

Direct IO commands for the TAMS 1800 Series Switches

The TAMS 1800 Series Switches are controlled via SCPI (Standard Commands for Programmable Instruments) and IEEE 448.2 commands.

Detailed information on SCPI is available at www.sepiconsortium.org.

Conventions

Long and short form commands

We'll show a command string like

STATus

This means that the short form of the command is STAT. The long form of the command is STATUS. The string STATU is not allowed – the commands must be the short form or the complete long form, not something in the middle. The device is not case sensitive: you can actually send “status” and get the same result as “STATUS”.

Optional parts

Optional parts of the command are in square brackets:

[ROUTE:]CLOSe <relay_list>

This means that you can send any of the following commands. They are all equivalent. They all close relay 1.

```
ROUT:CLOS (@1)
ROUTE:CLOSE (@1)
CLOS (@1)
CLOSE (@1)
```

Combining commands into one string

You may combine commands into one string by separating commands with a semicolon, and paying careful attention to the placement of the colon character. The colon character separates one level of a command from the others, as in

```
TRIGGER:SOURCE EXTERNAL
```

In this example, TRIGGER is the first level and SOURCE EXTERNAL is the second level. When two commands are combined in one string, as in

```
TRIGGER:SOURCE EXTERNAL ; COUNT 10
```

then the LACK of a colon after the semicolon causes this to be equivalent to

```
TRIGGER:SOURCE EXTERNAL ; :TRIGGER:COUNT 10
```

A semicolon separator must be followed by either one of these:

1. A complete command starting with a colon and a first level command
2. A command at the same level as the last command before the semicolon

The commands that begin with a star character (“*”) can be used without regard to colons and levels.

When the colon is not given, it is as though the first part of the first command is repeated, which is very useful for shortening command strings when the two commands share the same first part.

When a colon is used after the semicolon, then the complete second command must be given, as in:

```
TRIGGER:SOURCE EXTERNAL ; :CLOSE (@1)
```

The spaces around the semicolon are not needed. They are only here for printing clarity.

Entering numbers

The TAMS 1800 Series Switches accept whole numbers with no spaces or commas or exponents or signs. See the specific commands for the range of numbers allowed.

Command Reference

ABORT

Abort stops a scan in progress. If no scan is in progress, the command is ignored. See Scanning Overview.

CONFigure:EXTernal [:TRIGger] [:OUTput] 0

CONFigure:EXTernal [:TRIGger] [:OUTput] OFF

These two commands turn off the external trigger output. By default, it is off. Example short commands are:

```
CONF:EXT 0
CONF:EXT OFF
```

Example long commands are:

```
CONFIGURE:EXTERNAL:TRIGGER:OUTPUT 0
CONFIGURE:EXTERNAL:TRIGGER:OUTPUT OFF
```

CONFigure:EXTernal [:TRIGger] [:OUTput] 1

CONFigure:EXTernal [:TRIGger] [:OUTput] ON

These two commands turn on the external trigger output. By default, it is off. Example short commands are:

```
CONF:EXT 1
CONF:EXT ON
```

Example long commands are:

```
CONFIGURE:EXTERNAL:TRIGGER:OUTPUT 1
CONFIGURE:EXTERNAL:TRIGGER:OUTPUT ON
```

When the external trigger is on, the external trigger pin will be driven to logic low for approximately 2-4uS when all of the following occur:

1. Scanning mode is on (see Scanning Overview)
2. A relay is closed and settled
3. Any delay set by the DELAY command is finished

INITiate

This command starts a scanning operation. See Scanning Overview. See ABORT to stop a scan.

[ROUTe:][CHANnel:]DELAy <milliseconds>

Set the channel delay, in milliseconds. See Scanning Overview. The default value at power on and after a *RST is 0. This example sets the delay to 30mS:

```
DEL 30
```

The delay can be set from 0 to 60000 milliseconds. If the device is not scanning, the delay is ignored.

[ROUTe:][CHANnel:]DELAy?

Query the channel delay in milliseconds. See Scanning Overview. Example:

```
DEL 200
DEL?
```

the device would return

```
200
```

[ROUTe:]CLOSE <relay-list>

Close the relays in the list. See relay_list below. Examples:

```
CLOSE (@1)           -- close relay number 1
CLOSE (@1:5)         -- close relays 1, 2, 3, 4, and 5
CLOSE (@1, 2, 3, 10:15, 60) -- close relays 1, 2, 3,
                          10, 11, 12, 13, 14, 15, and 60
```

The device will close the relays as fast as possible. The delay parameter set, if any, has no effect on the CLOSE command. The switch device will not wait until each relay is closed and settled before moving to the next relay or returning. If you need to wait until the relays have settled, see the *OPC and *WAI commands.

[ROUTe:]CLOSE ? <relay_list>

Return a comma-separated list of “1”s for each closed relay and “0”s for each open relay. See relay_list below. Example:

```
OPEN ALL
CLOSE (@1, 3, 5)
CLOSE? (@1:5)
```

The device would return:

```
1,0,1,0,1
```

[ROUTE:]CLOSE:STATE ?

Return the list of closed relays. Examples:

```
OPEN ALL
CLOSE (@1, 3, 5)
CLOSE?
```

The device would return:

```
1,3,5
```

[ROUTE:]OPEN <relay_list>

Open the relays in the list. See relay_list below. Examples:

```
OPEN (@1)          -- open relay number 1
OPEN (@1:5)        -- open relays 1, 2, 3, 4, and 5
OPEN (@1, 2, 3, 10:15, 60)  -- open relays 1, 2, 3,
                             10, 11, 12, 13, 14, 15, and 60
```

The device will open the relays as fast as possible. The delay parameter set, if any, has no effect on the OPEN command. The switch device will not wait until each relay is opened and settled opening the next relay or before returning. If you need to wait until the relays have settled, see the *OPC and *WAI commands.

[ROUTE:]OPEN ? <relay_list>

Return a comma-separated list of “1”s for each opened relay and “0”s for each closed relay. See relay_list below. Example:

```
OPEN ALL
CLOSE (@1, 3, 5)
OPEN? (@1:5)
```

The device would return:

```
0,1,0,1,0
```

[ROUTE:]SCAN [:LIST] <relay_list>

Set the scan list. See relay_list below. See Scanning Overview below. Examples:

```
SCAN (@1:5)      -- set the scan list to 1, 2, 3, 4, 5
SCAN (@1:3,10)  -- set the scan list to 1, 2, 3, 10
```

[ROUTE:]SCAN [:LIST] ?

Return a comma-separated scan list. See Scanning Overview below. Examples:

```
SCAN (@1:3,10)  -- set the scan list to 1, 2, 3, 10
SCAN ?
```

The device would return:

```
1,2,3,10
```

[ROUTE:]SCAN CLEAR

Clear the scan list. See Scanning Overview below.

[ROUTE:]SCAN:SIZE ?

Return the number of relays in the scan list. See Scanning Overview below. Examples:

```
SCAN (@1:3,10)  -- set the scan list to 1, 2, 3, 10
SCAN:SIZE ?
```

The device would return:

```
4
```

STATus:OPERation:ENABLE <mask>

STATus:OPERation:ENABLE ?

Set and query the Status Operation Enable mask. See SCPI Status Commands.

Valid mask values are from 0 – 255.

STATus:OPERation [:EVENT] ?

Query the Status Operation Event register. See SCPI Status Commands. This register is cleared on read.

STATus:PRESet

Clear the Status Enable register to 0. See SCPI Status Commands

SYSTEM:ERRor?

Return the last error message from the device. This also clears the error message.

Example messages:

```
+0,"No error"  
+1,"No scan list"
```

For a table of common errors and remedies, see the Installation and Operation Manual.

SYSTEM:STATe:DELeTe <state-number>

Delete the given state number, making it available for storing another instrument state. State-number can be from 100-120 inclusive.

SYSTEM:STATe:DELeTe ALL

Delete all state numbers, making them available for storing another instrument state.

SYSTEM:VERSion?

Return the version of the SCPI standard to which this device adheres.

TRIGger:IMMediate

Send a trigger to the device. Trigger source should be HOLD. See Scanning Overview.

TRIGger:COUNT <number>

TRIGger:COUNT ?

Set or query the trigger count. Default is 1. Valid values are 1 – 65000. See Scanning Overview.

TRIGger:SOURce <source>

TRIGger:SOURce ?

Set or query the trigger source. The default at power on or after a *RST is IMMEDIATE. See Scanning Overview. The <source> must be one of the following:

- EXTernal – Trigger will come from the Trigger Input line
- IMMEDIATE – The device will trigger itself immediately
- TIMer – The device will trigger itself at a fixed time interval
- BUS – Trigger will come from the host with a *TRG or GroupExecuteTrigger
- HOLD – Trigger will come from the host with a TRIG:IMM command
- MIX – Trigger will come from either a BUS source or an EXTernal source

TRIGger:TIMer <milliseconds>

TRIGger:TIMer ?

Set or query the Trigger Timer in milliseconds. During a scan, if the Trigger Source is TIMer, then N milliseconds after a relay is closed, the device will trigger itself for that relay to be opened and the next relay to be closed. See Scanning Overview. The range of timers is 1 millisecond to 60000 milliseconds.

During a scan, if the Trigger Source is TIMer, and the DELay has been set to a non-zero value, then N milliseconds after a relay is closed and the DELay has been observed, the device will trigger itself for that relay to be opened and the next relay to be closed.

This example will set the timer to 50mS and then cause the device to return “50”:

```
TRIG:TIM 50
TRIG:TIM?
```

***CLS**

This clears the Standard Event Register (ESR), the Status Byte (STB), any output message to be sent to the controller, and the error number.

***ESE <value>**

***ESE?**

These commands set and query the ESE register. The value of this register is from 0-255. The ESE register serves as a “mask” between the ESR register and bit 5 of the STB register. See 488 Status.

***ESR?**

This command queries the value of the ESR register. The value will be between 0 and 255. The ESR register will then be cleared after the read. See 488 Status.

***IDN?**

This command returns a string that identifies the manufacturer, model, serial number, and firmware revision of the device. For example,

```
*IDN?
```

causes the device to return

```
TAMS INC,1805A,SN00001,1.0
```

***OPC**

This command (Output Complete) can be used to synchronize the device with the host program. This command will cause the device to wait until all pending commands have completed. Then it will set bit 0 of register ESR to the value “1”. See 488 Status. See Waiting Until Commands are Complete.

***OPC?**

This command (Output Complete) can be used to synchronize the device with the host program. This command will cause the device to wait until all pending commands have completed. Then it will set bit 0 of register ESR to the value “1”. Then it will cause the device to return a “1”. See 488 Status. See Waiting Until Commands are Complete.

***RCL <state>**

This command (Recall state) can be used to recall a saved state. See *SAV.

***RST**

This command (Reset) resets the device. The following actions occur:

- All status registers are cleared
- All relays are opened.
- Any scan is aborted
- Delay = 0
- Scan/Trigger count = 1
- Trigger source is set to IMMEDIATE
- Trigger Output is turned off
- Trigger Timer = 0

***SAV <state>**

This command (Save state) can be used to save the state of the relays in a state register. The state registers are numbered from 100 – 120. This can be an efficient way to move from one state to another. See *RCL and SYSTem:STATe:DELEte.

***SRE <value>**

***SRE?**

These commands set and query the SRE register. The value must be between 0 and 255. The SRE register serves as a mask between the STB register and the SRQ logic. In order to get an SRQ, the appropriate bit of SRE must set. See 488 Status.

***STB?**

This command queries the STB or Status Byte register. The value will be between 0 and 255. See 488 Status.

***TRG**

This command sends a trigger from the host to the device. See Scanning Overview.

***TST?**

This command performs a quick self-test and returns the result. The state of the relays is not disturbed.

***WAI**

This command can be used to synchronize the device with the host program. This command will cause the device to wait until all pending commands have completed. See Waiting Until Commands are Complete.

Relay_list

A relay_list is defined as:

(@ <relay_list2>)

where relay_list2 is defined as a comma separated list of relay-elements. A relay element is defined as:

relay_number OR
relay_number : relay_number

The colon specifies a range of relay numbers, inclusive. For example,

CLOSE (@1:5)
CLOSE (@1,2,3,4,5)
CLOSE (@1,2:4,5)

are all equivalent.

For the TAMS 1848A 4 x 8 Matrix Switch, the relays are numbered as follows:

11, 12, 13, 14, 15, 16, 17, 18
21, 22, 23, 24, 25, 26, 27, 28
31, 32, 33, 34, 35, 36, 37, 38
41, 42, 43, 44, 45, 46, 47, 48

These are valid ranges for this device:

11:18, 21:22, 31:37, 41:48

and this is NOT valid:

11:48 -- there is no relay 19, 20, 29, 30, 39, 40

Scanning Overview

Scanning is a procedure where a list of relays is, one by one, closed, used in a measurement, and then opened. The scan continues until each relay in the list has been used in a measurement.

Scanning is a very powerful, flexible, high-speed way of making a measurement across a number of channels. For example, you can make a voltage measurement across each channel via hardware triggering and synchronization between the switch and the voltmeter, with no interference by the host computer.

Before a scan is started, the following should be set as needed:

Set the list of relays to be scanned

-- Which relays are going to be stepped through?

Set the channel delay

-- After a relay is closed and settled, how long of a delay before the device sends trigger output?

Set the trigger source

-- Where will the trigger come from?

EXTernal – from the Trigger Input line

IMMEDIATE – the device will trigger itself immediately

TIMER – the device will trigger itself at a fixed time interval

BUS – from the host with a *TRG or GroupExecuteTrigger

HOLD – from the host with a TRIG:IMM command

MIX – either a BUS source or an EXTernal source

Set the trigger timer, if trigger source = timer

Set the scan/trigger count

-- How many times does the device run through the relay-list?

Set the external trigger on or off

-- Does the device send an external trigger after every relay close?

Let's walk through an example. This sequence of commands:

```
SCAN (@1:5)           -- scan channels 1, 2, 3, 4, 5
DELAY 30              -- pause for 30mS after each channel is closed
TRIG:SOURCE MIX      -- trigger can come from the host or the Trigger In
                    port
TRIG:COUNT 2        -- scan the list twice
CONF:EXT ON          -- send an external trigger after each relay
```



```
SCAN (@1:5)::TRIG:SOURCE TIM;TIM 30
INIT
CLOSE? (@1:5)
```

The CLOSE query might be expected to return “0,0,0,0,1” indicating “the scan is finished, only the last channel is closed”. But a more likely result may be “1,0,0,0,0” since the INIT command returns immediately. If you really want to wait until the scan is completed, code this:

```
SCAN (@1:5)::TRIG:SOURCE TIM;TIM 30
INIT;*OPC
CLOSE? (@1:5)
```

*OPC will cause the device to wait until the INIT command is completed (which will be after the entire scan is completed). The device will then return the expected

```
0,0,0,0,1
```

indicating that every channel was scanned, and the last channel was left closed.

We could also use the query form of *OPC?, which also waits until the command is complete, and then causes the device to return a “1”.

```
SCAN (@1:5)::TRIG:SOURCE TIM;TIM 30
INIT;*OPC?
```

Here we must read the device and verify the “1” is returned. Then we can query the switches:

```
CLOSE? (@1:5)
```

*OPC? is useful when the host needs to wait until the device is done processing the command. Note that in this *OPC example:

```
SCAN (@1:5)::TRIG:SOURCE TIM;TIM 30
INIT;*OPC
CLOSE? (@1:5)
```

the “CLOSE? (@1:5) will be sent to the device while the scan is proceeding, but the device won’t act upon this command until after the INIT has finished. In the *OPC? example,

```
SCAN (@1:5)::TRIG:SOURCE TIM;TIM 30
INIT;*OPC?
```

the read command will be suspended until the “1” is returned after the INIT and *OPC? have been processed.

*WAI is very similar to *OPC, so our example could also be coded:

```
SCAN (@1:5)::TRIG:SOURCE TIM;TIM 30
INIT;*WAI
CLOSE? (@1:5)
```

For long scans, you probably have to adjust the timeout value of the IO library, since many default to 2 seconds or so. Set the timeout value to be larger than the expected time of the scan.

SCPI Status Commands

The SCPI Status Commands can be used to query if the device is waiting for a trigger or has started a scan. Additionally, the device could be setup (with the 488.2 Status Commands) to provide an SRQ on these two events. Most programs don't need this detailed functionality, but it is included for advanced applications.

The SCPI Status commands pertain to three registers. Each register is 8 bits.

The Status Condition register is set by the device itself, and cannot be queried.
The Status Operation Event register is set by the device itself, and it can be queried.
The Status Enable register can be set and queried by the host.

The only two bits that are defined in these registers are bit 0 (the LSB, value weight “1”), which is set if and only if the device is waiting for a trigger, and bit 4 (value weight “16”) which is set when the scan starts.

To query the status of the device:

```
STAT:OPER?
```

The device will respond with one of 4 values from the Status Operation Event register:

0	-- the device is not scanning and is not waiting for a trigger
1	-- the device is waiting for a trigger
16	-- the device has started a scan
17	-- the device has started a scan and is waiting for a trigger

Then the device will clear the Status Operation Event register.

To set up the device to provide an SRQ when scan starts or when waiting for trigger, bit 7 (value weight 128) of the 488 Status Register Enable must also be set.

To set up the device to provide an SRQ when the device is waiting for a trigger:

```
STAT:OPER:ENABLE 1
*SRE 128
```

To set up the device to provide an SRQ when the scan starts:

```
STAT:OPER:ENABLE 16
*SRE 128
```

To set up the device to provide an SRQ when the device is waiting for a trigger or the scan starts:

```
STAT:OPER:ENABLE 17
*SRE 128
```

To disable SRQs from the SCPI Status system:

```
STAT:OPER:ENABLE 0
*SRE 128
```

488 Status

The 488 Status commands can be used to detect the following events:

- has device power been interrupted?
- has a device error occurred?
- has the device operation completed?
- is there a message available for the host?

The bit definitions for these events are as follows:

Event	Register.bit	Mask	Bit Weight
Device power on	ESR.7	ESE.7	128
Device Command error	ESR.5	ESE.5	32
Device Execution error	ESR.4	ESE.4	16
Device Query error	ESR.2	ESE.2	4
Operation Complete	ESR.0	ESE.0	1
Message available	STB.4	SRE.4	16

For example, to query if there has been an error,

*ESR?

will cause the device to return the value of the ESR register, which your program would then examine to determine the nature of the error. Of course, your program could always just issue an error query such as

SYST:ERR?

and get the exact error.

To determine if the device has a message to send back to the host,

*STB?

will return the Status Byte STB register, and the test program can then examine bit 4.

To get an SRQ on these events in register ESR, the appropriate mask bits must be set. The mask bit register for ESR is ESE. In addition, bit 5 of SRE must be set as well. For example to get an SRQ on Operation Complete, send the following commands.

*ESE 1 -- set bit 1 of the Standard Event Enable register
*SRE 32 -- set bit 5 of the Status Register Enable register

To get an SRQ on error, send the following commands.

*ESE 52 -- set bits 2,4, and 5
*SRE 32 -- set bit 5