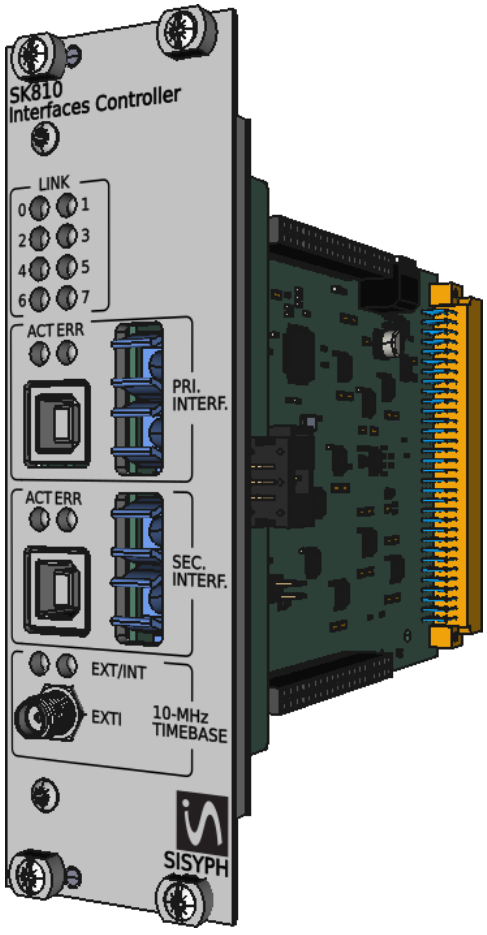
Programming Guide

SK810 *Tarn*
Interfaces Controller

SK-Series Modules



SISYPH
Signals & Systems for Physics

www.sisyph.com

General Information

Important Notice

Information in this document is subject to change without notice.

Signals and Systems for Physics
BP90406
16 place saint-Georges
F31000 Toulouse France
Phone (+33) 781 547 391
www.sisyph.com

Scope

This document describes operating the SK810 module over the remote interfaces.

Contents

General Information **2**
Important Notice . . . 2
Scope . . . 2

1 Introduction **4**
1.1 Power-on Configuration . . . 4
1.2 Buffers . . . 4
1.3 Link model . . . 4
1.4 Command syntax . . . 5
1.5 Examples . . . 5

2 List of Commands **6**
2.1 Instrument Settings commands . . . 8
2.2 Instrument Configuration commands . . . 12
2.3 Instrument Monitoring commands . . . 14
2.4 Status Reporting commands . . . 18
2.5 Interface commands . . . 35
2.6 Memory commands . . . 44

3 Status Model **46**
3.1 Master Summary Status (MSTS) . . . 47
3.2 Master Summary Enable (MSTE) . . . 47
3.3 Event Status (EVTS) . . . 48
3.4 Event Enable (EVTE) . . . 48
3.5 Instrument Status (INSS) . . . 49
3.6 Instrument Enable (INSE) . . . 49
3.7 Instrument Condition (INSC) . . . 49
3.8 /STATUS Lines (STAS) . . . 50
3.9 /STATUS Lines Enable (STAE) . . . 50
3.10 /CTS Lines (CTSS) . . . 51
3.11 /CTS Lines Enable (CTSE) . . . 51
3.12 Overload Status (OVLS) . . . 52
3.13 Overload Enable (OVLE) . . . 52
3.14 Overload Condition (OVLC) . . . 52
3.15 Communication Status (COMS) . . . 53
3.16 Communication Enable (COME) . . . 53
3.17 Last Command Error (LCMD) . . . 54
3.18 Last Execution Error (LEXE) . . . 54
3.19 Last Instrument Error (LINS) . . . 54
3.20 Last User Request (LURQ) . . . 54

4 Index of commands **55**

5 Document Revision History **56**
5.1 Version Number . . . 56
5.2 Revision History . . . 56

1 Introduction

The SK810 Interfaces Controller is a part of the SK-Series Platform on which the modular instruments are assembled. The platform provides power distribution, clock synchronization, individual module status and computer interface. Power distribution excepted, all these functionalities are under control of the SK810. It provides indeed, i) buffered, multiplexed communications between a host computer and up to 8 instruments of the SK-Series Modules through both optical fibre and USB interfaces; ii) clock sources selection; iii) status lines reporting and iv) monitoring functionalities.
The host computer communicates with the SK810 through its Primary and Secondary host interfaces, which can be either USB or optical fibre. All remote interfaces are active and available simultaneously on the SK810. Data is encoded on these host interfaces as asynchronous data using standard UART timing protocol : 8 data bits, no parity, one stop bit, no hardware hand-shaking, 9600 baud or 115 200 baud depending on the module settings. Whatever the baudrate selected for the remote control interfaces, the SK810 always communicates with the modular instruments at a fixed baudrate of 9600 baud.
Remote operation of the SK810 is through a simple command language documented in this chapter. Both set and query forms of most commands are supported, allowing the user complete control of the module from a remote computer. While SK810 has no direct instrumentation itself, its Primary interface acts as a transparent link to one of up to 8 downstream instruments plugged to the SK-Platform backplane's slots. During linked operation, the Secondary interface is still available to process the SK810's specific commands.

1.1 Power-on Configuration

The settings for the Primary and Secondary remote interfaces are 9600 baud (or 115 200 baud) with no parity and no hardware flow control, and local echo disabled (`CONS 0`). Refer to the SK810 *User's Guide* on-line for details on the selection of the host interfaces baudrate.
Most of the instrument settings are stored in non-volatile memory and can be retreived using the appropriate commands. At power-on the instrument returns to the state noted in the command descriptions. Reset values (`*RST` command) of parameters are shown in **boldface**.

1.2 Buffers

Except during link mode, the SK810 stores incoming bytes from the remote interfaces in separate 128-byte buffers. Characters accumulate in the input buffer until a command terminator (either `<CR>` or `<LF>`) is received, at which point the message is parsed and executed. Query responses from the instrument are sent when they are ready without any flow control nor output buffering. The input buffer is automatically flushed upon detecting an overflow, and an error is recorded in the EVTS status register.

1.3 Link model

The SK810 uses a link framework for providing communications between a host computer and the downstream instruments assembled in a SK-Series platform connected by USB or optical fibre. In this model, when a link is established, the Primary interface is linked to a single instrument : data bytes received from the Primary interface (USB or optical fibre) are relayed directly to the instrument and response data are relayed back to the Primary remote interface. When linked, the front panel indicator for the selected instrument is illuminated.
The Secondary interface, which can not be linked, remains available for regular commanding to the SK810. This interface can be used to reconfigure the SK810 or to query the status registers, or any other command documented here. The linked Primary interface, however, will not be processed (parsed) by the SK810's

commands transmitted to the SK810 *via* the linked remote interface will be relayed byte-for-byte to the linked instrument, and not interpreted as SK810 commands.
The link state can be exited by transmitting the escape character from the host computer to the linked Primary interface. The escape character is the "!" character (ASCII code 33). When operating in not-linked mode, the Primary interface can process regular SK810's commands; there is no difference between Primary and Secondary interfaces in this mode.

1.4 Command syntax

The four letter mnemonic (shown in CAPS) in each command sequence specifies the command. The rest of the sequence consists of parameters. The command parser accepts only uppercase mnemonics.
Commands may take either set or query form, depending on whether the `?` character follows the mnemonic. *Set only* commands are listed without the `?`, *query only* commands show the `?` after the mnemonic, and *optionally query* commands are marked with a `(?)`.
Parameters shown in `{ }` and `[ ]` are not always required. Parameters in `{ }` are only required to set a value, and should be omitted for queries. Parameters in `[ ]` are optional in both set and query commands. Parameters listed without any surrounding characters are always required.
Do not send `()` or `{ }` or `[]` as part of the command.
Multiple parameters are separated by commas. Multiple commands may be sent on one command line by separating them with semicolons `;` so long as the input buffer does not overflow. Commands are terminated by either `<CR>` or `<LF>` characters. Null commands and whitespace are ignored. Execution of the command does not begin until the command terminator is received.
The following table summarizes the notation used in the command descriptions:

Symbol	Definition
b	Boolean
i, m, n	Unsigned integers
u, v	Signed integers
(?)	Required for queries; illegal for set commands.
p	Parameter always required.
{*p*}	Required parameter for set commands; illegal for queries.
[*p*]	Optional parameter for both set and query forms.

1.5 Examples

Each command is provided with a simple example illustrating its usage. In these examples, all data sent by the host computer to the instrument are set as `straight teletype font`, while responses received the host computer from the instrument are set as *`slanted teletype font`*. The usage examples vary with respect to set/query, optional parameters, and token formats. These examples are not exhaustive, but are intended to provide a convenient starting point for user programming.

2 List of Commands

This section provides syntax and operational descriptions for remote commands.

2.1 Instrument Settings commands **8**
RTSS (/RTS Lines Status) 8
SLTS (/SLOT Lines Status) 9
SLTE (/SLOT Lines Enable) 10
LINK (Link) 11
2.2 Instrument Configuration commands **12**
PCFG (Power Monitoring Configuration) 12
SYNS (Synchronization Source Select) 13
2.3 Instrument Monitoring commands **14**
PMON (Power Supply Voltages) 14
PWGD (Power Good Signal) 15
TDIE (Die Temperature) 16
XCKD (External Clock Detected) 17
2.4 Status Reporting commands **18**
*CLS (Clear Status Registers) 18
MSTS (Master Summary Status) 19
MSTE (Master Summary Enable) 20
EVTS (Event Status) 21
EVTE (Event Enable) 22
COMS (Communications Status) 23
COME (Communications Enable) 24
OVLS (Overload Status) 25
OVLE (Overload Enable) 26
OVLC (Overload Condition) 27
INSS (Instrument Status) 28
INSE (Instrument Enable) 29
INSC (Instrument Condition) 30
STAS (/Status Lines Status) 31
STAE (/Status Lines Enable) 32
CTSS (/CTS Lines Status) 33
CTSE (CTS Lines Enable) 34
2.5 Interface commands **35**
*RST (Reset) 35
*OPC (Operation Complete) 36
CONS (Console Mode) 37
*IDN (Identify) 38
LINS (Last Instrument Error Status) 39
LURQ (Last User Request Status) 40
LCMD (Last Command Error Status) 41
LEXE (Last Execution Error Status) 42
TERM (Response Termination) 43

2.6 Memory commands 44
*RCL (Recall Settings) 44
*SAV (Save Current Settings) 45

2.1 Instrument Settings commands

The Instrument Settings commands provide control of the instrument's physical parameters.

RTSS (/RTS Lines Status)

Group	Instrument Settings commands
Action	Set/Query
Syntax	`RTSS(?)[`*n*`]{`*m*`}`
Description	Set (query) the /RTS line [bit-mask *n*] {to bit-mask *m*}.
	This command is used to drive the /RTS lines of the backplane slots. The execution of `RTSS 0` will de-assert all lines while querying `RTSS? 0` will be interpreted as the query `RTSS?` (preferred syntax). The set-form command will also de-assert the lines outside the bit-mask. For instance, `RTSS 33` will assert the lines /RTS#0 and /RTS#5. This functionality of driving the /RTS Lines is provided for hardware flow control purpose even if the SK-Series modules do not actually implement this feature. Nevertheless, the RTSS command could be useful, for instance, when the /RTS lines are used to synchronize the instruments from the host computer.
Power-on value	0
Reset value (`*RST`)	no change
Memory (`*SAV`, `*RCL`)	commands not apply
Example	`RTSS 2`; `RTSS? 2`
	2
Related commands	CTSS, STAS, SLTS .

SLTS (/SLOT Lines Status)

Group	Instrument Settings commands
Action	Query only
Syntax	`SLTS? [`*n*`]`
Description	Read the /SLOT Lines Status register [bit-mask *n*].
	The SLTS register value corresponds to the asserted /SLOT lines, which are driven by the slave modules plugged to the backplane. An internal task reads the logical level of all /SLOT lines and updates the SLTS register with the complement of the reading. This value is updated every 100ms. For example, if only positions 0 and 2 were occupied, the related slot lines would be asserted, *i.e.* /SLOT0 = 0 and /SLOT2 = 0, and executing the `SLTS?` command should return $2^2+2^0 = 5$. It is recommended to read the SLTS register's value before attempting to establish a link.
Power-on value	0
Reset value (`*RST`)	no change
Memory (`*SAV`, `*RCL`)	commands not apply
Example	`SLTS?`
	2
Related commands	SLTE, LINK.

SLTE (/SLOT Lines Enable)

Group	Instrument Settings commands
Action	Set/Query
Syntax	`SLTE(?)[`n`]{`m`}`
Description	Set (query) the /SLOT Lines Enable register [bit-mask n] {to bit-mask m}. The SLTE register is used to specify which slot will be linked upon executing the LINK command. The execution of `SLTE 0` will clear all bits while querying `SLTE? 0` will be interpreted as the byte query `SLTE?` (preferred syntax). The set-form command will also clear the bits outside the bit-mask. Because SLTE is the command used to establish a link, it is recommended prior to execute the SLTS command to check whether if the slot is really occupied. Since only one module can be linked at a time, the bit-mask used to set the linked slot must correspond to a single bit, therefore $\{\mathbf{0}, 1, 2, 4, 8, 16, 32, 64, 128\}$ are the only accepted input arguments for the set-form command. For instance, in order to link the slot#5 to the Primary interface, the command `SLTE 32` must be executed. Changing the SLTE register's value is not allowed when a link is established.
Allowed range	$m \in \{\mathbf{0}, 1, 2, 4, 8, 16, 32, 64, 128\}$ where : $\mathbf{0} \longleftrightarrow$ none of the slots can be linked, $1 \longleftrightarrow$ Slot#0 allowed for link, $2 \longleftrightarrow$ Slot#1 allowed for link, $4 \longleftrightarrow$ Slot#2 allowed for link, $8 \longleftrightarrow$ Slot#3 allowed for link, $16 \longleftrightarrow$ Slot#4 allowed for link, $32 \longleftrightarrow$ Slot#5 allowed for link, $64 \longleftrightarrow$ Slot#6 allowed for link, $128 \longleftrightarrow$ Slot#7 allowed for link.
Power-on value	0
Reset value (`*RST`)	**0**
Memory (`*SAV`, `*RCL`)	0
Example	`SLTE 32`
Related commands	SLTS, LINK.

LINK (Link)

Group	Instrument Settings commands
Action	Set/Query
Syntax	`LINK(?){`*b*`}`
Description	Set (query) the link {to *b*} between the Primary Interface and the specified slot. The LINK command controls (queries) the operation of the link between the Primary interface and the slot previously specified by the SLTE byte's content. The link is established (resp. disabled) when the boolean argument `b` is 1 (resp. 0). Upon executing `LINK 1`, the link is transparent and no incoming characters will be parsed. Before establishing the link, a test is performed to compare SLTS and SLTE registers in order to prevent any link attempt with an unoccupied slot. If the link can not be established an error is reported. To terminate a link session from the Primary linked remote interface, just send the escape character `!` (ASCII code 33). Because during linked mode operation the Secondary interface still operates as a regular remote interface, the established link can also be destroyed by sending the `LINK 0` command *via* this interface.
Allowed range	$b \in \{\mathbf{0}\,(\text{not-linked}), 1\,(\text{linked})\}$
Power-on value	0
Reset value (`*RST`)	**0**
Memory (`*SAV`, `*RCL`)	0
Example	`LINK 1; LINK ?` *1*
Related commands	

2.2 Instrument Configuration commands

The Instrument Configuration commands provide control of the instrument's physical functionalities.

PCFG (Power Monitoring Configuration)

Group	Configuration commands
Action	Set/Query
Syntax	`PCFG(?){`*m*`}`
Description	Configure (query) the under-voltage detector {to *m*}. The DC-power supply voltages are monitored periodically (100 ms) to detect an under-voltage operation. The input argument of the set command specifies what power supplies are monitored by the under-voltage detector. This setting allows the user to discard some power supplies from the test, preventing to signal an error to the modules mounted on the backplane. This could be useful when some power supplies are not present. For each monitored power supply, the threshold value is set to 10 % below the nominal level.
Allowed range	$m \in \{0, \mathbf{1}, 2, 3, 4\}$ where: $0 \longleftrightarrow$ all power supplies are monitored; $\mathbf{1} \longleftrightarrow$ detection restricted to ± 15 V and $+5$ V only; $2 \longleftrightarrow +24$ V discarded from detection; $3 \longleftrightarrow -5$ V discarded from detection; $4 \longleftrightarrow$ under-voltage detection disabled.
Power-on value	restored from non-volatile memory.
Reset value (`*RST`)	**1**
Memory (`*SAV`, `*RCL`)	commands apply.
Example	`PCFG 1; PCFG?` *1*
Related commands	PMON, PWGD

SYNS (Synchronization Source Select)

Group	Configuration commands
Action	Set/Query
Syntax	`SYNS(?){`*m*`}`
Description	Set (query) the Synchronization signal source {to *m*}. The set-form command provides the user with the selection of what clock source is used to drive the Synchronization lines of the back-plane. The internal clock is provided by the micro-controller time base (10 MHz) whereas the external source is routed from the the front-panel SMA connector. Because the command does not check whether the external signal is present or not, the presence of the external time-base should be checked *via* the XCKD command before selecting this source.
Allowed range	$m \in \{0, \mathbf{1}, 2\}$ where : $0 \longleftrightarrow$ no clock selected, the Synchro lines are not driven; $\mathbf{1} \longleftrightarrow$ internal clock (10 MHz) selected; $2 \longleftrightarrow$ external clock selected;
Power-on value	restored from non-volatile memory.
Reset value (`*RST`)	**1**
Memory (`*SAV`, `*RCL`)	commands apply.
Example	`SYNS 2`
Related commands	XCKD

2.3 Instrument Monitoring commands

The Instrument Monitoring commands provide the host computer with the last measurements of the instrument's physical parameters.

PMON (Power Supply Voltages)

Group	Monitoring commands
Action	Query only
Syntax	`PMON?`*m*
Description	Return the last measurement of the specified power supply m, in mV. The power supply voltages are periodically acquired (100 ms) by an internal task. The command returns the last reading of the power supply specified by the input argument.
Allowed range	$m \in \{0, 1, 2, 3, 4\}$ where: $0 \longleftrightarrow -15$ V power supply reading; $1 \longleftrightarrow +15$ V power supply reading; $2 \longleftrightarrow -5$ V power supply reading; $3 \longleftrightarrow +24$ V power supply reading; $4 \longleftrightarrow +5$ V power supply reading.
Example	`PMON? 0` *-14901*
Related commands	PCFG, PWGD

PWGD (Power Good Signal)

Group	Monitoring commands
Action	Query only
Syntax	`PWGD?`*m*
Description	query the Power Good line status. `PWGD?` queries whether the Power Good line is asserted (1) or not (0). This line is driven by the under-voltage detector whose operation is controlled through the PCFG command. If none of the monitored voltages were under their minimal values, the Power Good line would be asserted and the execution the `PWGD?` command would return 1. When at least one under-voltage condition matches, the Power Good line is driven low and the `PWGD?` command returns 0. The command actually checks whether the PUV flag, which stored in the Instrument status (INSS) byte, is set.
Example	`PWGD?` *1*
Related commands	PMON

TDIE (Die Temperature)

Group	Monitoring commands
Action	Query only
Syntax	`TDIE?`
Description	Return the die temperature. `TDIE?` returns the last measurement of the temperature (in K) of the die provided by the MCU on-chip sensor. The precision is about ±1 K. This reading can be used to get an approached value of the main printed circuit board's temperature where the MCU is mounted. This measurement is automatically updated every 100 ms.
Example	`TDIE?` *298*
Related commands	

XCKD (External Clock Detected)

Group	Monitoring commands
Action	Query only
Syntax	`XCKD?`
Description	`XCKD?` queries whether transitions have been detected (1) on the External Clock input or not (0). Only transitions are detected, the frequency value of the signal at the SMA input connector is not measured. Actually, the command tests the XCK flag's value, which is stored in the Instrument Status (INSS) byte. `XCKD?` should be executed before the selection of the external source for the Synchronization time-base. The clock detection circuit is sampled every 500 ms to update the XCK flag. Because the XCKD command reads the Instrument Status register, clearing this flag may be required to erase an older clock failure report. For example, the `*CLS` command can be used for this purpose. Whatever the command used to clear the XCK flag, the `XCKD?` command must be invoked - at least - 500 ms later to get a refreshed value.
Example	`XCKD?` *1*
Related commands	SYNS

2.4 Status Reporting commands

The Status commands query and configure registers associated with status reporting of the instrument.

*CLS (Clear Status Registers)

Group	Status reporting commands
Action	Query only
Syntax	`*CLS`
Description	Clear immediately all status registers, which are : CTSS, STAS, LEXE, LCMD, LINS, LURQ, INSS, OVLS, COMS and EVTS.
Example	`*CLS`
Related commands	

MSTS (Master Summary Status)

Group	Status reporting commands
Action	Query only
Syntax	`MSTS? [n]`
Description	Return the Master Summary Status register [bit-mask n]. The execution of the `MSTS?` query – without the optional bit-mask n – always causes the /STATUS signal to be de-asserted. Note that `MSTS? n` will not clear /STATUS, even if bit $i \mid n = 2^i$ is the only bit presently causing the /STATUS signal.
Power-on value	0
Example	`MSTS?; MSTS? 128;` *129* *128*
Related commands	MSTE

MSTE (Master Summary Enable)

Group	Status reporting commands
Action	Set/Query
Syntax	`MSTE(?)[`*n*`]{`*m*`}`
Description	Set (query) the Master Summary Enable register [bit-mask *n*] {to bit-mask *m*}. The set-form command will clear the bits outside the bit-mask.
Power-on value	0
Example	`MSTE 128; MSTE?` *128*
Related commands	MSTS

EVTS (Event Status)

Group	Status reporting commands
Action	Query only
Syntax	`EVTS? [`*n*`]`
Description	Read the Event Summary Status register [bit-mask *n*].
Power-on value	1
Example	`EVTS?`
	4
Related commands	EVTE

EVTE (Event Enable)

Group	Status reporting commands
Action	Set/Query
Syntax	`EVTE(?)[`*n*`]{`*m*`}`
Description	Set (query) the Event Summary Enable register [bit-mask *n*] {to bit-mask *m*}. The set-form command will clear the bits outside the bit-mask.
Power-on value	0
Example	`EVTE 4; EVTE?` *4*
Related commands	EVTS

COMS (Communications Status)

Group	Status reporting commands
Action	Query only
Syntax	`COMS? [`*n*`]`
Description	Read the Communications Status register [bit-mask *n*].
Power-on value	0
Example	`COMS?`
	0
Related commands	COME

COME (Communications Enable)

Group	Status reporting commands
Action	Set/Query
Syntax	`COME(?)[`*n*`]{`*m*`}`
Description	Set (query) the Communications Enable register [bit-mask *n*] {to bit-mask *m*}. The set-form command will clear the bits outside the bit-mask.
Power-on value	0
Example	`COME 1`
Related commands	COMS

OVLS (Overload Status)

Group	Status reporting commands
Action	Query only
Syntax	`OVLS? [`*n*`]`
Description	Read the Overload Status register [bit-mask *n*].
Power-on value	0
Example	`OVLS?`
	2
Related commands	OVLE, OVLC.

OVLE (Overload Enable)

Group	Status reporting commands
Action	Set/Query
Syntax	`OVLE(?)[`*n*`]{`*m*`}`
Description	Set (query) the Overload Enable register [bit-mask *n*] {to bit-mask *m*}. The set-form command will clear the bits outside the bit-mask.
Power-on value	0
Example	`OVLE 2`
Related commands	OVLS, OVLC.

OVLC (Overload Condition)

Group	Status reporting commands
Action	Query only
Syntax	`OVLC? [n]`
Description	Read the Overload Condition register [bit-mask *n*]. The values of the bits in the OVLC condition register are determined by the current (real-time) condition of the events defined in the OVLS status register. Reading the condition register does not affect the register.
Power-on value	0
Example	`OVLC?` *2*
Related commands	OVLS, OVLE.

INSS (Instrument Status)

Group	Status reporting commands
Action	Query only
Syntax	`INSS? [`*n*`]`
Description	Read the Instrument Status register [bit-mask *n*].
Power-on value	0
Example	`INSS?`
	1
Related commands	LINS, INSE, INSC.

INSE (Instrument Enable)

Group	Status reporting commands
Action	Set/Query
Syntax	`INSE(?)[`*n*`]{`*m*`}`
Description	Set (query) the Instrument Enable register [bit-mask *n*] {to bit-mask *m*}. The set-form command will clear the bits outside the bit-mask.
Power-on value	0
Example	`INSE 2`
Related commands	LINS, INSS, INSC.

INSC (Instrument Condition)

Group	Status reporting commands
Action	Query only
Syntax	`INSC? [`*n*`]`
Description	Read the Instrument Condition register [bit-mask *n*]. The values of the bits in the INSC condition register are determined by the current (real-time) condition of the events defined in the INSS status register. Reading the condition register does not affect the register.
Power-on value	0
Example	`INSC?` *2*
Related commands	LINS, INSE, INSS.

STAS (/Status Lines Status)

Group	Status reporting commands
Action	Query only
Syntax	`STAS? [`*n*`]`
Description	Read the /Status Lines register [bit-mask *n*].
	The STAS register's value is updated 100 ms by an internal task reading the /Status Lines of the backplane. These lines are driven by the slave modules to signal a service request.
Power-on value	0
Example	`STAS?`
	1
Related commands	STAE

STAE (/Status Lines Enable)

Group	Status reporting commands
Action	Set/Query
Syntax	`STAE(?)[`*n*`]{`*m*`}`
Description	Set (query) the /Status Lines Enable register [bit-mask *n*] {to bit-mask *m*}. The set-form command will clear the bits outside the bit-mask.
Power-on value	0
Example	`STAE 2`
Related commands	STAS

CTSS (/CTS Lines Status)

Group	Status reporting commands
Action	Query only
Syntax	`CTSS? [n]`
Description	Read the /CTS Lines register [bit-mask *n*]. The CTSS register's value is updated 100 ms by an internal task reading the /CTS Lines of the backplane. These lines are driven by the slave modules. This /CTS Lines reading functionality is provided for hardware hand-shaking purpose even if SK-Series modules do not implement this feature.
Power-on value	0
Example	`CTSS?` *1*
Related commands	CTSE

CTSE (CTS Lines Enable)

Group	Status reporting commands
Action	Set/Query
Syntax	`CTSE(?)[`*n*`]{`*m*`}`
Description	Set (query) the /CTS Lines Enable register [bit-mask *n*] {to bit-mask *m*}. The set-form command will clear the bits outside the bit-mask.
Power-on value	0
Example	`CTSE 2`
Related commands	CTSS

2.5 Interface commands

The Interface commands provide control over the interface between the instrument and the host computer.

*RST (Reset)

Group	Interface commands
Action	Set only
Syntax	`*RST`
Description	Reset the instrument to its default configuration. When a parameter is affected by the *RST command, its value is reset according to the information given by the Reset value field within the related command section. Whereas status registers are unaffected by *RST, the content of some conditions registers may have been modified upon resetting the intrument.
Example	`*RST`
Related commands	*RCL, *SAV.

*OPC (Operation Complete)

Group	Interface commands
Action	Set/Query
Syntax	`*OPC(?)`
Description	Set the OPC flag in the EVTS register.
	The query form `*OPC?` returns `1` when complete, but does not affect the EVTS register.
Example	`*OPC?`
	1
Related commands	

CONS (Console Mode)

Group	Interface commands
Action	Set/Query
Syntax	`CONS(?){`*m*`}`
Description	Set (query) the Console mode {to *m*}.
	`CONS 1` causes each character received to be returned to the host computer.
Allowed range	$m \in \{\mathbf{0}\,(\text{disabled}), 1\,(\text{enabled})\}$
Reset (`*RST`) value	**0**
Power-on value	0
Example	`CONS 1`
	1
Related commands	

*IDN (Identify)

Group	Interface commands
Action	Query only
Syntax	`*IDN?`
Description	Read the device identification string. This string is formatted as: Signals and Systems for Physics, model SK810, hw Rppx, fw Rqqy, s/n dddddd. In this string, `SK810` is the model number, `Rnnx` and `Rppy` are revision numbers identifying the hardware or the firmware versions and `dddddd` is the 6-digit serial number.
Example	`*IDN?` *`Signals and Systems for Physics, model SK810, hw R24B, fw R24A, s/n 123456.`*
Related commands	

LINS (Last Instrument Error Status)

Group	Status reporting commands
Action	Query only
Syntax	`LINS?`
Description	Query the last execution instrument error. `LINS?` returns the unique code number associated with this event.
Valid codes are	0 ⟷ no execution error since last `LINS?`; 1 ⟷ on-chip ADC error; 10 ⟷ detected hardware is in invalid condition.
Power-on value	0
Example	`LINS?` *0*
Related commands	LCMD, LEXE, LURQ.

LURQ (Last User Request Status)

Group	Interface commands
Action	Query only
Syntax	`LURQ?`
Description	Query the last User request. `LURQ?` returns the unique code number associated with this event.
Valid codes are	0 ⟷ No User request since last `LURQ?`
Power-on value	0
Example	`LURQ?` *0*
Related commands	LCMD, LEXE, LINS.

LCMD (Last Command Error Status)

Group	Interface commands
Action	Query only
Syntax	`LCMD?`
Description	Query the last command error. `LCMD?` returns the unique code number associated with this error.
Valid codes are	0 ⟷ No execution error since last `LCMD?` 1 ⟷ Illegal (unknown) command. 2 ⟷ Illegal query. 3 ⟷ Illegal set (read-only command). 4 ⟷ Extra parameter. 5 ⟷ Missing parameter. 6 ⟷ Null command.
Power-on value	0
Example	`*RST?;LCMD?` *2*
Related commands	LURQ, LEXE, LINS.

LEXE (Last Execution Error Status)

Group	Interface commands
Action	Query only
Syntax	`LEXE?`
Description	Query the last execution error. `LEXE?` returns the unique code number associated with this error.
Valid codes are	0 ⟷ No execution error since last `LEXE?` 1 ⟷ Invalid parameter. 2 ⟷ Argument value out-of-range. 3 ⟷ The execution causes some parameters to be adapted or clamped. 4 ⟷ A conflict due to the current operation has been avoided. 5 ⟷ No change upon executing the command. 6 ⟷ The operation was aborted due to a fault condition.
Power-on value	0
Example	`CONS2;LEXE?;LEXE?` *1* *0*
Related commands	LURQ, LCMD, LINS.

TERM (Response Termination)

Group	Interface commands
Action	Set/Query
Syntax	`TERM(?){`*m*`}`
Description	Set (query) the termination sequence {to m}.
	The termination sequence is appended to all query responses sent by the instrument. It is constructed of ASCII character(s) `<CR>` (carriage return) or `<LF>` (line feed).
Allowed range	$m \in \{1, 2, 3, 4\}$ where:
	$1 \longleftrightarrow$ `<CR>` character appended,
	$2 \longleftrightarrow$ `<LF>` character appended,
	3 $\longleftrightarrow$ both `<CR>` and `<LF>` characters appended,
	$4 \longleftrightarrow$ no character appended.
Power-on value	3
Reset (`*RST`) value	**3**
Example	`TERM?`
	3
Related commands	

2.6 Memory commands

The Memory commands allow the User to save and recall the instrument's settings in non-volatile memory.

*RCL (Recall Settings)

Group	Memory commands
Action	Set only
Syntax	`*RCL`
Description	Recall the settings stored in the non-volatile memory.
Example	`*RCL`
Related commands	*RST, *SAV.

*SAV (Save Current Settings)

Group	Memory commands
Action	Set only
Syntax	`*SAV`
Description	Save the current settings in the non-volatile memory.
Example	`*SAV`
Related commands	*RCL, *RST.

3 Status Model

The complete block diagram of the status register array is available online at the related product page. There are four categories of registers in this model :

Last Event registers These four read registers (LINS, LCMD, LURQ and LEXE) store the last event that they monitor. A query command i) return the last registered event since the previous query and ii) clears the register's content.

Condition registers These read-only registers correspond to the real-time condition of some underlying physical properties under monitoring. Queries return the latest value of the property, and have no other effect.
Condition register names end with C.

Status registers These read-only registers record the occurrence of defined events. If the event occurs, the corresponding status bit is set to 1. Upon querying a status register, any set bits within it are cleared. These are sometimes known as sticky bits since once set, a bit can only be cleared by reading its value. Status register names end with S.

Enable registers These read/write registers define a bitwise mask for their corresponding status register. If any bit position is set in a status register while the same bit position is also set in the enable register, then the corresponding summary bit is set in either the Event Summary or Master Summary register. Enable register names end with E.

3.1 Master Summary Status (MSTS)

The Master Summary Status (MSTS) is the top-level summary register of the status model. When masked by the Master Summary Status Enable (MSTE) register, a bit set in the Status Byte causes the /STATUS signal to be asserted on the DIN41612 connector. This register is queried with the MSTS?[n] command.

Weight $n = 2^i$	**Bit** i	**Flag**	**Description**
1	0	MSS	Master Summary Status. Indicates whether one or more of the enabled status messages in the Status Byte register is true.
2	1	COM	Communication Summary Bit. Indicates whether one or more of the enabled flags in the Communication Status register has become true.
4	2	EVT	Event Summary Bit. Indicates whether one or more of the enabled flags in the Event Status register is true.
8	3	RFU	Undefined (read 0).
16	4	CTS	Indicates whether one or more of the enabled flags in the /CTS lines Status register is true.
32	5	STA	Indicates whether one or more of the enabled flags in the /STATUS lines register is true.
64	6	INS	Instrument Summary Bit. Indicates whether one or more of the enabled flags in the Instrument Status register is true.
128	7	OVL	Overload Summary Bit. Indicates whether one or more of the enabled flags in the Overload Status register is true.

3.2 Master Summary Enable (MSTE)

Each bit in the MSTE register corresponds one-to-one with a bit in the MSTS register, and acts as a bitwise AND of the MSTS flags to generate the MSS flag. Bit 0 of the MSTE is undefined—setting it has no effect, and reading it always returns 0. This register is set and queried with the MSTE(?) command and cleared at power-on.

3.3 Event Status (EVTS)

The Event Status register consists of 8 event flags. These flags are set by the corresponding event, and cleared only by reading or with the *CLS command ("sticky bits"). Querying the single bit i with the command EVTS? n where the bit-mask $n = 2^i$ will only clear the bit i. For instance, issuing the command `EVTS?128` will clear the bit 7 (INS) only.

Weight $n = 2^i$	Bit i	Flag	Description
1	0	PON	Power On event. Indicates that an off-to-on transition has occurred.
2	1	OPC	Operation Complete. Set by the *OPC command.
4	2	CMD	Command Error event. Indicates an error detected by the command parser. The error code can be queried with LCMD?
8	3	EXE	Execution Error event. Indicates an error in a command that was successfully parsed. The error code can be queried with LEXE?
16	4	RXQ	Reception Buffer event. Indicates that the RX buffer has been cleared.
32	5	TXQ	Transmission Buffer event. Indicates that the TX buffer has been cleared.
64	6	URQ	User Request event. Indicates that a User request has occured. The request code can be queried with LURQ?
128	7	INS	Instrument event. Indicates whether one or more of the enabled flags in the Instrument Status register is true.

3.4 Event Enable (EVTE)

Each bit in the EVTE register corresponds one-to-one with a bit in the EVTS register, and acts as a bitwise AND of the EVTS flags to generate the EVT flag in the Master Summary Status (MSTS) register. This register is set and queried with the EVTE command and cleared at power-on. For instance, issuing the command `EVTE 128` enable the bit 7 (INS) only.

3.5 Instrument Status (INSS)

The Instrument Status register consists of 8 event flags. These flags are set by the corresponding event, and cleared only by reading or with the *CLS command ("sticky bits"). Querying the single bit i with the command INSS? n where the bit-mask $n = 2^i$ will only clear the bit i. For instance, issuing the command `INSS?4` will clear the bit 2 (LNK) only.

Weight	Bit	Flag	Description
$n = 2^i$	i		
1	0	XCK	External Clock event. No transitions have been detected on the External Clock input.
2	1	PUV	Under-Voltage event. At least one power supply voltage is under its minimal operating level.
4	2	LNK	Interlock event. Indicates that the link has been abnormally broken.
8	3	RFU	Undefined (read 0).
16	4	RFU	Undefined (read 0).
32	5	RFU	Undefined (read 0).
64	6	RFU	Undefined (read 0).
128	7	RFU	Undefined (read 0).

3.6 Instrument Enable (INSE)

Each bit in the INSE register corresponds one-to-one with a bit in the INSS register, and acts as a bitwise AND of the INSS flags to generate the INS flag in the Master Summary Status (MSTS) register. This register is set and queried with the INSE command and cleared at power-on.

3.7 Instrument Condition (INSC)

Each bit in the INSC register corresponds one-to-one with a bit in the INSS register. The bits in the INSC register reflect the real-time values of their corresponding signals. Reading the entire register, or individual bits within it, does not affect the value of INSC. This register is queried with the INSC command and cleared at power-on.

3.8 /STATUS Lines (STAS)

The /STATUS Lines register consists of 8 event flags. These flags are set when the corresponding /STATUS line has been asserted by the module mounted on the related slot, and cleared only by reading or with the *CLS command ("sticky bits"). Querying the single bit i with the command STAS? n where the bit-mask $n = 2^i$ will only clear the bit i. These flags are periodically updated by an internal task (100 ms).

Weight $n = 2^i$	**Bit** i	**Flag**	**Description**
1	0	STA0	/STATUS#0 line asserted.
2	1	STA1	/STATUS#1 line asserted.
4	2	STA2	/STATUS#2 line asserted.
8	3	STA3	/STATUS#3 line asserted.
16	4	STA4	/STATUS#4 line asserted.
32	5	STA5	/STATUS#5 line asserted.
64	6	STA6	/STATUS#6 line asserted.
128	7	STA6	/STATUS#7 line asserted.

3.9 /STATUS Lines Enable (STAE)

Each bit in the STAE register corresponds one-to-one with a bit in the STAS register, and acts as a bitwise AND of the STAS flags to generate the STA flag in the Master Summary Status (MSTS) register. This register is set and queried with the STAE command and cleared at power-on.

3.10 /CTS Lines (CTSS)

The /CTS Lines register consists of 8 event flags. These flags are set when the corresponding /CTS line has been asserted by the module mounted on the related slot, and cleared only by reading or with the *CLS command ("sticky bits"). Querying the single bit i with the command CTSS? n where the bit-mask $n = 2^i$ will only clear the bit i. These flags are periodically updated by an internal task (100 ms).

Weight $n = 2^i$	**Bit** i	**Flag**	**Description**
1	0	CTS0	/CTS#0 line asserted.
2	1	CTS1	/CTS#1 line asserted.
4	2	CTS2	/CTS#2 line asserted.
8	3	CTS3	/CTS#3 line asserted.
16	4	CTS4	/CTS#4 line asserted.
32	5	CTS5	/CTS#5 line asserted.
64	6	CTS6	/CTS#6 line asserted.
128	7	CTS6	/CTS#7 line asserted.

3.11 /CTS Lines Enable (CTSE)

Each bit in the CTSE register corresponds one-to-one with a bit in the CTSS register, and acts as a bitwise AND of the CTSS flags to generate the CTS flag in the Master Summary Status (MSTS) register. This register is set and queried with the CTSE command and cleared at power-on.

3.12 Overload Status (OVLS)

The Overload Status register consists of 8 event flags. These event flags are set by the corresponding event, and cleared only by reading or with the *CLS command ("sticky bits"). Querying the single bit i with the command OVLS? n where the bit-mask $n = 2^i$ will only clear the bit i.
Because the OVLS register is not used in the SK810 , querying this register always returns 0. Therefore, the corresponding summary bit in the MSTS register (bit OVL) is never set whatever the value of the OVLE register.

Weight $n = 2^i$	Bit i	Flag	Description
1	0	RFU	Undefined (read 0).
2	1	RFU	Undefined (read 0).
4	2	RFU	Undefined (read 0).
8	3	RFU	Undefined (read 0).
16	4	RFU	Undefined (read 0).
32	5	RFU	Undefined (read 0).
64	6	RFU	Undefined (read 0).
128	7	RFU	Undefined (read 0).

3.13 Overload Enable (OVLE)

Each bit in the OVLE register corresponds one-to-one with a bit in the OVLS register, and acts as a bitwise AND of the OVLS flags to generate the OVL flag in the Master Summary Status (MSTS) register.

3.14 Overload Condition (OVLC)

Each bit in the OVLC register corresponds one-to-one with a bit in the OVLS register. The bits in the OVLC register reflect the real-time values of their corresponding signals. Reading the entire register, or individual bits within it, does not affect the value of OVLC. This register is queried with the OVLC command and cleared at power-on.
Because the OVLC register is not used in the SK810 , querying this register always returns 0.

3.15 Communication Status (COMS)

The Communication Status register consists of 8 event flags. These flags are set by the corresponding event, and cleared only by reading or with the *CLS command ("sticky bits"). Querying the single bit i with the command COMS? n where the bit-mask $n = 2^i$ will only clear the bit i.
Because the COMS register is not used in the SK810 , querying this register always returns 0. Therefore, the corresponding summary bit in the MSTS register (bit COM) is never set whatever the value of the COME register.

Weight $n = 2^i$	**Bit** i	**Flag**	**Description**
1	0	PRY	Parity violation.
2	1	COL	Bus collision.
4	2	RFU	Undefined (read 0).
8	3	RFU	Undefined (read 0).
16	4	RFU	Undefined (read 0).
32	5	RFU	Undefined (read 0).
64	6	RFU	Undefined (read 0).
128	7	RFU	Undefined (read 0).

3.16 Communication Enable (COME)

Each bit in the COME register corresponds one-to-one with a bit in the COMS register, and acts as a bitwise AND of the COMS flags to generate the COM flag in the Master Summary Status (MSTS) register. This register is set and queried with the COME command and cleared at power-on.

3.17 Last Command Error (LCMD)

The LCMD register holds the last error detected by the command parser. The related error code can be retrieved by the command LCMD?. When such an error is detected, the corresponding bit in the Event Status register is set (bit CMD in EVTS).

3.18 Last Execution Error (LEXE)

The LEXE register holds the last error detected during the execution of a command. The related error code can be retrieved by the command LEXE?. When such an error is detected, the corresponding bit in the Event Status register is set (bit EXE in EVTS).

3.19 Last Instrument Error (LINS)

The LINS register holds the last error detected during the opertation of the instrument. The related error code can be retrieved by the command LINS?. When such an error is detected, the corresponding bit in the Event Status register is set (bit INS in EVTS).

3.20 Last User Request (LURQ)

The LURQ register holds the last User's request. The related request code can be retrieved by the command LURQ?. When such a request is reported, the corresponding bit in the Event Status register is set (bit URQ in EVTS).
Because the LURQ register is not used in the SK810, querying this register always returns 0 and the corresponding summary bit in the Event Status register is never set (bit URQ in EVTS).

4 Index of commands

Instrument Configuration commands
- PCFG (Power Monitoring Configuration), 12
- SYNS (Synchronization Source Select), 13

Instrument Monitoring commands
- PMON (Power Supply Voltages), 14
- PWGD (Power Good Signal), 15
- TDIE (Die Temperature), 16
- XCKD (External Clock Detected), 17

Instrument Settings commands
- LINK (Link), 11
- RTSS (/RTS Lines Status), 8
- SLTE (/SLOT Lines Enable), 10
- SLTS (/SLOT Lines Status), 9

Interface commands
- *IDN (Identify), 38
- *OPC (Operation Complete), 36
- *RST (Reset), 35
- CONS (Console Mode), 37
- LCMD (Last Command Error Status), 41
- LEXE (Last Execution Error Status), 42
- LINS (Last Instrument Error Status), 39
- LURQ (Last User Request Status), 40
- TERM (Response Termination), 43

Memory commands
- *RCL (Recall Settings), 44
- *SAV (Save Current Settings), 45

Status Reporting commands
- *CLS (Clear Status Registers), 18
- COME (Communications Enable, 24
- COMS (Communications Status), 23
- CTSE (CTS Lines Enable), 34
- CTSS (/CTS Lines Status), 33
- EVTE (Event Enable), 22
- EVTS (Event Status), 21
- INSC (Instrument Condition), 30
- INSE (Instrument Enable), 29
- INSS (Instrument Status), 28
- MSTE (Master Summary Enable), 20
- MSTS (Master Summary Status), 19
- OVLC (Overload Condition), 27
- OVLE (Overload Enable), 26
- OVLS (Overload Status), 25
- STAE (/Status Lines Enable), 32
- STAS (/Status Lines Status), 31

5 Document Revision History

5.1 Version Number

This document is identified by SK810-SU01-P24A.

5.2 Revision History

P24A (2024-02-14)

Initial version.