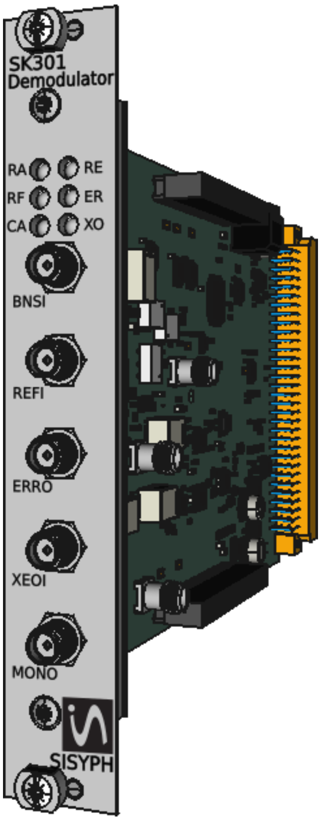
Programming Guide

SK301 *Vidourle* RF Demodulator

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 SK301 RF Demodulator module over the serial interface.

Contents

1 Introduction

Remote operation of the SK301 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.

1.1 Power-on Configuration

The settings for serial interface are 9600 baud with no parity and no hardware flow control, and local echo disabled (`CONS 0`).
Most of the instrument settings are stored in non-volatile memory and can be retrieved 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

The instrument stores incoming bytes from the host interface in a 128-byte input buffer. 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 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.

1.4 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

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

LPFS (Low-Pass Filter)

Group	Instrument Settings commands
Action	Set/Query
Syntax	`LPFS(?){`*n*`}`
Description	Set (query) the low-pass filter state {to n}.
Allowed range	$n \in \{\mathbf{0}, 1, 2\}$ where : $\mathbf{0} \longleftrightarrow$ no filtering (filter bypassed), $1 \longleftrightarrow f_c = 30\,\text{MHz}$, $2 \longleftrightarrow f_c = 3\,\text{MHz}$.
Power-on value	Restored from non-volatile memory (*cf* *SAV).
Reset value	**0**
Example	`LPFS 2; LPFS? 2` *2*
Related commands	IFFE, RFFE.

OFSS (DAC Offset Voltage)

Group	Instrument Settings commands
Action	Set/Query
Syntax	`OFSS(?){`u`}`
Description	Set (query) the Offset voltage {to u}, in µV. A 12-bit DAC is used to offset the Error output voltage with a resolution better than 10 µV.
Allowed range	$-12\,000\,\mu V \leq u \leq +12\,000\,\mu V$.
Power-on value	Restored from non-volatile memory (*cf* *SAV).
Reset value	**0**
Example	`OFSS -5000; OFSS?` *-5000*
Related commands	OFSE.

2.2 Instrument Configuration commands

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

RFFE (RF Notch-Filter Enable)

Group	Configuration commands
Action	Set/Query
Syntax	`RFFE(?){`*b*`}`
Description	Set (query) the RF Notch-Filter's state {to *b*}. The RF notch-filter's center frequency is tuned to 60 MHz. When $b = 0$ (resp. 1), the filter is disabled (resp. enabled).
Allowed range	$b \in \{\mathbf{0}, 1\}$.
Power-on value	Restored from non-volatile memory (*cf* *SAV).
Reset value (`*RST`)	**0**
Example	`RFFE 1; RFFE?`
	1
Related commands	IFFE, LPFS.

IFFE (IF Notch-Filter Enable)

Group	Configuration commands
Action	Set/Query
Syntax	`IFFE(?){`*b*`}`
Description	Set (query) the IF Notch-Filter's state {to b}. The IF notch-filter's center frequency is tuned to 30 MHz. When $b = 0$ (resp. `1`), the filter is disabled (resp. enabled).
Allowed range	$b \in \{\mathbf{0}, 1\}$.
Power-on value	Restored from non-volatile memory (*cf* *SAV).
Reset value (`*RST`)	**0**
Example	`IFFE 1; IFFE?` *1*
Related commands	LPFS, RFFE.

OFSE (DAC Offset Enable)

Group	Configuration commands
Action	Set/Query
Syntax	`OFSE(?){`*b*`}`
Description	Set (query) the state of the DAC Offset functionality {to b}. This command is provided to control the DAC offset functionality. When $b = 0$ (resp. `1`), the DAC offset functionality is disabled (resp. enabled).
Allowed range	$b \in \{\mathbf{0}, 1\}$.
Power-on value	Restored from non-volatile memory (*cf* *SAV).
Reset value (`*RST`)	**0**
Example	`OFSE 1; OFSE?`
	1
Related commands	OFSS.

CALE (Calibration Input Enable)

Group	Configuration commands
Action	Set/Query
Syntax	`CALE(?){`*b*`}`
Description	Set (query) the Calibration input state {to b}. This command is provided to use the Calibration input instead of the mixer IF as source of the error signal. When $b = 0$ (resp. 1), the calibration input is disabled (resp. enabled).
Allowed range	$b \in \{\mathbf{0}, 1\}$.
Power-on value	Restored from non-volatile memory (*cf* *SAV).
Reset value (`*RST`)	**0**
Example	`CALE 1; CALE?` *1*
Related commands	XEOE.

XEOE (External Offset Input Enable)

Group	Configuration commands
Action	Set/Query
Syntax	`XEOE(?){`*b*`}`
Description	Set (query) the External offset input state {to *b*}.
	The External Offset input can be used to offset the error voltage. When $b = 0$ (resp. 1), the External Offset input is disabled (resp. enabled).
Allowed range	$b \in \{\mathbf{0}, 1\}$.
Power-on value	Restored from non-volatile memory (*cf* *SAV).
Reset value (`*RST`)	**0**
Example	`XEOE 1; XEOE?`
	1
Related commands	CALE.

MONS (Monitoring Output)

Group	Configuration commands
Action	Set/Query
Syntax	`MONS(?){`n`}`
Description	Set (query) the source of the monitoring output signal {to n}. This command is used to select which signal is routed to the MONO coaxial connector.
Allowed range	$n \in \{\mathbf{0}, 1, \ldots, 6\}$ where : $\mathbf{0} \longleftrightarrow$ Analog ground voltage (0 V); $1 \longleftrightarrow$ Error, full-bandwidth; $2 \longleftrightarrow$ Low-pass filtered Error, $f_c = 1\,\mathrm{kHz}$; $3 \longleftrightarrow$ Low-pass filtered Error, $f_c = 200\,\mathrm{kHz}$; $4 \longleftrightarrow$ AC-coupled Error, $f_c = 10\,\mathrm{Hz}$; $5 \longleftrightarrow$ Mixer-RF power detector; $6 \longleftrightarrow$ Mixer-LO power detector.
Power-on value	Restored from non-volatile memory (*cf* *SAV).
Reset value (`*RST`)	**0**
Memory (`*SAV`, `*RCL`)	commands apply.
Example	`MONS 1; MONS?` *1*
Related commands	

2.3 Instrument Monitoring commands

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

RMON (Remote Monitoring)

Group	Monitoring commands
Action	Query only
Syntax	`RMON?`*n*
Description	Return to the host computer the last measurement of the parameter specified by n. These parameters are periodically sampled (100 ms) by an internal task.
Allowed range	$n \in \{0, 1, 2, 3\}$ where : 0 ⟷ Error positive peak-voltage in mV; 1 ⟷ Error negative peak-voltage in mV; 2 ⟷ Mixer RF-input power in mdBm; 3 ⟷ Mixer LO-input power in mdBm.
Example	`RMON? 1` *-17*
Related commands	

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	

STMS (Streamed Channels Selection)

Group	Monitoring commands
Action	Set/Query
Syntax	`STMS(?){`*m*`}`
Description	Set (query) the channels selection register {to bit-mask m}. In order to stream the channel i to output, the command must be invoked with $m = 2^i$ as argument. For instance, `STMS 3` will be executed to stream the Error positive and negative peak-voltages to output. Data are output on a single line where a comma delimiter is used to separate channel data. The rightmost position in the row is occupied by the channel with the lowest weight. Data streaming operation is enabled using the STME command while the number of measurements that will streamed out is set by the STMN command.
Allowed range	$m \in \{\mathbf{1}, \ldots, 15\}$, where m can be any combination of : $m_0 = \mathbf{1} = 2^0 \longleftrightarrow$ Error positive peak-voltage [mV]; $m_1 = 2 = 2^1 \longleftrightarrow$ Error negative peak-voltage [mV]; $m_2 = 4 = 2^2 \longleftrightarrow$ Mixer RF-input power [mdBm]; $m_3 = 8 = 2^3 \longleftrightarrow$ Mixer LO-input power [mdBm].
Power-on value	Restored from non-volatile memory (*cf* *SAV).
Reset value	**1**

Example	`STMN 5; STMS 3 ; STME 1`
	`-628,611`
	`-637,616`
	`-670,579`
	`-641,613`
	`-633,616`

Related commands	STME, STMN.

STME (Data Streaming Enable)

Group	Monitoring commands
Action	Set/Query
Syntax	`STME(?){`b`}`
Description	Set (query) the state of the data streaming functionality {to b}. If $b = 0$ (resp. 1), data streaming is disabled (resp. enabled). When data streaming is enabled, the number of measurements that will be streamed out is specified using the STMN command.
Allowed range	$b \in \{\mathbf{0}, 1\}$.
Power-on value	0
Reset value (`*RST`)	**0**
Example	`STME 1; STME?`
	1
Related commands	STMS, STMN.

STMN (Number of Streamed Measurements)

Group	Monitoring commands
Action	Set/Query
Syntax	`STMN(?){`n`}`
Description	Set (query) the number of measurements to be streamed out {to n}. When $n = 0$, measurements will be output indefinitely, until the STME command is invoked. If $n > 0$, data streaming will stop once n measurements will have been output. Measurements are streamed out at a constant rate of approximately one measurement per second.
Allowed range	$\mathbf{0} \leq m \leq 10000$.
Reset value	**0**

Example	`STMN 1000; STMN?`
	`1000`

Related commands	STME, STMS.

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.

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 SK301, hw Rppx, fw Rqqy, s/n dddddd. In this string, `SK301` 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 SK301, 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; 20 ⟷ some parameters have been be adapted or clamped; 21 ⟷ some functionalities have been be disabled.
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 ⟷ `<CR>` character appended, 2 ⟷ `<LF>` character appended, **3** ⟷ both `<CR>` and `<LF>` characters appended, 4 ⟷ 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	Bit	Flag	Description
$n = 2^i$	i		
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	RFU	Undefined (read 0).
32	5	RFU	Undefined (read 0).
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	**Bit**	**Flag**	**Description**
$n = 2^i$	i		
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?1` will clear the bit 0 only.

Weight	**Bit**	**Flag**	**Description**
$n = 2^i$	i		
1	0	PUV	Power Supply Under-Voltage. At least, one power supply is under its low-level threshold.
2	1	IKS	Internal 10-MHz clock source used. The module is not synchronized to the platform's timebase. *The platform's timebase synchronization feature is not yet implemented. The bit is therefore always set (read 1).*
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).

3.6 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 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. For instance, issuing the command `OVLS?2` will clear the bit 1 only.

Weight	**Bit**	**Flag**	**Description**
$n = 2^i$	i		
1	0	MRF	Mixer-RF Power Alarm. The measured power of the RF signal reaches its upper-limit (+3 dBm).
2	1	MLO	Mixer-LO Power Alarm. The measured power of the LO signal reaches its upper-limit (+10 dBm).
4	2	ERP	Error Positive Alarm. The positive peak-value of the Error signal reaches its upper-limit (+100 mV).
8	3	ERN	Error Negative Alarm. The negative peak-value of the Error signal reaches its lower-limit (−100 mV).
16	4	RFU	Undefined (read 0).
32	5	RFU	Undefined (read 0).
64	6	RFU	Undefined (read 0).
128	7	RFU	Undefined (read 0).

3.9 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.

3.10 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.

3.11 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 SK301 , 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	**Bit**	**Flag**	**Description**
$n = 2^i$	i		
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).

3.12 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.

3.13 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).

3.14 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).

3.15 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).

3.16 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 SK301, 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

5 Document Revision History

5.1 Version Number

This document is identified by SK301-SU01-P24A.

5.2 Revision History

P24A (2024-03-18)

Initial version.