ISC-2425-25+ Industrial System Controller Programming Manual

AN-50-002

www.minicircuits.com

Table of Contents

Mini-Circuits®

1. Introduction

The ISC-2425-25+ Industrial System Controller and Small Signal Generator board is a powerful device with many capabilities.

The ISC-series small signal generator boards can be operated by sending text commands over their serial interface(s). These commands are part of a non-proprietary command protocol made to be both legible by humans and suitable for process automation through software. The command set enables users to get started quickly and communicate directly with ISC-2425-25+ boards using nothing more than basic equipment.

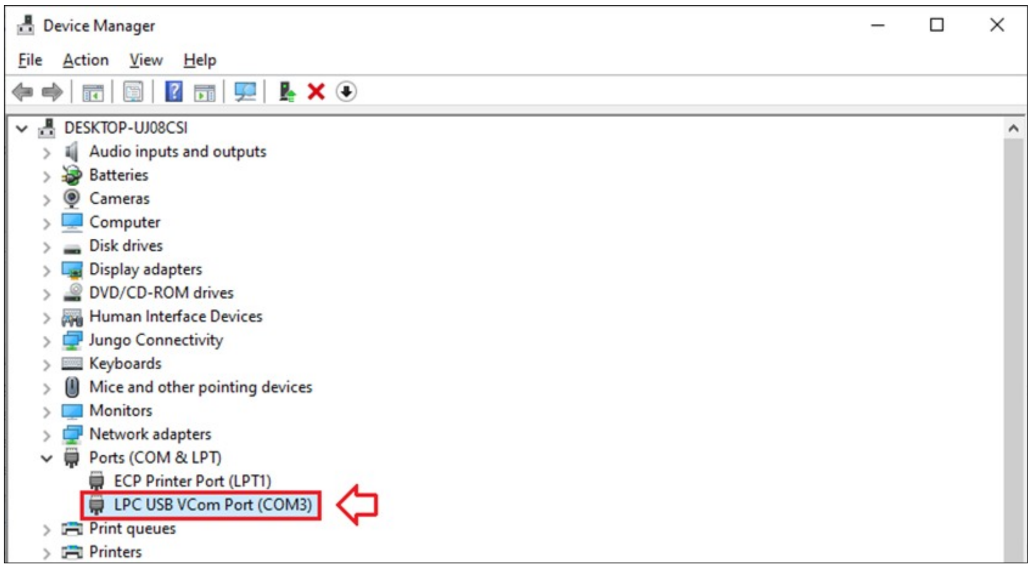
The purpose of this document is to provide detailed information about the use of commands supported by the firmware of the ISC-2425-25+ signal generator and controller boards.

1.1 Setting up communication

Setting up the communication between user and ISC-2425-25+ board is a straightforward process.Below is an example using PuTTY on a Windows or Linux PC.

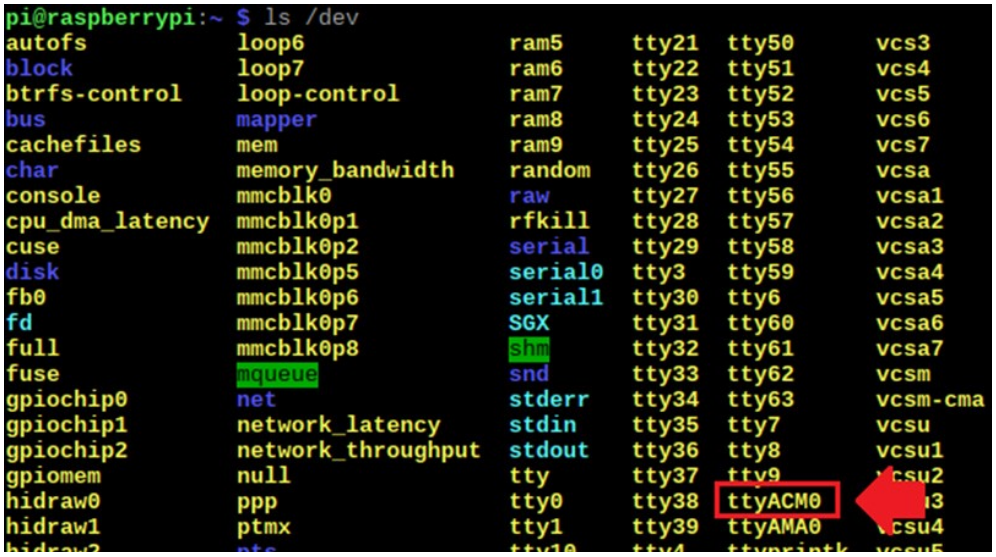
1. Plug the ISC-2425-25+ board into the PC using a mini-USB cable.
2. Find the port name of the device

- **Windows:**
  Open the 'Device Manager' in Windows and find the port name of your device. It will show up as an 'LPCUSB VCOM Port', followed by the port name.



- **Linux:**

  Open a terminal (CTRL + ALT + T) and use the "ls /dev" command to view available devices. The signalgenerator board should appear as a 'ttyACM' device followed by a number.



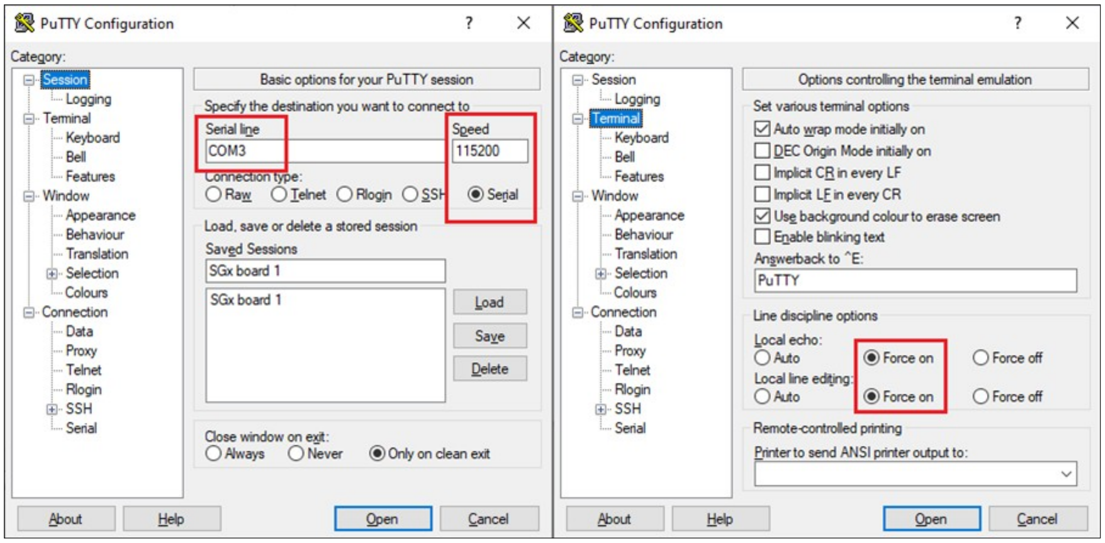
```
pi@raspberrypi:~ $ ls /dev
autofs           loop6              ram5     tty21  tty50       vcs3
block            loop7              ram6     tty22  tty51       vcs4
btrfs-control    loop-control       ram7     tty23  tty52       vcs5
bus              mapper             ram8     tty24  tty53       vcs6
cachefiles       mem                ram9     tty25  tty54       vcs7
char             memory_bandwidth   random   tty26  tty55       vcsa
console          mmcblk0            raw      tty27  tty56       vcsa1
cpu_dma_latency  mmcblk0p1          rfkill   tty28  tty57       vcsa2
cuse             mmcblk0p2          serial   tty29  tty58       vcsa3
disk             mmcblk0p5          serial0  tty3   tty59       vcsa4
fb0              mmcblk0p6          serial1  tty30  tty6        vcsa5
fd               mmcblk0p7          SGX      tty31  tty60       vcsa6
full             mmcblk0p8          shm      tty32  tty61       vcsa7
fuse             mqueue             snd      tty33  tty62       vcsm
gpiochip0        net                stderr   tty34  tty63       vcsm-cma
gpiochip1        network_latency    stdin    tty35  tty7        vcsu
gpiochip2        network_throughput stdout   tty36  tty8        vcsu1
gpiomem          null               tty      tty37  tty9        vcsu2
hidraw0          ppp                tty0     tty38  ttyACM0       3
hidraw1          ptmx               tty1     tty39  ttyAMA0     vcsu4
hidraw2          pts                tty10    tty4   ttyprintk   vcsu5
```

3. Open PuTTY on your PC and provide the necessary information for the connection with the device.

Baud rate:	115200
Data bits:	8
Parity bits:	0
Stop bits:	1
Flow control:	none

It is highly recommended to configure the settings ‘Local echo’ and ‘Local line editing’ to ‘Force on’.

Remark: Linux requires the full path to the port (e.g. “/dev/ttyACM0”).



PuTTY Configuration — Session: Basic options for your PuTTY session. Specify the destination you want to connect to. Serial line: COM3. Speed: 115200. Connection type: Raw, Telnet, Rlogin, SSH, [✓ Serial]. Saved Sessions: SGx board 1. Close window on exit: Always, Never, [✓ Only on clean exit].

PuTTY Configuration — Terminal: Options controlling the terminal emulation. Set various terminal options: [✓ Auto wrap mode initially on], DEC Origin Mode initially on, Implicit CR in every LF, Implicit LF in every CR, [✓ Use background colour to erase screen], Enable blinking text. Answerback to ^E: PuTTY. Line discipline options: Local echo: Auto, [✓ Force on], Force off. Local line editing: Auto, [✓ Force on], Force off.

Save the session, so that it won’t need to be reconfigured again in the future and press ‘Open’ to start aconnection with the ISC-2425-25+ board.

A blank terminal window will pop up. Communication with the ISC board should now be established.

1.2 Sending a command

The various commands of the command set can be divided up into three types:

- **Set commands** – These are used to write the setpoints and configurations of the ISC board and typical-ly end with the letter 'S'. For example: setting the RF power level, enabling DLL mode, etc.
- **Get commands** – These are used to read the values of the ISC board and, with a few exceptions, typical-ly end with the letter 'G'. For example: printing out setpoints, getting the temperature, etc.
- **Miscellaneous commands** – A few remaining commands that don't fall in either of the above catego-ries. They will usually execute more miscellaneous actions such resetting the board or performing a frequency sweep.

The template for a command is as follows:

```
$command,channel_identifier,parameter1,parameter2,etc...\r\n
```

The protocol uses the '$' symbol as an indicator for the start of a command. A message sent without the '$' symbol will not be recognized as a command. Similarly, '\r\n' (new line) marks the end of a command. Without at least a '\r' or a '\n' the device will keep waiting for more bytes indefinitely.

There is one parameter that is used in every command: the channel identifier.

Each ISC board is assigned a numeric channel identifier. The default value is 1, but when using more than one ISC board in a multi-channel setup, it may be desirable to assign a unique number to each device beforehand so that they can be differentiated during runtime.

When sending a command, it is necessary to always include the correct channel number of the board, otherwise the message is ignored under the assumption that it is intended for a different device.

There is also the possibility to use the channel identifier '0', which is accepted by every channel regardless of their assigned number. However, the user is required to have good understanding of their system, lest they unintentionally misconfigure multiple channels to the same values.

1.3 Receiving a response

The ISC board provides feedback when a command succeeds.A

successful set command will always reply with an OK:

```
$command,channel_identifier,OK\r\n
```

A successful get command will simply return the requested information:

```
$command,channel_identifier,parameter1,parameter2,etc...\r\n
```

Miscellaneous commands do not have a standardized way of relaying successful results but will in mostcases be similar to the above examples or some kind of combination of both.

It is also possible for the execution of a command to fail. There can be several reasons for this:

- A mistake could be made when writing out a command.
- The command may not be executable on the particular configuration of the board.
- A runtime problem occurs.
- Etc.

If something goes wrong during command execution, the ISC board will reply with an error code, indicatingthat the action has failed. An error response looks as follows:

```
$command,channel_identifier,ERR##\r\n
```

The possible error codes and their respective descriptions are listed in the following:

Hex code	Error Description
0x01	Reserved
0x02	The serial message exceeded the maximum length.
0x03	The serial message had too few arguments.
0x04	The message had too many arguments.
0x05	The system could not accept this message is the current mode.
0x06	The system was busy and cannot process this message at this time.
0x07	The message was recognized but is not yet implemented in the codebase.
0x10	An argument was in error with the lower nibble indicating the argument number.
0x11	Argument 1 was invalid / out of range.
0x12	Argument 2 was invalid / out of range.
0x13	Argument 3 was invalid / out of range.
0x14	Argument 4 was invalid / out of range.
0x15	Argument 5 was invalid / out of range.
0x16	Argument 6 was invalid / out of range.
0x17	Argument 7 was invalid / out of range.
0x18	Argument 8 was invalid / out of range.
0x19	Argument 9 was invalid / out of range.
0x7E	Command execution failed.
0x7F	An error occurred that is not covered by any of the other error codes.

An example of an error response would be the following

Input:	$VER,1,1
Output:	$VER,1,ERR04

Hex code 0x04 indicates that the input command had too many arguments. This is correct, as the $VER command requires no additional arguments beyond the channel identifier.

1.4 Protocol

The protocol of the command set is straight forward:

1. Send a command.
2. Wait until a full response with a ‘\r\n’ at the end is read.

It is instrumental to always wait for a complete reply from the device before sending another command. Sending commands without waiting for a full reply can result in communication problems. Sending commands in quick succession without waiting for the response may additionally overburden the ISC board with a growing list of tasks to complete, leaving it no time to perform safety operations and resultingin an automatic reset to break the cycle.

2. Information request commands

These commands can provide the user with useful information about the ISC board.

2.1 $IDN – Get board identity

This command returns the identity of the ISC board.

Identity name template

ISC-[frequency_low][frequency_high]-[power]+

- **[frequency_low]** – Lower frequency limit (only first 2 digits).
- **[frequency_high]** – Upper frequency limit (only first 2 digits).
- **[power]** – Maximum RF output power of the signal generator board in dBm.

Syntax:

Input:	$IDN,[channel]
Output:	$IDN,[channel],[manufacturer],[ISC board],[serial number]

- **[channel]** – Channel identification number.
- **[manufacturer]** – Name of the manufacturer:

  Mini-Circuits
- **[ISC board]** – The type of ISC board. The following types are possible:

  ISC-2425-25+
- **[serial number]** – unique serial number of the board.

Example:

Input:	$IDN,1
Output:	$IDN,1,Mini-Circuits,ISC-2425-25+,MN0000102101

This indicates the connected ISC board is an ISC-2425-25+ for use in the 2450 MHz ISM band, with serialnumber MN0000102101.

2.2 $RTG – Get uptime of the ISC board

This command returns the uptime of the ISC board since its initialization. The uptime count restarts whenthe board is reset.

Syntax:

Input:	$RTG,[channel]
Output:	$RTG,[channel],[uptime]

- **[channel]** – Channel identification number.
- **[uptime]** – The uptime in seconds.

Example:

Input:	$RTG,1
Output:	$RTG,1,51

This indicates that the ISC board has been running for 51 seconds.

2.3 $VER – Get firmware version

This command returns the current version of the firmware.

Syntax:

Input:	$VER,[channel]
Output:	$VER,[channel],[manufacturer identifier],[major revision],[minor revision],[Build],(hotfix),[date stamp], [time stamp]

- **[channel]** – Channel identification number.
- **[manufacturer identifier]** – Firmware developer identifier.
- **[major revision]** – The version's major revision number.
- **[minor revision]** – The version's minor revision number.
- **[build]** – The version's build number.
- **(hotfix)** – (optional) The version's hotfix number.
- **[date stamp] –** The date on which the firmware was compiled.
- **[time stamp]** – The time at which the firmware was compiled.

Example:

Input:	$VER,1
Output:	$VER,1,Mini-Circuits,1,11,2,Aug 25 2021,01:45:36

This indicates the firmware version is 1.11.2, compiled on August 25th, 2021.

3. Error handling commands

Commands for handling errors during device operation.

3.1 $ST – Request status

The $ST command is used to monitor the status of the ISC board.

It returns hexadecimal codes that can be compared against bitmasks to provide an overview of the state ofaffairs.

Error status:
ISC boards have a safety feature called the 'Safe Operating Area' (SOA). If a fault occurs during operation, the SOA raises an error and takes action to protect the system. This is indicated by a red LED on the board. An error on the ISC board is accompanied by an informative error code which can be used to trace the problem.

Remark:
To ensure the code can be viewed it stays in memory until manually cleared away. For safety reasons some errors block the RF power output of the ISC board until cleared ($ERRC).

The following table provides an overview of the error status bitmask:

Hex	Error / Warning Status	Description / Action
0x0	No error.	Status is nominal
0x1	Unspecified Error.	RF output OFF (blocking); Reset controller.
0x2	High temperature in the PA.	Unit throttles output power (with autogain enabled).
0x4	Shutdown Temperature in the PA.	RF output OFF (blocking)
0x8	High reflection.	Warning only; no action.
0x10	Shutdown reflection.	RF output OFF (blocking).
0x20	Reset detected.	Warning only; no action.
0x40	Temperature read-out error.	RF output OFF (blocking).
0x80	Power measurement failure.	RF output OFF (blocking).
0x100	RF output enable failure.	Indication message.
0x200	Multiplexer failure (I2C).	RF output OFF (blocking).
0x400	External shutdown triggered.	RF output OFF (non-blocking).
0x800	Reserved.	
0x1000	I2C communication problem has occurred.	RF output OFF (blocking)in case of critical measurements.
0x2000	SPI communication problem has occurred.	RF output OFF (blocking)in case of critical measurements.
0x4000	IQ Conversion Error.	RF output OFF (blocking).
0x8000	SOA Measurement Error.	RF output OFF (blocking).
0x10000	External Communication Watchdog Timeout.	RF output OFF (blocking)
0x20000	Calibration missing	RF output OFF (blocking)
Hex	**Error / Warning Status**	**Description / Action**

0x40000	Reserved	-
0x80000	SOA high dissipation	Warning only; no action.
0x100000	SOA shutdown dissipation	RF output OFF (blocking).
0x200000	EEPROM content incompatible with firmware version	RF output OFF (blocking).
0x400000	Internal PA Error	RF output OFF (blocking).
0x800000	PA Reset Failure	RF output OFF (blocking).
0x1000000	High Current	RF output OFF (blocking).

Note: "blocking" here means that the RF output is switched off and it remains off until the $ERRC command clears the error. Only then can the RF output be switched on again.

Syntax:

Input:	$ST,[channel]
Output:	$ST,[channel],[reserved],[error status code]

- **[channel]** – Channel identification number.
- **[reserved]** – Reserved. Will always return 0.
- **[error status code]** – A hexadecimal value representing various errors which have occurred on the ISCboard.

Example:

Input:	$ST,1
Output:	$ST,1,0,460

The codes returned by the command represent combinations of several statuses. To decipher a code, it must be converted from hexadecimal to binary and compared against the bitmask to see which bits are high.

The error status code is 0x460. The code is comprised of three separate errors summing up to 0x460:

- 0x20 = 0b000000**1**00000 – Reset detected.
- 0x40 = 0b00000**1**000000 – Temperature read-out error.
- 0x400 = 0b0**1**0000000000 – External shutdown triggered.

The error status code will remain in memory until an error clear command has been sent.

Syntax 2:

This command supports an alternative syntax which displays errors in a more user-friendly format.

Input:	$ST,[channel],(output mode)
Output:	$ST,[channel],[error status message]... $ST,[channel],OK

- **[channel]** – Channel identification number.
- **(output mode)** – (optional)0 –

  bitmasked error codes

  1 – list of legible errors messages

  Remark: this alters command's syntax
- **[error status message]** – A hexadecimal value representing various errors which have occurred on theISC board.

Example:

Input:	$ST,1,1
Output:	$ST,1,RESET_DETECTED $ST,1,TEMPERATURE_MEASUREMENT_FAILURE $ST,1,EXTERNAL_SHUTDOWN_DETECTED $ST,1,OK

This time the optional (output mode) argument is used. The ISC board has encountered the followingproblems:

- Reset detected.
- Temperature read-out error.
- External shutdown triggered.

The error status code will remain in memory until an error clear command has been sent.

3.2 $ERRC – Clear error

An error on the ISC board is accompanied by an informative error code ($ST). To ensure the code can be viewed, it stays in memory until manually cleared away. For safety purposes as long as a non-zero error status remains, depending on the exact value, further RF output of the ISC board may be blocked.

The $ERRC command clears the error state and resets the protective systems that impede the ISC board while an error is present.

The process of status checking and error clearing is the core of error handling. The error status of the ISC board is polled to retrieve current information about any errors that have occurred during operation. This information can be used to decide an appropriate response. After the problem has been resolved, the errors reported by the ISC board should be cleared to signal to the device that the error status is no longer actual.

A situation may occur where an error state appears to persist despite attempts to clear it. In fact, the error is getting cleared properly but is immediately reactivated again because the underlying cause of the error is still present.

Syntax:

Input:	$ERRC,[channel]
Output:	$ERRC,[channel],OK

- **[channel]** – Channel identification number.

Example:

Input:	$ERRC,1
Output:	$ERRC,1,OK

This clears the error state of the ISC board.

If the root cause of the problem has been resolved, the error code will be 0 the next time the error state of the ISC board is polled. If the root cause of the problem remains (e.g. the external shutdown remains triggered), the error state will keep reverting back to a non-zero value until it has been dealt with appropriately.

4. System-level- and configuration commands

System commands are important for (pre)configuring the system and ensuring its continuous operation.

4.1 $CHANS – Set channel identification number

Every ISC board is assigned a numeric value as a channel identifier for communication. The default value of the identifier is ‘1’, which serves its purpose in single-channel systems. In setups that deploy more than one ISC board it is often necessary to assign a unique number to each individual board beforehand, so that they can all be commanded as separate entities. An ISC board will not respond to commands written for a different channel.

The $CHANS command assigns a channel identification number to an ISC board.

Syntax:

Input:	$CHANS,[channel],[new channel]
Output:	$CHANS,[channel],OK

- **[channel]** – Channel identification number. (default = 1)
- **[new channel]** – New desired channel identification number.

Legacy Syntax:
This syntax still works, but its use is strongly discouraged.

Input:	$CHANS,[channel]
Output:	$CHANS,[channel],OK

- **[channel]** – Channel identification number. (default = 1)

Example:

Input:	$CHANS,1,2
Output:	$CHANS,2,OK

This assigns the channel identification number ‘2’ to the ISC board.

From now on, this board can only be addressed by providing ‘2’ as the [channel] argument for commands.

E.g. “$VER,2”

4.2 $CHANG – Get channel identifier

This command returns the channel number assigned to the ISC board.

Remark: This command does not adhere to established syntax.It does not accept a channel argument at the input.

Syntax:

Input:	$CHANG
Output:	$CHANG,[channel]

- **[channel]** – Channel identification number. (default = 1)

Example:

Input:	$CHANG
Output:	$CHANG,2

This indicates the channel number assigned to the ISC board is the number '2'.

4.3 $CSS – Set clock source

This command sets the clock source configuration of the ISC board.

An ISC board can either use its own internal 10MHz Crystal Controlled Oscillator (XCO), or it can accept an external clock reference from another ISC board. The clock signal can be transmitted and received using a Low Voltage Differential Signaling (LVDS) transceiver.

The clock source is required to synchronize signal phase of ISC boards in coherent multi-channel systems.

Syntax:

Input:	$CSS,[channel],[clock source]
Output:	$CSS,[channel],OK

- **[channel]** – Channel identification number.
- **[clock source]** – Numeric value assigned clock source configuration.

0 – Standalone (default):

- Use onboard XCO.
- Do not output reference signal.

1 – Master:

- Use onboard XCO.
- Output reference signal to slaves using LVDS.

2 – Slave:

- Use external clock reference from LVDS.
- Do not output reference signal.

3 – Slave inline:

- Use external clock reference from LVDS.
- Output reference signal to slaves using LVDS.

4 – Reserved. Do not use.

5 – Reserved. Do not use.

The default state is standalone.

Example:

Input:	$CSS,1,3
Output:	$CSS,1,OK

This configures the clock source to inline slave mode. See the Apps Note on Coherence for more details.

4.4 $CSG – Get clock source

This command returns the clock source configuration of the ISC board.

Syntax:

Input:	$CSG,[channel]
Output:	$CSG,[channel],[clock source]

- **[channel]** – Channel identification number.
- **[clock source]** – Numeric value assigned clock source configuration.

  0 – Standalone (default):
  - Use onboard XCO.
  - Do not output reference signal.

  1 – Master:
  - Use onboard XCO.
  - Output reference signal to slaves using LVDS.

  2 – Slave:
  - Use external clock reference from LVDS.
  - Do not output reference signal.

  3 – Slave inline:
  - Use external clock reference from LVDS.
  - Output reference signal to slaves using LVDS.

  4 – Reserved. Parameter should be ignored.5 –

  Reserved. Parameter should be ignored.

Example:

Input:	$CSG,1
Output:	$CSG,1,1

This indicates the clock source is configured to master mode.

4.5 $RST – Execute system reset

This command executes a reset of the ISC board. All

board settings will return to their default states.

Following a reset, whether intentional or as the result of a fault, the 'reset detected' error flag (0x20) will beraised.

Syntax:

Input:	$RST,[channel]
Output:	$RST,[channel],OK

- **[channel]** – Channel identification number.

Example:

Input:	$RST,1
Output:	$RST,1,OK

After sending the $RST command, the board returns an OK and then promptly resets.The 'reset

detected' error flag (0x20) is raised.

5. Basic commands

Common commands for basic device operation.

5.1 $ECS – Set RF output enabled / disabled

This command turns RF output of the ISC board ON or OFF.

Syntax:

Input:	$ECS,[channel],[enable]
Output:	$ECS,[channel],OK

- **[channel]** – Channel identification number.
- **[enable]** – The desired setting of the RF output.
  - 0 – OFF (default)
  - 1 – ON

Example

Input:	$ECS,1,1
Output:	$ECS,1,OK

This enables the RF power output of the ISC board.

5.2 $ECG – Get RF output enable state

This command returns the enable state of the ISC board's RF output.

Syntax:

Input:	$ECG,[channel]
Output:	$ECG,[channel],[enable]

- **[channel]** – Channel identification number.
- **[enable]** – The desired setting of the RF output.
  - 0 – OFF
  - 1 – ON

Example:

Input:	$ECG,1
Output:	$ECG,1,1

This indicates the ISC board's RF power output is enabled

5.3 $FCS – Set frequency

This command sets the frequency of the ISC board's RF output to the desired value in MHz.

Syntax:

Input:	$FCS,[channel],[frequency]
Output:	$FCS,[channel],OK

- **[channel]** – Channel identification number.
- **[frequency]** – The desired frequency setting for the RF signal.

Example:

Input:	$FCS,1,2450
Output:	$FCS,1,OK

This sets the frequency to 2450 MHz.

5.4 $FCG – Get frequency

This command returns the frequency of the ISC board's RF output in MHz.

Syntax:

Input:	$FCG,[channel]
Output:	$FCG,[channel],[frequency]

- **[channel] –** Channel identification number.
- **[frequency]** – The current frequency of the RF signal.

Example:

Input:	$FCG,1
Output:	$FCG,1,2450.000

This indicates the frequency is set to 2450 MHz

5.5 $PCS – Set phase

This command sets the phase of the ISC board's RF output in degrees (°). The phase set is referenced to theselected clock source (see $CSS command).

Syntax:

Input:	$PCS,[channel],[phase]
Output:	$PCS,[channel],OK

- **[channel]** – Channel identification number.
- **[phase] –** The desired phase value in degrees (°); between 0 and 359.

Example:

Input:	$PCS,1,25
Output:	$PCS,1,OK

This sets the phase to 25°.

5.6 $PCG – Get phase

This command returns the current phase value of the ISC board's RF output in degrees (°).

Syntax:

Input:	$PCG,[channel]
Output:	$PCG,[channel],[degrees]

- **[channel]** – Channel identification number.
- **[phase]** – The desired phase value in degrees (°)

Example:

Input:	$PCG,1
Output:	$PCG,1,25.00

This indicates the phase is configured to 25°

5.7 $PTG – Get PA temperature

This command returns the temperature of the power amplifier (PA).

Syntax:

Input:	$PTG,[channel]
Output:	$PTG,[channel],[temperature]

- **[channel]** – Channel identification number.
- **[temperature]** – The current temperature in degrees Celsius.

Example:

Input:	$PTG,1
Output:	$PTG,1,51

This indicates that the current temperature of the attached PA is 51 °C.

5.8 $GCS – Set VGA attenuation in dB

This command sets the attenuation of the variable gain amplifier (VGA) which regulates the ISC board's power output. The higher the value, the lower the power output.

Remark: Under normal conditions, both the VGA and the IQ modulator are used to regulate power output of the ISC board, thus the actual power output is a combination of both. The IQ modulator is controlled using the $MCS command (see chapter 5.10).

Remark: To use this command, auto-gain must be disabled first (see $AGES).

Syntax:

Input:	$GCS,[channel],[attenuation]
Output:	$GCS,[channel],OK

- **[channel]** – Channel identification number.
- **[Attenuation]** – The attenuation value of the DVGA.
  - Attenuation Range: 0 – 31.5 dB
  - Minimum step size: 0.5 dB

Example:

Input:	$GCS,1,7
Output:	$GCS,1,OK

This will set the attenuation of the VGA to 7dB.

5.9 $GCG – Get VGA attenuation in dB

This command returns the configured attenuation value of the VGA which regulates the ISC board's poweroutput. The higher the value, the lower the power output.

Syntax:

Input:	$GCG,[channel]
Output:	$GCG,[channel],[attenuation

- **[channel]** – Channel identification number.
- **[Attenuation]** – The attenuation value of the DVGA.
  - Attenuation Range: 0 – 31.5 dB
  - Minimum step size: 0.5 dB

Example:

Input:	$GCG,1
Output:	$GCG,1,10

This indicates the configured attenuation of the VGA is 10dB.a

5.10 $MCS – Set magnitude in percent

This command sets the magnitude setting of the IQ modulator, which regulates the ISC board's power output. The higher the value, the higher the power output.

Remark: Under normal conditions, both the VGA and the IQ modulator are used to regulate power output of the ISC board, thus the actual power output is a combination of both. The VGA is controlled using the $GCScommand (see chapter 5.8).

Remark: To use this command, auto-gain must be disabled first.

Syntax:

Input:	$MCS,[channel],[magnitude]
Output:	$MCS,[channel],OK

- **[channel]** – Channel identification number.
- **[magnitude]** – The desired magnitude of the IQ modulator in percent (%).

Example:

Input:	$MCS,1,75
Output:	$MCS,1,OK

This will set the magnitude of the IQ modulator to 75%.

5.11 $MCG – Get magnitude in percent

This command gets the magnitude of the IQ modulator.

Syntax:

Input:	$MCG,[channel]
Output:	$MCG,[channel],[magnitude]

- **[channel]** – Channel identification number.
- **[magnitude]** – The current magnitude configuration of the IQ modulator in percent.

Example:

Input:	$MCG,1
Output:	$MCG,1,75

This indicates the magnitude of the IQ modulator is configured to 75%.

5.12 $PWRSGDS – Set ISC board's output power in dBm

This command provides a coarse method to regulate the small signal output power of the ISC board byautomatically configuring the values of the VGA and IQ Modulator to roughly the desired dBm value.

Remark: To use this command, auto-gain must be disabled first (see $AGES).

Syntax:

Input:	$PWRSGDS,[channel],[power dBm]
Output:	$PWRSGDS,[channel],OK

- **[channel]** – Channel identification number.
- **[magnitude]** –The desired small signal output in dBm.

Example:

Input:	$PWRSGDS,1,20
Output:	$PWRSGDS,1,OK

This will set the power output of the ISC board to about 20 dBm (0.1 watt).

5.13 $PPG – Get PA forward and reflected power in watt

This command returns the forward and reflected power of the power amplifier in watt.

Syntax:

Input:	$PPG,[channel]
Output:	$PPG,[channel],[forward power],[reflected power]

- **[channel]** – Channel identification number.
- **[forward power]** – The RF power output of the PA in watt.
- **[reflected power]** – The RF power reflected back into the PA in watt.

Example:

Input:	$PPG,1
Output:	$PPG,1,200.00000,40.00000

This indicates the PA is outputting 200W of forward power, but 40W is being reflected back into the PA.This corresponds to an reflection of 20%.

5.14 $PPDG – Get PA forward and reflected power in dBm

This command returns the forward and reflected power of the power amplifier in dBm.

Syntax:

Input:	$PPDG,[channel]
Output:	$PPDG,[channel],[forward power],[reflected power]

- **[channel]** – Channel identification number.
- **[forward power]** – The RF power output of the PA in watt.
- **[reflected power]** – The RF power reflected back into the PA in watt.

Example:

Input:	$PPDG,1
Output:	$PPDG,1,50.00000,30.00000

This indicates the PA is outputting 50dBm of forward power, but 30dBm is being reflected back.This corresponds to an S11 of -20dB.

5.15 $PWRS – Set PA output power setpoint in watt

This command sets the amplifier chain's output power setpoint to the desired value in watt.

Syntax:

Input:	$PWRS,[channel],[power watt]
Output:	$PWRS,[channel],OK

- **[channel]** – Channel identification number.
- **[power watt]** – The desired power value for the RF signal in watt.

Example:

Input:	$PWRS,1,250
Output:	$PWRS,1,OK

This sets the output power setpoint to 250 W.

5.16 $PWRG – Get PA output power setpoint in watt

This command returns the configured output power setpoint in watt.

Syntax:

Input:	$PWRG,[channel]
Output:	$PWRG,[channel],[power watt]

- **[channel]** – Channel identification number.
- **[power watt]** – The current output power value for the RF signal in watt.

Example:

Input:	$PWRG,1
Output:	$PWRG,1,300.000000

This indicates that the output power setpoint is configured to 300 W.

5.17 $PWRDS – Set PA output power setpoint in dBm

This command sets the output power setpoint to the desired value in dBm.

Syntax:

Input:	$PWRDS,[channel],[power dBm]
Output:	$PWRDS,[channel],OK

- **[channel]** – Channel identification number.
- **[power dBm]** – The desired power value for the RF signal in dBm.

Example:

Input:	$PWRDS,1,50
Output:	$PWRDS,1,OK

This sets the output power setpoint to 50 dBm.

5.18 $PWRDG – Get PA output power setpoint in dBm

This command returns the configured output power setpoint in dBm.

Syntax:

Input:	$PWRDG,[channel]
Output:	$PWRDG,[channel],[power dBm]

- **[channel]** – Channel identification number.
- **[power dBm]** – The current power value for the RF signal in dBm.

Example:

Input:	$PWRDG,1,
Output:	$PWRDG,1,50.000000

This indicates that the output power setpoint is configured to 50 dBm.

5.19 $SWP – Perform S11 sweep with RF output power in watt

This command performs an S11 frequency sweep across a band provided by the user. The output powerduring the sweep is provided in watt.

Remark: The completion time of the command will increase as the number of frequency steps increases.This can make it seem as if the ISC board has become un-responsive for some time.

Remark: This command offers two output modes, which have different output syntaxes.

Syntax:

Input:	$SWP,[channel],[start frequency],[stop frequency],[step frequency],[power watt],[output mode]
Output (mode 0):	$SWP,[channel],[start frequency],[stop frequency],[step frequency],[power watt],[output mode]
Output (mode 1):	$SWP,[channel],[measurement frequency],[forward power], [reflected power]

- **[channel]** – Channel identification number.
- **[start frequency]** – The beginning of the sweep bandwidth in MHz.
- **[stop frequency]** – The end of the sweep bandwidth in MHz.
- **[step frequency]** – The size of the steps taken between each measurement in MHz.
- **[power watt]** – The output power at which the sweep is performed in watt.
- **[output mode]** – Dictates what the sweep will output.0 –

  Return all sweep results.

  It returns a large number of responses in one go: one for every measurement, as well as an OK messageat the end. All responses arrive simultaneously when the command finishes.

  1 – Returns only the best match sweep result. Also, the current frequency and the DLL start frequency(see chapter 6 for the DLL commands and also the AppNote on the digital locked loop algorithm) will automatically be set to the resulting best match frequency.
- **[measurement frequency]** – The frequency at which is measured in MHz. The output frequency value isrounded to the 2nd decimal.
- **[forward power]** – The forward power measured at [frequency] in watt.
- **[reflected power]** – The reflected power measured at [frequency] in watt.

Example 1:

Input:	$SWP,1,2400,2500,10,100,0
Output:	$SWP,1,2400,100.01,20.12 $SWP,1,2410,99.84,20.08 $SWP,1,2420,99.88,19.55 $SWP,1,2430,99.97,19.77 $SWP,1,2440,100.10,19.08 $SWP,1,2450,99.87,18.17 $SWP,1,2460,99.99,7.85 $SWP,1,2470,99.91,2.15 $SWP,1,2480,100.00,6.89 $SWP,1,2490,99.84,14.41 $SWP,1,2500,99.83,18.96 $SWP,1,OK

This executes an S11 sweep in the 2400 - 2500 MHz range with a step size of 10 MHz, at 100 W outputpower.

At 2400 MHz the forward power is 100.01 W and the reflected power is 20.12 W.At 2410

MHz the forward power is 99.84 W and the reflected power is 20.08 W. etc...

When the sweep has run through the entire frequency band "$SWP,1,OK" is returned to indicate it is finished.

Example 2:

Input:	$SWP,1,2400,2500,10,100,1
Output:	$SWP,1,2470,99.91,2.15

This executes an S11 sweep in the 2400 - 2500 MHz range with a step size of 10 MHz, at 100 W outputpower. However this time output mode 1 is used, which returns only the best match result.

At 2470 MHz the forward power is 99.91 W and the reflected power is 2.15 W.

5.20 $SWPD – Perform S11 sweep in dBm

This command performs an S11 frequency sweep across a band provided by the user. The output powerduring the sweep is provided in dBm.

Remark: The completion time of the command will increase as the number of frequency steps increases.This can make it seem as if the ISC board has become un-responsive for a period of time.

Remark: This command offers two output modes, which have different output syntaxes.

Syntax:

Input:	$SWPD,[channel],[start frequency],[stop frequency], [step frequency],[power dBm],[output mode]
Output (mode 0):	$SWP,[channel],[measurement frequency],[forward power], [reflected power] ... $SWP,[channel],OK
Output (mode 1):	$SWP,[channel],[measurement frequency],[forward power], [reflected power]

- **[channel]** – Channel identification number.
- **[start frequency]** – The beginning of the sweep bandwidth in MHz.
- **[stop frequency]** – The end of the sweep bandwidth in MHz.
- **[step frequency]** – The size of the steps taken between each measurement in MHz.
- **[power dBm]** – The output power at which the sweep is performed in dBm.
- **[output mode]** – Dictates what the sweep will output.0 –

  Return all sweep results.

    It returns a large number of responses in one go: one for every measurement, as well as an OK messageat the end. All responses arrive simultaneously when the command finishes.

  1 – Returns only the best match sweep result. Also, the current frequency and the DLL start frequency(see chapter 6 for the DLL commands and also the AppNote on the digital locked loop algorithm) will automatically be set to the resulting best match frequency.
- **[frequency]** – The frequency at which is measured in MHz. The output frequency value is rounded tothe 2nd decimal.
- **[forward power]** – The forward power measured at [frequency] in dBm.
- **[reflected power]** – The reflected power measured at [frequency] in dBm.

Example:

Input:	$SWPD,1,2400,2500,10,50,0
Output:	$SWPD,1,2400,50.00,43.04 $SWPD,1,2410,49.99,43.03 $SWPD,1,2420,49.99,42.91 $SWPD,1,2430,49.99,42.96 $SWPD,1,2440,50.00,42.81 $SWPD,1,2450,49.99,42.59 $SWPD,1,2460,50.00,38.95 $SWPD,1,2470,49.99,33.32 $SWPD,1,2480,50.00,38.38 $SWPD,1,2490,49.99,41.59 $SWPD,1,2500,49.99,42.78 $SWPD,1,OK

This executes an S11 sweep in the 2400 - 2500 MHz range with a step size of 10 MHz, at 50 dBm output power.

At 2400MHz the forward power is 49.99 dBm and the reflected power is 43.04 dBm. At 2410MHz

the forward power is 49.84 dBm and the reflected power is 43.03 dBm.etc...

When the sweep has run through the entire frequency band “$SWPD,1,OK” is returned to indicate it is finished.

Example 2:

Input:	$SWPD,1,2400,2500,10,50,0
Output:	$SWPD,1,2470,49.99,33.32

This executes an S11 sweep in the 2400 - 2500 MHz range with a step size of 10 MHz, at 50 dBm output power. However, this time output mode 1 is used, which returns only the best match result.

At 2470 MHz the forward power is 49.99 dBm and the reflected power is 33.32 dBm.

6. DLL commands

Commands to operate the 'Digital Locked Loop' (DLL) mode.

DLL is a feature which tunes the frequency of the output signal towards the best match in a frequency bandwhere the RF system's return losses are below a user-defined threshold.

6.1 $DLES – Set DLL enabled / disabled

This command turns the DLL mode ON or OFF.

Syntax:

Input:	$DLES,[channel],[enable]
Output:	$DLES,[channel],OK

- **[channel]** – Channel identification number.
- **[enable]** – The desired enable state of the DLL mode.0 – OFF

  (default)

  1 – ON

Example:

Input:	$DLES,1,1
Output:	$DLES,1,OK

This turns on DLL mode.

6.2 $DLEG – Get DLL enable state

This command indicates whether the DLL mode is currently turned ON or OFF.

Syntax:

Input:	$DLEG,[channel]
Output:	$DLEG,[channel],[enable]

- **[channel]** – Channel identification number.
- **[enable]** – The current enable state of the DLL mode.0 – OFF

  (default)

  1 – ON

Example:

Input:	$DLEG,1
Output:	$DLEG,1,1

6.3 $DLCS – Set DLL configuration

This command configures the parameters of the DLL mode.

Syntax:

Input:	$DLCS,[channel],[lower frequency],[upper frequency], [start frequency],[step frequency],[threshold],[main delay]
Output:	$DLCS,[channel],OK

- **[channel]** – Channel identification number.
- **[lower frequency]** – The lower boundary of the bandwidth for DLL in MHz.
- **[upper frequency]** – The upper boundary of the bandwidth for DLL in MHz.
- **[start frequency]** – The frequency at which the DLL starts its activities in MHz.
- **[step frequency]** – The step size of the DLL in MHz.
- **[threshold]** – The match/efficiency threshold in dB to be met before DLL latches onto a frequency.
- **[main delay]** – The delay between complete runs of the DLL in ms.

Example:

Input:	$DLCS,1,2400,2500,2410,5,0.5,25
Output:	$DLCS,1,OK

This sets lower frequency to 2400 MHz, upper frequency to 2500 MHz, start frequency to 2410 MHz, stepfrequency to 5MHz, threshold to 0.5 dB and the main delay to 25 ms.

6.4 $DLCG – Get DLL configuration

This command returns the configured parameters of the DLL mode.

Syntax:

Input:	$DLCG,1
Output:	$DLCG,[channel],[lower frequency],[upper frequency], [start frequency],[step frequency],[threshold],[main delay]

- **[channel]** – Channel identification number.
- **[lower frequency]** – The lower boundary of the bandwidth for DLL in MHz.
- **[upper frequency]** – The upper boundary of the bandwidth for DLL in MHz.
- **[start frequency]** – The frequency at which the DLL starts its activities in MHz.
- **[step frequency]** – The step size of the DLL in MHz.
- **[threshold]** – The match/efficiency threshold in dB to be met before DLL latches onto a frequency.
- **[main delay]** – The delay between complete runs of the DLL in ms.

Example:

Input:	$DLCG,1
Output:	$DLCG,1,2400.000000,2500.000000,2410.000000,5.000000,0.500000,0

This indicates that DLL is configured to function within the range of 2400-2500 MHz, starting off initially ata frequency of 2410MHz. It makes 5MHz steps with a threshold of 0.5dB. There is no delay.

7. PWM commands

Commands to operate the pulse width modulation (PWM) mode.

Pulse Width Modulation allows the user to modulate the RF signal, and in turn the average power output ofthe system, by turning the signal ON and OFF at a set rate.

The following two parameters are used to achieve this:

- PWM frequency – Dictates how often the signal switches between ON and OFF.
- PWM duty cycle – Dictates the time ratio between ON and OFF of each period.

Depending on the duty cycle, the average power output of the system will decrease to a percentage of its set output. For example, 50% duty cycle at 80W results in an average RF power output of 40W.

To ensure sufficient measurement time of the RF signal, the pulses generated by any PWM scheme must not be too short. Hence, the permissible PWM frequency and duty cycle are dependent on each other. The PWM frequency can vary between 1000 Hz – 19800Hz. To ensure accurate power readings (and therefore accurate power output), the minimum value of the duty cycle changes along with the PWM frequency according to the following formula:

*Duty Cycle_min=ROUNDUP(fPWM * T_(min pulse)/10,000)*

Where:

- DC_{min} is the minimum duty cycle as a percentage.
- f_{PWM} is the PWM frequency between 1000 and 19800 Hz.
- T_{min} pulse is the minimum pulse length in microseconds.T_{min}

pulse = 50 µs.

This means that at 1000 Hz, the minimum duty cycle is 5% and at 19800 Hz it is 99%. Going over thisfrequency value would effectively disable PWM, as the minimum duty cycle becomes 100%.

The user needs to keep these limitations in mind – the system will not check for the limits. For a reasonablecontrol range, PWM frequencies between 1 and 2 kHz are recommended.

7.1 $DCS – Set PWM duty cycle

This command sets the PWM duty cycle between 0% and 100%.

Remark: This command doubles as a PWM ON/OFF switch. Setting the duty cycle to 100% is the same asturning PWM off entirely, thus there is no dedicated PWM ON/OFF command.

Syntax:

Input:	$DCS,[channel],[duty cycle]
Output:	$DCS,[channel],OK

- **[channel]** – Channel identification number.
- **[duty cycle]** – A value between 0 and 100 that sets the duty cycle in percent. (default = 100%)

Example:

Input:	$DCS,1,50
Output:	$DCS,1,OK

This sets the PWM duty cycle to 50%.

Please note that the usable PWM frequency range in connection with the duty cycle parameter is limitedto allow a minimum pulse width needed for proper functioning of the autogain functionality (see below).

7.2 $DCG – Get all PWM settings

This command returns all the settings relating to PWM.

Syntax:

Input:	$DCG,[channel]
Output:	$DCG,[channel],[frequency],[reserved],[trigger mode], [reserved],[reserved],[reserved],[reserved],[reserved], [duty cycle]

- **[channel]** – Channel identification number.
- **[frequency]** – The current PWM frequency.
- **[reserved]** – Reserved. Parameter should be ignored.
- **[trigger mode]** – The current operational mode of the PWM triggering.

  1 – Free running

  2 – Reserved. Parameter should be ignored.3 –

  Reserved. Parameter should be ignored.
- **[Reserved]** – Reserved. Parameter should be ignored.
- **[Reserved]** – Reserved. Parameter should be ignored.
- **[Reserved]** – Reserved. Parameter should be ignored.
- **[Reserved]** – Reserved. Parameter should be ignored.
- **[Reserved]** – Reserved. Parameter should be ignored.
- **[duty cycle]** – The current duty cycle percentage value.

Example:

Input:	$DCG,1
Output:	$DCG,1,1000,0,1,255,255,255,255,0.000000,50

This indicates that the operational mode is configured to 'free running'.

The PWM signal is configured to a frequency of 1000 Hz with a duty cycle of 50%.There is no correction factor and no delay.

7.4 $DCG – Get all PWM settings

This command returns all the settings relating to PWM.

Syntax:

Input:	$DCG,[channel]
Output:	$DCG,[channel],[frequency],[correction factor],[trigger mode], [reserved],[reserved],[reserved],[reserved],[delay],[duty cycle]

- **[channel]** – Channel identification number.
- **[frequency]** – The current PWM frequency.
- **[correction factor]** – The current correction factor for the PWM signal.
- **[trigger mode]** – The current operational mode of the PWM triggering.1 – Free running

  2 – Reserved. Parameter should be ignored.3 –

  Reserved. Parameter should be ignored.
- **[Reserved]** – Reserved. Parameter should be ignored.
- **[Reserved]** – Reserved. Parameter should be ignored.
- **[Reserved]** – Reserved. Parameter should be ignored.
- **[Reserved]** – Reserved. Parameter should be ignored.
- **[delay]** – The current percentage-based delay of the PWM signal.
- **[duty cycle]** – The current duty cycle percentage value.

Example:

Input:	$DCG,1
Output:	$DCG,1,1000,0,1,255,255,255,255,0.000000,50

This indicates that the operational mode is configured to 'free running'.

The PWM signal is configured to a frequency of 1000Hz with a duty cycle of 50%.There is no correction factor and no delay.

8. Safe Operating Area (SOA) configuration commands

The following commands provide a powerful suite of parameters to protect the generator system against thermal- and RF overstress. Furthermore, watchdogs can be activated to check hardware and software against hang-ups – also with respect to higher-level system controllers.

The user should take very good care upon using the commands. Wrong configuration may disable protective functions and the unit might be damaged during operation in adverse conditions.

8.1 $SOA – Set SOA configuration

This command configures the enable state of the SOA's protection systems.SOA has

the following protection systems in place:

- Protection against high temperatures.
- Protection against software timeouts / freezes.
- Protection against excessive reflection.
- Auto-disable RF Power if the board status is not polled frequently enough.

Remark: This command does not adhere to established syntax. Its output is unique.

Syntax:

Input:	$SOA,[channel],[temperature enable],[watchdog enable], [reflection enable],[external watchdog enable],[dissipation enable]
Output:	$SOA Tmp:[0/1] S11:[0/1] eWD:[0/1] Diss:[0/1]

- **[channel]** – Channel identification number.
- **[temperature enable]** – Enable state of the temperature protection system.0 – OFF

  1 – ON
- **[watchdog enable]** – Enable state of the software timeout/freeze protection system.This parameter is ignored. Software watchdog is always enabled.
- **[reflection enable]** – Enable state of the RF power reflection protection system.0 – OFF

  1 – ON
- **[external watchdog enable]** – Enable state of the board status polling protection system.0 – OFF

  1 – ON

- **[dissipation enable]** – Enables the dissipation protection, i.e., a maximum amount of dissipated power inside the amplifier can be set. The dissipated power is the sum of the reflected RF power and the dissi-pation due to the RF generation process.

  In the return line:
- **Tmp = [temperature enable]**
- **S11 = [reflection enable]**
- **eWD = [external watchdog enable]**

Example:

Input:	$SOA,1,0,0,0,1,0
Output:	$SOA Tmp:0 S11:0 eWD:1 Diss:0

This enables the external watchdog, but disables the other protection systems, except the softwarewatchdog which is permanently enabled.

8.2 $SOG – Get SOA configuration

This command returns the enable state of the SOA's protection systems.

Syntax:

Input:	$SOG,[channel]
Output:	$SOA Tmp:[0/1] S11:[0/1] eWD:[0/1] Diss:[0,1]

- **[channel]** – Channel identification number.In the

  return line:
- **Tmp = [temperature enable]** – Enable state of the temperature protection system.
- **S11 = [reflection enable]** – Enable state of the reflection protection system.
- **eWD = [external watchdog enable]** – Enable state of the board status polling protection system.

Example:

Input:	$SOG,1
Output:	$SOA Tmp:0 S11:0 eWD:1 Diss:0

This indicates the external watchdog protection system is enabled, but the other protections systems aredisabled, except for the software watchdog, which is permanently enabled.

8.3 $SPS – Set reflected power SOA configuration

This command configures the reflected power values at which SOA takes action.One of the features of the SOA is protection against excessive reflected power.

Excessive reflection occurs when there is a bad match at the output and RF returns to the generator.The SOA has two reactions to excessive dissipation, depending on the severity:

- **If the reflection is high, but still tolerable:**
  Raise a 'High Reflection' error.
- **If the reflection is dangerously high:**
  Raise a 'Shutdown Reflection' error and shutdown RF power.

Syntax:

Input:	$SPS,[channel],[high reflection],[shutdown reflection]
Output:	$SPS,[channel],OK

- **[channel]** – Channel identification number.
- **[high reflection]** – The reflection value in dBm at which the 'High Reflection' situation is signaled by theSOA. It will be reported upon an $ST command.
- **[shutdown reflection]** – The reflection value in dBm at which the 'Shutdown Reflection' reaction isperformed by the SOA: RF will be switched off and the corresponding error bit will be set.

Example:

Input:	$SPS,1,53,54
Output:	$SPS,1,OK

This sets the 'High Reflection' and 'Shutdown Reflection' protection values to 53 dBm (200W) and 54 dBm(250W) respectively.

If the limit of 53 dBm reflection is exceeded, the corresponding bit is set in the status word.If the limit of 54 dBm reflection is exceeded, the RF is shut down by SOA.

8.4 $SPG – Get reflected power SOA configuration

This command returns the reflection values at which SOA takes action.

Syntax:

Input:	$SPG,[channel]
Output:	$SPG,[channel],[high reflection],[shutdown reflection]

- **[channel]** – Channel identification number.
- **[high reflection]** – The reflection value in dBm at which the 'High Reflection' reaction is performed bythe SOA.
- **[shutdown reflection]** – The reflection value in dBm at which the 'Shutdown Reflection' reaction isperformed by the SOA.

Example:

Input:	$SPG,1,
Output:	$SPG,1,53.000000,54.000000

This indicates the 'High Reflection' and 'Shutdown Reflection' protection values are configured to 53 dBm(200W) and 54 dBm (250W) respectively.

8.5 $STS – Set Temperature SOA Configuration

This command configures the temperature values at which SOA takes action.One of the

features of the SOA is protection against excessive temperatures.

Excessive temperatures can occur for any number of reasons: side effects of high RF power reflection,faulty cooling, excessive use, etc.

The SOA has two reactions to excessive temperatures, depending on the severity:

- If the temperature is high, but still tolerable:

  Raise a 'High Temperature' error.

- If the temperature is dangerously high:

  Raise a 'Shutdown Temperature' error and shutdown RF power.

Syntax:

Input:	$STS,[channel],[high temperature],[shutdown temperature]
Output:	$STS,[channel],OK

- **[channel]** – Channel identification number.
- **[high temperature]** – The temperature value in °C at which the 'High Temperature' situation is signaled by the SOA. The corresponding bit in the status word is set and can be read with an $ST command.
- **[shutdown temperature]** – The temperature value in °C at which the 'Shutdown Temperature' reaction is performed by the SOA. The generator will be switched off and the corresponding error bit will be setin the status word.

Example:

Input:	$STS,1,80,90
Output:	$STS,1,OK

This sets the 'High Temperature' and 'Shutdown Temperature' protection values to 80°C and 90°Crespectively.

If the limit of 80°C temperature is exceeded, the status bit is set.

If the limit of 90°C temperature is exceeded, the RF is shut down by SOA and the error bit is set.

8.6 $STG – Get Temperature SOA Configuration

This command returns the temperature values at which SOA takes action.

Syntax:

Input:	$STG,[channel]
Output:	$STG,[channel],[high temperature],[shutdown temperature]

- **[channel]** – Channel identification number.
- **[high temperature]** – The temperature value in °C at which the 'High Temperature' situation is signaledby the SOA.
- **[shutdown temperature]** – The temperature value in °C at which the 'Shutdown Temperature' reactionis performed by the SOA.

Example:

Input:	$STG,1,
Output:	$STG,1,80.0,90.0

This indicates the 'High Temperature' and 'Shutdown Temperature' protection values are configured to 80°C and 90°C respectively.

8.7 $SDS – Set Dissipation SOA Configuration

This command configures the dissipation value at which SOA takes action, as well as the grace period allotted before SOA shuts down RF in case of communication failure with power supplies.

One of the features of the SOA is protection against excessive power dissipation inside a generator.

Excessive power dissipation occurs when an RF system draws a disproportionate amount of current from its power supply (PSU) relative to the amount RF energy that is transmitted into a load. The problem can be further aggravated by reflected RF power as well.

Dissipation is measured in watt. The formula used to calculate it is the following

Dissipation(W) = PSU_Power(W) - FWD_Power(W) + RFL_Power(W)

The SOA has two reactions to excessive dissipation, depending on the amount:

- **If the dissipation is high, but still tolerable:**
  Raise a 'High Dissipation' warning. The respective bit is set in the status word and can be read via a $STcommand.
- **If the dissipation is dangerously high:**
  Raise a 'Shutdown Dissipation' error and shutdown RF power. The respective bit is set in the statusword.

Calculating the dissipation value necessitates communication with a power supply to ascertain the amount of power being drawn. This presents an opportunity for communication failure. Failure to acquire the PSU power means the dissipation cannot be calculated. Normally this would result in an immediate halt of RF power by the SOA, however this may not always be the desired behavior.

To counteract the occasional communication hiccup, a grace timeout period can be allotted to the dissipation SOA, providing an opportunity to recover from the communication fault.

Syntax:

Input:	$SDS,[channel],[high dissipation],[shutdown dissipation], [graceperiod]
Output:	$SDS,[channel],OK

- **[channel]** – Channel identifier number.
- **[high dissipation]** – The dissipation value in watt at which the 'High Dissipation' reaction is performedby the SOA. The corresponding bit is set in the status word.
- **[shutdown dissipation]** – The dissipation value in watt at which the 'Shutdown Dissipation' reaction is performed by the SOA. The corresponding bit is set in the status word and the RF output is switched off.
- **[grace period]** – (Optional) The grace timeout period in milliseconds during which communication withthe power supply may be reestablished.

Example:

Input:	$SDS,1,1000,2000,10
Output:	$SDS,1,OK

This sets the 'High Dissipation' and 'Shutdown Dissipation' protection values to 1kW and 2kW respectively,while the 'Grace Period' is set to 5 seconds.
If the limit of 1kW dissipation is exceeded, the respective status bit is set.
If the limit of 2kW dissipation is exceeded, the respective status bit is set and the RF is shut down by SOA.
If a communication fault occurs with the power supply, RF shutdown is delayed by 10 milliseconds, duringwhich the dissipation SOA attempts to re-establish communication.