)

Read back all the measured adsb gicb bd s 6,1 ST2 items.

Description: All values are returned in decimal except for the MB field which is returned as a 14 hexadecimal digits (56-bit field) string, and the res field which is returned as a 12 hexadecimal digits string.

Example: `ADSB:GICB:BD61:ST2:[DATA]?`

:ADSB
:GICB
:BD61
:ST1
:STARt

Parameters: None

Description: This starts the adsb gicb bds 6,1 ST1 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD61:ST1:STAR`

Start adsb gicb bds 6,1 ST1 test.

:ADSB
:GICB
:BD61
:ST2
:STARt

Parameters: None

Description: This starts the adsb gicb bds 6,1 ST2 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD61:ST2:STAR`

Start adsb gicb bds 6,1 ST2 test.

:ADSB

:GICB

:BD62

:ST0

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type, count state, count, addr state, addr, period state, period, vert data source state, vert data source, vert mode ind state, vert mode ind, tgt alt cap state, tgt alt cap, tgt alt type state, tgt alt type, tcasacas state, tcasacas, tgt alt state, tgt alt, sil state, sil, tgt hdg state, tgt hdg, nic baro state, nic baro, hor data av state, hor data av, hor mode ind state, hor mode ind, nac state, nac, empricode state, empricode, raa state, raa, ME state, ME

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

period state: [PASS | FAIL | INV | NDAT | NAV]

period: real

vert data src state: [PASS | FAIL | INV | NDAT | NAV]

vert data src: [NAV | MFCU | HALT | FRN]

vert mode ind state: [PASS | FAIL | INV | NDAT | NAV]

vert mode ind: [UNKN | ACQ | CAPT | RES]

tgt alt cap state: [PASS | FAIL | INV | NDAT | NAV]

tgt alt cap: [HALT | HAAC | HAAF | RES]

tgt alt type state: [PASS | FAIL | INV | NDAT | NAV]

[:DATA]? (cont)

Returned values:

tgt alt type: [FL | MSL]

tcasacas state: [PASS | FAIL | INV | NDAT | NAV]

tcasacas: integer

tgt alt state: [PASS | FAIL | INV | NDAT | NAV]

tgt alt: integer

sil state: [PASS | FAIL | INV | NDAT | NAV]

sil: integer

tgt hdg state: [PASS | FAIL | INV | NDAT | NAV]

tgt hdg: integer

nic baro state: [PASS | FAIL | INV | NDAT | NAV]

nic baro: integer

hor data av state: [PASS | FAIL | INV | NDAT | NAV]

hor data av: [NVAL | MFCU | MAIN | FRN]

hor mode ind state: [PASS | FAIL | INV | NDAT | NAV]

hor mode ind: [NAV | ACQ | MAIN | RES]

nac state: [PASS | FAIL | INV | NDAT | NAV]

nac: integer

empricode state: [PASS | FAIL | INV | NDAT | NAV]

empricode: [NEM | GEM | LMEM | MFU | NCOM | UINT | DAIR | RES]

raa state: [PASS | FAIL | INV | NDAT | NAV]

raa: integer

ME state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB
:GICB
:BD62
:ST0
[:DATA]? (cont)

Description: Read back all the measured adsb monitor bds 6,2 ST0 items.

All values are returned in decimal except for those fields that are character response data, and the ME field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:MON:BD62:ST0:ST0:DATA?`

:ADSB
:GICB
:BD62
:ST1
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type, addr state, addr, subtype state, subtype, sil state, sil, sil sup state, sil sup, nac state, nac, lnav state, lnav, vnav state, vnav, mcp state, mcp, autop state, autop, ahme state, ahme, ap mode, tcas op state, tacs op, selected state, selected alt, selected alt type state, selected alt type, baro pressure state, baro pressure, nic baro state, nic baro, selected heading state, selected heading, reversed adsr fly state, reversed adsr fly, count state, count, period state, period

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

subtype state: [PASS | FAIL | INV | NDAT | NAV]

subtype: integer

sil state: [PASS | FAIL | INV | NDAT | NAV]

sil: integer

sil sup state: [PASS | FAIL | INV | NDAT | NAV]

sil sup: integer

nac state: [PASS | FAIL | INV | NDAT | NAV]

nac: integer

lnav state: [PASS | FAIL | INV | NDAT | NAV]

lnav: integer

Returned values: vnav state: [PASS | FAIL | INV | NDAT | NAV]

vnav: integer

mcp state: [PASS | FAIL | INV | NDAT | NAV]

mcp: integer

autop state: [PASS | FAIL | INV | NDAT | NAV]

autop: integer

ahme state: [PASS | FAIL | INV | NDAT | NAV]

ahme: integer

ap mode:

tcas op state: [PASS | FAIL | INV | NDAT | NAV]

tcas op: integer

selected state: [PASS | FAIL | INV | NDAT | NAV]

selected alt: integer

selected alt type state: [PASS | FAIL | INV | NDAT | NAV]

selected alt type: integer

baro pressure state: [PASS | FAIL | INV | NDAT | NAV]

baro pressure: integer

nic baro state: [PASS | FAIL | INV | NDAT | NAV]

nic baro: integer

selected heading state: [PASS | FAIL | INV | NDAT | NAV]

selected heading: integer

reversed adsr fly state: [PASS | FAIL | INV | NDAT | NAV]

reversed adsr fly: integer

:ADSB
:GICB
:BD62
:ST1
[:DATA]? (cont)

Description: Read back all the measured adsb monitor bds 6,2 ST1 items.

All values are returned in decimal except for those fields that are character response data, and the ME field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:MON:BD62:ST1:DATA?`

:ADSB

:GICB

:BD62

:STARt

Parameters: None

Description: This starts the adsb gicb bds6,2 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD62:STAR`

Start adsb gicb bds6,2 test.

:ADSB
:GICB
:BD65
:AIR
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type, addr state, addr, subtype state, subtype, version number state, version number, capabilities class state, capabilities class, arv state, arv, ts state, ts, cdti state, cdti,uat state, uat, tc state, tc, adsr state (capabilities class bit 20), adsr (capabilities class bit 20), not tcas state, not tcas, operational mode state, operational mode, ra state, ra, system design assurance state, system design assurance, single antenna flag state, single antenna flag, atc services state, atc services, ident state, ident, horizontal reference direction state, horizontal reference direction, nic-a state, nic-a, baq state, baq, nic baro state, nic baro, sil supplement state, sil supplement, sil state, sil, nacp state, nacp, adsr state (bit 56), adsr (bit 56).

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

nacp state: [PASS | FAIL | INV | NDAT | NAV]

nacp: integer

baq state: [PASS | FAIL | INV | NDAT | NAV]

baq: integer

sil state: [PASS | FAIL | INV | NDAT | NAV]

sil: integer

sil sup: integer

cdti state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB

:GICB

:BD65

:AIR

[:DATA]? (cont)

Returned values:

cdti: integer

arv state: [PASS | FAIL | INV | NDAT | NAV]

arv: integer

ts state: [PASS | FAIL | INV | NDAT | NAV]

ts: integer

tc state: [PASS | FAIL | INV | NDAT | NAV]

tc: integer

ra state: [PASS | FAIL | INV | NDAT | NAV]

ra: integer

op modes sub state: [PASS | FAIL | INV | NDAT | NAV]

op modes sub: integer

ver num state: [PASS | FAIL | INV | NDAT | NAV]

ver num: string. Maximum length of 15 characters excluding quotes.

not tcas state: [PASS | FAIL | INV | NDAT | NAV]

not tcas: integer

cap class state: [PASS | FAIL | INV | NDAT | NAV]

cap class: integer

nic baro state: [PASS | FAIL | INV | NDAT | NAV]

nic baro: integer

hor res dir state: [PASS | FAIL | INV | NDAT | NAV]

hor res dir: [TNOR | MNOR]

ident state: [PASS | FAIL | INV | NDAT | NAV]

ident: integer

:ADSB

:GICB

:BD65

:AIR

[:DATA]? (cont)

Returned values: recatc serv state: [PASS | FAIL | INV | NDAT | NAV]

recatc serv: integer

nic-a state: [PASS | FAIL | INV | NDAT | NAV]

nic-a: integer

subtype state: [PASS | FAIL | INV | NDAT | NAV]

subtype: [AIRB | SURF | RES]

surtype: integer

nic-c: integer

nacv: integer

ant off: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

UAT: integer

ADSR: integer

SDA: integer

SAF: integer

:ADSB

:GICB

:BD65

:AIR

[:DATA]? (cont)

Description: Read back all the measured adsb gicb bds 6,5 items.

All values are returned in decimal except for those fields that are character response data, the ver number field which is returned as a 15 character string, and the MB field which is returned as 14 hexadecimal digits (56-bit field).

Valid ver num fields are:
"DO-260 "
"DO-260A "
"DO-260B"
"RESERVED"

Example: `ADSB:GICB:BD65?`

:ADSB

:GICB

:BD65

:SUR

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type, addr state, addr, subtype state, subtype, version number state, version number, capabilities class state, capabilities class, b2low state, b2low, cdti state, cdti, uat state, uat, nic-c state, nic-c, nacv state, nacv, operational mode state, operational mode, ra state, ra, system design assurance state, system design assurance, single antenna flag state, single antenna flag, atc services state, atc services, ident state, ident, antenna offset state, antenna offset, horizontal reference direction state, horizontal reference, nic-a state, nic-a, poa state, poa, trk hdg state, trk hdg, sil supplement state, sil supplement, sil state, sil, nacp state, nacp, adsr state (bit 56), adsr (bit 56), len width state, len width.

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

nacp state: [PASS | FAIL | INV | NDAT | NAV]

nacp: integer

baq state: [PASS | FAIL | INV | NDAT | NAV]

baq: integer

sil state: [PASS | FAIL | INV | NDAT | NAV]

sil: integer

cdti state: [PASS | FAIL | INV | NDAT | NAV]

cdti: integer

:ADSB

:GICB

:BD65

:SUR

[:DATA]? (cont)

Returned values:

ra state: [PASS | FAIL | INV | NDAT | NAV]

ra: integer

op modes sub state: [PASS | FAIL | INV | NDAT | NAV]

op modes sub: integer

ver num state: [PASS | FAIL | INV | NDAT | NAV]

ver num: string. Maximum length of 15 characters excluding quotes.

cap class state: [PASS | FAIL | INV | NDAT | NAV]

cap class: integer

hor res dir state: [PASS | FAIL | INV | NDAT | NAV]

hor res dir: [TNOR | MNOR]

ident state: [PASS | FAIL | INV | NDAT | NAV]

ident: integer

recatc serv state: [PASS | FAIL | INV | NDAT | NAV]

recatc serv: integer

nia state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB

:GICB

:BD65

:SUR

[:DATA]? (cont)

Returned values: nia: integer

subtype state: [PASS | FAIL | INV | NDAT | NAV]

subtype: [AIRB | SURF | RES]

surtype: integer

nic-c: integer

nacv: integer

ant off: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb gicb bds 6,5 items.

All values are returned in decimal except for those fields that are character response data, the ver number field which is returned as a 15 character string, and the MB field which is returned as 14 hexadecimal digits (56-bit field).

Valid ver num fields are:
"DO-260 "
"DO-260A"
"DO-242B"
"RESERVED"

Example: `ADSB:GICB:BD65?`

:ADSB
:GICB
:BD63
:STARt

Parameters: None

Description: This starts the adsb gicb bds6,3 test. The test will run continuously until stopped.

Example: `ADSB:GICB:BD63:STAR`

Start adsb gicb bds6,3 test.

:ADSB
:GICB
:STOP

Parameters: None

Description: This stops the currently running adsb gicb test.

Example: `ADSB:GICB:STOP`

Stop adsb gicb test.

:ADSB
:MONitor
:ALL

Parameters: None

Description: This starts running all the adsb monitor tests. The tests will run continuously until stopped.

Results can be read back using the individual data query commands for each bds test.

Example: `ADSB:MON:ALL`

Start all the adsb monitor tests.

:ADSB

:MONitor

:BD05

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR1>,<NR1>, <NR1>, <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type, count state, count, addr state, addr, period state, period, latitude state, latitude degrees, latitude minutes, latitude seconds, latitude direction, longitude state, longitude degrees, longitude minutes, longitude seconds, longitude direction, saf state, saf, t state, t, surv state, surv, baro pressure alt state, baro pressure alt, gnss alt state, gnss alt, NIC, Rc, ME state, ME

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

period state: [PASS | FAIL | INV | NDAT | NAV]

period: real

latitude state: [PASS | FAIL | INV | NDAT | NAV]

latitude degrees: integer. Values are in the range 0 to 90.

latitude minutes: integer. Values are in the range 0 to 59.

latitude seconds: integer. Values are in the range 0 to 59.

latitude direction: [NORT | SOUT]

longitude state: [PASS | FAIL | INV | NDAT | NAV]

longitude degrees: integer. Values are in the range 0 to 180.

longitude minutes: integer. Values are in the range 0 to 59.

:ADSB

:MONitor

:BD05

[:DATA]? (cont)

Returned values: longitude seconds: integer. Values are in the range 0 to 59.

longitude direction: [EAST | WEST]

saf state: [PASS | FAIL | INV | NDAT | NAV]

saf: integer

t state: [PASS | FAIL | INV | NDAT | NAV]

t: integer

surv state: [PASS | FAIL | INV | NDAT | NAV]

surv: integer

baro pressure alt state: [PASS | FAIL | INV | NDAT | NAV]

baro pressure alt: integer

gnss alt state: [PASS | FAIL | INV | NDAT | NAV]

gnss alt: integer

NIC, Rc: Refer to tables in Operation Manual

ME state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb monitor bds0,5 items.

All values are returned in decimal except for the ME field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:MON:BD05?`

:ADSB

:MONitor

:BD05

:STARt

Parameters: None

Description: This starts the adsb monitor bds0,5 test. The test will run continuously until stopped.

Example: `ADSB:MON:BD05:STAR`

Start adsb monitor bds0,5 test.

:ADSB

:MONitor

:BD06

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR1>,<NR1>, <NR1>, <CRD>, <CRD>, <NR1>, <NR1>, <NR1>, <CRD>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type, count state, count, addr state, addr, period state, period, latitude state, latitude degrees, latitude minutes, latitude seconds, latitude direction, longitude state, longitude degrees, longitude minutes, longitude seconds, longitude direction, movement state, movement, t state, t, hdg state, hdg, NIC, Rc, ME state, ME

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

period state: [PASS | FAIL | INV | NDAT | NAV]

period: real

latitude state: [PASS | FAIL | INV | NDAT | NAV]

latitude degrees: integer. Values are in the range 0 to 90.

latitude minutes: integer. Values are in the range 0 to 59.

latitude seconds: integer. Values are in the range 0 to 59.

latitude direction: [NORT | SOUT]

longitude state: [PASS | FAIL | INV | NDAT | NAV]

longitude degrees: integer. Values are in the range 0 to 180.

longitude minutes: integer. Values are in the range 0 to 59.

:ADSB
:MONitor
:BD06
[:DATA]? (cont)

Returned values: longitude seconds: integer. Values are in the range 0 to 59.

longitude direction: [EAST | WEST]

movement state: [PASS | FAIL | INV | NDAT | NAV]

movement: string. Maximum length of 17 characters excluding quotes.

t state: [PASS | FAIL | INV | NDAT | NAV]

t: integer

hdg state: [PASS | FAIL | INV | NDAT | NAV]

hdg: integer

NIC, Rc: Refer to tables in Operation Manual

ME state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb monitor bds0,6 items.

All values are returned in decimal except for the ME field which is returned as 14 hexadecimal digits (56-bit field) and the movement field which is returned as a string.

Valid movement fields are:

"NO INFO"
"STOPPED"
"0.125 kt to <1 kt"
"1 kt to <2 kt"
"2 kt to <15 kt"
"15 kt to <70 kt"
"70 kt to <100 kt"
"100 kt to <175 kt"
"175 kt"
"DECELERATING"
"ACCELERATING"
"BACKING UP"

Example: `ADSB:MON:BD06?`

:ADSB

:MONitor

:BD06

:STARt

Parameters: None

Description: This starts the adsb monitor bds0,6 test. The test will run continuously until stopped.

Example: `ADSB:MON:BD06:STAR`

Start adsb monitor bds0,6 test.

:ADSB

:MONitor

:BD08

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR2>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>

overall state, type state, type, count state, count, addr state, addr, ME state, ME, period state, period, AIS state, AIS, flight ID state, flight ID, emitcatset state, emitcatset, emitcat state, emitcat

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

ME state: [PASS | FAIL | INV | NDAT | NAV]

ME: string. Maximum length of 14 characters excluding quotes.

period state: [PASS | FAIL | INV | NDAT | NAV]

period: real

AIS state: [PASS | FAIL | INV | NDAT | NAV]

AIS: string. Maximum length of 12 characters excluding quotes.

flight ID state: [PASS | FAIL | INV | NDAT | NAV]

flight ID: string. Maximum length of 8 characters excluding quotes.

emitcatset state: [PASS | FAIL | INV | NDAT | NAV]

emitcatset: integer

[:DATA]? (cont)

Returned values: emitcat state: [PASS | FAIL | INV | NDAT | NAV]

emitcat: string. Maximum length of 25 characters excluding quotes.

Description: Read back all the measured adsb monitor bds0,8 items.

All values are returned in decimal except for the ME field which is returned as a 14 hexadecimal digits (56-bit field) string, the AIS field which is returned as a 12 hexadecimal digits string, and the flight ID and emitcat fields which are returned as strings.

Valid emitcat fields are:

"NO ADS-B EMITTER INFO"
"LIGHT"
"SMALL"
"LARGE"
"HIGH VORTEX"
"HEAVY"
"HIGH PERFORMANCE"
"ROTOCRAFT"
"GLIDER/SAILPLANE"
"LIGHTER-THAN-AIR"
"PARACHUTIST/SKYDIVER"
"ULTRALIGHT/HANG-GLIDER"
"UNMANNED AERIAL VEHICLE"
"SPACE VEHICLE"
"SURFACE EMERGENCY VEHICLE"
"SURFACE SERVICE VEHICLE"
"FIXED GND/TETHERED OBSTR"
"CLUSTER OBSTR"
"LINE OBSTR"
"RESERVED"

Example: `ADSB:MON:BD08?`

:ADSB
:MONitor
:BD08
:STARt

Parameters: None

Description: This starts the adsb monitor bds0,8 test. The test will run continuously until stopped.

Example: `ADSB:MON:BD08:STAR`

Start adsb monitor bds0,8 test.

:ADSB

:MONitor

:BD09

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR1>, <CRD>, <CRD>, <NR1>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type, count state, count, addr state, addr, period state, period, ew vel state, ew velocity, ew velocity direction, ns vel state, ns velocity, ns velocity direction, nac state, nac, subtype state, subtype, vrate state, vrate, hdg state, hdg, geodiffbaro state, geodiffbaro, source state, source, intent state, intent, airspeed state, airspeed, airspeedtype state, airspeedtype, ifrcapflag state, ifrcapflag, ME state, ME

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

period state: [PASS | FAIL | INV | NDAT | NAV]

period: real

ew vel state: [PASS | FAIL | INV | NDAT | NAV]

ew vel: integer

ew vel direction: [EAST | WEST]

ns vel state: [PASS | FAIL | INV | NDAT | NAV]

ns vel: integer

ns vel direction: [NORT | SOUT]

nac state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB

:MONitor

:BD09

[:DATA]? (cont)

Returned values:

nac: integer

subtype state: [PASS | FAIL | INV | NDAT | NAV]

subtype: integer

vrate state: [PASS | FAIL | INV | NDAT | NAV]

vrate: integer

hdg state: [PASS | FAIL | INV | NDAT | NAV]

hdg: real

geodiffbaro state: [PASS | FAIL | INV | NDAT | NAV]

geodiffbaro: integer

source state: [PASS | FAIL | INV | NDAT | NAV]

source: [GEO | BARO]

intent state: [PASS | FAIL | INV | NDAT | NAV]

intent: integer

airspeed state: [PASS | FAIL | INV | NDAT | NAV]

airspeed: integer

airspeed type state: [PASS | FAIL | INV | NDAT | NAV]

airspeed type: [IAS | TAS]

ifrcap flag state: [PASS | FAIL | INV | NDAT | NAV]

ifrcap flag: integer

ME state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB
:MONitor
:BD09
[:DATA]? (cont)

Description: Read back all the measured adsb monitor bds0,9 items.

All values are returned in decimal except for source and airspeed type fields which are character response data, and the ME field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:MON:BD09?`

:ADSB
:MONitor
:BD09
:STARt

Parameters: None

Description: This starts the adsb monitor bds0,9 test. The test will run continuously until stopped.

Example: `ADSB:MON:BD09:STAR`

Start adsb monitor bds0,9 test.

:ADSB

:MONitor

:BD61

:ST1

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR2>, <CRD>, <CRD>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>

overall state, type state, type, count state, count, addr state, addr, ME state, ME, MODEacode, period state, period, empricode state, empricode, res state, res, subtype state, subtype

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

ME state: [PASS | FAIL | INV | NDAT | NAV]

ME: string. Maximum length of 14 characters excluding quotes.

MODEacode:

period state: [PASS | FAIL | INV | NDAT | NAV]

period: real

empricode state: [PASS | FAIL | INV | NDAT | NAV]

empricode: [NEM | GEM | LMEM | MFU | NCOM | UINT | DAIR | RES]

res state: [PASS | FAIL | INV | NDAT | NAV]

res: string. Maximum length of 12 characters excluding quotes.

subtype state: [PASS | FAIL | INV | NDAT | NAV]

subtype: integer

:ADSB

:MONitor

:BD61

:ST1

[:DATA]? (cont)

Description: Read back all the measured adsb monitor bds 6,1 ST1 items.

All values are returned in decimal except for the ME field which is returned as a 14 hexadecimal digits (56-bit field) string, and the res field which is returned as a 12 hexadecimal digits string.

Example: `ADSB:MON:BD61:ST1:[DATA]?`

:ADSB
:MONitor
:BD61
:ST2
[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR2>, <CRD>, <CRD>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>

ARA, overall state, type state, type, count state, count, addr state, addr, ME state, ME, MTE, period state, period, empricode state, empricode, RAC, RAT, res state, res, subtype state, subtype, TID, TIDA, TIDB, TIDR, TTI

Returned values: ARA:
overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

ME state: [PASS | FAIL | INV | NDAT | NAV]

ME: string. Maximum length of 14 characters excluding quotes.

MTE:

period state: [PASS | FAIL | INV | NDAT | NAV]

period: real

RAC:

RAT:

empricode state: [PASS | FAIL | INV | NDAT | NAV]

subtype state: [PASS | FAIL | INV | NDAT | NAV]

subtype: integer

:ADSB
:MONitor
:BD61
:ST2
[:DATA]? (cont)

Returned values: TID:

TIDA:

TIDB:

TIDR:

TTI:

Description: Read back all the measured adsb monitor bds 6,1 ST2 items.

All values are returned in decimal except for the ME field which is returned as a 14 hexadecimal digits (56-bit field) string, and the res field which is returned as a 12 hexadecimal digits string.

Example: `ADSB:MON:BD61:ST2:[DATA]?`

:ADSB
:MONitor
:BD61
:ST1
:STARt

Parameters: None

Description: This starts the adsb monitor bds 6,1 ST1 test. The test will run continuously until stopped.

Example: `ADSB:MON:BD61:ST1:STAR`

Start adsb monitor bds 6,1 ST1 test.

:ADSB
:MONitor
:BD61
:ST2
:STARt

Parameters: None

Description: This starts the adsb monitor bds 6,1 ST2 test. The test will run continuously until stopped.

Example: `ADSB:MON:BD61:ST2:STAR`

Start adsb monitor bds 6,1 ST2 test.

:ADSB

:MONitor

:BD62

:ST0

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type, count state, count, addr state, addr, period state, period, vert data source state, vert data source, vert mode ind state, vert mode ind, tgt alt cap state, tgt alt cap, tgt alt type state, tgt alt type, tcasacas state, tcasacas, tgt alt state, tgt alt, sil state, sil, tgt hdg state, tgt hdg, nic baro state, nic baro, hor data av state, hor data av, hor mode ind state, hor mode ind, nac state, nac, empricode state, empricode, raa state, raa, ME state, ME

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

period state: [PASS | FAIL | INV | NDAT | NAV]

period: real

vert data src state: [PASS | FAIL | INV | NDAT | NAV]

vert data src: [NAV | MFCU | HALT | FRN]

vert mode ind state: [PASS | FAIL | INV | NDAT | NAV]

vert mode ind: [UNKN | ACQ | CAPT | RES]

tgt alt cap state: [PASS | FAIL | INV | NDAT | NAV]

tgt alt cap: [HALT | HAAC | HAAF | RES]

tgt alt type state: [PASS | FAIL | INV | NDAT | NAV]

:MONitor

:BD62

:ST0

[:DATA]? (cont)

Returned values:

tgt alt type: [FL | MSL]

tcasacas state: [PASS | FAIL | INV | NDAT | NAV]

tcasacas: integer

tgt alt state: [PASS | FAIL | INV | NDAT | NAV]

tgt alt: integer

sil state: [PASS | FAIL | INV | NDAT | NAV]

sil: integer

tgt hdg state: [PASS | FAIL | INV | NDAT | NAV]

tgt hdg: integer

nic baro state: [PASS | FAIL | INV | NDAT | NAV]

nic baro: integer

hor data av state: [PASS | FAIL | INV | NDAT | NAV]

hor data av: [NVAL | MFCU | MAIN | FRN]

hor mode ind state: [PASS | FAIL | INV | NDAT | NAV]

hor mode ind: [NAV | ACQ | MAIN | RES]

nac state: [PASS | FAIL | INV | NDAT | NAV]

nac: integer

empricode state: [PASS | FAIL | INV | NDAT | NAV]

empricode: [NEM | GEM | LMEM | MFU | NCOM | UINT | DAIR | RES]

raa state: [PASS | FAIL | INV | NDAT | NAV]

raa: integer

ME state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB

:MONitor

:BD62

:ST0

[:DATA]? (cont)

Description: Read back all the measured adsb monitor bds 6,2 ST0 items.

All values are returned in decimal except for those fields that are character response data, and the ME field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:MON:BD62:ST0:ST0:DATA?`

:ADSB

:MONitor

:BD62

:ST1

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type, addr state, addr, subtype state, subtype, sil state, sil, sil sup state, sil sup, nac state, nac, lnav state, lnav, vnav state, vnav, mcp state, mcp, autop state, autop, ahme state, ahme, ap mode, tcas op state, tacs op, selected state, selected alt, selected alt type state, selected alt type, baro pressure state, baro pressure, nic baro state, nic baro, selected heading state, selected heading, reversed adsr fly state, reversed adsr fly, count state, count, period state, period

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

subtype state: [PASS | FAIL | INV | NDAT | NAV]

subtype: integer

sil state: [PASS | FAIL | INV | NDAT | NAV]

sil: integer

sil sup state: [PASS | FAIL | INV | NDAT | NAV]

sil sup: integer

nac state: [PASS | FAIL | INV | NDAT | NAV]

nac: integer

lnav state: [PASS | FAIL | INV | NDAT | NAV]

lnav: integer

:ADSB

:MONitor

:BD62

:ST1

[:DATA]? (cont)

Returned values: vnav state: [PASS | FAIL | INV | NDAT | NAV]

vnav: integer

mcp state: [PASS | FAIL | INV | NDAT | NAV]

mcp: integer

autop state: [PASS | FAIL | INV | NDAT | NAV]

autop: integer

ahme state: [PASS | FAIL | INV | NDAT | NAV]

ahme: integer

ap mode:

tcas op state: [PASS | FAIL | INV | NDAT | NAV]

tcas op: integer

selected state: [PASS | FAIL | INV | NDAT | NAV]

selected alt: integer

selected alt type state: [PASS | FAIL | INV | NDAT | NAV]

selected alt type: integer

baro pressure state: [PASS | FAIL | INV | NDAT | NAV]

baro pressure: integer

nic baro state: [PASS | FAIL | INV | NDAT | NAV]

nic baro: integer

selected heading state: [PASS | FAIL | INV | NDAT | NAV]

selected heading: integer

reversed adsr fly state: [PASS | FAIL | INV | NDAT | NAV]

reversed adsr fly: integer

:ADSB
:MONitor
:BD62
:ST1
[:DATA]? (cont)

Returned values: count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

period state: [PASS | FAIL | INV | NDAT | NAV]

period: real

Description: Read back all the measured adsb monitor bds 6,2 ST1 items.

All values are returned in decimal except for those fields that are character response data, and the ME field which is returned as 14 hexadecimal digits (56-bit field).

Example: `ADSB:MON:BD62:ST1:DATA?`

:ADSB
:MONitor
:BD62
:STARt

Parameters: None

Description: This starts the adsb monitor bds6,2 test. The test will run continuously until stopped.

Example: `ADSB:MON:BD62:STAR`

Start adsb monitor bds 6,2 test.

:ADSB

:MONitor

:BD65

:AIR

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type,count state, count, period state, period, addr state, addr, subtype state, subtype, version number state, version number, capabilities class state, capabilities class, arv state, arv, ts state, ts, cdti state, cdti,uat state, uat, tc state, tc, adsr state (capabilities class bit 20), adsr (capabilities class bit 20), not tcas state, not tcas, operational mode state, operational mode, ra state, ra, system design assurance state, system design assurance, single antenna flag state, single antenna flag, atc services state, atc services, ident state, ident, horizontal reference direction state, horizontal reference direction, nic-a state, nic-a, baq state, baq, nic baro state, nic baro, sil supplement state, sil supplement, sil state, sil, nacp state, nacp, adsr state (bit 56), adsr (bit 56).

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

period state: [PASS | FAIL | INV | NDAT | NAV]

period: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

nacp state: [PASS | FAIL | INV | NDAT | NAV]

nacp: integer

baq state: [PASS | FAIL | INV | NDAT | NAV]

baq: integer

:ADSB

:MONitor

:BD65

:AIR

[:DATA]? (cont)

Returned values: sil state: [PASS | FAIL | INV | NDAT | NAV]

sil: integer

sil sup: integer

cdti state: [PASS | FAIL | INV | NDAT | NAV]cdti: integer

arv state: [PASS | FAIL | INV | NDAT | NAV]

arv: integer

ts state: [PASS | FAIL | INV | NDAT | NAV]

ts: integer

tc state: [PASS | FAIL | INV | NDAT | NAV]

tc: integer

ra state: [PASS | FAIL | INV | NDAT | NAV]

ra: integer

op modes sub state: [PASS | FAIL | INV | NDAT | NAV]

op modes sub: integer

ver num state: [PASS | FAIL | INV | NDAT | NAV]

ver num: string. Maximum length of 15 characters excluding quotes.

not tcas state: [PASS | FAIL | INV | NDAT | NAV]

not tcas: integer

cap class state: [PASS | FAIL | INV | NDAT | NAV]

cap class: integer

nic baro state: [PASS | FAIL | INV | NDAT | NAV]

nic baro: integer

:MONitor

:BD65

:AIR

[:DATA]? (cont)

Returned values:

hor res dir state: [PASS | FAIL | INV | NDAT | NAV]

hor res dir: [TNOR | MNOR]

ident state: [PASS | FAIL | INV | NDAT | NAV]

ident: integer recatc serv state: [PASS | FAIL | INV | NDAT | NAV]

recatc serv: integer

nic-a state: [PASS | FAIL | INV | NDAT | NAV]

nic-a: integer

subtype state: [PASS | FAIL | INV | NDAT | NAV]

subtype: [AIRB | SURF | RES]

surtype: integer

nic-c: integer

nacv: integer

ant off: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

UAT: integer

ADSR: integer

SDA: integer

SAF: integer

:ADSB
:MONitor
:BD65
:AIR
[:DATA]? (cont)

Description: Read back all the measured adsb gicb bds 6,5 items.

All values are returned in decimal except for those fields that are character response data, the ver number field which is returned as a 15 character string, and the MB field which is returned as 14 hexadecimal digits (56-bit field).

Valid ver num fields are:

"DO-260 "
"DO-260A "
"DO-260B"
"RESERVED"

Example: `ADSB:MON:BD65?`

:ADSB

:MONitor

:BD65

:SUR

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <ARBITRARY ASCII RESPONSE DATA>

overall state, type state, type, count state, count, period state, period, addr state, addr, subtype state, subtype, version number state, version number, capabilities class state, capabilities class, b2low state, b2low, cdti state, cdti, uat state, uat, nic-c state, nic-c, nacv state, nacv, operational mode state, operational mode, ra state, ra, system design assurance state, system design assurance, single antenna flag state, single antenna flag, atc services state, atc services, ident state, ident, antenna offset state, antenna offset, horizontal reference direction state, horizontal reference, nic-a state, nic-a, poa state, poa, trk hdg state, trk hdg, sil supplement state, sil supplement, sil state, sil, nacp state, nacp, adsr state (bit 56), adsr (bit 56), len width state, len width.

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

type state: [PASS | FAIL | INV | NDAT | NAV]

type: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

period state: [PASS | FAIL | INV | NDAT | NAV]

period: integer

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

nacp state: [PASS | FAIL | INV | NDAT | NAV]

nacp: integer

baq state: [PASS | FAIL | INV | NDAT | NAV]

baq: integer

:ADSB

:MONitor

:BD65

:SUR

[:DATA]? (cont)

Returned values: sil state: [PASS | FAIL | INV | NDAT | NAV]

sil: integer

cdti state: [PASS | FAIL | INV | NDAT | NAV]

cdti: integer ra state: [PASS | FAIL | INV | NDAT | NAV]

ra: integer

op modes sub state: [PASS | FAIL | INV | NDAT | NAV]

op modes sub: integer

ver num state: [PASS | FAIL | INV | NDAT | NAV]

ver num: string. Maximum length of 15 characters excluding quotes.

cap class state: [PASS | FAIL | INV | NDAT | NAV]

cap class: integer

hor res dir state: [PASS | FAIL | INV | NDAT | NAV]

hor res dir: [TNOR | MNOR]

ident state: [PASS | FAIL | INV | NDAT | NAV]

ident: integer

recatc serv state: [PASS | FAIL | INV | NDAT | NAV]

recatc serv: integer

nia state: [PASS | FAIL | INV | NDAT | NAV]

:ADSB
:MONitor
:BD65
:SUR
[:DATA]? (cont)

Returned values: nia: integer

subtype state: [PASS | FAIL | INV | NDAT | NAV]

subtype: [AIRB | SURF | RES]

surtype: integer

nic-c: integer

nacv: integer

ant off: integer

MB state: [PASS | FAIL | INV | NDAT | NAV]

Description: Read back all the measured adsb gicb bds 6,5 items.

All values are returned in decimal except for those fields that are character response data, the ver number field which is returned as a 15 character string, and the MB field which is returned as 14 hexadecimal digits (56-bit field).

Valid ver num fields are:
"DO-260 "
"DO-260A"
"DO-242B"
"RESERVED"

Example: `ADSB:MON:BD65?`

:ADSB

:MONitor

:BD63

:STARt

Parameters: None

Description: This starts the adsb monitor bds6,3 test. The test will run continuously until stopped.

Example: `ADSB:MON:BD63:STAR`

Start adsb monitor bds6,3 test.

:ADSB

:MONitor

:STOP

Parameters: None

Description: This stops the currently running adsb monitor test.

Example: `ADSB:MON:STOP`

Stop adsb monitor test.

:ADSB
:SETup
:GENerate

Parameters: <CPD>

df

Valid values: df: [DF17 | DF18]. Values other than those stated are rejected and an error generated.

Description: Select whether to use DF17 or DF18 for adsb generate testing.

Example: `ADSB:SET:GEN DF18`

Select DF18 for generate tests.

:ADSB
:SETup
:GENerate?

Parameters: none

Response: <CRD>

df

Returned values: df: [DF17 | DF18]

Description: Determine if using DF17 or DF18 for adsb generate testing.

Example: `ADSB:SET:GEN?`

:ADSB
:SETup
:GICB

Parameters: <CPD>

df

Valid values: df: [DF20 | DF21]. Values other than those stated are rejected and an error generated.

Description: Select whether to use DF20 or DF21 for adsb gicb testing.

Example: `ADSB:SET:GICB DF21`

Select DF21 for gicb tests.

:ADSB
:SETup
:GICB?

Parameters: none

Response: <CRD>

df

Returned values: df: [DF20 | DF21]

Description: Determine if using DF20's or DF21's for adsb gicb testing.

Example: `ADSB:SET:GICB?`

:ADSB
:SETup
:MONitor

Parameters: <CPD>

df

Valid values: df: [DF17 | DF18]. Values other than those stated are rejected and an error generated.

Description: Select whether to use DF17 or DF18 for adsb monitor testing.

Example: `ADSB:SET:MON DF18`

Select DF18 for monitor tests.

:ADSB
:SETup
:MONitor?

Parameters: none

Response: <CRD>

df

Returned values: df: [DF17 | DF18]

Description: Determine if using DF17's or DF18's for adsb gicb testing.

Example: `ADSB:SET:MON?`

UAT COMMANDS

The IFR 6000 provides flight line test capability for receiving (UAT MON mode), decoding and displaying full UAT DO-282B messages from 978 MHz transceivers.

Capability to generate (UAT GEN mode) full DO-282B message transmissions for testing UAT transceivers is provided. The supported messages are FIS-B, TIS-B, and ADS-B.

UAT SUBSYSTEM

- UAT
    - FISB
        - GENerate
            - DATa\?
            - DAYTime\?
            - REPort\?
            - SLOTid\?
            - STARt
            - STATion\?
        - STOP
    - TISB
        - GENerate
            - ALT\?
            - DATA?
            - HDG\?
            - LAT\?
            - LON\?
            - SIteid\?
            - STARt
            - TARGets\?
        - STOP

- UAT
    - ADSB
        - GENerate
            - ALT\?
            - DATA\?
            - HDG\?
            - LAT\?
            - LON\?
            - STARt
            - TARGets\?
        - MONitor
            - PTC0
            - PTC1
            - PTC2
            - STARt
            - STOP
        - STOP
    - GPS
        - STATus?
        - DATA?
        - STOP
        - STARt

:UAT

:FISB

:GENerate

:DATa?

Parameters: <ASCII DATA>

FIS-B encoded data

Valid values: Any METAR or TAF encoded data

Description: Represents the encoded METAR or TAF data selected to be generated. Note that the selection of the data is determined by the Report and Station selected.

Example: `17018G23KT 10SM FEW220 25/07 A2998 RMK A02 SLP147 T02500067,`

`37 38 59.80 N, 97 25 59.00 W`

:UAT

:FISB

:GENerate

:DAYTime

Parameters: <NRf>, <NRf>, <NRf>

day, hour, minute

Valid values: day: integer. Valid values are 1 to 31. Values outside range are rejected and an error generated.

hour: integer. Valid values are 0 to 23. Values outside range are rejected and an error generated.

minute: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

Description: Set the Zulu time to be encoded for transmission.

Example: `UAT:FISB:GEN:DAYT 22, 07, 54`

Set day to 22nd of the month, 0754 Zulu time

UAT

:FISB

:GENerate

:DAYTime?

Parameters: None

Response: <NR1>, <NR1>, <NR1>

day, hour, minute

Returned values: day: integer. Values are in the range 1 to 31.

hour: integer. Values are in the range 0 to 23.

minute: integer. Values are in the range 0 to 59.

Description: Retrieves the day of the month and Zulu time.

Example: `UAT:FISB:GEN:DAYT?`

:UAT

:FISB

:GENerate

:REPort

Parameters: <NRf>

report type

Valid values: report type: [0 | 1]. 0=METAR, 1=TAF. Values other than those stated are rejected and an error generated.

Description: Select report type to be encoded for transmission.

Example: `UAT:FISB:GEN:REP 1`

Select report type to TAF.

UAT

:FISB

:GENerate

:REPort?

Parameters: None

Response: <CRD>

report type

Returned values: report type: [METAR | TAF]

Description: Retrieves the currently selected FIS-B report type.

Example: `UAT:FISB:GEN:REP?`

:UAT

:FISB

:GENerate

:SLOTid

Parameters: <NRf>

slot ID

Valid values: slot id: integer. Valid values are 0 to 31. Values outside range are rejected and an error generated.

Description: Set the slot ID of the FIS-B message.

Example: `UAT:FISB:GEN:SLOT 7`

Select the slot ID to 7.

UAT

:FISB

:GENerate

:SLOTid?

Parameters: None

Response: <NR1>

slot ID

Returned values: slot ID: integer. Values are in the range 0 to 31.

Description: Retrieves the slot ID of the FIS-B message.

Example: `UAT:FISB:GEN:SLOT?`

:UAT

: FISB

:GENerate

:STARt

Parameters: None

Description: This runs all the FIS-B generate test. Use UAT:FISB:STOP to stop the testing.

Example: `UAT:FISB:GEN:STAR`

Run the FIS-B generate tests.

:UAT

:FISB

:GENerate

:STATion

Parameters: <NRf>

station
0 = KMCI
1 = KAUS
2 = KFFC
3 = KBNA
4 = PANC

Valid values: station: integer. Valid values are 0 to 4. Values outside range are rejected and an error generated.

Description: Selects the station (airport) for the FIS-B message. Note that this also determines the weather data and position that will be encoded for transmission.

Example: `UAT:FISB:GEN:STAT 3`

Set station to KFFC for either report type METAR or TAF.

UAT

:FISB

:GENerate

:STATion?

Parameters: None

Response: <NR1>

station

Returned values: station: integer. Values are in the range 0 to 4.

Description: Retrieves the currently selected station (airport).

Example: `UAT:FISB:GEN:STAT?`

:UAT

:TISB

:GENerate

:ALT

Parameters: <NRf>

altitude

Valid values: altitude: integer. Valid values are -900 to 20000 in 25 ft increment. Values outside range are rejected and an error generated.

Description: Sets the UUT altitude in feet.

Example: `UAT:TISB:GEN:ALT -800`

Select the UUT altitude to -800 ft.

UAT

:TISB

:GENerate

:ALT?

Parameters: None

Response: <NR1>

altitude

Returned values: altitude: integer. Values are in the range -900 to 20000.

Description: Retrieves the setting of the UUT altitude in feet

Example: `UAT:TISB:GEN:ALT?`

:UAT

:TISB

:GENerator

:DATA

Parameters: <NRf>,<NRf>,<NRf>,<NRf>,<CPD>,<NRf>
target, bearing, range, altitude, altitude rate, heading

Valid values: target: integer. Valid values are 1 to 5. This is the number of targets to be encoded for transmission. Values outside range are rejected and an error generated.

bearing: integer. Valid values are 0 to 359. This is the bearing of the target relative to the UUT in degrees. Values outside range are rejected and an error generated.

range: integer. Valid values are 0.0 to 40.0 in 0.1 nm increment. This is the range of the target relative to the UUT in nm. Values outside range are rejected and an error generated.

altitude: integer. Valid values are -35 to 35 (x 100) in 100 ft increment. This is the altitude of the target relative to the UUT in feet. Values outside range are rejected and an error generated.

altitude rate: integer. [CLIMb | DESCend | LEVel]. Values outside range are rejected and an error generated.

heading: integer. Valid values are 0 to 359. This is the bearing of the target relative to the UUT in degrees. Values outside range are rejected and an error generated.

Description: Set the target position and direction.

Example: `UAT:TISB:GEN:DATA 1, 85, 15, 6, DESC, 190`

Set target to 1, bearing to 85°, range to 15.0 nm, altitude to 600 ft, altitude rate to descend, and heading to 190°. All parameters are relative to the UUT.

:UAT

:TISB

:GENerate

:DATA?

Parameters: <NRf>

target

Response: <NR1>,<NR1>,<NR1>,<NR1>,<CRD>,<NR1>

target, bearing, range, altitude, altitude rate, heading

Returned values: target: integer. Values are in the rang 1 to 5.

bearing: integer. Values are in the range 0 to 359 degrees.

range: float. Values are in the range 0.0 to 40.0 nm.

altitude: integer. Values are in the range -35 to 35 (x 100 ft).

altitude rate: integer. [CLIMb | DESCend | LEVel]

heading: integer. Values are in the range 0 to 359°.

Description: Retrieves the target setup.

Example: `UAT:TISB:GEN:DATA? 4`

Retrieve the target setup of target 4.

:UAT

:TISB

:GENerate

:HDG

Parameters: <NRf>

uut heading

Valid values: uut heading: integer. Valid values are 0 to 359°. Values outside range are rejected and an error generated.

Description: Sets the UUT heading in degrees.

Example: `UAT:TISB:GEN:HDG 190`

Set the heading to 190°.

UAT

:TISB

:GENerate

:HDG?

Parameters: None

Response: <NR1>

uut heading

Description: Retrieves the UUT heading in degrees.

Example: `UAT:TISB:GEN:HDG?`

:UAT

:TISB

:GENerate

:LAT

Parameters: <NRf>, <NRf>, <NRf>, <CPD>

degrees, minutes, seconds, direction

Valid values: degrees: integer. Valid values are 0 to 90. Values outside range are rejected and an error generated.

minutes: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

seconds: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

direction: [NORTh | SOUTh]. Values other than those stated are rejected and an error generated.

Description: Set the UUT latitude.

Example: `UAT:TISB:GEN:LAT 75, 12, 55, nort`

Set latitude 75 degrees, 12 minutes and 55 seconds north.

:UAT

:TISB

:GENerate

:LAT?

Parameters: None

Response: <NR1>, <NR1>, <NR1>, <CRD>

degrees, minutes, seconds, direction

Returned values: degrees: integer. Values are in the range 0 to 90.

minutes: integer. Values are in the range 0 to 59.

seconds: integer. Values are in the range 0 to 59.

direction: [NORT | SOUT]

Description: Retrieves the UUT latitude.

Example: `UAT:TISB:GEN:LAT?`

:UAT

:TISB

:GENerate

:LONgitude

Parameters: <NRf>, <NRf>, <NRf>, <CPD>

degrees, minutes, seconds, direction

Valid values: degrees: integer. Valid values are 0 to 180. Values outside range are rejected and an error generated.

minutes: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

seconds: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

direction: [EAST | WEST]. Values other than those stated are rejected and an error generated.

Description: Sets the UUT longitude.

Example: `UAT:TISB:GEN:LON 135, 32, 5, west`

Set longitude 135 degrees, 32 minutes and 5 seconds west.

:UAT

:TISB

:GENerate

:LON?

Parameters: None

Response: <NR1>, <NR1>, <NR1>, <CRD>

degrees, minutes, seconds, direction

Returned values: degrees: integer. Values are in the range 0 to 180.

minutes: integer. Values are in the range 0 to 59.

seconds: integer. Values are in the range 0 to 59.

direction: [EAST | WEST]

Description: Retrieves the UUT longitude.

Example: `UAT:TISB:GEN:LON?`

:UAT

:TISB

:GENerate

:SIteid

Parameters: <NRf>

site id

Valid values: site id: integer. Valid values are 0 to 15. Values outside range are rejected and an error generated.

Description: Sets the site ID to be encoded to the TIB-B message.

Example: `UAT:TISB:GEN:SI 5`

Set the site ID to 5.

UAT

:TISB

:GENerate

:SIteid?

Parameters: None

Response: <NR1>

site id

Returned values: site id: integer. Values in the range 0 to 15.

Description: Retrieves the site ID.

Example: `UAT:TISB:GEN:SI?`

:UAT

:TISB

:GENerate

:STARt

Parameters: None

Description: This starts the TIS-B generate test. The test will run continuously until stopped. Use UAT:TISB:STOP to stop the testing.

Example: `UAT:TISB:GEN:STAR`

Start TIS-B generate test.

UAT

:TISB

:STOP

Parameters: None

Description: This stops the currently running UAT TIS-B generate test.

Example: `UAT:TISB:GEN:STOP`

Stop UAT TIS-B gen test.

:UAT

:TISB

:GENerate

:TARGets

Parameters: <NRf>

target

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

Description: Sets the number of targets to be created.

Example: `UAT:TISB:GEN:TARG 3`

Set the number of targets to 3.

UAT

:TISB

:GENerate

:TARGets?

Parameters: None

Response: <NR1>

target

Returned values: target: integer. Values in the range 1 to 5.

Description: Retrieves the number of targets to be created.

Example: `UAT:TISB:GEN:TARG?`

:UAT

:ADSB

:GENerate

:ALT

Parameters: <NRf>

altitude

Valid values: altitude: integer. Valid values are -900 to 20000 in 25 ft increment. Values outside range are rejected and an error generated.

Description: Sets the UUT altitude in feet.

Example: `UAT:ADSB:GEN:ALT -800`

Select the UUT altitude to -800 ft.

UAT

:ADSB

:GENerate

:ALT?

Parameters: None

Response: <NR1>

altitude

Returned values: altitude: integer. Values are in the range -900 to 20000.

Description: Retrieves the setting of the UUT altitude in feet

Example: `UAT:ADSB:GEN:ALT?`

:UAT

:ADSB

:GENerator

:DATA

Parameters: <NRf>,<NRf>,<NRf>,<NRf>,<CPD>,<NRf>
target, bearing, range, altitude, altitude rate, heading

Valid values: target: integer. Valid values are 1 to 5. This is the number of targets to be encoded for transmission. Values outside range are rejected and an error generated.

bearing: integer. Valid values are 0 to 359. This is the bearing of the target relative to the UUT in degrees. Values outside range are rejected and an error generated.

range: integer. Valid values are 0.0 to 40.0 in 0.1 nm increment. This is the range of the target relative to the UUT in nm. Values outside range are rejected and an error generated.

altitude: integer. Valid values are -35 to 35 (x 100) in 100 ft increment. This is the altitude of the target relative to the UUT in feet. Values outside range are rejected and an error generated.

altitude rate: integer. [CLIMb | DESCend | LEVel]. Values outside range are rejected and an error generated.

heading: integer. Valid values are 0 to 359. This is the bearing of the target relative to the UUT in degrees. Values outside range are rejected and an error generated.

Description: Set the target position and direction.

Example: `UAT:ADSB:GEN:DATA 1, 85, 15, 6, DESC, 190`

Set target to 1, bearing to 85°, range to 15.0 nm, altitude to 600 ft, altitude rate to descend, and heading to 190°. All parameters are relative to the UUT.

:UAT

:ADSB

:GENerate

:DATA?

Parameters: <NRf>

target

Response: <NR1>,<NR1>,<NR1>,<NR1>,<CRD>,<NR1>

target, bearing, range, altitude, altitude rate, heading

Returned values: target: integer. Values are in the range 1 to 5.

bearing: integer. Values are in the range 0 to 359 degrees.

range: float. Values are in the range 0.0 to 40.0 nm.

altitude: integer. Values are in the range -35 to 35 (x 100 ft).

altitude rate: integer. [CLIMb | DESCend | LEVel]

heading: integer. Values are in the range 0 to 359°.

Description: Retrieves the target setup.

Example: `UAT:ADSB:GEN:DATA? 4`

Retrieve the target setup of target 4.

:UAT

:ADSB

:GENerate

:HDG

Parameters: <NRf>

uut heading

Valid values: uut heading: integer. Valid values are 0 to 359°. Values outside range are rejected and an error generated.

Description: Sets the UUT heading in degrees.

Example: `UAT:ADSB:GEN:HDG 190`

Set the heading to 190°.

UAT

:ADSB

:GENerate

:HDG?

Parameters: None

Response: <NR1>

uut heading

Description: Retrieves the UUT heading in degrees.

Example: `UAT:ADSB:GEN:HDG?`

:UAT

:ADSB

:GENerate

:LAT

Parameters: <NRf>, <NRf>, <NRf>, <CPD>

degrees, minutes, seconds, direction

Valid values: degrees: integer. Valid values are 0 to 90. Values outside range are rejected and an error generated.

minutes: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

seconds: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

direction: [NORTh | SOUTh]. Values other than those stated are rejected and an error generated.

Description: Set the UUT latitude.

Example: `UAT:ADSB:GEN:LAT 75, 12, 55, nort`

Set latitude 75 degrees, 12 minutes and 55 seconds north.

UAT

:ADSB

:GENerate

:LAT?

Parameters: None

Response: <NR1>, <NR1>, <NR1>, <CRD>

degrees, minutes, seconds, direction

Returned values: degrees: integer. Values are in the range 0 to 90.

minutes: integer. Values are in the range 0 to 59.

seconds: integer. Values are in the range 0 to 59.

direction: [NORT | SOUT]

Description: Retrieves the UUT latitude.

Example: `UAT:ADSB:GEN:LAT?`

:UAT

:ADSB

:GENerate

:LONgitude

Parameters: <NRf>, <NRf>, <NRf>, <CPD>

degrees, minutes, seconds, direction

Valid values: degrees: integer. Valid values are 0 to 180. Values outside range are rejected and an error generated.

minutes: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

seconds: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

direction: [EAST | WEST]. Values other than those stated are rejected and an error generated.

Description: Sets the UUT longitude.

Example: `UAT:ADSB:GEN:LON 135, 32, 5, west`

Set longitude 135 degrees, 32 minutes and 5 seconds west.

:UAT

:ADSB

:GENerate

:LON?

Parameters: None

Response: <NR1>, <NR1>, <NR1>, <CRD>

degrees, minutes, seconds, direction

Returned values: degrees: integer. Values are in the range 0 to 180.

minutes: integer. Values are in the range 0 to 59.

seconds: integer. Values are in the range 0 to 59.

direction: [EAST | WEST]

Description: Retrieves the UUT longitude.

Example: `UAT:ADSB:GEN:LON?`

:UAT

:ADSB

:GENerate

:STARt

Parameters: None

Description: This starts the ADS-B generate test. The test will run continuously until stopped. Use UAT:ADSB:STOP to stop the testing.

Example: `UAT:ADSB:GEN:STAR`

Start ADS-B generate test.

UAT

:ADSB

:STOP

Parameters: None

Description: This stops the currently running UAT ADS-B generate test.

Example: `UAT:ADSB:GEN:STOP`

Stop UAT ADS-B gen test.

:UAT

:ADSB

:GENerate

:TARGets

Parameters: <NRf>

target

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

Description: Sets the number of targets to be created.

Example: `UAT:ADSB:GEN:TARG 3`

Set the number of targets to 3.

UAT

:ADSB

:GENerate

:TARGets?

Parameters: None

Response: <NR1>

target

Returned values: target: integer. Values in the range 1 to 5.

Description: Retrieves the number of targets to be created.

Example: `UAT:ADSB:GEN:TARG?`

:UAT

:ADSB

:MONitor

:PTC[0 | 1 | 2]?

Parameters: None

Response: <see list below>

state vector test status, auxiliary vector test status, mode status test status, target state test status, payload status, payload, address qualifier status, address qualifier, address status, address, latitude status, latitude direction, latitude degrees, latitude minutes, latitude seconds, longitude status, longitude direction, longitude degrees, longitude minutes, longitude seconds, altitude status, altitude, altitude type status, altitude type, nic status, nic, air ground state status, air ground state, north south velocity status, north south velocity, east west velocity status, east west velocity, vertical velocity status, vertical velocity, vertical velocity source status, vertical velocity source, ground speed status, ground speed, track angle heading type status, track angle heading type, track angle heading status, track angle heading, av length status, av length, av width status, av width, gps offset lateral status, gps offset lateral, gps offset longitudinal status, gps offset longitudinal, tisb site id status, tisb site id, utc status, utc, uplink feedback status, uplink feedback, emitter category status, emitter category, call sign status, call sign, emergency priority status, emergency priority, mops version status, mops version, sil status, sil, tx mso status, tx mso, sda status, sda, nacp status, nacp, nacv status, nacv, nic baro status, nic baro, cap uat in status, cap uat in, cap 1090es in status, cap 1090es in, cap tcas op status, cap tcas op, op tcas ra active status, op tcas ra active, op ident active status, op ident active, op rx atc services status, op rx atc services, csid status, csid, sil supp status, sil supp, geometric vertical accuracy status, geometric vertical accuracy, single antenna status, single antenna, nic supp status, nic supp, selected altitude type status, selected altitude type, selected heading state status, selected heading state, mcp fcu mode status, mcp fcu mode, auto pilot status, auto pilot, vnav mode status, vnav mode, altitude hold status, altitude hold, approach mode status, approach mode, lnav mode status, lnav mode, selected altitude status, selected altitude, barometric pressure status, barometric pressure, selected heading status, selected heading, auxiliary altitude status , auxiliary altitude, power status, power watts, power dbw, frequency status, frequency, flight id status, flight id

Returned values: state vector test status: string. [NRUN=Not Run | AVAI=Available].

auxiliary vector test status: string. [NRUN=Not Run | AVAI=Available].

mode status test status: string. [NRUN=Not Run | AVAI=Available].

target state test status: string. [NRUN=Not Run | AVAI=Available].

payload status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available].

payload: integer. Values in the range 0 to 6.

address qualifier status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available].

address qualifier: integer. [0=ADS-B ICAO | 1=ADS-B TEMP | 2=TIS-B/ADS-R | 3=TIS-B IDENT | 4=SURFACE | 5= ADS-B BEACON | 6=NON-ICAO | 7=RESERVED]

address status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available].

address: integer. Decimal representation of the aircraft address.

latitude status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

latitude direction: integer. [0=North | 1=South]. Applicable to State Vector.

latitude degrees: integer. Values in the range 0 to 90 in degrees. Applicable to State Vector.

latitude minutes: integer. Values in the range 0 to 59 in minutes. Applicable to State Vector.

latitude seconds: integer. Values in the range 0 to 5999 in seconds (x 100). Applicable to State Vector.

longitude status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

longitude direction: integer. [0=East | 1=West]. Applicable to State Vector.

longitude degrees: integer. Values in the range 0 to 180 in degrees. Applicable to State Vector.

longitude minutes: integer. Values in the range 0 to 59 in minutes. Applicable to State Vector.

longitude seconds: integer. Values in the range 0 to 5999 in seconds (x 100). Applicable to State Vector.

altitude status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

altitude: integer. Values in the range -1000 to 101325 in feet. Applicable to State Vector.

altitude type status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

altitude type: integer. [0=Barometric Pressure | 1=Geometric]. Applicable to State Vector.

nic status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

nic: integer. Values in the range 0 to 15. See Table 1. Applicable to State Vector.

Table 1

NIC	NIC Supplement Flag	Horizontal Containment
0		Rc Unknown
1		Rc < 20 nm
2		Rc < 8 nm
3		Rc < 4 nm
4		Rc < 2 nm
5		Rc < 1 nm
6	0	Rc < 0.6 nm
6	1	Rc < 0.3 nm
7		Rc < 0.2 nm
8		Rc < 0.1 nm
9		Rc < 0.0405 nm
10		Rc < 0.0135 nm
11		Rc < 0.004 nm
12		Reserved
13		Reserved
14		Reserved
15		Reserved

air ground state status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

air ground state: integer. [0=Subsonic | 1=Supersonic | 2=Ground | 3=Reserved]. Applicable to State Vector.

north south velocity status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

north south velocity: integer. For value range, see Table 2. North for values greater than zero; otherwise, the direction is South. Applicable to State Vector.

Table 2

Subsonic (kts)	Supersonic (kts)
N/S Velocity not available	N/S Velocity not available
0	0
1	4
2	8
…	…
1021	4084
>1021.5	>4086

east west velocity status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

east west velocity: integer. For value range, see Table 3. East for values greater than zero; otherwise, the direction is West. Applicable to State Vector.

Table 3

Subsonic (kts)	Supersonic (kts)
E/W Velocity not available	E/W Velocity not available
0	0
1	4
2	8
…	…
1021	4084
>1021.5	>4086

vertical velocity status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

vertical velocity: integer. For value range, see Table 4. Up for values greater than zero; otherwise, the direction is Down. Applicable to State Vector.

Table 4

Vertical Rate (feet/minute)
Vertical Rate information not available
0
64
128
…
32576
>32608

vertical velocity source status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

vertical velocity source:integer. [0=Geometric | 1=Barometric]. Applicable to State Vector.

ground speed status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

ground speed: integer. For value range, see Table 5. East for values greater than zero; otherwise, the direction is West. Applicable to State Vector.

Table 5

Ground Speed (kts)
Ground Speed information not available
0
1
2
…
1021
>1021.5

track angle heading type status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

track angle heading type: integer. [0=No Data | 1=True Track | 2=Magnetic | 3= True]. Applicable to State Vector.

track angle heading status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

track angle heading: integer. For value range, see Table 6. Applicable to State Vector.

Table 6

Track Angle/Heading (degrees / 100)
0
70
140
210
…
35859
35929

av length status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

av length: integer. For value range, see Table 7. Applicable to State Vector.

Table 7

A/V Length (meter)
15
25
35
45
55
65
75
85

av width status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

av width: integer. For value range, see Table 8. Applicable to State Vector.

Table 8

A/V Width (x 10 meter)
230
285
340
330
380
395
450
520
595
670
725
800
900

gps offset lateral status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

gps offset lateral: integer. For value range, see Table 9. Applicable to State Vector.

Table 9

GPS Lateral Offset(raw data)	Direction	GPS Lateral Offset (m)
0	Left	No Data
1	Left	2
2	Left	4
3	Left	6
4	Right	0
5	Right	2
6	Right	4
7	Right	6

gps offset longitudinal status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

gps offset longitudinal: integer. For value range, see Table 10. Applicable to State Vector.

Table 10

GPS Longitudinal Offset(raw data)	GPS Longitudinal Offset (m)
0	No Data
1	Position Offset Applied by Sensor
2	2
3	4
4	6
…	…
31	60

tisb site id status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

tisb site id: decimal.   Values in the range 0 to 15. Applicable to State Vector.

utc status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to State Vector.

utc: integer. [0=Non-UTC Coupled | 1=UTC Coupled]. Applicable to State Vector.

uplink feedback status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available].

uplink feedback: decimal.   Values in the range 0 to 7. Applicable to State Vector.

emitter category status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

emitter category: decimal. For value range, see Table 11. Applicable to Mode Status.

Table 11

Emitter Category	Meaning
0	No aircraft type information
1	Light (ICAO) < 15500 lbs
2	Small – 15500 to 75000 lbs
3	Large – 75000 to 300000 lbs
4	High Vortex Large
5	Heavy (ICAO) > 300000 lbs
6	Highly maneuverable > 5G
7	Rotorcraft
8	Unassigned
9	Glider/Sailplane
10	Lighter than air
11	Parachutist/Sky diver
12	Ultra light/Hang glider/Paraglider
13	Unassigned
14	Unmanned aerial vehicle
15	Space/Transatmospheric vehicle
16	Unassigned
17	Surface vehicle – Emergency
18	Surface vehicle – Service
19	Point obstacle (includes tethered balloons)
20	Cluster obstacle
21	Line obstacle
22	Reserved
…	…
39	Reserved

call sign status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

call sign: string. Value consists of eight characters. Applicable to Mode Status.

emergency priority status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available].

emergency priority: decimal. For value range, see Table 12. Applicable to Mode Status.

Table 12

Emergency Priority	Meaning
0	No emergency/Not reported
1	General emergency
2	Lifeguard/Medical emergency
3	Minimum fuel
4	No communications
5	Unlawful interference
6	Downed aircraft
7	Reserved

mops version status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

mops version: decimal. For value range, see Table 13. Applicable to Mode Status.

Table 13

UAT MOPS Version	Meaning
0	Reserved
1	Conforms to RTCA DO-282A
2	Conforms to RTCA DO-282B
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved

sil status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

sil: decimal. For value range, see Table 14. Applicable to Mode Status.

Table 14

SIL	Meaning (per flight hour or per sample)
0	Unknown
1	$<= 1x10^{-3}$
2	$<= 1x10^{-5}$
3	$<= 1x10^{-7}$

tx mso status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

tx mso: decimal. Values in the range 0 to 63. Applicable to State Vector.

sda status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

sda: decimal. For value range, see Table 15. Applicable to Mode Status.

Table 15

SDA	Failure Condition	Probability of undected fault (per flight hour or per sample)	Design Assurance Level
0	Unknown	$> 1x10^{-3}$	N/A
1	Minor	$<= 1x10^{-3}$	D
2	Major	$<= 1x10^{-5}$	C
3	Hazardous	$<= 1x10^{-7}$	B

nacp status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

nacp: decimal. For value range, see Table 16. Applicable to Mode Status.

Table 16

NACp	95% Horizontal Accuracy Bound (nm)
0	>= 10
1	< 10
2	< 4
3	< 2
4	< 1
5	< 0.5
6	< 0.3
7	< 0.1
8	< 0.05
9	< 0.0162
10	< 0.0054
11	< 0.0016
12	Reserved
13	Reserved
14	Reserved
15	Reserved

nacv status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

nacv: decimal. For value range, see Table 17. Applicable to Mode Status.

Table 17

NACv	Horizontal Velocity Error (m/s)
0	Unknown or >= 10
1	< 10
2	< 3
3	< 1
4	< 0.3
5	Reserved
6	Reserved
7	Reserved

nic baro status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

nic baro: integer. [0=Gilham not cross-checked | 1=Gilham cross-checked or non-Gilham]. Applicable to Mode Status.

cap uat in status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

cap uat in: integer. [0=No | 1=Yes]. Applicable to Mode Status.

cap 1090es in status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

cap 1090es in: integer. [0=No | 1=Yes]. Applicable to Mode Status.

cap tcas op status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

cap tcas op: integer. [0=No | 1=Yes]. Applicable to Mode Status.

op tcas ra active status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

op tcas ra active: integer. [0=No | 1=Yes]. Applicable to Mode Status.

op ident active status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

op ident active: integer. [0=No | 1=Yes]. Applicable to Mode Status.

op rx atc services status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

op rx atc services: integer. [0=Not receiving ATC | 1=Receiving ATC]. Applicable to Mode Status.

csid status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

csid: integer. [0=Flight ID | 1=Call sign]. Applicable to Mode Status.

sil supp status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

sil sup: integer. [0=per hour | 1=per sample]. Applicable to Mode Status.

geometric vertical accuracy status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

geometric vertical accuracy: decimal.   For value range, see Table 18. Applicable to Mode Status.

Table 18

GVA	Meaning (m)
0	Unknown or > 150
1	<= 150
2	<= 45
3	Reserved

single antenna status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

single antenna: integer. [0=Diversity | 1=Non-diversity]. Applicable to Mode Status.

nic supp status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

nic sup: decimal.   For value range, see Table 1. Applicable to Mode Status.

selected altitude type status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Target State.

selected altitude type: integer. [0=MCP/FCU | 1=FMS]. Applicable to Target State.

selected heading state status: string. [PASS=Valid Measurement | FAIL=Invalid

Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Target State.

selected heading state: integer. [0=Not Available | 1=Valid]. Applicable to Target State.

mcp fcu mode status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Target State.

mcp fcu mode: integer. [0=Invalid | 1=Valid]. Applicable to Target State.

auto pilot status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Target State.

auto pilot: integer. [0=Not engaged | 1=Engaged]. Applicable to Target State.

vnav mode status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Target State.

vnav mode: integer. [0=Not engaged | 1=Engaged]. Applicable to Target State.

altitude hold status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Target State.

altitude hold: integer. [0=Not engaged | 1=Engaged]. Applicable to Target State.

approach mode status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Target State.

approach mode: integer. [0=Not engaged | 1=Engaged]. Applicable to Target State.

lnav mode status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Target State.

lnav mode: integer. [0=Not engaged | 1=Engaged]. Applicable to Target State.

selected altitude status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Target State.

selected altitude: decimal. Values in the range 0 to 65472 in 32 ft increments. Applicable to Target State.

barometric pressure status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Target State.

barometric pressure: decimal. Values in the range 8000 to 12080 (x 10 mb). Applicable to Target State.

selected heading status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Target State.

selected heading: decimal. Values in the range -1793 to 1793 (x 10 degrees). Applicable to Target State.

auxiliary altitude status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Auxiliary State Vector.

auxiliary altitude: integer. Values in the range -1000 to 101325 in feet. Applicable to Auxiliary State Vector.

power status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available].

power watts: integer. Measured power (x 10 W).

power dbw: integer. Measured power (x 10 dBm).

frequency status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available].

frequency: integer. Measured frequency (x 100 MHz).

flight id status: string. [PASS=Valid Measurement | FAIL=Invalid Measurement | INV=No Measurement | NDAT=No Display | NAV=Not Available]. Applicable to Mode Status.

flight id: string. Value consists of eight characters. Applicable to Mode Status.

Description: Retrieves the first[0], second[1], or third[2] received payload.

Example: `UAT:ADSB:MON:PTC0?`

:UAT

:GPS

:STATus?

Parameters: None

Response: <NR1>

status

Returned values: status: integer. Values: 0 = No GPS Sync; 1 = GPS Sync.

Description: Retrieves the GPS synchronization status.

Example: `UAT:GPS:STAT?`

:UAT

:GPS

:DATA?

Parameters: None

Response: <CRD>,<ASCII DATA>,<CRD>,<ASCII DATA>,<CRD>,
<ASCII DATA>,<CRD>,<ASCII DATA>
number of satellites status, number of satellites, time status, time, latitude status, latitude, longitude status, longitude

Returned values: number of satellites status: [PASS | FAIL]

number of satellites: integer. Values in the range 0 to 32.

time status: [PASS | FAIL]

time: hh:mm:ss. hh = hour (0 to 23), mm = minute (0 to 59), ss = second (0 to 59)

latitude status: [PASS | FAIL]

latitude: dd:mm:ss.ss c. dd = degree (0 to 90), mm = minute (0 to 59), ss.ss = second (0 to 59.99), c = cardinal direction [N | S]

longitude status: [PASS | FAIL]

longitude: dd:mm:ss.ss c. dd = degree (0 to 180), mm = minute (0 to 59), ss.ss = second (0 to 59.99), c = cardinal direction [E | W]

Description: Retrieves the GPS data.

Example: `UAT:GPS:DATA?`

:UAT

:GPS

:STARt

Parameters: None

Description: This applies power to the GPS module so that GPS acquisition can start. The GPS data acquisition will run continuously until stopped. Use UAT:GPS:STOP to stop the GPS acquisition.

Example: `UAT:GPS:STAR`

Start GPS data acquisition.

UAT

:GPS

:STOP

Parameters: None

Description: This removes power from the GPS module to stop the GPS data acquisition.

Example: `UAT:GPS:STOP`

Stop GPS data acquisition.

THIS PAGE INTENTIONALLY LEFT BLANK.

DME COMMANDS

DME Mode provides flight line test capability for Distance Measuring Equipment Interrogators. All parameters normally required for DME testing are displayed on one main screen.

UUT interrogation parameters are clearly displayed in conjunction with Test Set reply parameters.

DME SUBSYSTEM

```
DME
    ANTenna
        GAIN\?
        RANGe\?
    CLOSs
        ANTenna
            MODE\?
            [VALue]\?
        [CURRent]\?
        DIRect\?
    DIAGnostic
        DATA?
        FREQuency
            CHANnel\?
            [RX]\?
            TX\?
        GENerate\?
        LEVel\?
        PRF\?
        RATTenuation\?
        [SELect]\?
        STARt
        STOP
    ECHO\?
    FREQuency
        CHANnel\?
        [RX]\?
        VOR\?
    IDENt
        [STATe]\?
        STRing\?
    LEVel
        [PORT]\?
        UUT
            LIMits?
            [VALue]\?
```

```
DME
    MEASure
        [DATA]?
        STARt
        STOP
    RANGe
        [CURRent]\?
        MAXimum\?
    RATE
        DIRection\?
        STATionary\?
        VALue\?
    REPLy\?
    SQUitter\?
```

:DME

:ANTenna

:GAIN

Parameters: <NRf>, <NRf>, <NRf>, <NRf>, <NRf>

gain960, gain1030, gain1090, gain1150, gain1220

Valid values: gain960: real. Valid values are 0.0 to 20.9. Values outside range are rejected and an error generated.
gain1030: real. Valid values are 0.0 to 20.9. Values outside range are rejected and an error generated.
gain1090: real. Valid values are 0.0 to 20.9. Values outside range are rejected and an error generated.
gain1150: real. Valid values are 0.0 to 20.9. Values outside range are rejected and an error generated.
gain1220: real. Valid values are 0.0 to 20.9. Values outside range are rejected and an error generated.

Description: Set the gain of the 6000 antenna at various frequencies in the DME band.

Example: `DME:ANT:GAIN 9.5, 9.7, 9.8, 10.0, 10.1`

Set gain of the 6000 antenna.

:DME

:ANTenna

:GAIN?

Parameters: None

Response: <NR2>, <NR2>, <NR2>, <NR2>, <NR2>

gain960, gain1030, gain1090, gain1150, gain1220

Returned values: gain962: real. Values are in the range 0.0 to 20.9.
gain1012: real. Values are in the range 0.0 to 20.9.
gain1062: real. Values are in the range 0.0 to 20.9.
gain1112: real. Values are in the range 0.0 to 20.9.
gain1162: real. Values are in the range 0.0 to 20.9.

Description: Determine the gain of the 6000 antenna at various frequencies in the DME band.

Example: `DME:ANT:GAIN?`

:DME

:ANTenna

:RANGe

Parameters: <NRf>

antenna range

Valid values: antenna range: real

Description: Set the range to the DME antenna on the aircraft. Rounded to nearest 0.5 meters or integral feet. In meters mode, range is 2.0 to 75.0 meters. In feet mode, range is 6 to 250 feet.

Example: `DME:ANT:RANG 5.5`

Set antenna range to 5.5 meters (assuming range entry mode is meters).

:DME

:ANTenna

:RANGe?

Parameters: None

Response: <NR2>

range

Returned values: range: real

Description: Determine the range to the DME antenna on the aircraft.

Example: `DME:ANT:RANG?`

:DME

:CLOSs

:ANTenna

:MODE

Parameters: <CPD>

ant cable loss mode

Valid values: ant cable loss mode: [UDEFined | L25 | L50 | L75]. Values other than those stated are rejected and an error generated.

Description: Select whether one of the VIAVI supplied cables is used to connect to the antenna (the 6000 has the cable loss programmed into it) or a user cable is used and its cable loss must be entered using DME:CLOS:ANT.

Example: `DME:CLOS:ANT:MODE L50`

The user is using a VIAVI supplied 50 ft cable and the 6000 automatically handles its cable loss..

:DME

:CLOSs

:ANTenna

:MODE?

Parameters: none

Response: <CRD>

ant cable loss mode

Returned values: ant cable loss mode: [UDEF | L25 | L50 | L75]

Description: Determine whether we are using a VIAVI supplied cable or a user defined cable to connect to the antenna.

Example: `DME:CLOS:ANT:MODE?`

:DME

:CLOSs

:ANTenna

[:VALue]

Parameters: <NRf>

cable loss

Valid values: cable loss: real

Description: This command sets the cable loss (in dB) of the cable used to connect to the test set antenna. This command will always set the cable loss of the antenna cable even if the current selection is direct connect. This value is only used if DME:CLOS:ANT:MODE is UDEF.

The DME cable loss value is kept separate from the transponder cable loss value.

The value is in units of dB.

Example: `DME:CLOS:ANT 1.7`

Inform the instrument of the loss of the cable connected to the 6000 antenna.

:DME

:CLOSs

:ANTenna

[:VALue]?

Parameters: None

Response: <NR2>

cable loss

Returned values: Cable loss: real

Description: Determine the loss of the antenna cable. This is the value used when DME:CLOS:ANT:MODE is set to UDEF.

Example: `DME:CLOS:ANT?`

:DME

:CLOSs

[:CURRent]

Parameters: <NRf>

cable loss

Valid values: Cable loss: real

Description: This command sets the cable loss (in dB) of the cable used to direct connect to the aircraft antenna or the cable used to connect to the test set antenna if performing over the air measurements.

The DME cable loss value is kept separate from the transponder cable loss value.

The value is in units of dB.

Example: `DME:CLOS 1.7`

Inform the instrument of the loss of the cable currently being used.

:DME

:CLOSs

[:CURRent]?

Parameters: None

Response: <NR2>

cable loss

Returned values: cable loss: real

Description: Determine the loss of the cable currently being used.

Example: `DME:CLOS?`

:DME

:CLOSs

:DIRect

Parameters: <NRf>

cable loss

Valid values: cable loss: real

Description: This command sets the cable loss (in dB) of the cable used to directly connect from the test set to the transponder under test. This command will always set the cable loss of the direct connect cable even if the current selection is antenna (over the air).

The DME cable loss value is kept separate from the transponder cable loss value.

The value is in units of dB.

Example: `DME:CLOS:DIR 1.7`

Inform the instrument of the loss of the direct connect cable.

:DME

:CLOSs

:DIRect?

Parameters: None

Response: <NR2>

cable loss

Returned values: cable loss: real

Description: Determine the loss of the direct connect cable.

Example: `DME:CLOS:DIR?`

:DME

:DIAGnostic

:DATA?

Parameters: none

Response: <CRD>, <NR1>

state, data

Returned values: state: [NRUN | PASS]
data: integer. Values are in the range 0 to 65535.

Description: Read the dsp measurement value – a raw reading of the power measured

This command is intended to be used in the DME diagnostics dsp mode.

Example: `DME:DIAG:DATA?`

Read the raw dsp value.

:DME

:DIAGnostic

:FREQuency

:CHANnel

Parameters: <CPD>

channel

Valid values: channel: [X | Y]. Values other than those stated are rejected and an error generated.

Description: Select the dme channel in use.

The value set is only used in DME diagnostics dsp or pulse modes.

In pulse mode, changing the value only affects the pulse spacing on replies sent by the 6000. It is possible to send “illegal” settings, eg a frequency of 962 MHz with a channel Y spacing.

In dsp mode the channel is necessary since channel X and Y use identical interrogation frequencies. We use the channel to set the receive hardware correctly.

Example: `DME:DIAG:FREQ:CHAN Y`

Select channel Y for DME diagnostics pulse and dsp modes.

:DME

:DIAGnostic

:FREQuency

:CHANnel?

Parameters: None

Response: <CRD>

channel

Returned values: channel: [X | Y]

Description: Determine the channel being used whilst in DME diagnostic mode.

Example: `DME:DIAG:FREQ:CHAN?`

:DME

:DIAGnostic

:FREQuency

[:RX]

Parameters: <NRf>

frequency

Valid values: Frequency: integer. Valid values are 962000000 to 1213000000. Values outside range are rejected and an error generated.

Description: Select the frequency to be used for DME replies – the value is in Hz.

Used for DME diagnostics cw and pulse modes. Sets the frequency that the 6000 transmits on.

Example: `DME:DIAG:FREQ 1104000000`

Set frequency to be 1104 MHz (channel 17Y) for diagnostic mode.

:DME

:DIAGnostic

:FREQuency

[:RX]?

Parameters: None

Response: <NR1>

frequency

Returned values: frequency: integer. Values are in the range 962000000 to 1213000000.

Description: Determine the frequency being used whilst in DME diagnostic mode – cw and pulse.

Example: `DME:DIAG:FREQ?`

:DME

:DIAGnostic

:FREQuency

:TX

Parameters: <NRf>

frequency

Valid values: Frequency: integer. Valid values are 1025000000 to 1150000000. Values outside range are rejected and an error generated.

Description: Select the frequency to be used for receiving interrogations– the value is in Hz.

Used in DME diagnostics dsp mode only, in conjunction with DME:DIAG:FREQ:CHAN.

Example: `DME:DIAG:FREQ:TX 1104000000;CHAN Y`

Set frequency to be 1104 MHz (channel 80Y) for diagnostic dsp mode.

:DME

:DIAGnostic

:FREQuency

:TX?

Parameters: none

Response: <NR1>

frequency

Returned values: Frequency: integer. Values are in the range 1025000000 to 1150000000.

Description: Determine the frequency being used whilst in DME diagnostic dsp mode.

Example: `DME:DIAG:FREQ:TX?`

:DME

:DIAGnostic

:GENerate

Parameters: <BOOLEAN PROGRAM DATA>

GEN IF during dsp measure state

Description: This sets whether the 6000 generates an output while performing dme diagnostic dsp measure. If set then a signal is output at the IF stage of the generate chain.

Example: `DME:DIAG:GEN ON`

Enable a signal at IF generate stage.

:DME

:DIAGnostic

:GENerate?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

GEN IF during dsp measure state

Description: Determine whether a signal is generated at the IF stage during dme diagnostics dsp measure.

Example: `DME:DIAG:GEN?`

:DME

:DIAGnostic

:LEVel

Parameters: <NRf>

RF level

Valid values: RF level: integer. Valid values are -2 to -115. Values outside range are rejected and an error generated.

Description: Set the RF level that DME replies are sent out at while in DME diagnostic CW and pulse modes. Units of dBm.Range depends on port – antenna or direct connect:
For direct connect can set –47 dBm to –115 dBm.
For antenna can set –2 dBm to –67 dBm.

Example: `DME:DIAG:LEV –76`

Set power level to be –76 dBm.

:DME

:DIAGnostic

:LEVel?

Parameters: None

Response: <NR1>

RF level

Returned values: RF level: integer. Values are in the range -2 to -115.

Description: Determine the RF power level of replies/CW signal in diagnostic mode.

Example: `DME:DIAG:LEV?`

:DME

:DIAGnostic

:PRF

Parameters: <NRf>

PRF

Valid values: PRF: integer. Valid values are 1 to 300. Values outside range are rejected and an error generated.

Description: Set the pulse repetition frequency (PRF) of replies to the dme equipment.

This is only used in the DME diagnostics pulse mode.

Example: `DME:DIAG:PRF 78`

Select the PRF for replies in diagnostics mode.

:DME

:DIAGnostic

:PRF?

Parameters: None

Response: <NR1>

PRF

Returned values: PRF: integer. Values are in the range 1 to 300.

Description: Determine the current PRF for diagnostics.

Example: `DME:DIAG:PRF?`

:DME

:DIAGnostic

:RATTenuation

Parameters: <NRf>

attenuation

Valid values: attenuation: integer. Valid values are 0 to 55. Values outside range are rejected and an error generated.

Description: Set the receiver attenuator to specified setting (units of dB). Since the attenuation is settable in discrete steps, the entered value will be rounded to the nearest valid value: 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55.

Example: `DME:DIAG:RATT 20`

Set the attenuator for the receiver.

:DME

:DIAGnostic

:RATTenuation?

Parameters: None

Response: <NR1>

attenuation

Returned values: attenuation: integer. Values are in the range 0 to 55.

Description: Determine the selected receiver attenuation.

Example: `DME:DIAG:RATT?`

:DME

:DIAGnostic

[:SELect]

Parameters: <CPD>

diagnostic mode

Valid values: Diagnostic mode: [CW | PULSe | DSP]. Values other than those stated are rejected and an error generated.

Description: Set the mode that DME diagnostics is operating in.
Output a CW signal or, output pulses, or read received level.

Example: `DME:DIAG CW`

Set DME diagnostics into CW mode.

:DME

:DIAGnostic

[:SELect]?

Parameters: none

Response: <CRD>

diagnostic mode

Returned values: diagnostic mode: [CW | PULS | DSP]

Description: Determine the DME diagnostics mode.

Example: `DME:DIAG?`

:DME

:DIAGnostic

:STARt

Parameters: None

Description: This starts the currently selected diagnostic test.

For the CW and pulse modes nothing is received. For dsp mode, use DME:DIAG:DATA? to read the value.

Example: `DME:DIAG:STAR`

Start diagnostics test.

:DME

:DIAGnostic

:STOP

Parameters: None

Description: This stops the current diagnostic test.

Example: `DME:DIAG:STOP`

Stop diagnostics test.

:DME

:ECHO

Parameters: <BOOLEAN PROGRAM DATA>

echo state

Description: This sets echo on or off. When on, an echo will be added to all replies.

Example: `DME:ECHO ON`

Turn echo on.

:DME

:ECHO?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

echo state

Description: Determine whether echo is on or off.

Example: `DME:ECHO?`

:DME

:FREQuency

:CHANnel

Parameters: <NRf>, <CPD>

chan no, chan sel

Valid values: chan no: integer. Valid values are 1 to 126. Values outside range are rejected and an error generated.

chan sel: [X | Y]. Values other than those stated are rejected and an error generated.

Description: Select the frequency to be used by channel.

Example: `DME:FREQ:CHAN 17, Y`

Select channel 17Y.

:DME

:FREQuency

:CHANnel?

Parameters: none

Response: <NR1>, <CRD>

chan no, chan sel

Returned values: chan no: integer. Values are in the range 1 to 126.

chan sel: [X | Y]

Description: Determine the channel selected.

Example: `DME:FREQ:CHAN?`

:DME

:FREQuency

[:RX]

Parameters: <NRf>

frequency

Valid values: Frequency: integer. Valid values are 962000000 to 1213000000. Values outside range are rejected and an error generated.

Description: Select the frequency to be used – the value is in Hz.

Example: `DME:FREQ 1104E6`

Set frequency to be 1104 MHz (channel 17Y).

:DME

:FREQuency

[:RX]?

Parameters: none

Response: <NR1>

frequency

Returned values: frequency: integer. Values are in the range 962000000 to 1213000000.

Description: Determine the frequency being used.

Example: `DME:FREQ?`

:DME

:FREQuency

:VOR

Parameters: <NRf>

VOR frequency

Valid values: VOR frequency: integer. Valid values are 108000000 to 117950000. Values outside range are rejected and an error generated.

Description: Select the frequency to be used by selecting the matching VOR frequency – the value is in Hz. The value is rounded to the nearest valid value.

Example:
```
DME:FREQ:VOR 108050000
or
DME:FREQ:VOR 108.05E6
```

Set frequency to be 1104 MHz (VOR frequency 108.05 MHz).

:DME

:FREQuency

:VOR?

Parameters: none

Response: <NR1>

VOR frequency

Returned values: VOR frequency: integer. Values are in the range 108000000 to 117950000.

Description: Determine the frequency being used – in terms of VOR frequency.

Example: `DME:FREQ:VOR?`

:DME

:IDENt

[:STATe]

Parameters: <BOOLEAN PROGRAM DATA>

ident state

Description: This sets ident on or off. When on, the ident characters will be output every 30 seconds.

Example: `DME:IDEN ON`

Turn ident on.

:DME

:IDENt

[:STATe]?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

ident state

Description: Determine whether ident is on or off.

Example: `DME:IDEN?`

:DME

:IDENt

:STRing

Parameters: <STRING PROGRAM DATA>

ident string

Valid values: ident string: string. Maximum length of 3 characters excluding quotes. Excess characters will be ignored.

Description: This sets the 3 characters to be output as an ident every 30 seconds if ident is on.

Example: `DME:IDEN:STR “IFR”`

Set ident characters.

:DME

:IDENt

:STRing?

Parameters: none

Response: <STRING RESPONSE DATA>

ident string

Returned values: ident string: string. Maximum length of 3 characters excluding quotes.

Description: Determine ident characters.

Example: `DME:IDEN:STR?`

:DME

:LEVel

[:PORT]

Parameters: <NRf>

RF level

Valid values: RF level: real. Valid values are -2 to -115. Values outside range are rejected and an error generated.

Description: Set the RF level that DME replies are sent out at. Units of dBm. Value is rounded to nearest 0.5 dBm.

This is the power level at the port of the instrument, not as seen at the aircraft under test.

The valid range depends on whether the antenna port or the direct connect port is selected.
For direct connect can set –47 dBm to –115 dBm.
For antenna can set –2 dBm to –67 dBm.

Example: `DME:LEV –76`

Set power level to be –76 dBm.

:DME

:LEVel

[:PORT]?

Parameters: none

Response: <NR2>

RF level

Returned values: RF level: real. Values are in the range -2 to -115.

Description: Determine the RF power level of replies. This is the power level at the port of the instrument, not as seen at the aircraft under test. Units of dBm.

Example: `DME:LEV?`

:DME

:LEVel

:UUT

:LIMits?

Parameters: None

Response: <NR2>, <NR2>

min limit, max limit

Returned values: min limit: real

max limit: real

Description: Read back the lower and upper limits of rf level at the uut.

Example: `DME:LEV:UUT:LIM?`

:DME

:LEVel

:UUT

[:VALue]

Parameters: <NRf>

RF level

Valid values: RF level: real

Description: Set the RF level that DME replies are sent out at. Units of dBm.

The value will be set to the nearest valid value.

This is the power level at the aircraft under test. The valid range will vary.

Example: `DME:LEV:UUT –76`

Set power level to be –76 dBm.

:DME

:LEVel

:UUT

[:VALue]?

Parameters: none

Response: <NR2>

RF level

Returned values: RF level: real

Description: Determine the RF power level of replies. This is the power level at the aircraft under test. Units of dBm.

Example: `DME:LEV:UUT?`

:DME

:MEASure

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>

overall state, freq state, freq, ERP state, ERP, prf state, prf, P1 width state, P1 width, P2 width state, P2 width, P1P2 spacing state, P1P2 spacing, channel indicator

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

freq state: [PASS | FAIL | INV | NDAT | NAV]

freq: integer. Values are in the range 0 to 1999990000.

ERP state: [PASS | FAIL | INV | NDAT | NAV]

ERP: real

prf state: [PASS | FAIL | INV | NDAT | NAV]

prf: integer

P1 width state state: [PASS | FAIL | INV | NDAT | NAV]

P1 width: real

P2 width state state: [PASS | FAIL | INV | NDAT | NAV]

P2 width: real

P1P2 spacing state: [PASS | FAIL | INV | NDAT | NAV]

P1P2 spacing: real

channel indicator: [X | Y | INV]

Description: Read back all the measured DME items.

Frequency is in Hz. It is rounded to nearest 10 kHz.

ERP is in dBm, dBw, or W. Use SYST:UNIT:POW to select.

P1, P2 and P1-P2 spacing are in micro-seconds.

PRF is in Hz.

Example: `DME:MEAS?`

:DME

:MEASure

:STARt

Parameters: None

Description: This starts dme measurements.

Example: `DME:MEAS:STAR`

Start dme measurements.

:DME

:MEASure

:STOP

Parameters: None

Description: This stops dme measurements.

Example: `DME:MEAS:STOP`

Stop dme measurements.

:DME

:RANGe

[:CURRent]

Parameters: <NRf>

range

Valid values: range: real. Valid values are 0 to 450. Values outside range are rejected and an error generated.

Description: Set the range of the aircraft in nautical miles. Rounded to nearest 0.01 nm.

This cannot be set larger than the max range set by :DME:RANG:MAX. An error will be generated if the value is greater than the maximum range.

Example: `DME:RANG 235.31`

Set range to 235.31 nautical miles.

:DME

:RANGe

[:CURRent]?

Parameters: None

Response: <NR2>

range

Returned values: range: real. Values are in the range 0 to 450.

Description: Determine the range to the aircraft in nautical miles.

Example: `DME:RANG?`

:DME

:RANGe

:MAXimum

Parameters: <NRf>

range

Valid values: range: real. Valid values are 0 to 450. Values outside range are rejected and an error generated.

Description: Set the maximum range of the aircraft in nautical miles. Rounded to nearest 0.01 nm. The current range cannot be set higher than this and will be clipped if necessary.

Example: `DME:RANG:MAX 335.5`

Set range to 335.5 nautical miles.

:DME

:RANGe

:MAXimum?

Parameters: None

Response: <NR2>

range

Returned values: range: real. Values are in the range 0 to 450.

Description: Determine the maximum range to the aircraft in nautical miles.

Example: `DME:RANG:MAX?`

:DME

:RATE

:DIRection

Parameters: <CPD>

direction

Valid values: direction: [IN | OUT]. Values other than those stated are rejected and an error generated.

Description: Set the direction of aircraft travel. Not used if stationary.

Example: `DME:RATE:DIR IN`

Set direction as IN.

:DME

:RATE

:DIRection?

Parameters: none

Response: <CRD>

direction

Returned values: direction: [IN | OUT]

Description: Determine the direction of flight. Can be read at any time, but if stationary, not being used.

Example: `DME:RATE:DIR?`

:DME

:RATE

:STATionary

Parameters: <BOOLEAN PROGRAM DATA>

stationary

Description: Set whether the aircraft is stationary at a fixed distance, or is flying.

If stationary, direction and rate are not used.

Example: `DME:RATE:STAT OFF`

Set into flying mode (not stationary).

:DME

:RATE

:STATionary?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

stationary

Description: Determine if stationary or flying.

Example: `DME:RATE:STAT?`

:DME

:RATE

:VALue

Parameters: <NRf>

rate

Valid values: rate: integer. Valid values are 10 to 6500. Values outside range are rejected and an error generated.

Description: Set the rate, in knots. Only used if not stationary. Rounded to nearest 10 knots.

Example: `DME:RATE:VAL 300`

Set rate to be 300 knots.

:DME

:RATE

:VALue?

Parameters: none

Response: <NR1>

rate

Returned values: rate: integer. Values are in the range 10 to 6500.

Description: Determine the rate. Can be read at any time, but if stationary, not being used.

Example: `DME:RATE:VAL?`

:DME

:REPLy

Parameters: <NRf>

percent reply

Valid values: percent reply: integer. Valid values are 0 to 100. Values outside range are rejected and an error generated.

Description: Set the percentage reply rate.

Example: `DME:REPL 100`

Set up to reply to all interrogations.

:DME

:REPLy?

Parameters: none

Response: <NR1>

percent reply

Returned values: percent reply: integer. Values are in the range 0 to 100.

Description: Determine what percentage of interrogations will be replied to.

Example: `DME:REPL?`

:DME

:SQUitter

Parameters: <BOOLEAN PROGRAM DATA>

squitter state

Description: This sets squitter on or off. When on, squitters will be output at a PRF of 2700 Hz.

Example: `DME:SQU ON`

Turn squitters on.

:DME

:SQUitter?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

squitter state

Description: Determine whether squitters are being sent or not.

Example: `DME:SQU?`

THIS PAGE INTENTIONALLY LEFT BLANK.

TCAS COMMANDS

TCAS Mode provides flight line test capability for TCAS I and II. ATCRBS and Mode S intruders are simulated, allowing the generation of proximity, TA and RA flight deck annunciations. TCAS Interrogator parametric measurements are displayed.

TCAS SUBSYSTEM

```
TCAS
    ANTenna
        [CONNect]\?
        GAIN\?
        HEIGht\?
        RANGe\?
    CLOSs
        ANTenna
            MODE\?
            [VALue]\?
        [CURRent]\?
        DIRect\?
    CONVerge\?
    DIAGnostic
        ACQuisition\?
        ADDRess\?
        ALTitude?
        DATA?
        GENerate\?
        RATTenuation\?
        SELect\?
        STARt
        STOP
        TLEVel\?
    INTRuder
        ADDRess\?
        ALTitude
            RATE\?
            RELative\?
            REPorting\?
            STARt\?
            STOP\?

TCAS
    INTRuder
        RANGe
            RATE\?
            STARt\?
            STOP\?
        [TYPE]\?
    MEASure
        [DATA]?
        RA?
        STARt
        STOP
        TCAS?
        UF0?
        UF16?
        WSHout?
    RBITs
        DF0
            SL\?
            VS\?
        DF11
            CA\?
        DF16
            ARA\?
            RAC\?
            RIA\?
            RIT\?
            SL\?
            VDS\?
            VS\?
    RESet
```

TCAS SUBSYSTEM

- TCAS
  - REPLy\?
  - SCENario
    - LIST?
    - LOAD
    - NUMBer?
    - READ?
    - SAVE
    - [SELect]
  - SQUitter\?
  - STATionary\?
  - TYPE\?
  - UUT
    - ADDRess
      - STATe\?
      - [VALue]\?
    - ALTitude
      - STATe\?
      - [VALue]\?
  - WSHout
    - [ATTenuation]\?
    - STATe\?

:TCAS

:ANTenna

[:CONNect]

Parameters: <CPD>

antenna connection

Valid values: antenna connection: [DIRect | ANTenna]. Values other than those stated are rejected and an error generated.

Description: Set whether the TCAS measurements are to be performed over the air using the test set antenna (ANTenna) or directly via a cable (DIRect).

In most cases TCAS testing will be performed over the air.

Does not affect DME and transponder measurements.

Example: `TCAS:ANT DIRECT`

Make measurements using a direct connection to the aircraft.

:TCAS

:ANTenna

[:CONNect]?

Parameters: None

Response: <CRD>

antenna connection

Returned values: antenna connection: [DIR | ANT]

Description: Determine whether measurements are being performed over the air or directly connected via a cable.

Example: `TCAS:ANT?`

:TCAS

:ANTenna

:GAIN

Parameters: <NRf>, <NRf>

gain1030, gain1090

Valid values: gain1030: real. Valid values are 0.0 to 20.9. Values outside range are rejected and an error generated.
gain1090: real. Valid values are 0.0 to 20.9. Values outside range are rejected and an error generated.

Description: Set the gain of the 6000 antenna at the tcas transmit and receive frequencies.

Example: `TCAS:ANT:GAIN 9.5, 9.7`

Set gain of the 6000 antenna.

:TCAS

:ANTenna

:GAIN?

Parameters: None

Response: <NR2>, <NR2>

gain1030, gain1090

Returned values: gain1030: real. Values are in the range 0.0 to 20.9.
gain1090: real. Values are in the range 0.0 to 20.9.

Description: Determine the gain of the 6000 antenna at the tcas transmit and receive frequencies.

Example: `TCAS:ANT:GAIN?`

:TCAS

:ANTenna

:HEIGht

Parameters: <NRf>

antenna height

Valid values: antenna height: real

Description: Set the height to the tcas antenna on the aircraft. Rounded to nearest 0.5 meters or integral feet. In meters mode, height is 0.5 to 30.0 meters. In feet mode, height is 1 to 99 feet.

Example: `TCAS:ANT:HEIG 5.5`

Set antenna range to 5.5 meters (assuming range entry mode is meters).

:TCAS

:ANTenna

:HEIGht?

Parameters: None

Response: <NR2>

height

Returned values: height: real

Description: Determine the height to the tcas antenna on the aircraft.

Example: `TCAS:ANT:HEIG?`

:TCAS

:ANTenna

:RANGe

Parameters: <NRf>

antenna range

Valid values: antenna range: real

Description: Set the range to the tcas antenna on the aircraft. Rounded to nearest 0.5 meters or integral feet. In meters mode, range is 2.0 to 75.0 meters. In feet mode, range is 6 to 250 feet.

Example: `TCAS:ANT:RANG 5.5`

Set antenna range to 5.5 meters (assuming range entry mode is meters).

:TCAS

:ANTenna

:RANGe?

Parameters: None

Response: <NR2>

range

Returned values: range: real

Description: Determine the range to the tcas antenna on the aircraft.

Example: `TCAS:ANT:RANG?`

:TCAS

:CLOSs

:ANTenna

:MODE

Parameters: <CPD>

ant cable loss mode

Valid values: ant cable loss mode: [UDEFined | L25 | L50 | L75]. Values other than those stated are rejected and an error generated.

Description: Select whether one of the VIAVI supplied cables is used to connect to the antenna (the 6000 has the cable loss programmed into it) or a user cable is used and its cable loss must be entered using TCAS:CLOS:ANT.

Example: `TCAS:CLOS:ANT:MODE L50`

The user is using a VIAVI supplied 50 ft cable and the 6000 automatically handles its cable loss..

:TCAS

:CLOSs

:ANTenna

:MODE?

Parameters: none

Response: <CRD>

ant cable loss mode

Returned values: ant cable loss mode: [UDEF | L25 | L50 | L75]

Description: Determine whether we are using a VIAVI supplied cable or a user defined cable to connect to the antenna.

Example: `TCAS:CLOS:ANT:MODE?`

:TCAS

:CLOSs

:ANTenna

[:VALue]

Parameters: <NRf>

cable loss

Valid values: cable loss: real

Description: This command sets the cable loss (in dB) of the cable used to connect to the test set antenna. This command will always set the cable loss of the antenna cable even if the current selection is direct connect. This value is only used if TCAS:CLOS:ANT:MODE is UDEF.

The tcas cable loss value is kept separate from the DME and transponder cable loss values.

The value is in units of dB.

Example: `TCAS:CLOS:ANT 1.7`

Inform the instrument of the loss of the cable connected to the 6000 antenna.

:TCAS

:CLOSs

:ANTenna

[:VALue]?

Parameters: None

Response: <NR2>

cable loss

Returned values: cable loss: real

Description: Determine the loss of the antenna cable. This is the value used when TCAS:CLOS:ANT:MODE is set to UDEF.

Example: `TCAS:CLOS:ANT?`

:TCAS

:CLOSs

[:CURRent]

Parameters: <NRf>

cable loss

Valid values: cable loss: real

Description: This command sets the cable loss (in dB) of the cable used to direct connect to the aircraft antenna or the cable used to connect to the test set antenna if performing over the air measurements.

The tcas cable loss value is kept separate from the DME and transponder cable loss values.

The value is in units of dB.

Example: `TCAS:CLOS 1.7`

Inform the instrument of the loss of the cable currently being used.

:TCAS

:CLOSs

[:CURRent]?

Parameters: None

Response: <NR2>

cable loss

Returned values: cable loss: real

Description: Determine the loss of the cable currently being used.

Example: `TCAS:CLOS?`

:TCAS

:CLOSs

:DIRect

Parameters: <NRf>

cable loss

Valid values: cable loss: real

Description: This command sets the cable loss (in dB) of the cable used to directly connect from the test set to the transponder under test. This command will always set the cable loss of the direct connect cable even if the current selection is antenna (over the air).

The tcas cable loss value is kept separate from the DME and transponder cable loss values.

The value is in units of dB.

Example: `TCAS:CLOS:DIR 1.7`

Inform the instrument of the loss of the direct connect cable.

:TCAS

:CLOSs

:DIRect?

Parameters: None

Response: <NR2>

cable loss

Returned values: cable loss: real

Description: Determine the loss of the direct connect cable.

Example: `TCAS:CLOS:DIR?`

:TCAS

:CONVerge

Parameters: <BOOLEAN PROGRAM DATA>

converge state

Description: This sets whether the intruder aircraft will fly directly at the aircraft under test. If set then the range stop will be forced to 0, the altitude stop will match that of the aircraft under test, and the altitude rate will be automatically calculated.

This provides an easy way of testing an intruder flying directly at the aircraft under test.

Example: `TCAS:CONV ON`

Fly directly at the aircraft under test.

:TCAS

:CONVerge?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

converge state

Description: Determine whether the intruder will fly directly at the aircraft under test. Setting converge on reduces the number of parameters that need setting up before performing a test.

Example: `TCAS:CONV?`

:TCAS

:DIAGnostic

:AQUisition

Parameters: <BOOLEAN PROGRAM DATA>

acquisition state

Description: This sets the state of the AQ bit that the test set transmits as part of DF0 and DF16.

Setting acquisition on will set the AQ bit to 1. Setting acquisition off will set the AQ bit to 0.

Example: `TCAS:ACQ ON`

Set AQ bit to 1 for DF0 and DF16 transmissions.

:TCAS

:DIAGnostic

:AQUisition?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

acquisition state

Description: Determine state of the DF0 and DF16 AQ bit.

Example: `TCAS:ACQ?`

:TCAS

:DIAGnostic

:ADDRess

Parameters: <NRf>

address

Valid values: address: integer. Valid values are 0 to 16777215. Values outside range are rejected and an error generated.

Description: Set the address to use in diagnostics mode

The address can also be entered in hexadecimal using #Hxxxxxx, or in octal using #Qxxxxxxxx, or in binary using #Bxxxxxxxxxxxxxxxxxxxxxxxx.

Example: `TCAS:DIAG:ADDR 238467`

Select an address for mode S interrogations in diagnostics mode.

:TCAS

:DIAGnostic

:ADDRess?

Parameters: None

Response: <NR1>

address

Returned values: address: integer. Values are in the range 0 to 16777215.

Description: Determine the selected manual transponder address.

Example: `TCAS:DIAG:ADDR?`

:TCAS

:DIAGnostic

:ALTitude

Parameters: <NRf>

altitude

Valid values: altitude: integer. Valid values are -1000 to 126700. Values outside range are rejected and an error generated.

Description: Set the altitude to use in diagnostics mode.

Can be set in steps of 100 ft only. Will round to nearest valid value.

Example: `TCAS:DIAG:ALT 38400`

Set an altitude for diagnostics mode.

:TCAS

:DIAGnostic

: ALTitude?

Parameters: None

Response: <NR1>

altitude

Returned values: altitude: integer. Values are in the range -1000 to 126700.

Description: Determine the altitude used during diagnostics.

Example: `TCAS:DIAG:ALT?`

:TCAS

:DIAGnostic

:DATA?

Parameters: none

Response: <CRD>, <NR1>

state, data

Returned values: state: [NRUN | PASS]
data: integer. Values are in the range 0 to 65535.

Description: Read the dsp measurement value – a raw reading of the power measured

This command is intended to be used in the TCAS diagnostics dsp mode.

Example: `TCAS:DIAG:DATA?`

Read the raw dsp value.

:TCAS

:DIAGnostic

:GENerate

Parameters: <BOOLEAN PROGRAM DATA>

GEN IF during dsp measure state

Description: This sets whether the test set generates an output while performing tcas diagnostic dsp measure. If set then a signal is output at the IF stage of the generate chain.

Example: `TCAS:DIAG:GEN ON`

Enable a signal at IF generate stage.

:TCAS

:DIAGnostic

:GENerate?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

GEN IF during dsp measure state

Description: Determine whether a signal is generated at the IF stage during tcas diagnostics dsp measure.

Example: `TCAS:DIAG:GEN?`

:TCAS

:DIAGnostic

:RATTenuation

Parameters: <NRf>

attenuation

Valid values: attenuation: integer. Valid values are 0 to 55. Values outside range are rejected and an error generated.

Description: Set the receiver attenuator to specified setting (units of dB). Since the attenuation is settable in discrete steps, the entered value will be rounded to the nearest valid value: 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55.

Example: `TCAS:DIAG:RATT 20`

Set the attenuator for the receiver.

:TCAS

:DIAGnostic

:RATTenuation?

Parameters: None

Response: <NR1>

attenuation

Returned values: attenuation: integer. Values are in the range 0 to 55.

Description: Determine the selected receiver attenuation.

Example: `TCAS:DIAG:RATT?`

:TCAS

:DIAGnostic

:SELect

Parameters: <CPD>

selected test

Valid values: selected test: [C | DF0 | DF11 | DF16 | CW | DSP]. Values other than those stated are rejected and an error generated.

Description: Selects which test to perform in diagnostics mode.

Example: `TCAS:DIAG:SEL DF16`

Select the DF16 test for diagnostics.

:TCAS

:DIAGnostic

:SELect?

Parameters: None

Response: <CRD>

selected test

Returned values: selected test: [C | DF0 | DF11 | DF16 | CW | DSP]

Description: Determine which test is being performed for tcas diagnostics.

Example: `TCAS:DIAG:SEL?`

:TCAS

:DIAGnostic

:STARt

Parameters: None

Description: This starts the currently selected diagnostic test. Replies will be sent at a fixed prf.

Example: `TCAS:DIAG:STAR`

Start diagnostics test.

:TCAS

:DIAGnostic

:STOP

Parameters: None

Description: This stops the current diagnostic test.

Example: `TCAS:DIAG:STOP`

Stop diagnostics test.

:TCAS

:DIAGnostic

:TLEVel

Parameters: <NRf>

level

Valid values: level: integer

Description: Set the level to transmit on. Always in units of dBm.

For the Antenna port, valid range is –2 dBm to –67 dBm
For the Direct Connect port, valid range is –47 dBm to –115 dBm.

Example: `TCAS:DIAG:TLEV –26`

Select the level for transmitted replies in diagnostics mode.

:TCAS

:DIAGnostic

:TLEVel?

Parameters: none

Response: <NR1>

level

Returned values: level: integer. Values are in the range -115 to -2.

Description: Determine the selected transmit level.

Example: `TCAS:DIAG:TLEV?`

:TCAS

:INTRuder

:ADDRess

Parameters: <NRf>

address

Valid values: address: integer. Valid values are 0 to 16777215. Values outside range are rejected and an error generated.

Description: Set the address to use for any mode S transmissions/replies.

The address can also be entered in hexadecimal using #Hxxxxxx, or in octal using #Qxxxxxxxx, or in binary using #Bxxxxxxxxxxxxxxxxxxxxxxxx.

Example: `TCAS:INTR:ADDR 238467`

Select an address for the intruder (test set) mode S interrogations.

:TCAS

:INTRuder

:ADDRess?

Parameters: None

Response: <NR1>

address

Returned values: address: integer. Values are in the range 0 to 16777215.

Description: Determine the intruder (test set) address.

Example: `TCAS:INTR:ADDR?`

:TCAS

:INTRuder

:ALTitude

:RATE

Parameters: <NRf>

altitude rate

Valid values: altitude rate: integer. Valid values are 0 to 10000. Values outside range are rejected and an error generated.

Description: Set the altitude rate to use when converge is off. If converge is on then the altitude rate is calculated by the test set.

Example: `TCAS:INTR:ALT:RATE 238`

Set the altitude rate to be 238 feet per minute.

:TCAS

:INTRuder

:ALTitude

:RATE?

Parameters: None

Response: <NR1>

altitude rate

Returned values: altitude rate: integer. Values are in the range 0 to 10000.

Description: Determine the altitude rate.

Example: `TCAS:INTR:ALT:RATE?`

:TCAS

:INTRuder

:ALTitude

: RELative

Parameters: <BOOLEAN PROGRAM DATA>

relative altitude

Description: This will set whether the returned altitude is absolute height or a relative height to the height of the aircraft under test. This only affects the altitude data that is returned in the TCAS:MEAS[:DATA]? command.

Example: `TCAS:INTR:ALT:REL ON`

Treat altitude to be relative to the aircraft under test.

:TCAS

:INTRuder

:ALTitude

:RELative?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

relative altitude

Description: Determine whether returned altitude is absolute or is relative to the altitude of the aircraft under test.

Example: `TCAS:INTR:ALT:REL?`

:TCAS

:INTRuder

:ALTitude

: REPorting

Parameters: <BOOLEAN PROGRAM DATA>

altitude reporting

Description: This will set whether altitude is returned by the intruder (test set) in its replies.

Example: `TCAS:INTR:ALT:REP ON`

Return altitude when requested.

:TCAS

:INTRuder

:ALTitude

:REPorting?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

altitude reporting

Description: Determine whether altitude is returned or not

Example: `TCAS:INTR:ALT:REP?`

:TCAS

:INTRuder

:ALTitude

:STARt

Parameters: <NRf>

altitude start

Valid values: altitude start: integer. Valid values are –127700 to 127700. Values outside range are rejected and an error generated.

Description: Set the starting altitude (in feet) of the intruder (test set). This value is relative to the aircraft under test. Can be set in steps of 100 ft only. Will round to nearest valid value.

If the aircraft under test altitude is being determined automatically then the altitude is not known until the test is started. Due to this only a limited amount of error trapping can be done (-127700 ft to +127700 ft). When the test is started, the intruder altitude will be clipped as necessary to keep it's absolute altitude between the valid range of –1000 ft to +126700 ft.

If the aircraft under test altitude is user entered then the intruder altitude will be accepted only if the entered value along with the user entered aircraft under test value will set the intruder aircraft altitude between the valid range of –1000 ft to +126700 ft.

Example: `TCAS:INTR:ALT:STAR –1000`

Set the starting altitude of the simulated intruder aircraft to –1000 ft below the aircraft under test.

:TCAS

:INTRuder

:ALTitude

:STARt?

Parameters: None

Response: <NR1>

altitude start

Returned values: altitude start: integer. Values are in the range -127700 to 127700.

Description: Determine the starting altitude of the simulated intruder. This is relative to the aircraft under test altitude.

Example: `TCAS:INTR:ALT:STAR?`

:TCAS

:INTRuder

:ALTitude

:STOP

Parameters: <NRf>

altitude stop

Valid values: altitude stop: integer. Valid values are -127700 to 127700. Values outside range are rejected and an error generated.

Description: Set the ending altitude (in feet) of the intruder (test set). This value is relative to the aircraft under test. Can be set in steps of 100 ft only. Will round to nearest valid value.

This value is not used if convergence is enabled. It will still be possible to set and read the value, but that value will not be used by the test set until convergence is disabled. While convergence is on, the intruder stop altitude is considered to be the same as the aircraft under test.

If the aircraft under test altitude is being determined automatically then the altitude is not known until the test is started. Due to this only a limited amount of error trapping can be done (-127700 ft to +127700 ft). When the test is started, the intruder altitude will be clipped as necessary to keep it's absolute altitude between the valid range of –1000 ft to +126700 ft.

If the aircraft under test altitude is user entered then the intruder altitude will be accepted only if the entered value along with the user entered aircraft under test value will set the intruder aircraft altitude between the valid range of –1000 ft to +126700 ft.

Example:
```
TCAS:INTR:ALT:STOP 51000
```

Set the ending altitude of the simulated intruder aircraft to 51000 ft above the aircraft under test.

:TCAS

:INTRuder

:ALTitude

:STOP?

Parameters: None

Response: <NR1>

altitude stop

Returned values: altitude stop: integer. Values are in the range -127700 to 127700.

Description: Determine the ending altitude of the simulated intruder.

If convergence is enabled, the value will still be returned, however the test set is not using that value.

Example: `TCAS:INTR:ALT:STOP?`

:TCAS

:INTRuder

:RANGe

:RATE

Parameters: <NRf>

range rate

Valid values: range rate: integer. Valid values are 0 to 1200. Values outside range are rejected and an error generated.

Description: Set the range rate to use.

Example: `TCAS:INTR:RANG:RATE 600`

Set the range rate to be 600 knots.

:TCAS

:INTRuder

:RANGe

:RATE?

Parameters: None

Response: <NR1>

range rate

Returned values: range rate: integer. Values are in the range 0 to 1200.

Description: Determine the range rate.

Example: `TCAS:INTR:RANG:RATE?`

:TCAS

:INTRuder

:RANGe

:STARt

Parameters: <NRf>

range start

Valid values: range start: real. Valid values are 0.00 to 260.00. Values outside range are rejected and an error generated.

Description: Set the starting range (in nautical miles) of the intruder (test set).

For an atcrbs intruder the minimum range allowed is 0.35 nm.

Example: `TCAS:INTR:RANG:STAR 22.5`

Set the starting range of the simulated intruder aircraft to 22.5 nautical miles.

:TCAS

:INTRuder

:RANGe

:STARt?

Parameters: None

Response: <NR2>

range start

Returned values: range start: real. Values are in the range 0.00 to 260.00.

Description: Determine the starting range of the simulated intruder.

Example: `TCAS:INTR:RANG:STAR?`

:TCAS

:INTRuder

:RANGe

:STOP

Parameters: <NRf>

range stop

Valid values: range stop: real. Valid values are 0.00 to 260.00. Values outside range are rejected and an error generated.

Description: Set the ending range (in nautical miles) of the intruder (test set).

For an atcrbs intruder the minimum range allowed is 0.35 nm.

This value is not used if convergence is enabled. The value can still be set but it will not be used until convergence is turned off using TCAS:CONV OFF.

Example: `TCAS:INTR:RANG:STOP 3.76`

Set the ending range of the simulated intruder aircraft to 3.76 nautical miles.

:TCAS

:INTRuder

:RANGe

:STOP?

Parameters: None

Response: <NR2>

range stop

Returned values: range stop: real. Values are in the range 0.00 to 260.00.

Description: Determine the ending range of the simulated intruder.

Note that this will return the range stop even if it is not being used because convergence is off.

Example: `TCAS:INTR:RANG:STOP?`

:TCAS

:INTRuder

[:TYPE]

Parameters: <CPD>

intruder type

Valid values: intruder type: [ATCRbs | MS]. Values outside range are rejected and an error generated.

Description: Select whether the test set is simulating an atcrbs equipped instruder aircraft or a mode S equipped intruder aircraft.

Example: `TCAS:INTR MS`

We are simulating an aircraft equipped with a mode S transponder.

:TCAS

:INTRuder

[:TYPE]?

Parameters: None

Response: <CRD>

intruder type

Returned values: intruder type: [ATCR | MS]

Description: Determine whether the test set is simulating an atcrbs or a mode S equipped aircraft.

Example: `TCAS:INTR?`

:TCAS

:MEASure

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR2>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <CRD>, <CRD>, <NR1>, <CRD>

overall state, freq state, freq, ERP state, ERP, range state, range, direction state, direction, altitude state, altitude, altitude direction state, altitude direction, tcas status state, tcas status, threat status, time to encounter, surveillance status

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

freq state: [PASS | FAIL | INV | NDAT | NAV]

freq: integer. Values are in the range 0 to 1999999000.

ERP state: [PASS | FAIL | INV | NDAT | NAV]

ERP: real

range state: [PASS | FAIL | INV | NDAT | NAV]

range: real

direction state: [PASS | FAIL | INV | NDAT | NAV]

direction: [IN | OUT | STOP]

altitude state: [PASS | FAIL | INV | NDAT | NAV]

altitude: integer. Values are in the range -1000 to 126700.

altitude direction state: [PASS | FAIL | INV | NDAT | NAV]

altitude direction: [UP | DOWN | STOP]

tcas status state: [PASS | FAIL | INV | NDAT | NAV]

tcas status: [TRAC | ACQ]

threat status: [TRAF | PROX | RES | NONE]

time to encounter: integer

surveillance status: [NONE | INT | OK]

:TCAS

:MEASure

[:DATA]? (cont)

Description: Read back the measured TCAS overview items. More detailed items can be read using other commands (TCAS:MEAS:DF0?, TCAS:MEAS:DF16?, TCAS:MEAS:RA?, and TCAS:MEAS:TCAS?).

Frequency is in Hz. It is rounded to nearest 1 kHz.

ERP is in dBm, dBw, or W. Use SYST:UNIT:POW to select.

range is in nautical miles.

altitude is in feet.

time to encounter is in seconds.

Example: `TCAS:MEAS?`

:TCAS

:MEASure

:RA?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <OCTAL NUMERIC RESPONSE DATA>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>

overall state, UDS state, UDS, RAT state, RAT, MTE state, MTE, AID state, AID, CAC state, CAC, ARA state, ARA, RAC state, RAC, count state, count

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

UDS state: [PASS | FAIL | INV | NDAT | NAV]

UDS: integer

RAT state: [PASS | FAIL | INV | NDAT | NAV]

RAT: integer

MTE state: [PASS | FAIL | INV | NDAT | NAV]

MTE: integer

AID state: [PASS | FAIL | INV | NDAT | NAV]

AID: integer

CAC state: [PASS | FAIL | INV | NDAT | NAV]

CAC: integer

ARA state: [PASS | FAIL | INV | NDAT | NAV]

ARA: integer

RAC state: [PASS | FAIL | INV | NDAT | NAV]

RAC: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

Description: Read back all the measured tcas resolution advisory broadcast items.

All values are returned as decimal integers except AID which is returned in octal.

Example: `TCAS:MEAS:RA?`

:TCAS

:MEASure

:STARt

Parameters: None

Description: This starts tcas measurements.

Example: `TCAS:MEAS:STAR`

Start tcas measurements.

:TCAS

:MEASure

:STOP

Parameters: None

Description: This stops tcas measurements.

Example: `TCAS:MEAS:STOP`

Stop tcas measurements.

:TCAS

:MEASure

:TCAS?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>

overall state, UDS state, UDS, MID state, MID, count state, count, interval state, interval

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

UDS state: [PASS | FAIL | INV | NDAT | NAV]

UDS: integer

MID state: [PASS | FAIL | INV | NDAT | NAV]

MID: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

interval state: [PASS | FAIL | INV | NDAT | NAV]

interval: real

Description: Read back all the measured tcas broadcast items. Interval is in seconds.

Example: `TCAS:MEAS:TCAS?`

:TCAS

:MEASure

:UF0?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>

overall state, RL state, RL, AQ state, AQ, count state, count, interval state, interval

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

RL state: [PASS | FAIL | INV | NDAT | NAV]

RL: integer

AQ state: [PASS | FAIL | INV | NDAT | NAV]

AQ: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

interval state: [PASS | FAIL | INV | NDAT | NAV]

interval: real

Description: Read back all the measured tcas UF0 items. Interval is in seconds.

Example: `TCAS:MEAS:UF0?`

:TCAS

:MEASure

:UF16?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>

overall state, RL state, RL, AQ state, AQ, UDS state, UDS, LCK state, LCK, MTB state, MTB, CVC state, CVC, VRC state, VRC, CHC state, CHC, HRC state, HRC, ESB state, ESB, MID state, MID, count state, count, interval state, interval

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

RL state: [PASS | FAIL | INV | NDAT | NAV]

RL: integer

AQ state: [PASS | FAIL | INV | NDAT | NAV]

AQ: integer

UDS state: [PASS | FAIL | INV | NDAT | NAV]

UDS: integer

LCK state: [PASS | FAIL | INV | NDAT | NAV]

LCK: integer

MTB state: [PASS | FAIL | INV | NDAT | NAV]

MTB: integer

CVC state: [PASS | FAIL | INV | NDAT | NAV]

CVC: integer

VRC state: [PASS | FAIL | INV | NDAT | NAV]

VRC: integer

CHC state: [PASS | FAIL | INV | NDAT | NAV]

CHC: integer

HRC state: [PASS | FAIL | INV | NDAT | NAV]

HRC: integer

:MEASure

:UF16? (cont)

Returned values: ESB state: [PASS | FAIL | INV | NDAT | NAV]

ESB: integer

MID state: [PASS | FAIL | INV | NDAT | NAV]

MID: integer

count state: [PASS | FAIL | INV | NDAT | NAV]

count: integer

interval state: [PASS | FAIL | INV | NDAT | NAV]

interval: real

Description: Read back all the measured tcas UF16 items. Interval is in seconds.

Example: `TCAS:MEAS:UF16?`

:TCAS

:MEASure

:WSHout?

Parameters: None

Response: <CRD>, <CRD>, <NR2>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR2>, <CRD>, <NR2>

overall state, range state, range, altitude state, altitude, no supp state, no supp, S1 state, S1, P2 state, P2, both state, both, spacing state, spacing, interval state, interval

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

range state: [PASS | FAIL | INV | NDAT | NAV]

range: real

altitude state: [PASS | FAIL | INV | NDAT | NAV]

altitude: integer. Values are in the range -1000 to 126700.

no supp state: [PASS | FAIL | INV | NDAT | NAV]

no supp: integer

S1 state: [PASS | FAIL | INV | NDAT | NAV]

S1: integer

P2 state: [PASS | FAIL | INV | NDAT | NAV]

P2: integer

both state: [PASS | FAIL | INV | NDAT | NAV]

both: integer

spacing state: [PASS | FAIL | INV | NDAT | NAV]

spacing: real

interval state: [PASS | FAIL | INV | NDAT | NAV]

interval: real

:TCAS

:MEASure

:WSHout? (cont)

Description: Read back all the measured tcas whisper/shout test items. Spacing is in milli-seconds. Interval is in seconds.

The range and altitude are the same as for TCAS:MEAS? and are included here for the users convenience.

Example: `TCAS:MEAS:WSH?`

:TCAS

:RBITs

:DF0

:SL

Parameters: <NRf>

df0 sl

Valid values: df0 sl: integer. Valid values are 0 to 7. Values outside range are rejected and an error generated.

Description: Set the SL field in DF0 transmissions.

Example: `TCAS:RBIT:DF0:SL 1`

Set the SL field to 1.

:TCAS

:RBITs

:DF0

:SL?

Parameters: None

Response: <NR1>

df0 sl

Returned values: df0 sl: integer. Values are in the range 0 to 7.

Description: Determine the SL field in DF0.

Example: `TCAS:RBIT:DF0:SL?`

:TCAS

:RBITs

:DF0

:VS

Parameters: <NRf>

df0 vs

Valid values: df0 vs: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the VS field in DF0 transmissions.

Example: `TCAS:RBIT:DF0:VS 0`

Set the VS bit to 0.

:TCAS

:RBITs

:DF0

:VS?

Parameters: None

Response: <NR1>

df0 vs

Returned values: df0 vs: integer. Values are in the range 0 to 1.

Description: Determine the VS field in DF0.

Example: `TCAS:RBIT:DF0:VS?`

:TCAS

:RBITs

:DF11

:CA

Parameters: <NRf>

df11 ca

Valid values: df11 ca: integer. Valid values are 0 to 7. Values outside range are rejected and an error generated.

Description: Set the CA field in DF11 transmissions.

Example: `TCAS:RBIT:DF11:CA 4`

Set the CA field to 4.

:TCAS

:RBITs

:DF11

:CA?

Parameters: None

Response: <NR1>

df11 ca

Returned values: df11 ca: integer. Values are in the range 0 to 7.

Description: Determine the CA field in DF11.

Example: `TCAS:RBIT:DF11:CA?`

:TCAS

:RBITs

:DF16

:ARA

Parameters: <NRf>

df16 ara

Valid values: df16 ara: integer. Valid values are 0 to 16383. Values outside range are rejected and an error generated.

Description: Set the ARA field in DF16 transmissions.

Example: `TCAS:RBIT:DF16:ARA 6391`

Set the ARA field to 6391.

:TCAS

:RBITs

:DF16

:ARA?

Parameters: None

Response: <NR1>

df16 ara

Returned values: df16 ara: integer. Values are in the range 0 to 16383.

Description: Determine the ARA field in DF16.

Example: `TCAS:RBIT:DF16:ARA?`

:TCAS

:RBITs

:DF16

:RAC

Parameters: <NRf>

df16 rac

Valid values: df16 rac: integer. Valid values are 0 to 15. Values outside range are rejected and an error generated.

Description: Set the RAC field in DF16 transmissions.

Example: `TCAS:RBIT:DF16:RAC 9`

Set the RAC field to 9.

:TCAS

:RBITs

:DF16

:RAC?

Parameters: None

Response: <NR1>

df16 rac

Returned values: df16 rac: integer. Values are in the range 0 to 15.

Description: Determine the RAC field in DF16.

Example: `TCAS:RBIT:DF16:RAC?`

:TCAS

:RBITs

:DF16

:RIA

Parameters: <NRf>

df16 ria

Valid values: df16 ria: integer. Valid values are 8 to 14. Values outside range are rejected and an error generated.

Description: Set the RI field in DF16 acquisition transmissions.

Example: `TCAS:RBIT:DF16:RIA 9`

Set the RIA field to 9.

:TCAS

:RBITs

:DF16

:RIA?

Parameters: None

Response: <NR1>

df16 ria

Returned values: df16 ria: integer. Values are in the range 8 to 14.

Description: Determine the RI field in DF16 acquisition transmissions.

Example: `TCAS:RBIT:DF16:RIA?`

:TCAS

:RBITs

:DF16

:RIT

Parameters: <NRf>

df16 rit

Valid values: df16 rit: integer. Valid values are 0 to 4. Values outside range are rejected and an error generated.

Description: Set the RI field in DF16 tracking (non acquisition) transmissions.

Only 0, 3, and 4 are valid values, all other numbers will give an error.

Example: `TCAS:RBIT:DF16:RIT 3`

Set the RIT field to 3.

:TCAS

:RBITs

:DF16

:RIT?

Parameters: None

Response: <NR1>

df16 rit

Returned values: df16 rit: integer. Values are in the range 0 to 4.

Description: Determine the RI field in DF16 tracking (non acquisition) transmissions.

Example: `TCAS:RBIT:DF16:RIT?`

:TCAS

:RBITs

:DF16

:SL

Parameters: <NRf>

df16 sl

Valid values: df16 sl: integer. Valid values are 0 to 7. Values outside range are rejected and an error generated.

Description: Set the SL field in DF16 transmissions.

Example: `TCAS:RBIT:DF16:SL 1`

Set the SL field to 1.

:TCAS

:RBITs

:DF16

:SL?

Parameters: None

Response: <NR1>

df16 sl

Returned values: df16 sl: integer. Values are in the range 0 to 7.

Description: Determine the SL field in DF16.

Example: `TCAS:RBIT:DF16:SL?`

:TCAS

:RBITs

:DF16

:VDS

Parameters: <NRf>

df16 vds

Valid values: df16 vds: integer. Valid values are 0 to 255. Values outside range are rejected and an error generated.

Description: Set the VDS field in DF16 transmissions.

Example: `TCAS:RBIT:DF16:VDS 21`

Set the VDS field to 21.

:TCAS

:RBITs

:DF16

:VDS?

Parameters: None

Response: <NR1>

df16 vds

Returned values: df16 vds: integer. Values are in the range 0 to 255.

Description: Determine the VDS field in DF16.

Example: `TCAS:RBIT:DF16:VDS?`

:TCAS

:RBITs

:DF16

:VS

Parameters: <NRf>

df16 vs

Valid values: df16 vs: integer. Valid values are 0 to 1. Values outside range are rejected and an error generated.

Description: Set the VS field in DF16 transmissions.

Example: `TCAS:RBIT:DF16:VS 0`

Set the VS bit to 0.

:TCAS

:RBITs

:DF16

:VS?

Parameters: None

Response: <NR1>

df16 vs

Returned values: df16 vs: integer. Values are in the range 0 to 1.

Description: Determine the VS field in DF16.

Example: `TCAS:RBIT:DF16:VS?`

:TCAS

:RBITs

:RESet

Parameters: none

Description: Set all the fields in the tcas replies to their default value.

See appendix E for a list of the default values.

Example: `TCAS:RBIT:RES`

Default all reply fields.

:TCAS

:REPLy

Parameters: <NRf>

percent reply

Valid values: percent reply: integer. Valid values are 0 to 100. Values outside range are rejected and an error generated.

Description: Set the percentage reply rate.

Example: `TCAS:REPL 100`

Set up to reply to all interrogations.

:TCAS

:REPLy?

Parameters: none

Response: <NR1>

percent reply

Returned values: percent reply: integer. Values are in the range 0 to 100.

Description: Determine what percentage of interrogations will be replied to.

Example: `TCAS:REPL?`

:TCAS

:SCENario

:LIST?

Parameters: None

Response: <STRING RESPONSE DATA>, ..., <STRING RESPONSE DATA>

first scenario, ..., last scenario

Returned values: first scenario: string
...
last scenario: string

Description: List all tcas scenarios available in the instrument. The data is returned as multiple strings. The first three characters of each string are a two digit store number and a space, the remaining data is the store name.

Example: `TCAS:SCEN:LIST?`

List all the tcas scenarios available.

:TCAS

:SCENario

:LOAD

Parameters: <NRf>, <ARBITRARY BLOCK PROGRAM DATA>

store number, store data

Valid values: store number: integer. Valid values are 10 to 25. Values outside range are rejected and an error generated.

Description: Load a single scenario from external controller. The built-in scenarios cannot be overwritten, nor can the last power down.

Example:
```
TCAS:SCEN:LOAD 19,
#32040100010105780000012C0000FFFFF2540000000001F4006400000096
0100000000000000000000000803000000000003000000000000000000000
0000000000000000000000000002D3335303020667420636F6C6C6973696F
6E0000000000000000000007CEA
```

Download store 19.

:TCAS

:SCENario

:NUMBer?

Parameters: none

Response: <NR1>

number of scenarios

Returned values: number of scenarios: integer

Description: This command is to be used in conjunction with :TCAS:SCEN:LIST. This command determines how many scenarios will be returned by the :TCAS:SCEN:LIST? command.

Example: `TCAS:SCEN:NUMB?`

Determine how many tcas scenarios are available in the instrument.

:TCAS

:SCENario

:READ?

Parameters: <NRf>

store number

Valid values: store number: integer. Valid values are 1 to 25. Values outside range are rejected and an error generated.

Response: <DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA>

store data

Returned values: store data: block data

Description: Read the data from the specified store.

Note that the built-in stores can be uploaded to an external controller, but they cannot be downloaded back into the box unless they are written back to one of the user stores.

It is not possible to read the current settings using this command. Save to a user store and then read that store.

Example: `TCAS:SCEN:READ? 7`

`Read store 7.`

Example response: #3208000000010101057800000012C0000FFFFFF380000000001F4006400000009601000000
00000000000000000803000000300030000000000000000000000000000000000000000000
0000002D32303020667420636F6C6C6973696F6E30000000300000000300000346E

:TCAS

:SCENario

:SAVE

Parameters: <NRf>, <STRING PROGRAM DATA>

store number, store name

Valid values: store number: integer. Valid values are 10 to 25. Values outside range are rejected and an error generated.

store name: string. Maximum length of 26 characters excluding quotes. Excess characters will be ignored.

Description: Save scenario into specified store. Any existing data in the store will be overwritten.

Example: `TCAS:SCEN:SAVE 19, "My test"`

Save current tcas setup into scenario store 19.

:TCAS

:SCENario

[:SELect]

Parameters: <NRf>

store number

Valid values: store number: integer. Valid values are 1 to 25. Values outside range are rejected and an error generated.

Description: Select a scenario. The data in the scenario store is copied to current settings. If the store does not contain valid data, an error will be generated.

Store 1 is the last power down store.
Stores 2 to 9 are default stores that are read only (fixed scenarios).
Stores 10 to 25 are user stores.

Example: `TCAS:SCEN 4`

Select store 4.

:TCAS

:SQUitter

Parameters: <BOOLEAN PROGRAM DATA>

squitter state

Description: This sets squitter on or off. When on, squitters will be output at a PRF of 1 Hz.

Example: `TCAS:SQU ON`

Turn squitters on.

:TCAS

:SQUitter?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

squitter state

Description: Determine whether squitters are being sent or not.

Example: `TCAS:SQU?`

:TCAS

:STATionary

Parameters: <BOOLEAN PROGRAM DATA>

stationary state

Description: This allows a toggle between the intruder flying towards/away from the aircraft under test and being stationary (hovering). This can be done while the test is being performed.

Example: `TCAS:STAT ON`

Stop the intruder from moving – it remains at a fixed distance from the aircraft under test.

:TCAS

: STATionary?

Parameters: none

Response: <BOOLEAN RESPONSE DATA>

stationary state

Description: Determine whether the intruder is flying or hovering.

Example: `TCAS:STAT?`

:TCAS

:TYPE

Parameters: <CPD>

tcas type

Valid values: tcas type: [TAS | I | II]. Values outside range are rejected and an error generated.

Description: Select whether we are testing TAS, TCAS I or TCAS II..

Example: `TCAS:TYPE II`

We are testing TCAS II.

:TCAS

:TYPE?

Parameters: None

Response: <CRD>

tcas type

Returned values: tcas type: [TAS | I | II]

Description: Determine whether we are testing TAS, TCAS I or TCAS II.

Example: `TCAS:TYPE?`

:TCAS

:UUT

:ADDRess

:STATe

Parameters: <CPD>

address state

Valid values: address state: [AUTO | MANual]. Values outside range are rejected and an error generated.

Description: Select whether the test set determines the mode S address of the aircraft under test automatically or the value set by TCAS:UUT:ADDR is used..

Example: `TCAS:UUT:ADDR:STAT AUTO`

Determine aircraft under test transponder address automatically.

:TCAS

:UUT

:ADDRess

:STATe?

Parameters: None

Response: <CRD>

address state

Returned values: address state: [AUTO | MAN]

Description: Determine whether the aircraft under test transponder address is user entered or automatically detected.

Example: `TCAS:UUT:ADDR:STAT?`

:TCAS

:UUT

:ADDRess

[:VALue]

Parameters: <NRf>

address

Valid values: address: integer. Valid values are 0 to 16777215. Values outside range are rejected and an error generated.

Description: Set the aircraft under test mode S transponder address for use when not automatically determining the transponder address.

The address can also be entered in hexadecimal using #Hxxxxxx, or in octal using #Qxxxxxxxx, or in binary using #Bxxxxxxxxxxxxxxxxxxxxxxxx.

Example: `TCAS:UUT:ADDR 238467`

Select the aircraft under test address for mode S interrogations.

:TCAS

:UUT

:ADDRess

[:VALue]?

Parameters: None

Response: <NR1>

address

Returned values: address: integer. Values are in the range 0 to 16777215.

Description: Determine the aircraft under test transponder address. Always returns a decimal number.

Example: `TCAS:UUT:ADDR?`

:TCAS

:UUT

:ALTitude

:STATe

Parameters: <CPD>

altitude state

Valid values: altitude state: [AUTO | MANual]. Values outside range are rejected and an error generated.

Description: Select whether the test set determines the altitude of the aircraft under test automatically or the value set by TCAS:UUT:ALT is used..

Example: `TCAS:UUT:ALT:STAT AUTO`

Determine aircraft under test altitude automatically.

:TCAS

:UUT

: ALTitude

:STATe?

Parameters: None

Response: <CRD>

altitude state

Returned values: altitude state: [AUTO | MAN]

Description: Determine whether the aircraft under test altitude is user entered or automatically detected.

Example: `TCAS:UUT:ALT:STAT?`

:TCAS

:UUT

: ALTitude

[:VALue]

Parameters: <NRf>

altitude

Valid values: altitude: integer. Valid values are -1000 to 126700. Values outside range are rejected and an error generated.

Description: Set the aircraft under test altitude for use when not automatically determining the altitude.

Example: `TCAS:UUT:ALT 2100`

Select the aircraft under test altitude.

:TCAS

:UUT

: ALTitude

[:VALue]?

Parameters: None

Response: <NR1>

altitude

Returned values: altitude: integer. Values are in the range -1000 to 126700.

Description: Determine the aircraft under test altitude.

Example: `TCAS:UUT:ALT?`

:TCAS

:WSHout

[:ATTenuation]

Parameters: <NRf>

attenuation

Valid values: attenuation: real

Description: This command sets the receive attenuation during atcrbs whisper-shout testing. The valid range is 0 to 50 dB in steps of 0.5 dB.

Example: `TCAS:WSH 21.5`

Set receive attenuation for whisper-shout test.

:TCAS

:WSHout

[:ATTenuation]?

Parameters: None

Response: <NR2>

attenuation

Returned values: attenuation: real

Description: Determine the receive attenuation for whisper-shout test.

Example: `TCAS:WSH?`

:TCAS

:WSHout

:STATe

Parameters: <BOOLEAN PROGRAM DATA>

whisper-shout state

Description: Select whether the whisper-shout test is on or off.

This can be turned on/off during a tcas test, or it can be set up before starting the test.

Example: `TCAS:WSH:STAT ON`

Perform whisper-shout tcas test.

:TCAS

:WSHout

:STATe?

Parameters: None

Response: <BOOLEAN RESPONSE DATA>

whisper-shout state

Description: Determine whether the whisper-shout tcas test is being performed.

Example: `TCAS:WSH:STAT?`

TIS COMMANDS

TIS Provides a five aircraft static flight simulation, using the Comm A protocol, to test the TIS (Traffic Information Service).

TIS SUBSYSTEM

TIS
- ANTenna
  - [CONNect]\?
  - GAIN\?
  - HEIGht\?
  - RANGe\?
- CLOSs
  - ANTenna
    - MODE\?
    - [VALue]\?
  - [CURRent]\?
  - DIRect\?
- MEASure
  - [DATA]?
  - STARt
  - STOP
- TARGet
  - ALTitude\?
  - ARATe\?
  - BEARing\?
  - HEADing\?
  - NUMBer\?
  - RANGe\?
  - TRAFfic\?
- UUT
  - ADDRess
    - STATe\?
    - [VALue]\?
  - HEADing\?

:TIS

:ANTenna

[:CONNect]

Parameters: <CPD>

antenna connection

Valid values: antenna connection: [DIRect | ANTenna]. Values other than those stated are rejected and an error generated.

Description: Set whether the TIS measurements are to be performed over the air using the test set antenna (ANTenna) or directly via a cable (DIRect).

Does not affect TCAS, DME and transponder measurements.

Example: `TIS:ANT DIRECT`

Make measurements using a direct connection to the aircraft.

:TIS

:ANTenna

[:CONNect]?

Parameters: None

Response: <CRD>

antenna connection

Returned values: antenna connection: [DIR | ANT]

Description: Determine whether measurements are being performed over the air or directly connected via a cable.

Example: `TIS:ANT?`

:TIS

:ANTenna

:GAIN

Parameters: <NRf>, <NRf>

gain1030, gain1090

Valid values: gain1030: real. Valid values are 0.0 to 20.9. Values outside range are rejected and an error generated.
gain1090: real. Valid values are 0.0 to 20.9. Values outside range are rejected and an error generated.

Description: Set the gain of the 6000 antenna at the tis transmit and receive frequencies.

Example: `TIS:ANT:GAIN 9.5, 9.7`

Set gain of the 6000 antenna.

:TIS

:ANTenna

:GAIN?

Parameters: None

Response: <NR2>, <NR2>

gain1030, gain1090

Returned values: gain1030: real. Values are in the range 0.0 to 20.9.
gain1090: real. Values are in the range 0.0 to 20.9.

Description: Determine the gain of the 6000 antenna at the tis transmit and receive frequencies.

Example: `TIS:ANT:GAIN?`

:TIS

:ANTenna

:HEIGht

Parameters: <NRf>

antenna height

Valid values: antenna height: real

Description: Set the height to the tis antenna on the aircraft. Rounded to nearest 0.5 meters or integral feet. In meters mode, height is 0.5 to 30.0 meters. In feet mode, height is 1 to 99 feet.

Example: `TIS:ANT:HEIG 5.5`

Set antenna range to 5.5 meters (assuming range entry mode is meters).

:TIS

:ANTenna

:HEIGht?

Parameters: None

Response: <NR2>

height

Returned values: height: real

Description: Determine the height to the tis antenna on the aircraft.

Example: `TIS:ANT:HEIG?`

:TIS

:ANTenna

:RANGe

Parameters: <NRf>

antenna range

Valid values: antenna range: real

Description: Set the range to the tis antenna on the aircraft. Rounded to nearest 0.5 meters or integral feet. In meters mode, range is 2.0 to 75.0 meters. In feet mode, range is 6 to 250 feet.

Example: `TIS:ANT:RANG 5.5`

Set antenna range to 5.5 meters (assuming range entry mode is meters).

:TIS

:ANTenna

:RANGe?

Parameters: None

Response: <NR2>

range

Returned values: range: real

Description: Determine the range to the tis antenna on the aircraft.

Example: `TIS:ANT:RANG?`

:TIS

:CLOSs

:ANTenna

:MODE

Parameters: <CPD>

ant cable loss mode

Valid values: ant cable loss mode: [UDEFined | L25 | L50 | L75]. Values other than those stated are rejected and an error generated.

Description: Select whether one of the VIAVI supplied cables is used to connect to the antenna (the 6000 has the cable loss programmed into it) or a user cable is used and its cable loss must be entered using TIS:CLOS:ANT.

Example: `TIS:CLOS:ANT:MODE L50`

The user is using a VIAVI supplied 50 ft cable and the 6000 automatically handles its cable loss..

:TIS

:CLOSs

:ANTenna

:MODE?

Parameters: none

Response: <CRD>

ant cable loss mode

Returned values: ant cable loss mode: [UDEF | L25 | L50 | L75]

Description: Determine whether we are using a VIAVI supplied cable or a user defined cable to connect to the antenna.

Example: `TIS:CLOS:ANT:MODE?`

:TIS

:CLOSs

:ANTenna

[:VALue]

Parameters: <NRf>

cable loss

Valid values: cable loss: real

Description: This command sets the cable loss (in dB) of the cable used to connect to the test set antenna. This command will always set the cable loss of the antenna cable even if the current selection is direct connect. This value is only used if TIS:CLOS:ANT:MODE is UDEF.

The tis cable loss value is kept separate from the tcas, DME and transponder cable loss values.

The value is in units of dB.

Example: `TIS:CLOS:ANT 1.7`

Inform the instrument of the loss of the cable connected to the 6000 antenna.

:TIS

:CLOSs

:ANTenna

[:VALue]?

Parameters: None

Response: <NR2>

cable loss

Returned values: cable loss: real

Description: Determine the loss of the antenna cable. This is the value used when TIS:CLOS:ANT:MODE is set to UDEF.

Example: `TIS:CLOS:ANT?`

:TIS

:CLOSs

[:CURRent]

Parameters: <NRf>

cable loss

Valid values: cable loss: real

Description: This command sets the cable loss (in dB) of the cable used to direct connect to the aircraft antenna or the cable used to connect to the test set antenna if performing over the air measurements.

The tis cable loss value is kept separate from the tcas, DME and transponder cable loss values.

The value is in units of dB.

Example: `TIS:CLOS 1.7`

Inform the instrument of the loss of the cable currently being used.

:TIS

:CLOSs

[:CURRent]?

Parameters: None

Response: <NR2>

cable loss

Returned values: cable loss: real

Description: Determine the loss of the cable currently being used.

Example: `TIS:CLOS?`

:TIS

:CLOSs

:DIRect

Parameters: <NRf>

cable loss

Valid values: cable loss: real

Description: This command sets the cable loss (in dB) of the cable used to directly connect from the test set to the transponder under test. This command will always set the cable loss of the direct connect cable even if the current selection is antenna (over the air).

The tis cable loss value is kept separate from the tcas, DME and transponder cable loss values.

The value is in units of dB.

Example: `TIS:CLOS:DIR 1.7`

Inform the instrument of the loss of the direct connect cable.

:TIS

:CLOSs

:DIRect?

Parameters: None

Response: <NR2>

cable loss

Returned values: cable loss: real

Description: Determine the loss of the direct connect cable.

Example: `TIS:CLOS:DIR?`

:TIS

:MEASure

[:DATA]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <STRING RESPONSE DATA>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <CRD>, <CRD>, <NR1>

overall state, addr state, addr, tail no state, tail no, alt state, alt, TSCR state, TSCR, TSDR state, TSDR, status state, status, info state, info

Returned values: overall state: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

addr state: [PASS | FAIL | INV | NDAT | NAV]

addr: integer

tail no state: [PASS | FAIL | INV | NDAT | NAV]

tail no: string. Maximum length of 6 characters excluding quotes.

alt state: [PASS | FAIL | INV | NDAT | NAV]

alt: integer

TSCR state: [PASS | FAIL | INV | NDAT | NAV]

TSCR: integer

TSDR state: [PASS | FAIL | INV | NDAT | NAV]

TSDR: integer

status state: [PASS | FAIL | INV | NDAT | NAV]

status: [ENAB | DIS | CONN]

info state: [PASS | FAIL | INV | NDAT | NAV]

info: integer

Description: Read back the measured TIS items.

range is in nautical miles.

altitude is in feet.

Example: `TIS:MEAS?`

:TIS

:MEASure

:STARt

Parameters: None

Description: This starts tis measurements.

Example: `TIS:MEAS:STAR`

Start tis measurements.

:TIS

:MEASure

:STOP

Parameters: None

Description: This stops tis measurements.

Example: `TIS:MEAS:STOP`

Stop tis measurements.

:TIS

:TARGet

:ALTitude

Parameters: <NRf>, <CPD>, <NRf>

target, direction, altitude

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

direction: [ABOVe | BELow | NALTitude]. Values other than those stated are rejected and an error generated.

altitude: integer

Description: Set the altitude of specified target aircraft relative to the aircraft under test.

The entered value is actually a set of ranges as defined below. Any altitude within the specified range will select that range. To indicate that the target aircraft is not reporting altitude, enter `NALT, 0`.

```
For target aircraft above aircraft under test:
         0 ft <= rel alt <  100 ft     ABOV, 0  ..  ABOV, 99
       100 ft <= rel alt <  200 ft     ABOV, 100  ..  ABOV, 
199
       200 ft <= rel alt <  300 ft     ABOV, 200  ..  ABOV, 
299
       300 ft <= rel alt <  400 ft     ABOV, 300  ..  ABOV, 
399
       400 ft <= rel alt <  500 ft     ABOV, 400  ..  ABOV, 
499
       500 ft <= rel alt <  600 ft     ABOV, 500  ..  ABOV, 
599
       600 ft <= rel alt <  700 ft     ABOV, 600  ..  ABOV, 
699
       700 ft <= rel alt <  800 ft     ABOV, 700  ..  ABOV, 
799
       800 ft <= rel alt <  900 ft     ABOV, 800  ..  ABOV, 
899
       900 ft <= rel alt < 1000 ft     ABOV, 900  ..  ABOV, 
999
      1000 ft <= rel alt < 1500 ft     ABOV, 1000  ..  ABOV, 
1499
      1500 ft <= rel alt < 2000 ft     ABOV, 1500  ..  ABOV, 
1999
      2000 ft <= rel alt < 2500 ft     ABOV, 2000  ..  ABOV, 
2499
      2500 ft <= rel alt < 3000 ft     ABOV, 2500  ..  ABOV, 
2999
      3000 ft <= rel alt < 3500 ft     ABOV, 3000  ..  ABOV, 
```

:TIS

:TARGet

:ALTitude (cont)

Description:

```
3499
       rel alt >= 3500 ft         ABOV, 3500

For target aircraft below aircraft under test:
          0 ft <  rel alt <  100 ft      BEL, 0  ..  BEL, 99
        100 ft <= rel alt <  200 ft      BEL, 100  ..  BEL,
199
        200 ft <= rel alt <  300 ft      BEL, 200  ..  BEL,
299
        300 ft <= rel alt <  400 ft      BEL, 300  ..  BEL,
399
        400 ft <= rel alt <  500 ft      BEL, 400  ..  BEL,
499
        500 ft <= rel alt <  600 ft      BEL, 500  ..  BEL,
599
        600 ft <= rel alt <  700 ft      BEL, 600  ..  BEL,
699
        700 ft <= rel alt <  800 ft      BEL, 700  ..  BEL,
799
        800 ft <= rel alt <  900 ft      BEL, 800  ..  BEL,
899
        900 ft <= rel alt < 1000 ft      BEL, 900  ..  BEL,
999
       1000 ft <= rel alt < 1500 ft      BEL, 1000  ..  BEL,
1499
       1500 ft <= rel alt < 2000 ft      BEL, 1500  ..  BEL,
1999
       2000 ft <= rel alt < 2500 ft      BEL, 2000  ..  BEL,
2499
       2500 ft <= rel alt < 3000 ft      BEL, 2500  ..  BEL,
2999
       rel alt >= 3000 ft         BEL, 3000
```

Example: `TIS:TARG:ALT 5, BEL, 2200`

Set relative altitude of target aircraft 5 to the range 2000..2499 ft below aircraft under test.

:TIS

:TARGet

:ALTitude?

Parameters: <NRf>

target

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

Response: <CRD>, <NR1>

direction, altitude

Returned values: direction: [ABOV | BEL | NALT]

altitude: integer. Values are in the range 0 to 3500.

Description: Determine the relative altitude of specified target aircraft.

The altitude is a range of values, a single value is returned that indicates which range the altitude is in:

```
For target aircraft above aircraft under test:
         0 ft <= rel alt <  100 ft            ABOV, 0
       100 ft <= rel alt <  200 ft            ABOV, 100
       200 ft <= rel alt <  300 ft            ABOV, 200
       300 ft <= rel alt <  400 ft            ABOV, 300
       400 ft <= rel alt <  500 ft            ABOV, 400
       500 ft <= rel alt <  600 ft            ABOV, 500
       600 ft <= rel alt <  700 ft            ABOV, 600
       700 ft <= rel alt <  800 ft            ABOV, 700
       800 ft <= rel alt <  900 ft            ABOV, 800
       900 ft <= rel alt < 1000 ft            ABOV, 900
      1000 ft <= rel alt < 1500 ft            ABOV, 1000
      1500 ft <= rel alt < 2000 ft            ABOV, 1500
      2000 ft <= rel alt < 2500 ft            ABOV, 2000
      2500 ft <= rel alt < 3000 ft            ABOV, 2500
      3000 ft <= rel alt < 3500 ft            ABOV, 3000
      rel alt >= 3500 ft               ABOV, 3500
```

TIS

:TARGet

:ALTitude? (cont)

Description:

```
For target aircraft below aircraft under test:
       0 ft <  rel alt <  100 ft            BEL, 1
     100 ft <= rel alt <  200 ft            BEL, 100
     200 ft <= rel alt <  300 ft            BEL, 200
     300 ft <= rel alt <  400 ft            BEL, 300
     400 ft <= rel alt <  500 ft            BEL, 400
     500 ft <= rel alt <  600 ft            BEL, 500
     600 ft <= rel alt <  700 ft            BEL, 600
     700 ft <= rel alt <  800 ft            BEL, 700
     800 ft <= rel alt <  900 ft            BEL, 800
     900 ft <= rel alt < 1000 ft            BEL, 900
    1000 ft <= rel alt < 1500 ft            BEL, 1000
    1500 ft <= rel alt < 2000 ft            BEL, 1500
    2000 ft <= rel alt < 2500 ft            BEL, 2000
    2500 ft <= rel alt < 3000 ft            BEL, 2500
    rel alt >= 3000 ft               BEL, 3000

For target aircraft not indicating altitude:
    NALT, 0
```

Example: `TIS:TARG:ALT? 2`

:TIS

:TARGet

:ARATe

Parameters: <NRf>, <CPD>

target, alt rate

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

alt rate: [UNUSed | CLIMb | DESCend | LEVel]. Values other than those stated are rejected and an error generated.

Description: Set the altitude rate field for specified target aircraft.

Example: `TIS:TARG:ARAT 5, DESC`

Set the altitude rate to descend for target aircraft 5.

:TIS

:TARGet

:ARATe?

Parameters: <NRf>

target

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

Response: <CRD>

alt rate

Returned values: alt rate: [UNUS | CLIM | DESC | LEV]

Description: Determine the altitude rate for specified target aircraft.

Example: `TIS:TARG:ARAT? 4`

:TIS

:TARGet

:BEARing

Parameters: <NRf>, <NRf>

target, bearing

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

bearing: integer. Valid values are 0 to 354. Values outside range are rejected and an error generated.

Description: Set the bearing of specified target aircraft. The value can be set in steps of 6 degrees. The entered value will be rounded to the nearest valid value.

Example: `TIS:TARG:BEAR 3, 100`

Set bearing of target aircraft 3 to 102 degrees (nearest valid value to 100).

:TIS

:TARGet

:BEARing?

Parameters: <NRf>

target

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

Response: <NR1>

bearing

Returned values: bearing: integer. Values are in the range 0 to 354.

Description: Determine the beraring of specified target aircraft.

Example: `TIS:TARG:BEAR? 5`

:TIS

:TARGet

:HEADing

Parameters: <NRf>, <NRf>

target, heading

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

heading: integer. Valid values are 0 to 315. Values outside range are rejected and an error generated.

Description: Set the heading of specified target aircraft. The value can be set in steps of 45 degrees. The entered value will be rounded to the nearest valid value.

Example: `TIS:TARG:HEAD 1, 100`

Set heading of target aircraft 1 to 90 degrees (nearest valid value to 100).

:TIS

:TARGet

:HEADing?

Parameters: <NRf>

target

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

Response: <NR1>

heading

Returned values: heading: integer. Values are in the range 0 to 315.

Description: Determine the heading of specified target aircraft.

Example: `TIS:TARG:HEAD? 4`

:TIS

:TARGet

:NUMBer

Parameters: <NRf>

no of targets

Valid values: no of targets: integer. Valid values are 0 to 5. Values outside range are rejected and an error generated.

Description: Set the number of target aircraft to be simulated.

Example: `TIS:TARG:NUMB 3`

Simulate 3 target aircraft.

:TIS

:TARGet

:NUMBer?

Parameters: none

Response: <NR1>

no of targets

Returned values: no of targets: integer. Values are in the range 0 to 5.

Description: Determine the number of simulated target aircraft.

Example: `TIS:TARG:NUMB?`

:TIS

:TARGet

:RANGe

Parameters: <NRf>, <NRf>

target, range

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

range: real. Valid values are 0.000 to 7.500. Values outside range are rejected and an error generated.

Description: Set the range of specified target aircraft. The entered value will be rounded to the nearest valid value. A range of 7.500 means greater than 7.000. Units of nautical miles.

Valid values are:
0, 0.125, 0.375, 0.625, 0.875, 1.125, 1.375, 1.625, 1.875, 2.25, 2.75, 3.5, 4.5, 5.5, 6.5, 7.5.

Example: `TIS:TARG:RANG 5, 1.8`

Set range of target aircraft 5 to 1.875 nm (nearest valid value to 1.8).

TIS

:TARGet

:RANGe?

Parameters: <NRf>

target

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

Response: <NR2>

range

Returned values: range: real. Values are in the range 0 to 7.5.

Description: Determine the range of specified target aircraft.

Example: `TIS:TARG:RANG? 1`

:TIS

:TARGet

:TRAFfic

Parameters: <NRf>, <CPD>

target, traffic

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

traffic: [PROXimity | TRAFfic]. Values other than those stated are rejected and an error generated.

Description: Set the traffic field for specified target aircraft.

Example: `TIS:TARG:TRAF 4, TRAF`

Set the traffic field to traffic for target aircraft 4.

:TIS

:TARGet

:TRAFfic?

Parameters: <NRf>

target

Valid values: target: integer. Valid values are 1 to 5. Values outside range are rejected and an error generated.

Response: <CRD>

traffic

Returned values: traffic: [PROX | TRAF]

Description: Determine the traffic field for specified target aircraft.

Example: `TIS:TARG:TRAF? 2`

:TIS

:UUT

:ADDRess

:STATe

Parameters: <CPD>

address state

Valid values: address state: [AUTO | MANual]. Values outside range are rejected and an error generated.

Description: Select whether the test set determines the mode S address of the aircraft under test automatically or the value set by TIS:UUT:ADDR is used..

Example: `TIS:UUT:ADDR:STAT AUTO`

Determine aircraft under test transponder address automatically.

:TIS

:UUT

:ADDRess

:STATe?

Parameters: None

Response: <CRD>

address state

Returned values: address state: [AUTO | MAN]

Description: Determine whether the aircraft under test transponder address is user entered or automatically detected.

Example: `TIS:UUT:ADDR:STAT?`

:TIS

:UUT

:ADDRess

[:VALue]

Parameters: <NRf>

address

Valid values: address: integer. Valid values are 0 to 16777215. Values outside range are rejected and an error generated.

Description: Set the aircraft under test mode S transponder address for use when not automatically determining the transponder address.

The address can also be entered in hexadecimal using #Hxxxxxx, or in octal using #Qxxxxxxxx, or in binary using #Bxxxxxxxxxxxxxxxxxxxxxxxx.

Example: `TIS:UUT:ADDR 238467`

Select the aircraft under test address for mode S interrogations.

:TIS

:UUT

:ADDRess

[:VALue]?

Parameters: None

Response: <NR1>

address

Returned values: address: integer. Values are in the range 0 to 16777215.

Description: Determine the aircraft under test transponder address. Always returns a decimal number.

Example: `TIS:UUT:ADDR?`

:TIS

:UUT

:HEADing

Parameters: <NRf>

heading

Valid values: heading: integer. Valid values are 0 to 354. Values outside range are rejected and an error generated.

Description: Set the heading of the aircraft under test. The value can be set in steps of 6 degrees. The entered value will be rounded to the nearest valid value.

Example: `TIS:UUT:HEAD 100`

Set heading of aircraft under test to 102 degrees (nearest valid value to 100).

:TIS

:UUT

:HEADing?

Parameters: none

Response: <NR1>

heading

Returned values: heading: integer. Values are in the range 0 to 354.

Description: Determine the heading of the aircraft under test.

Example: `TIS:UUT:HEAD?`

THIS PAGE INTENTIONALLY LEFT BLANK.

AENCODER COMMANDS

The Aencoder Mode allows testing of an altitude encoder. Testing may be done by directly connecting the instrument via a Breakout Box, or measured via a transponder. Refer to the 6000 Operations Manual, Appendix E for information regarding the Breakout Box.

AENCODER SUBSYSTEM

```
AENCoder
        [MEASure]
                [AENCoder]?
                XPDR?
        SELect\?
        STARt
        STOP
```

:AENCoder

[:MEASure]

[:AENCoder]?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>

overall status, raw data status, raw data, altitude status, altitude

Returned values: overall status: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

raw data status: [PASS | FAIL | INV | NDAT | NAV]

raw data: integer. Values are in the range 0 to 4094.

altitude status: [PASS | FAIL | INV | NDAT | NAV]

altitude: integer. Values are in the range -1000 to 126700.

Description: Read back the value from the altitude encoder.

The raw data will be 2 bytes (16 bits) in the following format:
$0_{(msb)}$ 0 0 0 A4 A2 A1 B4 B2 B1 C4 C2 C1 D4 D2 $0_{(lsb)}$.
This matches the format returned by transponder diagnostics.

If altitude valid is PASS then the altitude value will be a measurement in feet otherwise the returned value should be discarded.

Example: `AENC?`

Example response: PASS,PASS,8,PASS,-800

Only bit C1 is set, corresponding to an altitude of –800 ft.

:AENCoder

[:MEASure]

:XPDR?

Parameters: None

Response: <CRD>, <CRD>, <NR1>, <CRD>, <NR1>, <CRD>, <NR1>

overall status, raw data status, raw data, altitude status, altitude, id status, id

Returned values: overall status: [NRUN | NREP | PASS | WARN | FAIL | NCAP | ERR]

raw data status: [PASS | FAIL | INV | NDAT | NAV]

raw data: integer. Values are in the range 0 to 4094.

altitude status: [PASS | FAIL | INV | NDAT | NAV]

altitude: integer. Values are in the range -1000 to 126700.

id status: [PASS | FAIL | INV | NDAT | NAV]

id: integer. Values are in the range 0 to 4095.

Description: Read back the value from the altitude encoder. This command is intended to be used if the altitude is being measured via the tranponder. This returns the id from the tranponder along with the altitude data.

The raw data will be 2 bytes (16 bits) in the following format:

$0_{(msb)}$ 0 0 0   A4 A2 A1 B4   B2 B1 C4 C2   C1 D4 D2 $0_{(lsb)}$.

This matches the format returned by transponder diagnostics.

If altitude valid is PASS then the altitude value will be a measurement in feet otherwise the returned value should be discarded.

Example: `AENC:XPDR?`

Example response: PASS,PASS,8,PASS,-800,PASS,2216

Only bit C1 is set, corresponding to an altitude of –800 ft. The transponder id is 2216 (4250 in octal).

:AENCoder

:SELect

Parameters: <CPD>

connection

Valid values: connection: [AENCoder | XPDR]. Values other than those stated are rejected and an error generated.

Description: Set whether the altitude is to be read directly from the altitude encoder (attached to the 6000) or via a transponder.

Example: `AENC:SEL AENC`

An altitude encoder is directly connected to the test set.

:AENCoder

:SELect?

Parameters: none

Response: <CRD>

connection

Returned values: connection: [AENC | XPDR]

Description: Determine whether altitude encoder is directly connected, or the measurement will be performed via a transponder.

Example: `AENC:SEL?`

:AENCoder

:STARt

Parameters: None

Description: This starts the altitude encoder test. Selection of whether we are directly connected to an altitude encoder or we are measuring via a transponder should be done using AENC:SEL before starting the test.

Example: `AENC:STAR`

Start altitude encoder test.

:AENCoder

:STOP

Parameters: None

Description: This stops the current altitude encoder test.

Example: `AENC:STOP`

Stop altitude encoder test.

THIS PAGE INTENTIONALLY LEFT BLANK.

DISPLAY COMMANDS

Display Commands set the contrast settings and backlight brightness on the display panel.

DISPLAY SUBSYSTEM

DISPlay

BACKlight\?

CONTrast\?

:DISPLay

:BACKlight

Parameters: <NRf>

backlight

Valid values: backlight: integer. Valid values are 1 to 99. Values outside range are rejected and an error generated.

Description: Set the LCD backlight brightness.

Example: `DISP:BACK 99`

Set the LCD backlight brighness to maximum

:DISPlay

:BACKlight?

Parameters: none

Response: <NR1>

backlight

Returned values: backlight: integer. Values are in the range 1 to 99.

Description: Determine the current LCD backlight brightness setting.

Example: `DISP:BACK?`

:DISPLay

:CONTrast

Parameters: <NRf>

contrast

Valid values: contrast: integer. Valid values are 1 to 99. Values outside range are rejected and an error generated.

Description: Set the LCD contrast.

Example: `DISP:CONT 29`

Set the LCD contrast

:DISPlay

:CONTrast?

Parameters: none

Response: <NR1>

contrast

Returned values: contrast: integer. Values are in the range 1 to 99.

Description: Determine the current LCD contrast.

Example: `DISP:CONT?`

THIS PAGE INTENTIONALLY LEFT BLANK.

STATUS COMMANDS

The Status Mode is used to check the instrument's operational status and to determine when new measurement data is available.

STATUS SUBSYSTEM

STATus
- OPERation
  - CONDition?
  - ENABle\?
  - [EVENt]?
  - NTRansition\?
  - PTRansition\?
- PRESet
- QUEStionable
  - CONDition?
  - ENABle\?
  - [EVENt]?
  - NTRansition\?
  - PTRansition\?

:STATus

:OPERation

:CONDition?

Parameters: none

Response: <NR1>

register contents

Returned values: register contents: integer. Values are in the range 0 to 32767.

Description: Read the contents of the Operation Status Condition Register. This register returns the current state of the instrument. Reading the register does not affect its contents. This is a sixteen bit register.

Refer to Appendix A for the meaning of each bit.

Example: `STAT:OPER:COND?`

:STATus

:OPERation

:ENABle

Parameters: <NRf>

mask

Valid values: mask: integer. Valid values are 0 to 65535. Values outside range are rejected and an error generated.

Description: Sets the enable mask which allows true conditions in the Operation Status Event Register to be reported in the summary bit (bit 7 in the Status Byte Register).

If a bit is 1 in the Operation Status Enable Register and its associated event bit (in the Operation Status Event Register) makes a transition to true, a positive transition will occur in the associated summary bit if that bit was previously 0.

Bit 15 of the mask value supplied is ignored since bit 15 of the Operation Status Enable Register is always zero. This is a sixteen bit register.

Refer to Appendix A for the meaning of each bit.

Example: `STAT:OPER:ENAB 8`

Program the mask associated with the Operation Status Event Register with the value 8 (0000 0000 0000 1000 in binary) to enable a positive transition in the summary bit when the instrument has new data available to be read.

:STATus

:OPERation

:ENABle?

Parameters: none

Response: <NR1>

mask

Returned values: mask: integer. Values are in the range 0 to 32767.

Description: Read the mask from the Operation Status Enable Register. This is a sixteen bit register.

Example: `STAT:OPER:ENAB?`

:STATus

:OPERation

[:EVENt]?

Parameters: none

Response: <NR1>

event register contents

Returned values: event register contents: integer. Values are in the range 0 to 32767.

Description: Read the contents of the Operation Status Event Register. Reading the register will clear it. This is a sixteen bit register.

Refer to Appendix A for the meaning of each bit.

Example: `STAT:OPER?`

:STATus

:OPERation

:NTRansition

Parameters: <NRf>

negative transition mask

Valid values: negative transition mask: integer. Valid values are 0 to 65535. Values outside range are rejected and an error generated.

Description: Sets the negative transition filter which allows transitions from 1 to 0 in the Operation Status Condition Register to be latched into the Operation Status Event Register.

Bit 15 of the mask value supplied is ignored since bit 15 of the Operation Status Negative Transition Filter Register is always zero. This is a sixteen bit register.

Refer to Appendix A for the meaning of each bit.

Example: `STAT:OPER:NTR 16`

Program the negative transition filter associated with the Operation Status Register with the value 16 (0000 0000 0001 0000 in binary) to enable the event register to be set when the instrument has finished testing.

:STATus

:OPERation

:NTRansition?

Parameters: none

Response: <NR1>

negative transition mask

Returned values: negative transition mask: integer. Values are in the range 0 to 32767.

Description: Read the mask from the Operation Status Negative Transition Filter Register. This is a sixteen bit register.

Example: `STAT:OPER:NTR?`

:STATus

:OPERation

:PTRansition

Parameters: <NRf>

positive transition mask

Valid values: positive transition mask: integer. Valid values are 0 to 65535. Values outside range are rejected and an error generated.

Description: Sets the positive transition filter which allows transitions from 0 to 1 in the Operation Status Condition Register to be latched into the Operation Status Event Register.

Bit 15 of the mask value supplied is ignored since bit 15 of the Operation Status Positive Transition Filter Register is always zero. This is a sixteen bit register.

Refer to Appendix A for the meaning of each bit.

Example: `STAT:OPER:PTR 16`

Program the positive transition filter associated with the Operation Status Register with the value 16 (0000 0000 0001 0000 in binary) to enable the event register to be set when the instrument starts a test. There are currently no bits in the register that would be useful to wait on a positive transition.

:STATus

:OPERation

:PTRansition?

Parameters: none

Response: <NR1>

positive transition mask

Returned values: positive transition mask: integer. Values are in the range 0 to 32767.

Description: Read the mask from the Operation Status Positive Transition Filter Register. This is a sixteen bit register.

Example: `STAT:OPER:PTR?`

:STATus

:PRESet

Parameters: none

Description: Preset the Operation Status Enable Register and the Questionable Status Enable Register to zero.

Also defaults the Operation status transition filters and the Questionable status transition fileters.

Refer to Appendix A for the meaning of each bit.

Example: `STAT:PRES`

:STATus

:QUEStionable

:CONDition?

Parameters: none

Response: <NR1>

register contents

Returned values: register contents: integer. Values are in the range 0 to 32767.

Description: Read the contents of the Questionable Status Condition Register. This register returns the current state of the instrument. Reading the register does not affect its contents. This is a sixteen bit register.

Refer to Appendix A for the meaning of each bit.

Example: `STAT:QUES:COND?`

:STATus

:QUEStionable

:ENABle

Parameters: <NRf>

mask

Valid values: mask: integer. Valid values are 0 to 65535. Values outside range are rejected and an error generated.

Description: Sets the enable mask which allows true conditions in the Questionable Status Event Register to be reported in the summary bit (bit 3 in the Status Byte Register).

If a bit is 1 in the Questionable Status Enable Register and its associated event bit (in the Questionable Status Event Register) makes a transition to true, a positive transition will occur in the associated summary bit if that bit was previously 0.

Bit 15 of the mask value supplied is ignored since bit 15 of the Questionable Status Enable Register is always zero. This is a sixteen bit register.

Refer to Appendix A for the meaning of each bit.

Example: `STAT:QUES:ENAB 8`

No bits are yet defined in the questionable status register, so there is currently no useful example to give.

:STATus

:QUEStionable

:ENABle?

Parameters: none

Response: <NR1>

mask

Returned values: mask: integer. Values are in the range 0 to 32767.

Description: Read the mask from the Questionable Status Enable Register. This is a sixteen bit register.

Example: `STAT:QUES:ENAB?`

:STATus

:QUEStionable

[:EVENt]?

Parameters: none

Response: <NR1>

event register contents

Returned values: event register contents: integer. Values are in the range 0 to 32767.

Description: Read the contents of the Questionable Status Event Register. Reading the register will clear it. This is a sixteen bit register.

Refer to Appendix A for the meaning of each bit.

Example: `STAT:QUES?`

:STATus

:QUEStionable

:NTRansition

Parameters: <NRf>

negative transition mask

Valid values: negative transition mask: integer. Valid values are 0 to 65535. Values outside range are rejected and an error generated.

Description: Sets the negative transition filter which allows transitions from 1 to 0 in the Questionable Status Condition Register to be latched into the Questionable Status Event Register.

Bit 15 of the mask value supplied is ignored since bit 15 of the Questionable Status Negative Transition Filter Register is always zero. This is a sixteen bit register.

Refer to Appendix A for the meaning of each bit.

Example: `STAT:QUES:NTR 256`

No bits are yet defined in the questionable status register, so there is currently no useful example to give.

:STATus

:QUEStionable

:NTRansition?

Parameters: none

Response: <NR1>

negative transition mask

Returned values: negative transition mask: integer. Values are in the range 0 to 32767.

Description: Read the mask from the Questionable Status Negative Transition Filter Register. This is a sixteen bit register.

Example: `STAT:QUES:NTR?`

:STATus

:QUEStionable

:PTRansition

Parameters: <NRf>

positive transition mask

Valid values: positive transition mask: integer. Valid values are 0 to 65535. Values outside range are rejected and an error generated.

Description: Sets the positive transition filter which allows transitions from 0 to 1 in the Questionable Status Condition Register to be latched into the Questionable Status Event Register.

Bit 15 of the mask value supplied is ignored since bit 15 of the Questionable Status Positive Transition Filter Register is always zero. This is a sixteen bit register.

Refer to Appendix A for the meaning of each bit.

Example: `STAT:QUES:PTR 32`

No bits are yet defined in the questionable status register, so there is currently no useful example to give.

:STATus

:QUEStionable

:PTRansition?

Parameters: none

Response: <NR1>

positive transition mask

Returned values: positive transition mask: integer. Values are in the range 0 to 32767.

Description: Read the mask from the Questionable Status Positive Transition Filter Register. This is a sixteen bit register.

Example: `STAT:QUES:PTR?`

THIS PAGE INTENTIONALLY LEFT BLANK.

SYSTEM COMMANDS

System Commands are miscellaneous and not specific to operating modes.

SYSTEM SUBSYSTEM

SYSTem
- ANTenna
  - [CONNect]\?
- BATTery?
- CONTroller\?
- DATE\?
- ERRor
  - [NEXT]?
- OPTions?
- PDOWn\?
- SERial
  - BAUD\?
  - FCONtrol\?
- TEMPerature?
- TEST
  - PMEMory?
  - READ?
  - [RUN]?
- TIME\?
- UNITs
  - DISTance\?
  - POWer\?
- VERSion?

:SYSTem

:ANTenna

[:CONNect]

Parameters: <CPD>

antenna connection

Valid values: antenna connection: [DIRect | ANTenna]. Values other than those stated are rejected and an error generated.

Description: Set whether the measurements are to be performed over the air using the test set antenna (ANTenna) or directly via a cable (DIRect).

Applies to DME and transponder measurements.

Example: `SYST:ANT DIRECT`

Make measurements using a direct connection to the aircraft antenna.

:SYSTem

:ANTenna

[:CONNect]?

Parameters: None

Response: <CRD>

antenna connection

Returned values: antenna connection: [DIR | ANT]

Description: Determine whether measurements are being performed over the air or directly connected via a cable.

Example: `SYST:ANT?`

:SYSTem

:BATTery?

Parameters: none

Response: <CRD>, <NR1>

charge state, life

Returned values: charge state: [CHAR | DISC]

life: real

Description: Determine whether the battery is being charged or is discharging. If discharging, also returns the battery life in 6 minute increments (0.1 hour) as hours with one decimal place.

Example: `SYST:BATT?`

:SYSTem

:CONTroller

Parameters: <CPD>

controller

Valid values: controller: [SERial | NONE]. Values other than those stated are rejected and an error generated.

Description: This command sets how the instrument is being controlled. It can be controlled by a controller over RS232. It is also possible to select no controller.

If no controller is selected then the instrument cannot be controlled remotely. The RS232 is available to send test results out to from front panel operation.

If serial controller is selected then the instrument will accept remote commands from the RS232. The RS232 is not available to send test results out to from front panel operation.

Example: `SYST:CONT SER`

Select the RS232 for use by an external controller.

:SYSTem

:CONTroller?

Parameters: none

Response: <CRD>

controller

Returned values: controller: [SER | NONE]

Description: Determine whether the instrument is accepting external control and if so, where from.

Example: `SYST:CONT?`

:SYSTem

:DATE

Parameters: <NRf>, <NRf>, <NRf>

month, date, year

Valid values: month: integer. Valid values are 1 to 12. Values outside range are rejected and an error generated.

date: integer. Valid values are 1 to 31. Values outside range are rejected and an error generated.

year: integer. Valid values are 0 to 99. Values outside range are rejected and an error generated.

Description: Set the date.

The date is stored in the RTC chip. Data will only be lost if the Nvram battery dies

Example: `SYST:DATE 1, 21, 5`

:SYSTem

:DATE?

Parameters: None

Response: <NR1>, <NR1>, <NR1>

month, date, year

Returned values: month: integer. Values are in the range 1 to 12.

date: integer. Values are in the range 1 to 31.

year: integer. Values are in the range 0 to 99.

Description: Determine the date – read from the RTC chip.

Example: `SYST:DATE?`

:SYSTem

:ERRor

[:NEXT]?

Parameters: none

Response: <NR1>,<STRING RESPONSE DATA>

error number, error message string

Returned values: error number: integer
error message string: string

Description: Read the SCPI error number and error message from the head of the error queue.

Example: `SYST:ERR?`

Example Response: `-112,"Program mnemonic too long"`

:SYSTem

:OPTions?

Parameters: none

Response: <NR1>

options

Returned values: options: integer. Values are in the range 0 to 32767.

The returned value is a sixteen bit value *with each set bit representing the presence of an option:*

Bit Number	Option Present If Bit Set
0	MODE S
1	TCAS
2	ADS-B
3	Reserved
4	SPARE
5	SPARE
6	SPARE
7	SPARE
8	SPARE
9	SPARE
10	SPARE
11	SPARE
12	SPARE
13	SPARE
14	SPARE
15	Always zero

This command can be used instead of *OPT? if it is easier to decode than the text strings returned by *OPT?.

Description: Read hardware options present.

Example: `SYST:OPT?`

:SYSTem

:PDOWn

Parameters: <NRf>

powerdown timeout

Valid values: powerdown timeout: integer. Valid values are 0 to 20. Values outside range are rejected and an error generated.

Description: Set the timeout period. A value of 0 will turn off the auto-turnoff feature. Any other value will be clipped to the valid range of 5 to 20 and will cause the instrument to switch off after that many minutes of non operation (no keypress or remote command).

Example: `SYST:PDOW 10`

:SYSTem

:PDOWn?

Parameters: none

Response: <NR1>

powerdown timeout

Returned values: powerdown timeout: integer. Values are in the range 0 to 20.

Description: Determine the auto power-off timeout value.

Example: `SYST:PDOW?`

:SYSTem

:SERial

:BAUD

Parameters: <NRf>

baud rate

Valid values: baud rate: integer. Valid values are 9600 to 115200. Values outside range are clipped.

Description: Set the serial interface baud rate. The same rate is used for transmission and reception of data. The following values are valid and the nearest will be used:
9600, 19200, 38400, 57600, 115200

If this command is being sent over the RS232 port then it is recommended that it is the last command in the program message. A delay should be inserted before sending any further commands at the new baud rate.

Example: `SYST:SER:BAUD 19200`

Set RS232 serial interface baud rate to 19200.

:SYSTem

:SERial

:BAUD?

Parameters: none

Response: <NR1>

baud rate

Returned values: baud rate: integer

Description: Determine the serial interface baud rate.

Example: `SYST:SER:BAUD?`

:SYSTem

:SERial

:FCONtrol

Parameters: <CPD>

flow control method

Valid values: flow control method: [NONE | XON | HARDware]. Values other than those stated are rejected and an error generated.

Description: Set the flow control method for the RS232 serial port:

NONE	No flow control method in use. Data can be lost if the receiving device is slower than the transmitting device.
XON	Use software handshaking (XON and XOFF).
HARDware	Use hardware handshaking (RTS and CTS).

Note that to use hardware handshaking, the cable in use must contain the correct wires to support the hardware handshaking method.
Both handshaking methods will only work if both devices connected are using the specified method.

Example: `SYST:SER:FCON HARD`

Set RS232 serial interface to use hardware handshaking.

:SYSTem

:SERial

:FCONtrol?

Parameters: none

Response: <CRD>

flow control method

Returned values: flow control method: [NONE | XON | HARD]

Description: Determine the serial interface flow control method.

Example: `SYST:SER:FCON?`

:SYSTem

:TEMPerature?

Parameters: none

Response: <NR1>, <NR1>

ambient temperature, attenuator temperature

Returned values: ambient temperature: integer

attenuator temperature: integer

Description: Read the two temperature sensors in the box and return the results. Both values are in degrees centigrade.

Example: `SYST:TEMP?`

:SYSTem

:TEST

:PMEMory?

Parameters: none

Response: <CRD>, <NR1>, <NR1>, <NR1><NR1>, <CRD>, <NR1>

ppc ram test status, ppc ram error, ppc ram data upper, ppc ram data lower, ppc ram address, ppc flash test status, ppc flash error

Returned values: ppc ram test status: [NRUN | PASS | FAIL | ERR]
ppc ram error: integer
ppc ram data upper: integer
ppc ram data lower: integer
ppc ram address: integer
ppc flash test status: [NRUN | PASS | FAIL | ERR]
ppc flash error: integer

Description: Return part of the self-test results. The results of the ppc ram and flash tests are returned.

This command does not perform any tests, the results returned will be due to the last test (*TST? or SYST:TEST? PMEM).

Example: `SYST:TEST:PMEM?`

:SYSTem

:TEST

:READ?

Parameters: <CPD>

required test result

Valid values: required test result: [CRAM | CFLash | CCPLd | NVBattery | USB | FPGA | CPFLlash | RTC | EEPRom | PCOMms | PREMote | KEYPad | BATTery | RMIF | RMODule]. Values other than those stated are rejected and an error generated.

Response: <CRD>, <NR1>

selected test status, selected test error

Returned values: selected test status: [NRUN | PASS | FAIL | ERR]
selected test error: integer

Description: Return part of the self-test results. The results of the specified test is returned.

This command does not perform any tests, the results returned will be due to the last test (*TST? or SYST:TEST? xxxx).

Example: `SYST:TEST:READ? FPGA`

:SYSTem

:TEST

[:RUN]?

Parameters: <CPD>

required test

Valid values: required test: [CRAM | CFLash | CCPLd | NVBattery | USB | FPGA | CPFLlash | RTC | EEPRom | PCOMms | PMEMory | PREMote | KEYPad | BATTery | RMIF | RMODule]. Values other than those stated are rejected and an error generated.

Response: <NR1>

selected test result

Returned values: selected test error: integer

Description: Run a single selftest. Returns 0 when self test completes without any errors, and 1 if it completes with an error.

Use :SYST:TEST:READ? (or :SYST:TEST:PMEM? for PMEMory test) to return further failure data if the self test failed.

Example: `SYST:TEST? FPGA`

:SYSTem

:TIME

Parameters: <NRf>, <NRf>, <NRf>

hours, minutes, seconds

Valid values: hours: integer. Valid values are 0 to 23. Values outside range are rejected and an error generated.

minutes: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

seconds: integer. Valid values are 0 to 59. Values outside range are rejected and an error generated.

Description: Set the time.

The date is stored in the RTC chip. Data will only be lost if the Nvram battery dies

Example: `SYST:TIME 15, 12, 55`

:SYSTem

:TIME?

Parameters: None

Response: <NR1>, <NR1>, <NR1>

hours, minutes, seconds

Returned values: hours: integer. Values are in the range 0 to 23.

minutes: integer. Values are in the range 0 to 59.

seconds: integer. Values are in the range 0 to 59.

Description: Determine the time – read from the RTC chip.

Example: `SYST:TIME?`

:SYSTem

:UNITs

:DISTance

Parameters: <CPD>

distance units

Valid values: distance units: [FEET | METers]. Values other than those stated are rejected and an error generated.

Description: Selects the distance units to be either in feet or meters.

Example: `SYST:UNIT:DIST FEET`

Set distance units to be feet.

:SYSTem

:UNITs

:DISTance?

Parameters: none

Response: <CRD>

distance units

Returned values: distance units: [FEET | MET]

Description: Determine if the instrument is treating distances to be in units of feet or meters.

Example: `SYST:UNIT:DIST?`

:SYSTem

:UNITs

:POWer

Parameters: <CPD>

power units

Valid values: power units: [DBM | DBW | W]. Values other than those stated are rejected and an error generated.

Description: Selects the power units to be either in dBm, dBW, or Watts.

Example: `SYST:UNIT:POW DBW`

Set power units to be dBW.

:SYSTem

:UNITs

:POWer?

Parameters: none

Response: <CRD>

power units

Returned values: power units: [DBM | DBW | W]

Description: Determine if the instrument is treating power to be in units of dBm, dBW, or Watts.

Example: `SYST:UNIT:POW?`

:SYSTem

:VERSion?

Parameters: none

Response: <NR2>

scpi version

Returned values: scpi version: real

Description: Determine which SCPI version the instrument complies to. We will always return 1999.0

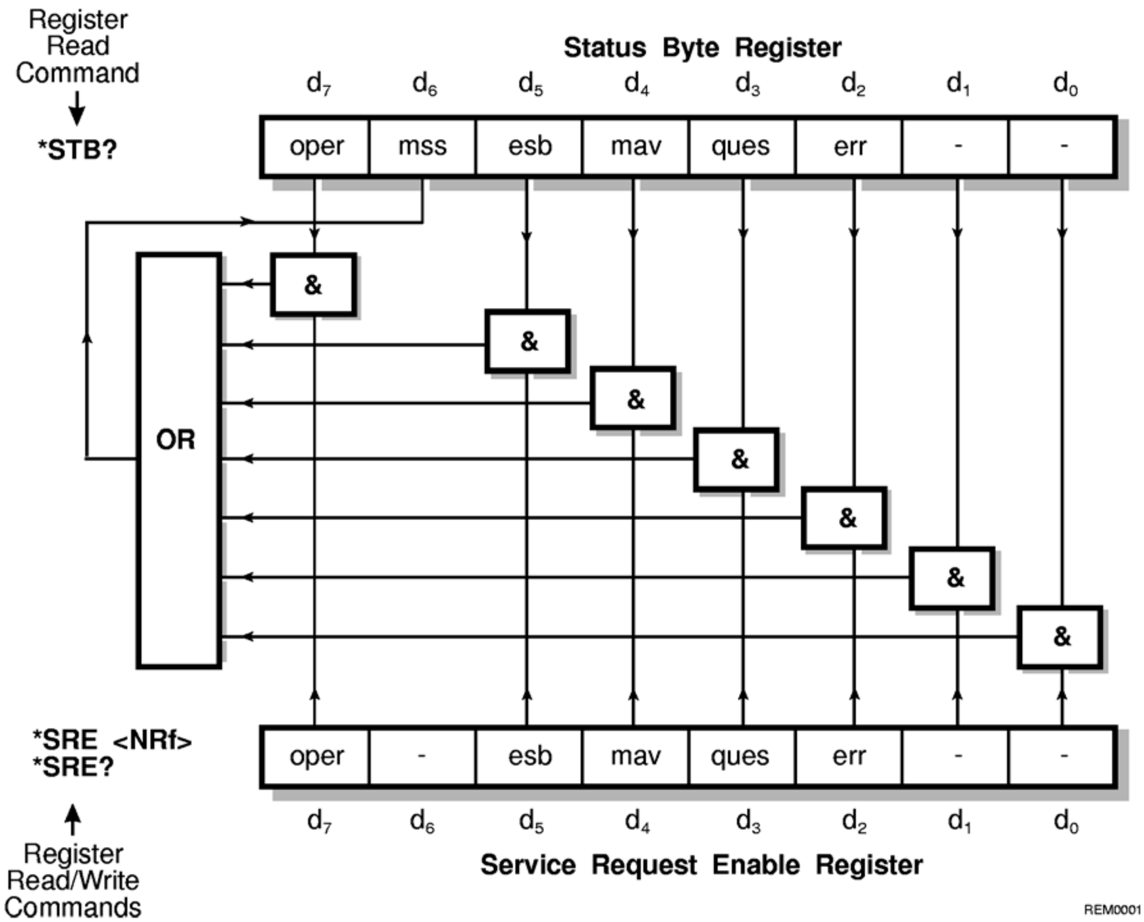
The 6000 is not SCPI compliant. We do however use many of the ideas from SCPI.

Example: `SYST:VERS?`

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX A – REMOTE STATUS REPORTING STRUCTURE

STATUS BYTE WHEN READ BY *STB



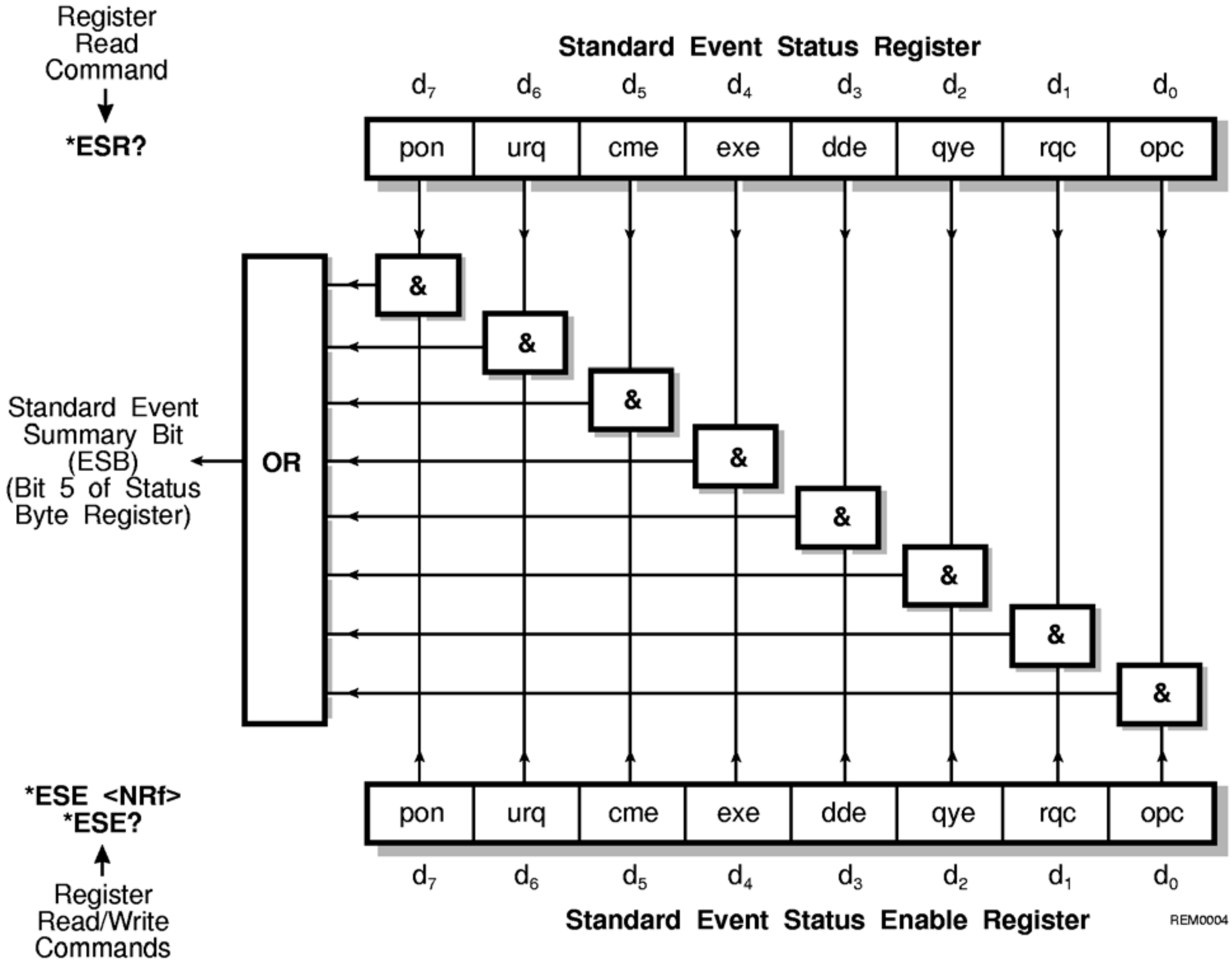
BIT	MNEM	DESCRIPTION
d_0	Not used	
d_1	Not used	
d_2	ERR	Error queue contains at least one error
d_3	QUES	Questionable Status Event Register Summary Bit
d_4	MAV	Message available in output queue (Queue not empty)
d_5	ESB	Standard Event Status Register Summary Bit
d_6	MSS	True when the device has at least one reason for requesting service
d_7	OPER	Operation Status Event Register Summary Bit

Notes...

When read by Serial Poll (rather than *STB?), d_6 contains RQS (Request Service) as defined in IEEE 488.2.

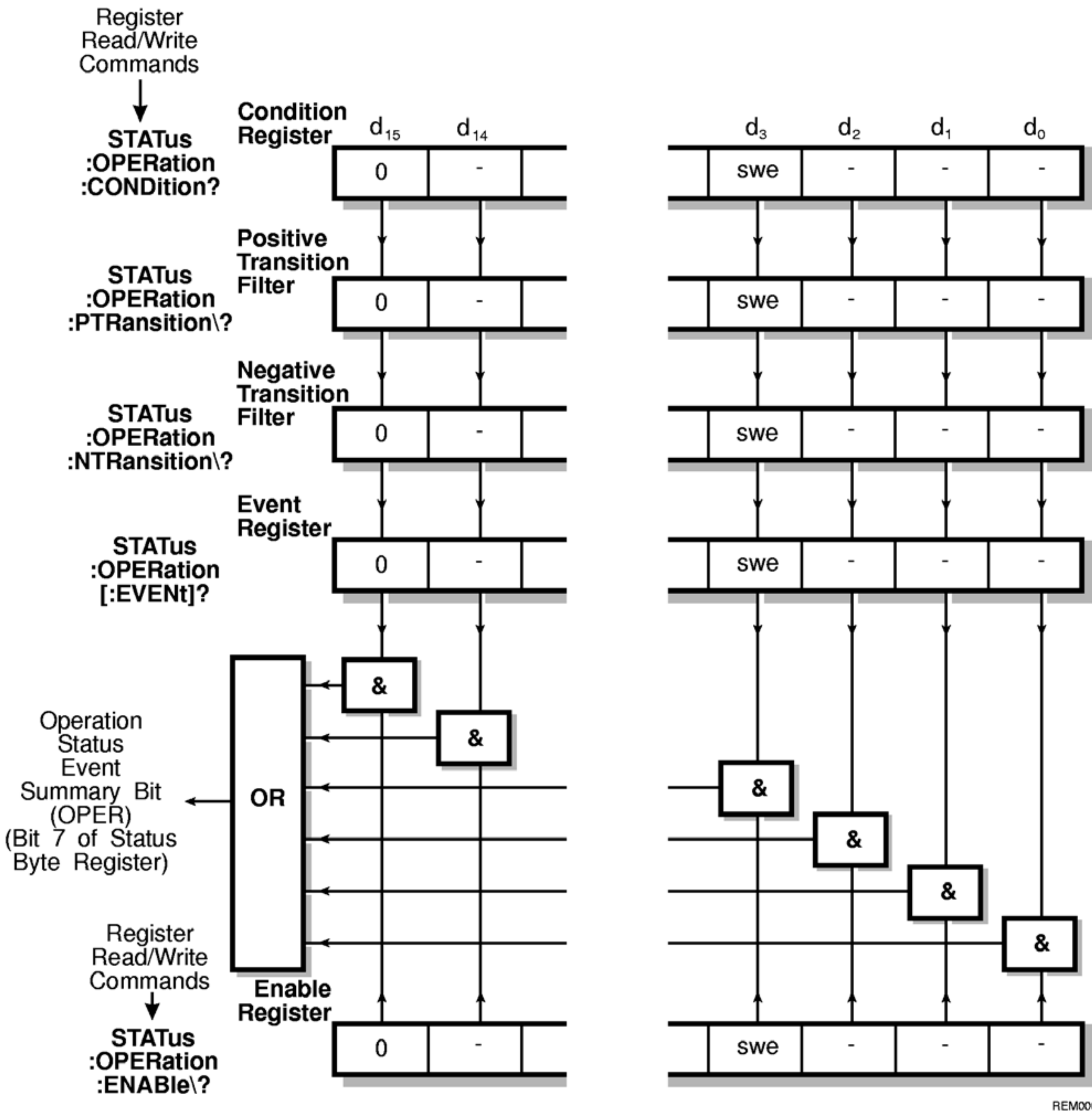
*SRE? always returns 0 for bit d_6.

STANDARD EVENT STATUS REGISTER (AS DEFINED IN IEEE 488.2)



BIT	MNEM	DESCRIPTION
d_0	OPC	Operation Complete
d_1	RQC	Request Control - Not implemented in this instrument
d_2	QYE	Query Error
d_3	DDE	Device-Specific Error
d_4	EXE	Execution Error
d_5	CME	Command Error
d_6	URQ	User Request - Not implemented in this instrument
d_7	PON	Power on

OPERATION STATUS CONDITION/EVENT/ENABLE REGISTERS



BIT	MNEM	DEFAULT TRANSITION	DESCRIPTION
d_0			Not Used
d_1			Not Used
d_2			Not Used
d_3	SWE	NEG	Sweeping – For auto test, same as the measuring bit. For all other tests, set at start of every time round test loop and cleared at the end of the loop. This can be used to determine when new data is available to be read back from the instrument.
d_4	MEAS	NEG	Measuring – Set when a test is started. Cleared when a test is stopped, or in the case of auto test (which is only run once, not continuously) cleared when the test completes. So this can be used to determine if a test is running or not.
d_5			Not Used
d_6			Not Used
d_7			Not Used
d_8			Not Used
d_9			Not Used
d_{10}			Not Used
d_{11}			Not Used
d_{12}			Not Used
d_{13}			Not Used
d_{14}			Not Used
d_{15}			Always zero

Notes...

The default transitions listed above are those set at power on. Note that the Operation Status Enable Register is cleared to all zeros at power on so it is necessary to enable the appropriate bits before the summary bit in the status byte register will be enabled.

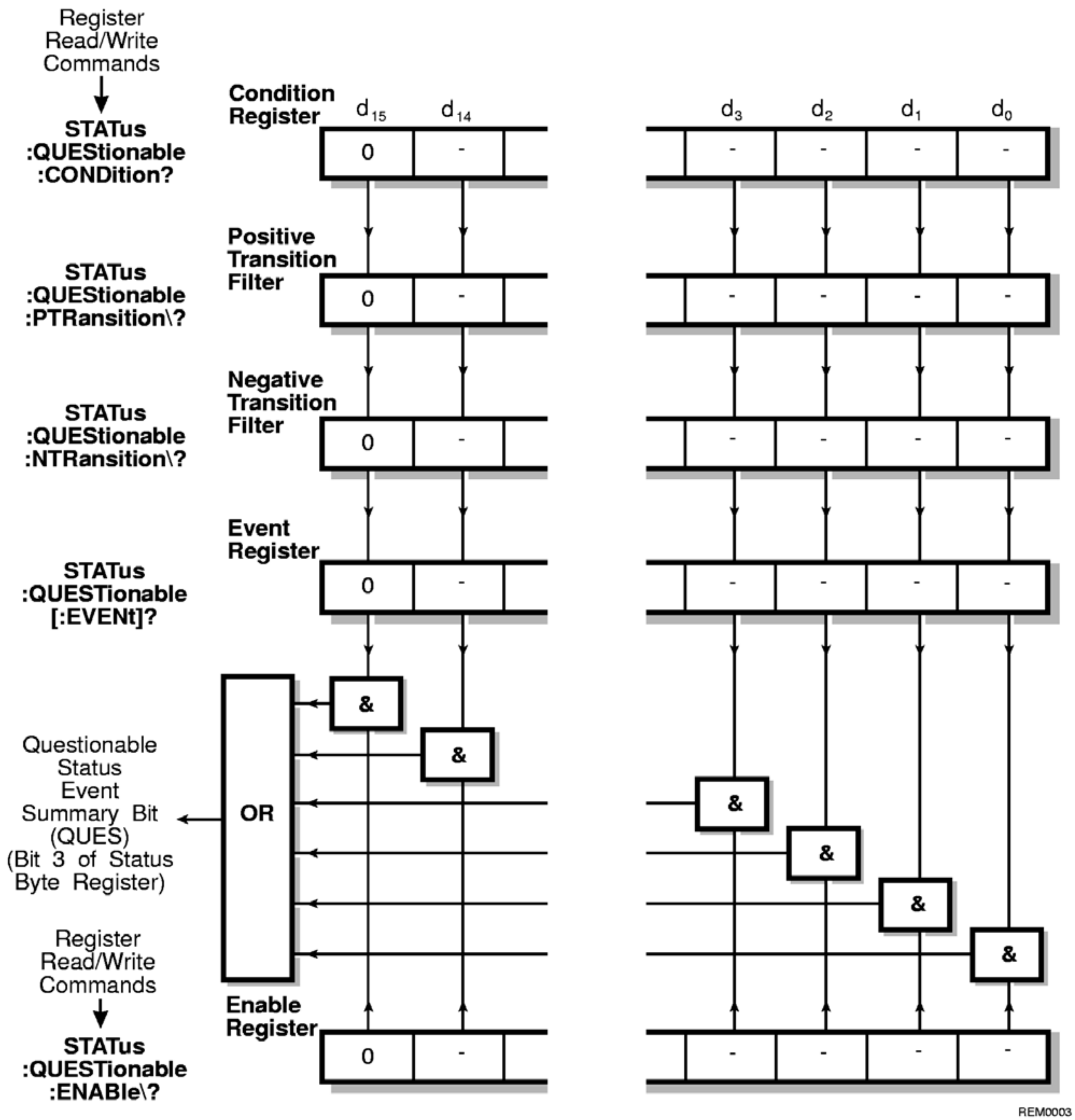
Each transition filter can be set independently giving four states:

Operation Status Event disabled
Operation Status Event set on positive transition in condition register
Operation Status Event set on negative transition in condition register
Operation Status Event set on positive or negative transitions in condition register

SWE is set at the start of each time round a test loop and cleared at the end of the test loop. For the auto test (which is only run once), it will be set when the command to start the test is received, and cleared when the auto test completes. For all other tests it will be set when the command is received to start the test and every time that the test sequence restarts and cleared at the end of the test sequence and when the command is received to stop the test. This bit can be used to indicate when new data is available to be read from the instrument. Note that for continuous tests data is only available when this bit goes low and MEAS is still high. Data is not updated when a test is stopped.

MEAS is set at the start of a test and cleared at the end of the test. For the auto test (which is only run once), it will be set when the command to start the test is received, and cleared when the auto test completes. For all other tests it will be set when the command is received to start the test and cleared when the command is received to stop the test. This bit indicates if a test is running or not.

QUESTIONABLE STATUS CONDITION/EVENT/ENABLE REGISTERS



BIT	MNEM	DEFAULT TRANSITION	DESCRIPTION
d_0			Not Used
d_1			Not Used
d_2			Not Used
d_3			Not Used
d_4			Not Used
d_5			Not Used
d_6			Not Used
d_7			Not Used
d_8	CAL	POS	Calibration suspect – at least one of TCXO, Generate, Receive, or Video Zero calibration is using default data
d_9			Not Used
d_{10}			Not Used
d_{11}			Not Used
d_{12}			Not Used
d_{13}			Not Used
d_{14}			Not Used
d_{15}			Always zero

Notes...

The default transitions listed above are those set at power on. Note that the Questionable Status Enable Register is cleared to all zeros at power on so it is necessary to enable the appropriate bits before the summary bit in the status byte register will be enabled.

CAL will be set to 1 if any of the four main calibrations are using default data. It will be set to 0 if all four are using valid data. The RTC cal is not reported in this bit since it does not affect validity of measurements.

APPENDIX B – OVERLAPPED COMMANDS

Currently this instrument does not have any overlapped commands.

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX C – EMULATION OF IEEE488.1

The RS232 interface does not have the extra control lines that the IEEE488.1 interface possesses, therefore it is necessary to emulate the extra functionality. This is done using two mechanisms, the first is the break facility and the second is by sending specific control codes over the RS232 interface which are interpreted to mean specific IEEE488.1 behavior.

This emulation of IEEE488.1 is only performed when the instrument is using the RS232 interface to receive commands from an external controller. If the RS232 is being used to output test data to a terminal emulator, this does not apply.

COMMANDS FROM CONTROLLER TO THE INSTRUMENT

There are four messages from the controller to the instrument:

Code sent	Meaning
break signal	Device clear
&POL	Perform serial poll
&DFC	Change to no serial flow control (same as SYST:SER:FCON NONE)
&HFC	Change to hardware flow control (same as SYST:SER:FCON HARD)
&SFC	Change to software flow control (same as SYST:SER:FCON XON)
&GTL	Return to local operation

The break signal acts as a device clear at any time. Once the device clear has been actioned a reply (&DCL<cr><lf>) is returned. This is necessary because there is no concept of bus holdoff on RS232.

The emulation codes are accepted at all times except within <ARBITRARY BLOCK PROGRAM DATA> or <STRING PROGRAM DATA> where the data is passed through unchanged. The emulation codes do not require terminating by a carriage return or linefeed – just send the specified 4 characters and the command will be accepted.

The instrument will enter remote from local on receipt of a byte over the RS232 interface.

RESPONSES/REQUESTS FROM THE INSTRUMENT TO THE CONTROLLER

There are three messages from the instrument to the controller:

Code sent	Meaning
&SRQ<cr><lf>	Request service (asynchronous)
&ddd<cr><lf>	Reply to &POL - STB & RQS sent as three decimal digits (000 - 255)
&DCL<cr><lf>	Acknowledges device clear completion

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX D – RESET VALUES

The following items are reset to the specified default values on receipt of *RST:

xpdr config: index 3 (which is GENERIC MODE S)
antenna: bottom
rf port: antenna
diversity: on
top ant height: 1 ft
top ant range: 12 ft
bottom ant height: 1 ft
bottom ant range: 12 ft
top ant height(meters): 2 m
top ant range(meters): 10 m
bottom ant height(meters): 2 m
bottom ant range(meters): 10 m
antenna cable loss: 0.1 dB
antenna cable loss mode: user defined
direct connect cable loss: 1.2 dB
ant gain (1030 MHz): 7.1 dBi
ant gain (1090 MHz): 6.1 dBi
uut address: AUTO
manual address: 0
diagnostics rx attenuation: 50 dB
diagnostics rf level: –50 dBm
diagnostics sls (both atcrbs and modeS): off
diagnostics prf: 50
diagnostics decoder: off
diagnostics dsp CW (Gen I/F): off
check xpdr capabilities: yes
power limit checks: far 43

dme echo: off
dme squitters: on
dme ident: on
dme percent reply: 100 %
dme rf level: -2 dBm (if antenna), -47 dBm (if direct connect)
dme range: 0 nm
dme rate: 10 kts
dme direction: out
dme freq: 978 MHz
dme ant range: 12 ft
dme ant range (meters): 10 m
dme max range: 400 nm
dme ident: IFR
dme antenna cable loss: 0.1 dB
dme antenna cable loss mode: user defined
dme direct connect cable loss: 1.2 dB
dme ant gain (960 MHz): 7.5 dBi
dme ant gain (1030 MHz): 7.1 dBi
dme ant gain (1090 MHz): 6.1 dBi
dme ant gain (1150 MHz): 5.0 dBi
dme ant gain (1220 MHz): 2.8 dBi
dme diagnostics rf level: -50 dBm
dme diagnostics channel: X
dme diagnostics prf: 50

The following items are reset to the specified default values on receipt of *RST:

dme diagnostics tx freq: 1025 MHz
dme diagnostics rx attenuation: 50 dB

dme diagnostics dsp CW (Gen I/F): off

The following items are reset to the specified default values on receipt of *RST:

tcas rf port: antenna
tcas ant range: 12 ft
tcas ant height: 1 ft
tcas ant range (meters): 10 m
tcas ant height (meters): 2 m
tcas uut address: auto
tcas manual address: 000000
tcas direct connect cable loss: 1.2 dB
tcas antenna cable loss mode: user defined
tcas antenna cable loss: 0.1 dB
tcas ant gain (1030 MHz): 7.1 dBi
tcas ant gain (1090 MHz): 6.1 dBi
tcas squitters: on
tcas altitude reporting: on
tcas test set address: A92493
tcas type: tcas 2
tcas percent reply: 100%
tcas intruder type: mode S
tcas range start: 10 nm
tcas range stop: 0 nm
tcas range rate: 300 kts
tcas altitude start: 1000 ft
tcas altitude stop: 0 ft
tcas altitude rate: 500 fpm
tcas converge: off
tcas uut altitude: 0 ft
tcas altitude detect: off
tcas reply fields defaulted: see Appendix E
tcas whisper/shout attenuation: 15 dB
tcas diagnostics rf level: -50 dBm
tcas diagnostics altitude: 10,000 ft
tcas diagnostics address: A92493
tcas diagnostics AQ: tracking
tcas diagnostics rx attenuation: 50 dB
tcas diagnostics gen i/f: off

The following items are reset to the specified default values on receipt of *RST:

tis rf point: antenna
tis antenna range: 12 ft
tis antenna height: 1 ft
tis antenna range (meters): 10 m
tis antenna height (meters): 2 m
tis direct connect cable loss: 1.2 dB
tis antenna cable loss mode: user defined
tis antenna cable loss: 0.1 dB
tis antenna gain (1030 MHz): 7.1 dBi
tis antenna gain (1090 MHz): 6.1 dBi
tis uut address: auto
tis manual address: 000000

All 5 tis targets are set identically
tis targets: 5
bearing: 0 degrees
range: 0 nm
altitude: 0 ft
altitude rate: level
heading: 0 degrees
traffic: proximity

The following items are reset to the specified default values on receipt of *RST:

tis uut heading: 0 degrees

adsb position decode: global
adsb latitude 0º 0 mn 0 sec North
adsb longitude 0º 0 mn 0 sec East
adsb gen: DF17
adsb mon: DF17
adsb gicb: DF20
adsb gen address: A92492
adsb gen 0,5 type: 9
period: 1 sec
latitude: 0º 0 min 0 sec North
longitude: 0º 0 min 0 sec East
SAF: 0
T: N/UTC
surveillance status: no info
barometric pressure altitude: 0 ft
gnss altitude: 0 ft
adsb gen 0,6 type: 5
period: 1 sec
latitude: 0º 0 min 0 sec North
longitude: 0º 0 min 0 sec East
movement: no info
T: N/UTC
heading: 0 degrees

The following items are reset to the specified default values on receipt of *RST:

adsb gen 0,8	type: 1
	period: 1 sec
	flight id: (blank)
	emitter category set: D
	emitter category: reserved
adsb gen 0,9	type: 19
	period: 1 sec
	east-west velocity: 0 knots, EAST
	north-south velocity: 0 knots, NORTH
	NACV: 0
	subtype: 1
	vertical rate: 0 ft/min
	geometric altitude difference from barometric: 0 ft
	source: geometric
	intent change: no
	ifr capability flag: no
	heading: 0 degrees
	air speed: 0 knots
	air speed type: ias
adsb gen 6,1	type: 28
	period: 1 sec
	reserved: 0
	subtype: 0
	emergency/priority code: no emergency

The following items are reset to the specified default values on receipt of *RST:

adsb gen 6,2	type: 29
	period: 1 sec
	vertical data/source info: not valid
	target altitude capability: holding altitude only
	vertical mode indicator: unknown
	SIL: 0
	target altitude type: flight level
	nic for baro: 0
	target altitude: 0 ft
	target heading: 0 degrees
	tcas/acas operational: yes
	RAA: no
	horizontal data available/source ind: not valid
	horizontal mode indicator: unknown
	NAC: 0

The following items are reset to the specified default values on receipt of *RST:

	emergency/priority code: no emergency
adsb gen 6,3	type: 31
	period: 1 sec
	subtype: 0
	NAC: 0
	BAQ: 0
	SIL: 0
	TC: 0
	CDTI: 0
	ARV: 0
	TS: 0
	RA: 0
	Version number: DO-260
	not-tcas: 0
	operational mode subfield: 0
	nic for baro: 0
	horizontal reference direction: true north
	ident: no
	track angle/heading: 0
	NIC: 0
	length/width codes: 0
	receiving atc services: 0
	b2 low: 0
	POA: 0

All adsb gen tests are disabled.

altitude encoder source: encoder

Power down timeout: 10 min
Erp units: dBm
Units: feet
Contrast and backlight are set to the values set in abm (the boot monitor)
Data dump format: comma delimited

All results are set to Not Run.

The following items are not changed by *RST:

RS232 baud rate
RS232 flow control

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX E – TCAS REPLY FIELD DEFAULT VALUES

The following items are reset to the specified default values on receipt of TCAS:RBIT:RES.

DFO

VS=0

SL=0

DF11

CA=0

DF16

VS=0

SL=0

RI(acquisition)=8

RI(tracking)=3

ARA=0

RAC=0

VDS=30 (hexadecimal)

THIS PAGE INTENTIONALLY LEFT BLANK.

INDEX

THIS PAGE INTENTIONALLY LEFT BLANK.

Distributed on CD #6093 L0

112286 Rev. B0

November 2019

VIAVI Solutions

North America:	**1.844.GO VIAVI / 1.844.468.4284**
Latin America	**+52 55 5543 6644**
EMEA	**+49 7121 862273**
APAC	**+1 512 201 6534**
All Other Regions:	**viavisolutions.com/contacts**