ThinkRF R5700/R5750

Real-Time Spectrum Analyzer with GNSS

Programmer's Guide
Version 1.1.0

May 15, 2019
Document no. 75-0033-190515

Important notice

The information in this guide is furnished for informational use only and is subject to change without notice. ThinkRF Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

Trademarks

ThinkRF, the ThinkRF logo and R5700/R5750 are trademarks of ThinkRF Corporation.

All other brand or product names are trademarks or registered trademarks of their respective companies or owners.

ThinkRF Corp

390 March Road
Kanata, ON K2K 0G7
(613) 369-5104

HARDWARE WARRANTY AND LIMITATION OF LIABILITY

Read this warranty carefully before you use the product.

R5700/R5750 (or R57x0 for short) Real Time Spectrum Analyzers with GNSS are warranted for workmanship and materials for a period of one (1) year from the date of shipment as identified by the Customer's packing slip or carrier waybill. ThinkRF reserves the right to void the warranty on any equipment that has been altered or damaged due to Customer negligence, unauthorized repair, misuse of equipment, evidence of physical or environmental damage, transportation abuse or removal of any ThinkRF identification labels or serial numbers.

It will remain the responsibility of the Customer, having obtained a Return Material Authorization (RMA) and shipping instructions from ThinkRF, to return, at the Customer's expense, the defective unit to ThinkRF's repair facilities. ThinkRF will incur shipping charges for the return of warranty repaired equipment. The RMA number can be secured by calling ThinkRF Customer Service and Support (1-613-369-5104). If the product does not fall within ThinkRF's warranty period or the product is found to be functioning as designed, then under the terms of ThinkRF's warranty policy, all costs of repairs and shipping will be charged directly to the Customer. ThinkRF will warrant repaired units for a period of 90 days from date of shipment from ThinkRF to the Customer. If the remaining period on the original hardware warranty is greater than 30 days, then ThinkRF will honor this remaining warranty period.

THINKRF EXPRESSLY DISCLAIMS ALL OTHER WARRANTIES AND CONDITIONS, WHETHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, WARRANTIES, CONDITIONS OR REPRESENTATIONS OF WORKMANSHIP, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, DURABILITY, OR THAT THE OPERATION OF THE HARDWARE OR LICENSED SOFTWARE WILL BE ERROR FREE. IN NO EVENT WILL THINKRF BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

USE OF PRODUCTS IN HIGH RISK ACTIVITIES

THINKRF PRODUCTS ARE INTENDED FOR STANDARD INDOOR COMMERCIAL USE. WITHOUT THE APPROPRIATE NETWORK DESIGN ENGINEERING, THEY MUST NOT BE USED FOR ANY "HIGH RISK ACTIVITY", as described in this paragraph. Customer acknowledges and agrees that the products supplied hereunder are not fault-tolerant and are not designed, manufactured or intended for use or resale as on-line control equipment in hazardous environments requiring fail safe performance including but not limited to the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of products could lead directly to death, personal injury, or severe physical or environmental damage, all of which are examples of "High Risk Activity". THINKRF AND ITS SUPPLIERS EXPRESSLY DISCLAIM ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR HIGH RISK ACTIVITIES.

GNU General Public License

This device contains free firmware: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. GNU General Public License is available at http://www.gnu.org/licenses.

Table of Contents

Abbreviations 7
List of Figures 8
List of Tables 9
Preface 10

Audience 10
Conventions 10
Obtaining Documentation and Releases 10
Document Feedback 11
Obtaining Technical Assistance 11

R57x0 Functional Overview 12

System Overview 12
The Architecture 15
RF Receiver Front-End 17
Direct-Conversion Receiver Technology 18
DC Offset Correction 18
IQ Offset Correction 18
Digital Signal Processing 20
Digital Down Converter 20
Triggers 21
Frequency Domain Triggering 21
Periodic Triggering 22
External Triggering 22
Capture Controller 22
Trace Capture Control 23
Sweep Capture Control 23
Synchronized Sweep 25

VITA-49 Radio Transport Protocol 27

Purpose 27
R57x0's VRT Overview 27
Packet Classes and Streams 28
Receiver Context Packet Class 28
Context Field Change Indicator 30
RF Reference Frequency 30
Gain 30
Digitizer Context Packet Class 30
Context Field Change Indicator 32
Bandwidth 32
RF Frequency Offset 32
Reference Level 32
Formatted GPS Geolocation 33
Extension Context Packet Class 35
Context Field Change Indicator 36
IQ Swapped Indicator 36
New Stream Start ID 37
New Sweep Start ID 37
IF Data Packet Class 37

Picosecond Timestamp Words Format 38
Data Payload Format 38
Trailer Word Format 40

SCPI Command Set 43

SCPI Language Overview 43
IEEE Mandated SCPI Commands 44
*CLS 44
*ESE/*ESE? 44
*ESR? 45
*IDN? 45
*OPC/*OPC? 45
*RST 45
*SRE/*SRE? 46
*STB? 46
*TST? 46
*WAI 47
SYSTem Commands 47
:SYSTem:ABORt 47
:SYSTem:CAPTure:MODE? 47
:SYSTem:COMMunicate:HISLip:SESSion? 48
:SYSTem:COMMunicate:LAN:APPLy 48
:SYSTem:COMMunicate:LAN:CONFigure 48
:SYSTem:COMMunicate:LAN:DNS 49
:SYSTem:COMMunicate:LAN:GATEway 49
:SYSTem:COMMunicate:LAN:IP 50
:SYSTem:COMMunicate:LAN:MTU 50
:SYSTem:COMMunicate:LAN:NETMask 51
:SYSTem:COMMunicate:NTP 51
:SYSTem:ERRor[:NEXT]? 52
:SYSTem:ERRor:ALL? 52
:SYSTem:ERRor:CODE[:NEXT]? 52
:SYSTem:ERRor:CODE:ALL? 53
:SYSTem:ERRor:COUNt? 53
:SYSTem:FLUSh 53
:SYSTem:LOCK:HAVE? 54
:SYSTem:LOCK:REQuest? 54
:SYSTem:OPTions? 55
:SYSTem:SYNC:MASTer 55
:SYSTem:SYNC:WAIT 56
:SYSTem:VERSion? 56
:SYSTem:DATE 56
:SYSTem:TIME 57
:SYSTem:TIME:ADJust 57
:SYSTem:TIME:SYNC 57
STATus Commands 59
:STATus:OPERation[:EVENt]? 60
:STATus:OPERation:CONDition? 60
:STATus:OPERation:ENABle 61
:STATus:OPERation:NTRansition 61
:STATus:OPERation:PTRansition 61
:STATus:PRESET 62
:STATus:QUEStionable[:EVENt]? 62
:STATus:QUEStionable:CONDition? 62
:STATus:QUEStionable:ENABle 63
:STATus:QUEStionable:NTRansition 63

:STATus:QUEStionable:PTRansition 63
:STATus:TEMPerature? 64
INPut Commands 64
:INPut:ATTenuator 64
:INPut:ATTenuator:VARiable 64
:INPut:GAIN 65
:INPut:GAIN:HDR 66
:INPut:MODE 66
SOURce Commands 67
:SOURce:REFerence:PLL 67
:SOURce:REFerence:PPS 67
SENSe Commands 68
[:SENSe]:DECimation 68
[:SENSe]:FREQuency:CENTer 69
[:SENSe]:FREQuency:IF? 70
[:SENSe]:FREQuency:LOSCillator? 70
[:SENSe]:FREQuency:SHIFt 70
[:SENSe]:LOCK:REFerence? 71
[:SENSe]:LOCK:RF? 71
GNSS Commands 72
:GNSS[:ENABle] 72
:GNSS:ADELay 72
:GNSS:CONStellation 73
:GNSS:POSition? 73
:GNSS:REFerence? 74
TRIGger Commands 74
:TRIGger:TYPE 74
:TRIGger:LEVel 75
:TRIGger:PERiodic 76
TRACe Commands 76
:TRACe:BLOCk:DATA? 77
:TRACe:BLOCk:PACKets 77
:TRACe:SPPacket 78
:TRACe:STReam:STARt 79
:TRACe:STReam:STOP 79
SWEep Commands 79
:SWEep:LIST:ITERations 81
:SWEep:LIST:STARt 81
:SWEep:LIST:STATus? 81
:SWEep:LIST:STOP 82
:SWEep:ENTRy:COPY 82
:SWEep:ENTRy:COUNt? 82
:SWEep:ENTRy:DELETE 82
:SWEep:ENTRy:NEW 83
:SWEep:ENTRy:READ? 83
:SWEep:ENTRy:SAVE 83
:SWEep:ENTRy:ATTenuator 84
:SWEep:ENTRy:ATTenuator:VARiable 84
:SWEep:ENTRy:DECimation 84
:SWEep:ENTRy:FREQuency:CENTer 84
:SWEep:ENTRy:FREQuency:STEP 85
:SWEep:ENTRy:FREQuency:SHIFt 85
:SWEep:ENTRy:GAIN:HDR 85
:SWEep:ENTRy:MODE 86

:SWEep:ENTRy:DWELl ... 86
:SWEep:ENTRy:PPBlock ... 86
:SWEep:ENTRy:SPPacket ... 86
:SWEep:ENTRy:TRIGger:LEVel ... 87
:SWEep:ENTRy:TRIGger:TYPE ... 87

Appendix A: Connecting to RTSA ... 88

Simple 2-port TCP/IP Connection ... 88
Connection Using HiSLIP ... 89

Appendix B: Protocols for Discovering RTSA ... 92

Discovery Using mDNS/DNS-SD ... 92
Discovery Using Broadcast UDP ... 92

Appendix C: SCPI Command Syntax ... 94

Entering Commands ... 94
Notation ... 95
Parameter types ... 95
Default Units ... 95

Appendix D: SCPI Status and Event Registers ... 97

Status Byte Register (SBR) ... 97
Standard Event Status Register (ESR) ... 97
Operational Status Register (OSR) ... 98
Questionable Status Register (QSR) ... 98
Output Queue ... 99
Error and Event Queue ... 99

Appendix E: SCPI Error Codes Used ... 100
Appendix F: SCPI Commands Quick Reference ... 101
R55x0 vs. R57x0 List of Changes ... 107
References ... 108
Document Revision History ... 109

Abbreviations

ADC	Analog-to-Digital Converter
API	Application Programming Interface
CIC	Cascaded Integrator-Comb
DC	Direct Current
DD	Direct Digitizer
DDC	Digital Down Converter
DDS	Direct Digital Synthesizer
DSP	Digital Signal Processing
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GUI	Graphical User Interface
HDR	High Dynamic Range
IBW	Instantaneous Bandwidth
IEEE	Institute of Electrical and Electronics Engineers
IF	Intermediate Frequency
IQ	In-phase and Quadrature
LAN	Local Area Network
MB	Mega-Bytes
MSB	Most Significant Byte
MSa	Mega-Samples
NB	Narrowband
NCO	Numerically Controlled Oscillator
NTP	Network Time Protocol
NTPD	Network Time Protocol Daemon
PLL	Phase-Locked Loop
RF	Radio Frequency
RFE	Receiver Front-End
RTSA	Real Time Spectrum Analyzer
Sa/s	Samples-per-Second
SCPI	Standard Commands for Programmable Instruments
SH	Super-Heterodyne
SHN	Super-Heterodyne with narrower bandwidth
TCP/IP	Transmission Control Protocol/Internet Protocol
TD	Time Domain
TSF	TimeStamp-Fractional
TSI	TimeStamp-Integer
TSM	TimeStamp Mode
UTC	Coordinated Universal Time
VCO	Voltage Control Oscillator
VRT	VITA-49 Radio Transport
WB	Wideband
ZIF	Zero Intermediate Frequency

List of Figures

Figure 1: R57x0 Functional Block Diagram 13
Figure 2: RF Receiver Front-end and Capture Controller Functional Block Diagram 16
Figure 3: DC Offset with Amplitude Roll-Off at +50MHz 18
Figure 4: IQ Offset Correction 19
Figure 5: DDC Functional Block Diagram 20
Figure 6: Association between Time and Frequency Domain 21
Figure 7: Synchronized Sweep using Sync-Word 25
Figure 8: Synchronized Sweep with a Missed Capture 26
Figure 9: Connectivity and 4 Different Packet Streams Supported by R57x0 27
Figure 10: An Example Illustrating Uninverted and Inverted Spectrums 42
Figure 11: SCPI Language Hierarchical or Tree Structure Example 43
Figure 12: SCPI Measurement Function Block 44
Figure 13: Status Reporting Structure with Status & Enable Registers 59
Figure 14: 2-port TCP/IP connection to RTSA 88
Figure 15: HiSLIP and TCP connections to RTSA 90

List of Tables

Table 1: System Level Control/Status Commands with GNSS 13
Table 2: Radio RFE Modes and DSP Data Output Formats 17
Table 3: RF Front-End Control/Status Commands 19
Table 4: Trigger Control/Status Commands 22
Table 5: Trace Capture Control Commands 23
Table 6: Sweep Capture Control/Status Interface 24
Table 7: The Categories of VRT Packet Streams Supported by ThinkRF's R57x0 27
Table 8: A List of Stream Identifiers as Used by ThinkRF for Different Packet Classes 28
Table 9: Receiver Context Packet Class Structure 29
Table 10: Receiver Context Indicator Field Positions 29
Table 11: Receiver Context Field Definition and Values 29
Table 12: RF Reference Frequency Word Format 30
Table 13: Gain Field Format 30
Table 14: Digitizer Context Packet Class Structure 31
Table 15: Digitizer Context Indicator Field Bit Positions 31
Table 16: Digitizer Context Field Values 32
Table 17: Bandwidth Word Format 32
Table 18: RF Frequency Offset Word Format 32
Table 19: Reference Level Field Format 33
Table 20: Formatted GPS Geolocation Fields 34
Table 21: Geolocation Angle (Degrees) Format 34
Table 22: Altitude Subfield Format 35
Table 23: Speed Over Ground Subfield Format 35
Table 24: Extension Context Packet Class Structure 35
Table 25: Extension Context Indicator Field Positions 36
Table 26: Extension Context Field Definition and Values 36
Table 27: New Stream Start ID Field Format 37
Table 28: New Sweep Start ID Field Format 37
Table 29: Output Data Width and Packing Method for Different Data Formats 37
Table 30: IF Data Class Field Values 38
Table 31: Stream Identifier Values for Different Data Output Formats 38
Table 32: 64-bit or Two Words Picosecond Timestamp Format 38
Table 33: $\{I_{14}Q_{14}\}$ Data Payload Arrangement with Upper 2-bit Signed Extended to $\{I_{16}\,Q_{16}\}$ 39
Table 34: $\{I_{14}\}$ Data Payload Arrangement with Upper 2-bit Signed Extended to $\{I_{16}\}$ 39
Table 35: $\{I_{24}\}$ Data Payload Arrangement with Upper 8-bit Signed Extended to $\{I_{32}\}$ 40
Table 36: Trailer Word Format 40
Table 37: Trailer Indicator and Enable Bits 40
Table 38: Conditions Causing Abnormal Indicator State and Suggested Resolution 41
Table 39: RTSA Option Codes and the Corresponding Description 55
Table 40: Performance of The Gain Settings of R57x0-418, 427 and Their Variants 65
Table 41: Maximum Threshold Level Where +/-3 dBm Error or Less Still Hold For A Given Attenuation Level 75
Table 42: Max, Min, and Required Multiples for SPP and Samples-per-word for Different Data Output Format 78
Table 43: HiSLIP Message Header Format 90
Table 44: ThinkRF Vendor Specific Message Type Value Definitions 90
Table 45: ThinkRF Data Channel Initialization Transaction 90

Preface

This preface describes the audience for, the organization of, and conventions used in this document. It also identifies related documentation and explains how to access electronic documentation.

Audience

This document is written for software developers wishing to develop and/or maintain a software interface to the R5700/R5750 and who have a basic understanding, familiarity and experience with network test and measurement equipment.

Conventions

This section describes the conventions used in this document.

Grayed-out Font
Indicates a command or a feature is not yet available in the current release.

`Courier Font`
Illustrates this is an example for a command or a concept.

Light Blue Font
Contains hyperlink to the referenced source that can be clicked on.

Normal Bold Font
When used within a sentence or a paragraph, it emphasizes an idea to be paid attention to particularly.

Red Font
Conveys special information of that section.

Note: This symbol means **take note**. Notes contain helpful suggestions or references to additional information and material.

Caution: This symbol means **be careful**. In this situation, you might do something that could result in equipment damage or loss of data.

Warning: This symbol means **danger**. You are in a situation that could cause bodily injury. Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with the standard practices for preventing accidents.

Obtaining Documentation and Releases

You can access the most current ThinkRF documentation and the latest release bundles at http://www.thinkrf.com/resources.

Document Feedback

Please send your comments about this document or our other documentation to support@thinkrf.com.

Thank you, we appreciate your comments.

Obtaining Technical Assistance

The ThinkRF Support website provides online documents for resolving technical issues with ThinkRF products at www.thinkrf.com/resources.

For all customers who hold a valid end-user license, ThinkRF provides technical assistance 9 AM to 5 PM Eastern Time, Monday to Friday. Contact us at www.thinkrf.com/support/ or by calling **+1.613.369.5104**.

Before contacting Support, please have the following information available:

- R57x0's serial number and product version, which are located on the identification label on the R57x0's underside.
- The firmware version running on the R57x0.
- Versions of ThinkRF software you are using, potentially including the S240, API libraries to third-party applications.
- The operating system and version you are using.

R57x0 Functional Overview

This section overviews the R57x0's functionality and protocols used, and summarizes the SCPI command sets for controlling the individual functions.

Note: This is a living and evolving document. We welcome your feedback.

The features and functionality described in this section **may** exist in the current product firmware release or are scheduled for a future product firmware release (grayed out commands and/or text). Please refer to Appendix F: SCPI Commands Quick Reference for the complete list of commands and the availability information. No hardware upgrade is required at each feature release (unless specified though unlikely).

System Overview

The R5700/R5750 (or R57x0 for short) Real Time Spectrum Analyzer (RTSA) with GNSS (Global Navigation Satellite System) provides the benefits of a high-performance software-defined RF receiver, digitizer and analyzer along with integrated GNSS technology offering location and time information in one package. With patent-pending software-defined RF receiver technology, the RTSA provides industry leading combined sensitivity, tuning range, wide instantaneous bandwidth (IBW) and scan rate. Additionally, it provides real-time sophisticated triggering and capture control. Figure 1 illustrates an RTSA solution system example.

The R57x0 is designed for stand-alone, remote and/or distributed wireless signal analysis. Whether using as a single unit or a network of radio sensors, R57x0 is ideal for monitoring, management and surveillance of transmitters, whether they are in-building or spread across a geographic area. Applications include, but are not limited to:

- 5G wireless technology;
- test and measurement;
- monitoring and surveillance;
- research;
- OEM integration.

The R57x0 hardware largely consists of:

- a hybrid super-heterodyne, direct-conversion and direct-digitization RF receiver front-end (RFE);
- receiver front end inputs and outputs to support clock synchronization, and IF outputs for high-end digitization;
- a 125 MSamples/sec 14-bit wideband (WB) ADC with a dynamic range of greater than 70 dB;
- a 325 kSamples/sec 24-bit narrowband (NB) ADC with a dynamic range in excess of 100 dB;
- a GNSS module with embedded 10MHz reference clock source for further RTSA's time synchronization;
- a Xilinx's Zynq FPGA with built-in dual-core ARM®-based processor, Gigabit Ethernet interface and custom embedded digital signal processing (DSP) logic;

- 1 GB of DDR3 shared between firmware and real-time caching of digitized data;
- a general purpose input/output (GPIO) port.

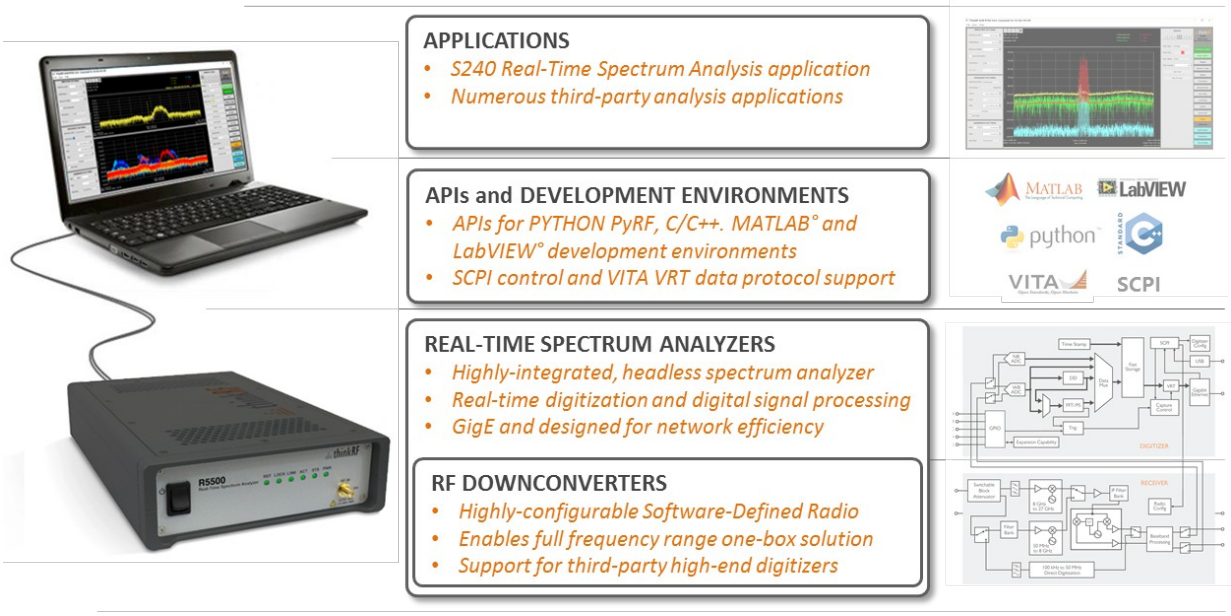


Figure 1: R57x0 Functional Block Diagram

ThinkRF's products conform with standardized protocols for interoperability. ThinkRF provides application programming interfaces (APIs) designed for easy integration with third-party applications. Standard protocols include the Standard Commands for Programmable Instruments (SCPI) protocol (page 43) for controlling and obtaining status from the RTSA and the VITA-49 Radio Transport (VRT) protocol (page 27) for digitized data and its associated context information.

In addition, API libraries, written in C/C++, Python, MATLAB and/or NI LabVIEW, are provided for quick interfacing, data acquisition and as well as for spectral analysis. The Python API is built within the PyRF development framework and is open-source under BSD licensing. PyRF handles the low-level details of real-time acquisition, signal processing and visualization, and provides feature rich libraries, example applications and source code, all specific to the requirements of signal analysis. Usage examples are provided through the available source codes of the Graphical User Interfaces (GUIs) or any applications included in each release package.

Refer to Appendix A for how to connect to an RTSA and Appendix B for the protocol on how to find any RTSAs available on the local network. The source code provided for the aforementioned APIs and GUIs/applications would serve as examples.

The R57x0 provides system level control and status commands as defined in Table 1.

Table 1: System Level Control/Status Commands with GNSS

SCPI Command	Description
:SYSTem	*Page 47*
:ABORt	Aborts the current data capturing process and puts the RTSA system into a normal manual mode (i.e. sweep, trigger, and streaming will be aborted)
:CAPTure	

SCPI Command	Description
:MODE?	Gets the current capture mode of the RTSA (i.e. sweeping, streaming or block mode)
:COMMunicate	
:HISLip	
:SESSion?	Returns the HiSLIP connection's session ID
:LAN<commands>	Subset of commands for configuring/querying RTSA's LAN settings
:ERRor	
[:NEXT]?	Returns the next error code and message from the SCPI error/event queue
:ALL?	Returns all the error codes and messages from the SCPI error/event queue
:CODE	
[:NEXT]?	Returns next the error code from the SCPI error/event queue
:ALL?	Returns all the error codes from the SCPI error/event queue
:COUNt?	Returns the number of errors in the SCPI error/event queue
:FLUSh	Clears the R57x0's internal data storage buffer of any remaining data that has not transferred out of the RTSA
:LOCK	
:HAVE?	Returns the current lock state of the task specified
:REQuest?	Requests the R57x0 to provide a lock on a specific task such that only the application that has the lock can perform the task
:OPTions?	Returns comma separated 3-digit values to represent the hardware option(s) or features available with a particular RTSA model
:SYNC	
:MASTer[?]	Sets an RTSA unit to be the master or slave for a synchronization trigger system with multiple units. Affects :TRIGger:TYPE PULSe or WORD.
:WAIT[?]	Sets the delay time in nanoseconds that the system must wait after receiving the trigger signal before performing data capture
:VERSion?	Returns the SCPI version number that the instrument complies with
:DATE[?]	Sets/reads date
:TIME[?]	Sets/reads time
:SYNC[?]	Sets/ gets the System time synchronization source via network or SCPI, or disable
:STATus	*Page 59*
:OPERation	
[:EVENt]?	Queries the Operation Status Register for any operation event
:CONDition?	Queries the Operation Condition Register for any operation event
:ENABle[?]	Enables or queries bits in the Operation Enable Register
:NTRansition[?]	Enables or queries bits in the Operation Negative Transition Register
:PTRansition[?]	Enables or queries bits in the Operation Positive Transition Register
:PRESET	Presets the R5500 (similar to *RST)
:QUEStionable	
[:EVENt]?	Queries the Questionable Status Register for any questionable event
:CONDition?	Queries the Questionable Condition Register for any questionable event
:ENABle[?]	Enables or queries bits in the Questionable Enable Register

SCPI Command	Description
:NTRansition[?]	Enables or queries bits in the Questionable Negative Transition Register
:PTRansition[?]	Enables or queries bits in the Questionable Positive Transition Register
:TEMPerature?	Returns the R57x0's internal ambient temperature
:GNSS	Page 72
[:Enable][?]	Enables or queries the status of the GNSS module
:ADELay[?]	Sets/reads antenna cable delay compensation
:CONStellation[?]	Sets one or two satellite constellations to be tracked or reads back
:POSition	Queries the last known GNSS position in degrees latitude, degrees longitude and altitude in meters
:REFerence?	Queries which timing reference source used to discipline the 10 MHz GNSS reference oscillator

See SCPI Command Set section (page 43 onward) for further details on the commands.

Caution pertaining to multi-users: See Appendix A: Connecting to RTSA for important notes on this caution.

The Architecture

The R57x0 is an integrated wireless radio receiver, digitizer/analyzer and GNSS technology. It has an embedded capture controller that enables users to:

- define and execute real-time and sophisticated triggers, traces and sweeps;
- configure the radio RFE and DSP in association with those traces or sweeps;
- obtain device's location position and time as provided by the GNSS through VRT packets (page 33); and
- have time-stamped with data output.

Traces and sweeps are controlled by the capture controller as illustrated in the Digitizer portion of Figure 2. A trace and a sweep are defined as a single (block or continuously streamed) capture and a series of captures, respectively, each with their associated hardware configurations.

When the GNSS module is enabled, GNSS position and time information is sent to users through VRT context packets roughly every second (see VRT's Formatted GPS Geolocation section). Besides two existing internal and external 10 MHz reference clock sources, the GNSS module also provides a third 10MHz reference clock source option to provide synchronized time-stamp for VRT packets (see SOURce Commands).

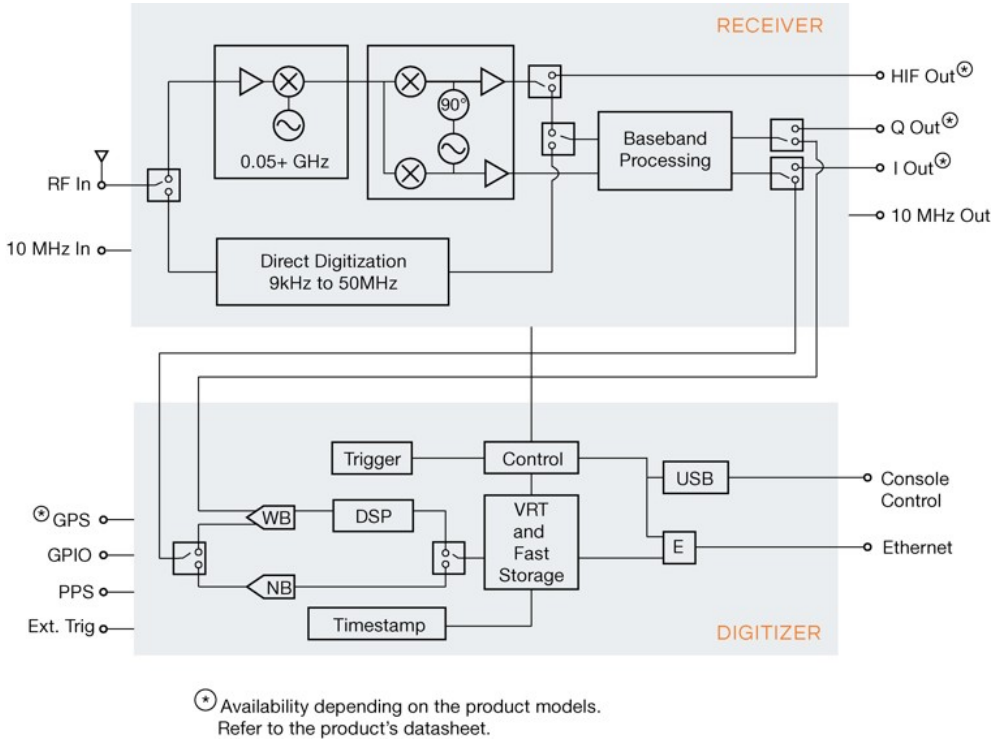


Figure 2: RF Receiver Front-end and Capture Controller Functional Block Diagram (*Note: The GPS (GNSS) option is included in this model)*

The R57x0 supports different RFE modes of operation and subsequent DSP capabilities as per Table 2 and as described in the following subsections.

Table 2: Radio RFE Modes and DSP Data Output Formats

Mode	Description	Freq Range (MHz)	IBW[3] (MHz)	Ouput Format (No DSP)	DSP Data Output Format		
					Decim-ation	Frequency Shift	IBW[3] (MHz)
ZIF	Zero-IF Receiver	50 - max	100	I_{14} Q_{14}	I_{14} Q_{14}	I_{14} Q_{14}	100 / decimation
SH	Super-Heterodyne Receiver	50 - max	40	I_{14}	I_{14} Q_{14}[1]	I_{14} Q_{14}	100 / decimation
SHN	SH Receiver with narrower BW	50 - max	10	I_{14}	I_{14} Q_{14}[1]	I_{14} Q_{14}	100 / decimation
HDR	High Dynamic Range Receiver	50 - max	0.1	I_{24}	–	–	–
DD	Direct Digitization Receiver	0.009 - 50[2]	50	I_{14}	I_{14}	I_{14} Q_{14}	No shifting: 50 / dec With shifting: 100 / dec

[1] For SH and SHN modes, when the decimation is used, a frequency shift of 35MHz will be applied automatically to bring the R57x0's center frequency back to the zero IF. Thus, the data output will be I and Q.

[2] In DD mode, there is no frequency tuning except for performing frequency shift. When decimation is applied, the decimation will be around the DD's center frequency of 0 Hz, plus the frequency shift value when used.

[3] IBW here refers also to the usable operating BW of the full hardware BW (which is of the sample rate, varies depending on the RFE mode and I/Q format).

R57x0 complies to VRT protocol for sending digitized IF data packets and their associated context information depending on the capture mode. It is very important to follow the VRT's IF Data Packet Class section (page 37) for the exact VRT data output formats as well as packing method.

RF Receiver Front-End

The Receiver portion of Figure 2 shows a block diagram of the RFE within the R57x0. The architecture consists of a super-heterodyne (SH) front-end with a back-end that utilizes an I/Q mixer similar to that in a direct-conversion (or zero-IF) receiver.

Depending on the frequency of the signals being analyzed, one of the three receiver signal processing paths is selected. Signals in the frequency range 9kHz to 50MHz are directly digitized, while all other signals are translated to the frequencies of the first IF block via one of the other two signal processing paths. The IF block consists of a bank of multiple SAW filters. SAW filter selection depends on the frequency of the input signal. The output of the SAW filter feeds the I/Q mixer.

The three signal processing paths are further classified into different modes of operation for the capture engine as shown in Table 2. The radio modes ZIF, SH, SHN and HDR support tuning the center frequency from 50MHz to the maximum frequency supported by the particular product model (ex. 8GHz, 18GHz, and 27GHz for R57x0-408, -418, and -427, respectively).

The ZIF, SH and SHN radio modes support a tuning resolution of 10Hz. Digital frequency shifting is then used to enhance the tuning resolution to the nearest 1Hz

(±0.23Hz). The frequency shifting technology used is an embedded Numerically Controlled Oscillator (NCO) based Direct Digital Synthesizer (DDS) as described in the Digital Down Converter subsection.

The HDR radio mode supports a tuning resolution of 10Hz. No further fine tuning is available.

The remaining radio mode, DD, supports 50MHz IBW Direct Digitization of the baseband from the external RF IN. Hence, this mode does not support frequency tuning of the radio although the DSP's frequency shift mode may be applied.

Direct-Conversion Receiver Technology

Direct-conversion (or ZIF) receivers are ideal for signal analysis of wideband waveforms, such as 4G/5G/LTE, Wi-Fi and Bluetooth. With that benefit comes the drawback of both IQ and DC offsets which are inherent to direct-conversion technology.

DC Offset Correction

The R57x0's WB ADC sampling rate is 125 MSa/s, intermediate frequency (IF) is 0 and the entire IF bandwidth is 125MHz. The analog filter results in an amplitude roll-off at approximately ±50MHz around the center frequency Fc, as illustrated in Figure 3.

Direct-conversion receivers have a DC offset at the center of the band. The offset is primarily compensated for in real-time in the receiver hardware but there always is some residual offset that (depending on the application and bandwidth of interest) might need to be compensated for in software. Several options such as calibration or dynamic offset compensation in software have been described in the open literature.

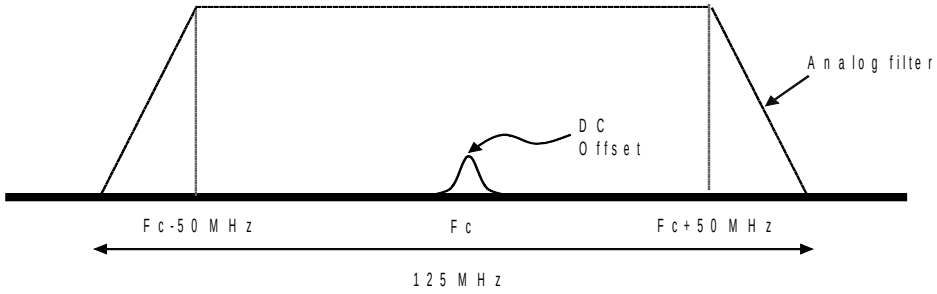


Figure 3: DC Offset with Amplitude Roll-Off at ±50MHz

If the application only needs to utilize up to 50MHz of IBW, a simple alternative to DC offset compensation is to use the SH mode of operation.

IQ Offset Correction

Direct-conversion receivers have phase and/or amplitude offsets between in-phase (I) and quadrature (Q) components of the baseband signal. Due to this, when an FFT is performed on digitized baseband data where there is a signal tone present, there will be an 'image' at the same frequency offset from the center frequency as the tone itself. This is illustrated in Figure 4.

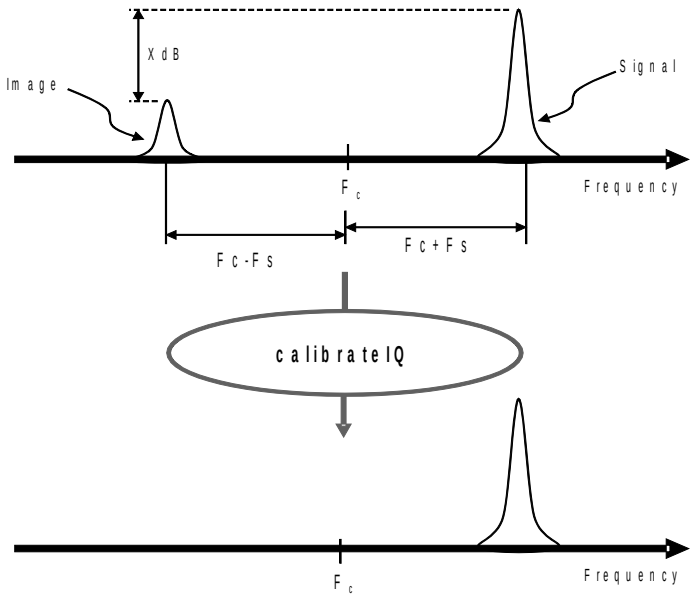


Figure 4: IQ Offset Correction

A correction algorithm would be needed to adjust this offset necessary for signal analysis, especially for the ZIF mode. The ThinkRF's APIs have included a correction.

Table 3: RF Front-End Control/Status Commands

SCPI Command	Description
:INPut	*Page 64*
:ATTenuator[?]	Enables/disables the front-end's attenuation for R57x0-**408** & their variants only
:VARiable[?]	Sets the variable attenuation for R57x0-**418 and -427** & their variants
:GAIN[?]	Sets the input gain stage for R57x0-**418, -427** & their variants
:HDR[?]	Sets gain level for the NB ADC of of the HDR signal path
:MODE[?]	Selects the receiver mode of operation
:SOURce	*Page 67*
:REFerence	
:PLL[?]	Selects the 10MHz reference clock source
:PPS[?]	Selects the PPS input source
[:SENSe]	*Page 68*
:DECimation[?]	Sets the decimation rate as an exponent of 2 (i.e. rate = 2^{level} where level = 0, 1, 2 - 10)
:FREQuency	
:CENTer[?]	Sets the center frequency of the RFE
:IF?	Queries the IF frequencies that are used for the current input mode and center frequency
:LOSCillator?	Gets the frequency of the external LO 1, 2 or 3 in corresponding to current the RTSA's center frequency
:SHIFt[?]	Sets the frequency shift value (not available for HDR mode)
:LOCK	
:REFerence?	Queries the lock status of the PLL reference clock

SCPI Command	Description
:RF?	Queries the lock status of the RFE's PLL

See SCPI Command Set section (page 43 onward) for further details on each set of commands.

Digital Signal Processing

The R57x0 has embedded DSP blocks to provide further signal processing capabilities, such as DDC with up to 10 levels of decimation and FFT computation.

Digital Down Converter

The DDC block takes the frequency band of interest and shifts it down in frequency, then provides decimation of the sampling rate to one that is lower and consistent with the bandwidth of the signal of interest. This enables channelization of signals having bandwidth smaller than the IBW.

Referring to Figure 5, the DDC has two major elements, an NCO (DDS) and a down sampling with filtering. The NCO generates a complex sinusoid, which is mixed with the IQ input using a complex multiplier, to shift or offset the signal spectrum from the selected carrier frequency. This process provides the frequency fine-tuning (and shifting) feature as mentioned in the previous subsections.

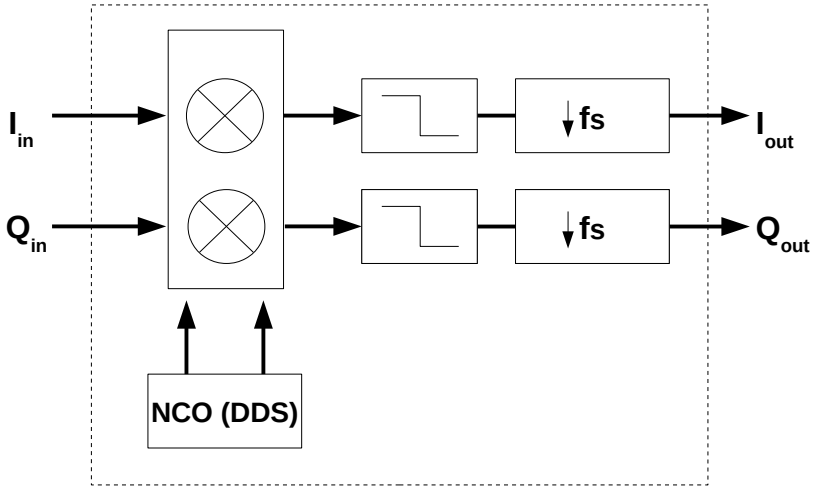


Figure 5: DDC Functional Block Diagram

The complex multiplication is then followed by either a finite impulse response (FIR) filter or cascaded integrator-comb (CIC) filters with a FIR filter combined. The CIC filter has a 'droop' associated with it in the passband. In order to compensate for this droop, the CIC filter is followed by a compensating FIR filter. Each filter type has its own decimator. This whole process effectively reduces the sample rate and filters the signal to remove adjacent channels, minimize aliasing, and maximize the received signal-to-noise ratio.

Note: The use of the NCO converts the in-phase signal (I data) input of the receiver's DD, SH and SHN processing paths to complex I and Q data output. See Table 2.

Triggers

Triggers provide a means of qualifying the storage of captured time domain IQ data based on an external, periodic or frequency domain event. Triggering can be considered a means of filtering signals of interest for the purposes of subsequent visualization and/or analysis.

The following describes the different types of triggers and their common controls. Selection of different types is mutually exclusive.

Frequency Domain Triggering

Frequency domain triggering relies on the embedded real-time FFT mechanism to transform the sampled signal from the time domain to the frequency domain. The R57x0 uses a 1024 point real-time FFT core embedded within the FPGA to transform 1024 time domain IQ samples to 1024 frequency domain FFT bins. Each bin is an average of the spectral activity over a range of 125MHz divided by the DDC decimation rate divided by the 1024 FFT points.

The frequency domain triggering supported by R57x0 is a level trigger type, used to capture any signal above the noise floor within a specified frequency range. The user defines a single amplitude level within a frequency range. The frequency range encompasses all FFT bins with center frequencies within the range defined by START and STOP. If the sampled signal amplitude exceeds the defined trigger level at any single sample within the defined frequency range, the trigger will occur and the subsequent IQ data capture will proceed.

Figure 6 illustrates the association of the time domain and the frequency domain for a 100 MHz IBW operation as an example. The internal frequency domain data lags the time domain data by a latency not shown here to simplify the explanation. After a trigger event is detected, the subsequent time domain IQ data is then stored to memory.

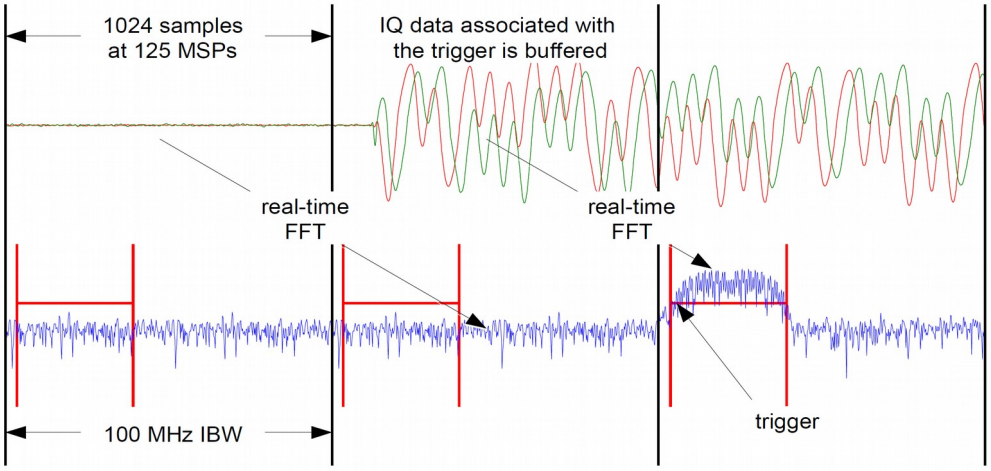


Figure 6: Association between Time and Frequency Domain

The measurable range of the input signal, and the corresponding allowable trigger level range, varies depending on the selected center frequency, the calibrated reference level and the attenuation setting. The threshold level error is approximately ±3 dBm or less when the trigger level is set within the range mentioned in the :TRIGger:LEVel command.

See TRIGger Commands (page 74) or SWEep's trigger (page 87) for further details.

Periodic Triggering

Periodic triggering provides a means of capturing a defined amount of IQ data on a periodic basis. Periodic triggering is typically used for statistical analysis of the captured signal.

External Triggering

External triggering provides a means of synchronized triggering based on the receiving of a trigger signal provided via the R57x0's GPIO. The trigger "signal" could be a single pulse, PPS or a sync-word. See Synchronized Sweep (page 25) for additional details.

Caution: The pulse and sync-word is applied to the GPIO's TRIG IN pin, while PPS is through PPS pin. Contact ThinkRF's Support for details on how to use the GPIO port prior to connecting anything to the port.

Table 4: Trigger Control/Status Commands

SCPI Command	Description
:TRIGger	*Page 74*
:TYPE[?]	Sets or disable the trigger type including LEVel | PERiodic | PPS | PULSe | WORD | NONE
:LEVel[?]	Sets the frequency range and amplitude of a frequency domain level trigger
:PERiodic[?]	Sets the time period of a periodic trigger

See TRIGger Commands (page 74) or SWEep's trigger (page 87) for further details.

Capture Controller

The Capture Controller provides a means of defining and performing simple traces and complex sweeps. For example, it allows for:

- the definition and execution of a complex sweep;
- the interruption of that sweep;
- the execution of a specific trace; and
- the resumption of the previous sweep.

Caution: The configurations of the capture engine associated with :TRACe and :SWEep commands are fully independent of each other. A :TRACe command uses the configurations of the capture engine based on the root :INPut, :SENSe and :TRIGger commands. It does not use the configurations based on the :SWEep command subset.

Note: Besides the information provided in the following sections, refer to appnotes "74-0050 Data Acquisition" and "74-0071 Sweep Synchronization with I and Q output" for more information, including examples.

Trace Capture Control

The :TRACe capture control initiates the capture, storage and conditionally the sending of IQ data through triggering when used. It supports both streaming and block mode capture.

The :TRACe:BLOCk (page 77) command initiates a block capture of continuous IQ data (available to be "pulled" from the R57x0 per command issued). Once it is issued, data will be stored instantly (conditional on triggering), contiguously and reliably and are available to be read. The maximum size of a block is limited by the memory device in the RTSA.

The :TRACe:STReam (page 79) command initiates the streaming of IQ data (which is "pushed" from the R57x0). Once it is issued, data packets will be sent instantly (conditional on triggering) and continuously on best effort basis (in other words, data might not be continuous from one packet to the next once the internal buffer is full).

The execution of the trace capture could be conditioned by the triggering. The triggering may be enabled or disabled via the :TRIGger:TYPE command, thereby, supporting free-run or triggered signal searches.

Table 5: Trace Capture Control Commands

SCPI Command	Description
:TRACe	*Page 76*
:BLOCk	
:DATA?	Initiates the sending of the IQ data captured
:PACKets[?]	Sets the number of IQ data packets to be captured per block (a block = :PACKets * SPP)
:SPPacket[?]	Defines the number of samples per VRT packet
:STReam	
:STARt	Initiates the capture, storage and streaming of IQ data
:STOP	Stops streaming

See TRACe Commands section (page 76) for further details.

Sweep Capture Control

The :SWEep capture control provides the ability to define and execute simple or complex sweeps. A sweep setup consists of defining a list or multiple lists and executing one of the defined lists, with each list consisting of one or more entries storing different capture engine configurations. A list may be edited, deleted and/or executed using the :SWEep:LIST command set.

The :SWEep:ENTRy commands provide the ability to define the capture engine configurations equivalent to most of :INPut, :SENSe, :TRACe and :TRIGger commands for each sweep entry. Sweep entries are identified by an index number and may be inserted, edited and/or deleted like rows in a table or spreadsheet.

There are slight differences between the configuration options for trace versus sweep captures. The sweep allows for definition of a range of center frequencies whereby the center frequency is incremented in frequency by a step value. Level triggers may be

defined over the entire range of center frequencies. Sweeping does not support time delayed triggers.

In addition, sweep mode data packets, whether VRT context or digitized data, are “streamed” or “pushed” from the RTSA (similar to :TRACe:STReam).

Table 6: Sweep Capture Control/Status Interface

SCPI Command	Description
:SWEep	*Page 79*
:LIST	
:ITERations[?]	Defines the number of times the list is repeated during execution
:STARt	Begins execution of the current sweep list from the first entry
:STATus?	Get the current sweep status
:STOP	Stops execution of the current sweep list
:ENTRy	*All entry commands operate on the current list*
:COPY	Copies the settings of an existing sweep entry into the current settings for quick editing
:COUNt?	Gets the number of entries available in the list
:DELETE	Deletes the specified entry or all entries
:NEW	Sets the sweep entry settings to default values
:READ?	Gets the settings of an existing sweep entry
:SAVE	Saves the current editing entry to the end of the list or before the specified ID location in the list when the integer value is given
:ATTenuator	As defined in :INPut:ATTenuator, page 64
:VAR[?]	As defined in :INPut:ATTenuator:VARiable, page 64
:DECimation[?]	As defined in [:SENSe]:DECimation, page 68
:FREQuency	
:CENTer[?]	Defines the center frequency of the RFE or a range of frequencies that are stepped by the value defined by the :FREQuency:STEP
:STEP[?]	Defines the amount of frequency that the center frequency is stepped by
:SHIFt[?]	As defined in [:SENSe]:FREQuency:SHIFt, page 70
:GAIN	
:HDR[?]	As defined in :INPut:GAIN:HDR, page 66
:MODE	As defined in :INPut:MODE, page 66
:DWELl[?]	Sets the maximum amount of time waited for a trigger to occur after which the trigger is aborted
:PPBlock[?]	Same as :TRACe:BLOCk:PACKets, page 77
:SPPacket[?]	As defined in :TRACe:SPPacket, page 78
:TRIGger	
:TYPE[?]	As defined in :TRIGger:TYPE, page 74
:LEVel[?]	As defined in :TRIGger:LEVel, page 75
:PERiodic[?]	As defined in :TRIGger:PERiodic, page 76

See SWEep Commands section (page 79) for further details.

Synchronized Sweep

The R57x0 supports a synchronized sweep function for the purposes of comparing the same signal received via multiple R57x0s.

Synchronized sweep is an extension of the external trigger capability. One of the R57x0s in a network is configured to be the master (:SYSTem:SYNC:MASTer ON) and the other R57x0s are configured as slaves (:SYSTem:SYNC:MASTer OFF). The master and slaves are configured with a sweep list, in which each sweep entry has a synchronization trigger type (:SWEep:ENTRy:TRIGger:TYPE PULSE | WORD). The synchronization trigger is generated and delivered from the master's GPIO to that of the slaves to indicate the beginning of a capture.

Figure 7 provides a synchronization trigger example using sync-word. The master sends the sync-word when the setup of its front-end has been completed. Master and slaves are also individually configured with a delay variable (:SYSTem:SYNC:WAIT <nsec> with a resolution of 8 nsec). This delay wait time accounts for the typical worst-case front-end setup time and for differences in the synchronization cable length. Master and slaves then begin the capture upon the expiration of the wait (or delay).

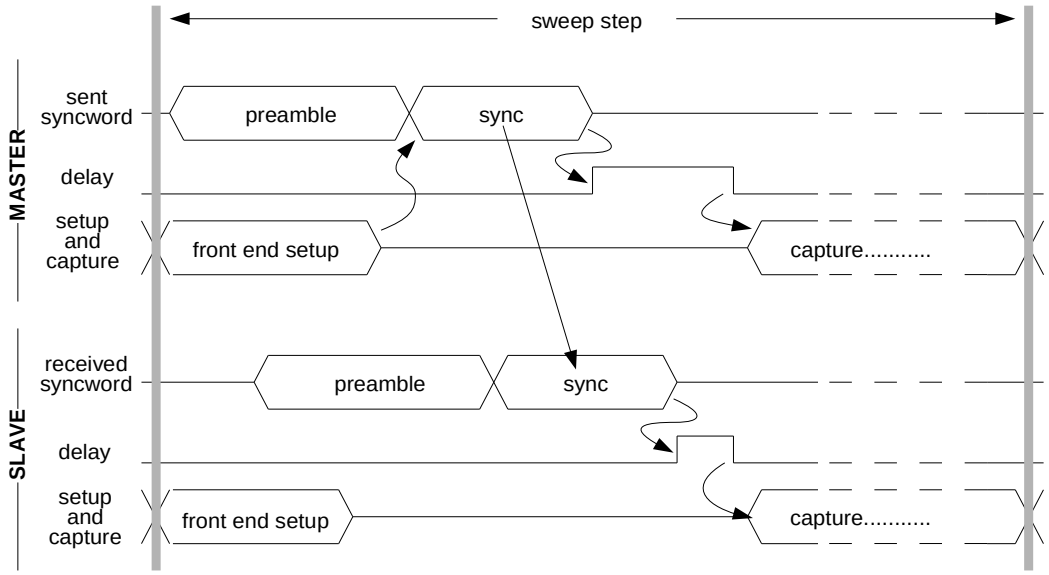


Figure 7: Synchronized Sweep using Sync-Word

The front-end setup time is typically of approximately 200 usec but is variable due to the embedded running processes. Referring to Figure 8, if the front-end setup time on one (or more) of the slaves is longer than the combined duration of the master's setup time plus the sync-word plus the slave's delay, then the slave will miss the beginning of the capture. The host-side application that is collating the capture data may recognize the missed capture by noting the timestamps and/or frequency of the capture data within the associated VRT Receiver Context packets. The rate of sweep versus the amount of missed captures may be balanced by adjusting the delay values.

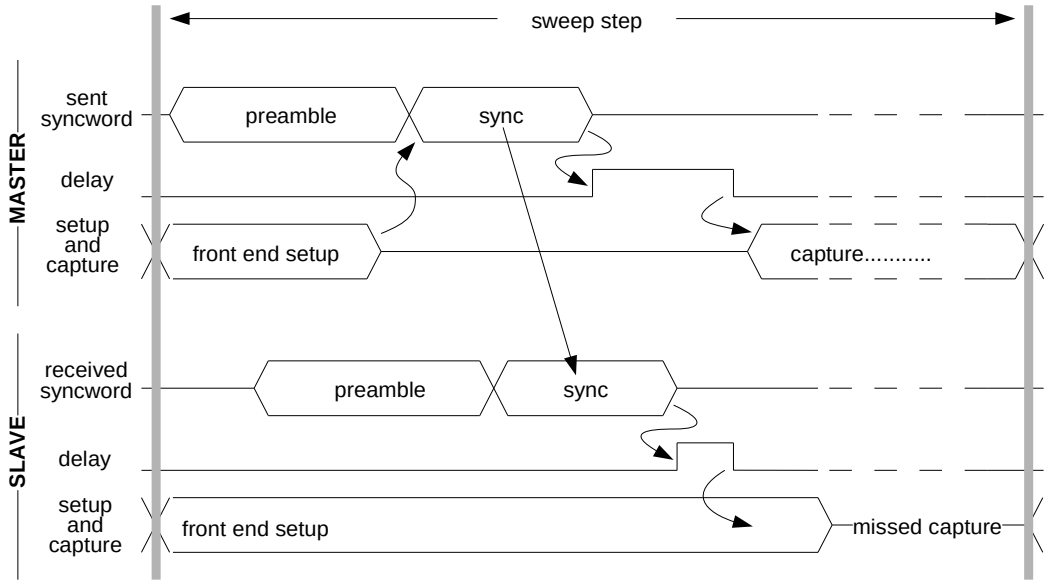


Figure 8: Synchronized Sweep with a Missed Capture

See SWEep Commands section (page 79) for further interface details or contact ThinkRF's Support for more information.

VITA-49 Radio Transport Protocol

The section describes the R57x0's VRT Information Class as per the "VITA Radio Transport (VRT) Draft Standard" Specification VITA-49.0 – 2007 Draft 0.21.

Purpose

Convey an arbitrary 100MHz of IF data and associated information from the R57x0 to another equipment using an industrial standard.

R57x0's VRT Overview

ThinkRF's VRT supports four different packet streams of information defined and organized as shown in Figure 9 and Table 7. The streams of packets are sent when the data capture is started. The context packets carry the R57x0 settings information associated with the immediately following IF data packets.

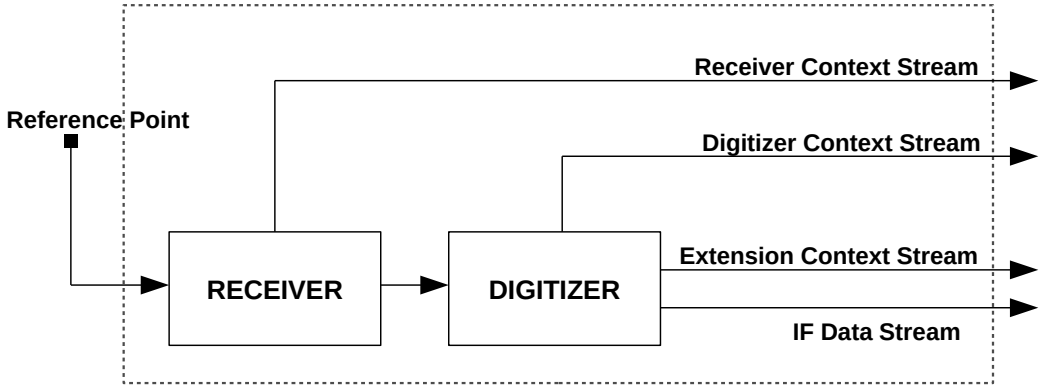


Figure 9: Connectivity and 4 Different Packet Streams Supported by R57x0

Table 7: The Categories of VRT Packet Streams Supported by ThinkRF's R57x0

Contents	Standard Formats	Custom Formats
Context	**IF Context Packet Stream** conveys metadata concerning IF Data Packet Stream and the settings *- Digitizer Context Packet Class Stream* *- Receiver Context Packet Class Stream*	**Extension Context Packet Stream** conveys additional Context concerning IF or Extension Data Packet Stream *- Extension Context Packet Class Stream*
Data	**IF Data Packet Stream** conveys discrete time sampled signal data *- IF Data Packet Class Stream*	**Extension Data Packet Stream** conveys any signal or data derived from a signal *- Currently not used*

Receiver Context Packet Class Stream

The Receiver Context Packet Class Stream is used to convey informational messages about changes in the configuration and status of the RF receiver in the R57x0.

Digitizer Context Packet Class Stream

The Digitizer Context Class Stream is used to convey information messages about changes in the configuration and status of the IF digitizer in the R57x0.

Extension Context Packet Class Stream

The Extension Context Packet Class Stream is used to convey metadata for the IF Data Packet Stream, which no provision has been made in the IF Context Packet Stream.

IF Data Packet Class Stream

The IF Data Packet Stream is used to convey complex IQ samples from the digitizer to devices external to the R57x0.

Table 8 summarizes numerically the list of Stream Identifiers used by ThinkRF for different Packet Class Stream. Each ID will be mentioned in the subsequent corresponding Packet Class sections.

Table 8: A List of Stream Identifiers as Used by ThinkRF for Different Packet Classes

Stream Identifier	Packet Class
0x90000001	Receiver Context
0x90000002	Digitizer Context
0x90000003	IF Data – $\{I_{14}Q_{14}\}$ Format
0x90000004	Extension Context
0x90000005	IF Data – $\{I_{14}\}$ Format
0x90000006	IF Data – $\{I_{24}\}$ Format

Packet Classes and Streams

This section describes in details the rules and structure of those Packet Classes and Streams. By definition, a series of packets instantiated from the same Packet Class form a Packet Stream. See Table 8 for the list of Stream Identifiers used to identify these different Packet Classes.

Note: All data words in each VRT packets are in big-endian order, and sent MSB first.

Receiver Context Packet Class

This Packet Class is a type of IF Context Packet Class. The packet information conveys changes in the configuration and status of the R57x0's RF receiver.

Table 9: Receiver Context Packet Class Structure

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pkt Type 0 1 0 0				C	R		T S M	TSI		TSF		Pkt Count				Pkt Size															
Stream Identifier (1 word)																															
Timestamp - Integer Seconds (1 word)																															
Timestamp - Integer Picoseconds (2 words)																															
Context Indicator Field (1 word)																															
Context Fields (Variable Size, see Table 11, 3rd column)																															

1. **Pkt Type** is **0100** to indicate this is a context packet.
2. **C** is set to **0** to indicate there is no Class Identifier in the packet.
3. **R** is **00** as they are reserved bits.
4. **TSM** (TimeStamp Mode) is **0**, indicating that context packet timestamps are precise.
5. **TSI** (TimeStamp-Integer) field is **01**, indicating that integer (seconds) part of the timestamps are in Coordinated Universal Time (UTC).
6. **TSF** (TimeStamp-Fractional) field is **10**, indicating that the fractional part of the timestamp measures in real time picosecond resolution.
7. **Pkt Count** starts at 0000 and increments once for each context packet, until reaching 1111 (or 15), where it shall rollover to 0000 on the next count.
8. **Pkt Size** indicates the total number of 32-bit words in the entire context packet, including all headers, the context indicator field and context sections.
9. **Stream Identifier** is the 32-bit word, **0x90000001**
10. **Timestamp - Integer Seconds** is in UTC format and represents the number of seconds occurred since Midnight, January 1, 1970, GMT.
11. **Timestamp - Integer Picoseconds** counts the number of picoseconds past since the last increment of the Timestamp seconds field. See the Picosecond Timestamp Words Format section for the format.
12. The **Context Indicator Field** follows the format indicated in Table 10.

Table 10: Receiver Context Indicator Field Positions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I	-			F	-			G	-																						

13. The **Context Fields** section contains a context field for every field that is indicated to be present in the Context Indicator Field. *The fields are arranged in the identical order of their occurrence in the Context Indicator Field.* See Table 11 for the definition and associated value of each field.

Table 11: Receiver Context Field Definition and Values

Indicator Bit Name	Bit Position	Context Field Type	# of Words in Context Fields	Period of Validity
I	31	Context Field Change Indicator	0	N/A
F	27	RF Reference Frequency	2	Persistent
G	23	Gain	1	Persistent

Context Field Change Indicator

The Context Field Change Indicator is used to indicate when some context value of the system has changed. One or more of the other bits in the indicator field will be also set, indicating which values have been changed and have their updated values in the context fields that follow. It is possible that a context packet may be sent where the Context Field Change Indicator is set to 0, indicating that no change has occurred.

RF Reference Frequency

The RF Reference Frequency communicates the frequency of origin for the signal. The value of the RF Reference Frequency is expressed in units of Hertz. The RF Reference Frequency sub-field uses the 64-bit, two's-complement format as shown in Table 12. This field has an integer and a fractional part, with the radix point to the right of bit 20 in the second 32-bit word. This gives the RF Reference Frequency a range of ±8.79THz with a resolution of 0.95µHz.

Table 12: RF Reference Frequency Word Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Integer RF Reference Value (43..12), Hz																															
Integer RF Ref. Value (11..0), Hz												Fractional RF Reference Value(19..0)																			

Gain

The gain is a 32-bit value that is split into two 16-bit values, representing the Stage 1 and Stage 2 gain values. The Stage 1 gain represents the amount of gain in the front-end system, the RF gain. The Stage 2 gain represents the amount of gain in the back-end system, the IF gain.

Each gain value is a signed two's-complement number, having two sub-fields, bits 15:7 being the integer value, and 6:0 being the fractional value. This gives each gain figure a range of ±256dB with a resolution of 1/128dB (0.0078125dB).

Table 13: Gain Field Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Integer IF									Fractional IF							Integer RF									Fractional RF							

Digitizer Context Packet Class

This Packet Class is a type of IF Context Packet Class. The packet information conveys changes in the configuration and status of the R57x0's IF digitizer.

Table 14: Digitizer Context Packet Class Structure

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pkt Type 0 1 0 0				C	R		T S M	TSI		TSF		Pkt Count				Pkt Size															
Stream Identifier (1 word)																															
Timestamp - Integer Seconds (1 word)																															
Timestamp - Integer Picoseconds (2 words)																															
Context Indicator Field (1 word)																															
Context Fields (Variable Size, see Table 16, 3rd column)																															

1. **Pkt Type** is **0100** to indicate this is a context packet.
2. **C** is **0** to indicate there is no Class Identifier in the packet.
3. **R** is **00** as they are reserved bits.
4. **TSM** is **0**, indicating that context packet timestamps are precise.
5. **TSI** field is **01**, indicating that integer (seconds) part of the timestamps are in UTC.
6. **TSF** field is **10**, indicating that the fractional part of the timestamp measures real time picosecond resolution.
7. **Pkt Count** starts at 0000 and increments once for each context packet, until reaching 1111 (or 15), where it shall rollover to 0000 on the next count.
8. **Pkt Size** indicates the size of the entire context packet, including all headers, the context indicator field and context sections.
9. **Stream Identifier** is the 32-bit word, **0x90000002**.
10. **Timestamp - Integer Seconds** is in UTC format and will represent the number of seconds occurred since Midnight, January 1, 1970 GMT.
11. **Timestamp - Integer Picoseconds** counts the number of picoseconds past since the last increment of the Timestamp seconds field. See the Picosecond Timestamp Words Format section for the format.
12. The **Context Indicator Field** follows the format indicated in Table 15.

Table 15: Digitizer Context Indicator Field Bit Positions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I		B			O		R										G														

13. The **Digitizer Context Fields** section contains a context field for every field that is indicated to be present in the Context Indicator Field. The fields are arranged in the identical order of their occurrence in the Context Indicator Field. See Table 16 for the definition and associated value of each field.

Table 16: Digitizer Context Field Values

Indicator Bit Name	Bit Position	Context Field Type	# of Words in Context Fields	Period of Validity
I	31	Context Field Change Indicator	0	N/A
B	29	Bandwidth	2	Persistent
O	26	RF Frequency Offset	2	Persistent
R	24	Reference Level	1	Persistent
G	14	Formatted GPS Geolocation	11	Persistent

Context Field Change Indicator

The Context Field Change Indicator is used to indicate when some context value of the system has changed. One or more of the other bits in the indicator field will be also set, indicating which values have been changed and have their updated values in the context fields that follow. It is possible that a context packet may be sent where the Context Field Change Indicator is set to 0, indicating that no change has occurred.

Bandwidth

The Bandwidth is used to indicate that the amount of spectrum that is currently view-able due to decimation settings.

The Bandwidth field has the 64-bit, two's-complement format as shown in Table 17. This field has an integer and a fractional part, with the radix point to the right of bit 20 in the second 32-bit word. This gives the RF Reference Frequency a range of ±8.79THz with a resolution of 0.95µHz.

Table 17: Bandwidth Word Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Integer Bandwidth (43..12), Hz																															
Integer Bandwidth (11..0), Hz												Fractional Bandwidth (19..0)																			

RF Frequency Offset

The RF Frequency Offset specifies the amount of frequency in Hz the received data has been shifted.

The RF Frequency Offset field has the 64-bit, two's-complement format as shown in Table 18. This field has an integer and a fractional part, with the radix point to the right of bit 20 in the second 32-bit word. This gives the RF Reference Frequency a range of ±8.79THz with a resolution of 0.95µHz.

Table 18: RF Frequency Offset Word Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Integer RF Reference Value (43..12), Hz																															
Integer RF Ref. Value (11..0), Hz												Fractional RF Reference Value(19..0)																			

Reference Level

The Reference Level provides a power level reference so that the magnitude of the received data can be calculated by a user. The reference level provided in the context

packet is adjusted according to the RFE's gain setting. However, the reference levels must also be adjusted accordingly any other conditions that might applied.

The absolute power level P (in dBm) is then computed using the following formula:

$$P = R + 20 * \log\left(IQ_{measured}\right)$$

with

$$IQ_{measured} = \sqrt{\left(I_{fft}^2 + Q_{fft}^2\right)}$$

where:

R = the reference level provided in the VRT context packet, dBm
$IQ_{measured}$ = as shown in the formula above. The $IQ_{measured}$ formula, however, is a simplified example as it doesn't include any corrections, such as IQ imbalance, DC offset or windowing
I_{fft} = the real component of the FFT (Fast Fourier Transform) computation applied on the VRT I data, which is normalized by dividing each Q by $2^{bit\_size - 1}$
Q_{fft} = the imaginary component of the FFT computation applied on the VRT Q data (when used), which is normalized by dividing each Q by $2^{bit\_size - 1}$

The Reference Level field uses the 32-bit format shown in Table 19 with the upper 16 bits reserved and is set to zero. The Reference Level field value is expressed in signed two's-complement format in the lower 16 bits of this field. This field has an integer and a fractional part, with the radix point to the right of bit 7.

Table 19: Reference Level Field Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Integer Ref. Level									Frac. Ref. Level						

The value of the Reference Level field has a range of nearly ±256dBm with a resolution of 1/128dBm (0.0078125dBm).

For examples, a Reference Level field value of:

- 0x0080 represents a reference level of +1dBm,
- 0xFF80 represents -1dBm,
- 0x0001 represents +0.0078125dBm, and
- 0xFFFF represents -0.0078125dBm.

Formatted GPS Geolocation

Note: For legacy specification reasons, such as with ANSI/VITA 49, the term GPS, instead of GNSS, is used in this section to refer generically to Global Navigation Satellite Systems.

The Formatted GPS Geolocation field communicates the location and time of the RTSA. The field uses eleven 32-bit words format as shown in Table 20 with the upper 16 bits reserved and is set to zero. The Reference Level field value is expressed in signed two's-complement format in the lower 16 bits of this field. This field has an integer and a fractional part, with the radix point to the right of bit 7.

Table 20: Formatted GPS Geolocation Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TSI		TSF		GPS Manufacturer OUI																							
Integer-second Timestamp of Position Fix (31..0)																															
Upper Fractional-second Timestamp of Position Fix (63..32)																															
Lower Fractional-second Timestamp of Position Fix (31..0)																															
Latitude (31..0), degrees																															
Longitude (31..0), degrees																															
Altitude (31..0), meters																															
Speed over Ground (31..0), meters/second																															
Heading Angle (31..0), degrees																															
Track Angle (31..0), degrees																															
Magnetic Variation (31..0), degrees																															

1. **TSI** field is **10**, indicating that Integer-second part of the timestamps are in GNSS time.
2. **TSF** field is **10**, indicating that the fractional part of the timestamp measures in real time picosecond resolution.
3. The **GPS Manufacturer OUI** subfield contains the 24-bit field for the IEEE-assigned Organizationally Unique Identifier of the GPS manufacturer.
4. The **Timestamp of Position Fix** subfields may differ from the Timestamp fields in the Digitizer Context packet that contains them. Note that the Timestamp of Position Fix fields refer to the time of the most recent GPS Position Fix, which may not be the current time or the Timestamp of the Digitizer/IF Context packet itself. These fields are useful for indicating the loss of the GPS signal.
5. The **Geolocation Angle Format** describes angles in units of degrees. The format uses the 32-bit, two's-complement format shown in Table 21. This field has an integer and a fractional part with the radix point to the right of bit 22. Hence, it has a possible range of ±512 degrees and a resolution of 2.38×10^{-7} degrees. Particular angular measurements allow various ranges such as 0 to 360 degrees, ±180 degrees, or ±90 degrees.

   This format is used for several other subfields within the GPS Geolocation field including **Latitude**, **Longitude**, **Heading Angle**, **Track Angle**, and **Magnetic Variation** subfields, which will be described further in the following points.

Table 21: Geolocation Angle (Degrees) Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Integer Angle, degrees										Fractional Angle, degrees																					

6. The **Latitude** subfield uses the Geolocation Angle Format shown in Table 21 with value ranging from -90.0 (South) to +90.0 (North) degrees.
7. The **Longitude** subfield uses the Geolocation Angle Format shown in Table 21 with value ranging from -180.0 (West) to +180.0 (East) degrees.
8. The **Altitude** subfield uses the 32-bit, two's-complement format shown in Table 22 with value expressed in units of meters. This subfield has an integer and a fractional part with the radix point to the right of bit 5. Hence, the subfield has a range of ±67108 kilometers and a resolution of 3.1 centimeters. In addition, the Altitude subfield references the height above the WGS-84 ellipsoid.

Table 22: Altitude Subfield Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Integer Altitude, meters																											Fractional Alt.					

9. The **Speed Over Ground** subfield uses the 32-bit, two's-complement format shown in Table 23 with value expressed in units of meters per second. It has an integer and a fractional part with the radix point to the right of bit 16. Thus, the subfield has a range of 0 to 65636 m/s and a resolution of 1.5e-5 m/s.

Table 23: Speed Over Ground Subfield Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Integer Speed, m/s																Fractional Speed, m/s															

10. The **Heading Angle** subfield uses the Geolocation Angle Format shown in Table 21. It expresses the platform's *orientation* with respect to true North in decimal degrees. Its value ranges from 0.0 to +359.999999761582 degrees.
11. The **Track Angle** subfield uses the Geolocation Angle Format shown in Table 21. It conveys the platform's *direction* of travel with respect to true North in decimal degrees. Its value ranges from 0.0 to +359.999999761582 degrees.
12. The Latitude, Longitude, Altitude, Speed Over Ground, Heading, Track Angle, and Magnetic Variation subfields take the value 0x7FFFFFFF when unspecified.

Extension Context Packet Class

This Packet Class conveys metadata concerning IF Data Packet Class that cannot be communicated in the IF Context Packet Class. See Table 24 for the organization of this context packet class.

Table 24: Extension Context Packet Class Structure

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pkt Type 0 1 0 1				C	R		T S M	TSI		TSF		Pkt Count				Pkt Size															
Stream Identifier (1 word)																															
Timestamp - Integer Seconds (1 word)																															
Timestamp - Integer Picoseconds (2 words)																															
Context Indicator Field (1 word)																															
Context Fields (Variable Size)																															

1. **Pkt Type** is **0101** to indicate this is an extension packet.
2. **C** is **0** to indicate there is no Class Identifier in the packet.
3. **R** is **00** because they are reserved bits.
4. **TSM** is **0**, indicating that context packet timestamps are precise.
5. **TSI** field is **01**, indicating that integer (seconds) part of the timestamps are in UTC.
6. **TSF** field is set to **10**, indicating that the fractional part of the timestamp measures in real time picosecond resolution.
7. **Pkt Count** starts at 0000 and increments once for each context packet, until reaching 1111, where it shall rollover to 0000 on the next count.
8. **Pkt Size** indicates the total number of 32-bit words in the entire context packet, including all headers, the context indicator field and context sections.

9. **Stream Identifier** is set to be **0x90000004**
10. **Timestamp - Integer Seconds** is in UTC format and represents the number of seconds occurred since Midnight, January 1, 1970, GMT.
11. **Timestamp - Integer Picoseconds** counts the number of picoseconds past since the last increment of the Timestamp seconds field. See the Picosecond Timestamp Words Format section for the format.
12. The **Context Indicator Field** follows the format indicated in Table 25.

Table 25: Extension Context Indicator Field Positions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I																												IQ	Sc	St	Sw

13. The **Context Fields** section contains a context field for every field that is indicated to be present in the Context Indicator Field. The fields are arranged in the identical order of their occurrence in the Context Indicator Field. See Table 26 for the definition and associated value of each field.

Table 26: Extension Context Field Definition and Values

Indicator Bit Name	Bit Position	Context Field	# of Words in Field	Period of Validity
I	31	Context Field Change Indicator	0	N/A
IQ	3	IQ Swapped Indicator	0	Persistent
Sc	2	N/A	N/A	N/A
St	1	New Stream Start ID	1	Persistent
Sw	0	New Sweep Start ID	1	Persistent

Context Field Change Indicator

The Context Field Change Indicator is used to indicate when some context value of the system has changed. One or more of the bits in the indicator field will then be set, indicating which values have been changed and have their updated values in the context field(s) that follow. It is possible that a context packet may be sent where the Context Field Change Indicator is set to 0, indicating that no change has occurred.

IQ Swapped Indicator

The IQ Swapped Indicator is used to indicate whether the two ADC data channels has been swapped due to the mixing of a high-side or low-side LO injection at a given center frequency. When the value is 1, the channels are swapped and 0 if not. This swapping is necessary to maintain the data output format to always be {I,Q} such that the spectral inversion is not required at the user-end during data processing.

This information, however, matters only when operating in the ZIF mode. The ZIF mode is a direct conversion receiver architecture that typically creates some artifacts, mainly due to IQ gain and phase imbalances. The artifacts are compensated by using a specific correction algorithm, which incorporates both time- and frequency-domain corrections.

Further information on the algorithm and its usage will be provided in a future release. Contact ThinkRF's Support for more information if necessary.

New Stream Start ID

The New Stream Start ID indicator indicates a new stream capture has started, any packets following this Context Packet belong to this new stream capture.

The value of the New Stream Start ID field uses the 32-bit unsigned integer format shown in Table 27.

Table 27: New Stream Start ID Field Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
New Stream Start ID																															

New Sweep Start ID

The New Sweep Start ID indicator indicates a new sweep has started, any packets following this Context Packet belong to this new sweep.

The value of the New Sweep Start ID field uses the 32-bit unsigned integer format shown in Table 28.

Table 28: New Sweep Start ID Field Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
New Sweep Start ID																															

IF Data Packet Class

The IF Data Packet Class conveys digitized IF Data from the digitizer to devices external to the R57x0. The payload data and its output format is dependent on the RFE modes of operation (:INPut:MODE or :SWEep:ENTRy:MODE). In addition to Table 2 (page 17), the following Table 29 describes the output data width and packing method for the different data type in order comply with VRT's 32-bit word output format:

Table 29: Output Data Width and Packing Method for Different Data Formats

Original Data Format	Binary Format Per Data Component	Signed Extension	Per 32-bit Word Packing Method
$\{I_{14}Q_{14}\}$	Signed 2-complement	$\{I_{16}Q_{16}\}$	$\{I_{16}Q_{16}\}$
$\{I_{14}\}$	Signed 2-complement	$\{I_{16}\}$	$\{\mathbf{I1}_{16}\mathbf{I2}_{16}\}$
$\{I_{24}\}$	Signed 2-complement	$\{I_{32}\}$	$\{I_{32}\}$

The different Stream Identifier values will be used to indicate these different formats.

The order of the fields in an IF Data packet is organized as shown in Table 30. Packets is transmitted in big-endian byte order.

Table 30: IF Data Class Field Values

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pkt Type				C	T	-		TSI		TSF		Pkt Count				Pkt Size															
Stream Identifier (1 word)																															
Timestamp - Integer Seconds (1 word)																															
Timestamp - Integer Picoseconds (2 words)																															
Data Payload (Variable Size)																															
Trailer (1 word)																															

1. **Pkt Type** is **0001**, indicating that an IF stream identifier is present.
2. **C** is set to **0**, indicating that there is no class identifier present.
3. **T** is set to **1**, indicating there is a trailer word in the packet.
4. **TSI** field is set to **01**, indicating that integer (seconds) part of the timestamps is in UTC.
5. **TSF** field is set to **10**, indicating that the fractional part of the timestamp measures in real time picosecond resolution.
6. **Pkt Count** starts at 0000, and increments once for each IF Data packet that is received, until reaching 1111, when it then wraps back to 0000 on the next count.
7. **Pkt Size** is the number of 32-bit words that are present in the packet, including all headers, data payload and trailer if available.
8. **Stream Identifier** has the values as shown in the following Table 31.

Table 31: Stream Identifier Values for Different Data Output Formats

Data Output Format	Stream Identifier
$\{I_{14}Q_{14}\}$	0x90000003
$\{I_{14}\}$	0x90000005
$\{I_{24}\}$	0x90000006

9. **Timestamp - Integer Seconds** is in UTC format and will represent the number of seconds occurred since Midnight, January 1st, 1970, GMT.
10. **Timestamp - Integer Picoseconds** counts the number of picoseconds past since the last increment of the Timestamp seconds field. See Table 32.
11. **Data Payload** contains the IF data from the R57x0, arranged in the format indicated in Table 33 to Table 35.
12. **Trailer** is included and arranged in the format described in Table 36.

Picosecond Timestamp Words Format

The two 32-bit words timestamp allotted for picoseconds are arranged as below.

Table 32: 64-bit or Two Words Picosecond Timestamp Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Timestamp – picoseconds, Word 1 (63..32)																															
Timestamp – picoseconds, Word 2 (31..0)																															

Data Payload Format

The data payload of an IF Data packet contains a contiguous sequence of the Data Samples from an IF Data Sample stream. The number of words in the data payload is

variable from packet to packet, and can be determined at the receiving end of the link from the Packet Size by subtracting the number of words dedicated to the header, trailer, and other additional fields. The presence or absence of these fields can be determined entirely from information in the header.

1. The maximum number of data payload 32-bit words is 2^{16}-16 and must be a multiple of 32. Limitation due to embedded data transferring engine.
2. The data payload consists of an integer number of contiguous 32-bit words.
3. IF Data Packets convey either the time domain in-phase (I or real) and/or quadrature (Q or imaginary) components forming the Complex Cartesian samples.

{$I_{14}Q_{14}$} Data Payload Format

4. Each I or Q data is a signed two's-complement 14-bit data with signed extended into 16-bit. Thus, each component is an integer ranging from -8192 to +8191 (or $\pm 2^{13}$).
5. The I-component is in the upper 16-bit of each data word followed by the Q-component in the lower 16-bit, as seen in Table 33.

Table 33: {$I_{14}Q_{14}$} Data Payload Arrangement with Upper 2-bit Signed Extended to {$I_{16}Q_{16}$}

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0bxx		Item 1 (Sample 1 I_{14})														0bxx		Item 2 (Sample 1 Q_{14})													
0bxx		Item 3 (Sample 2 I_{14})														0bxx		Item 4 (Sample 2 Q_{14})													
0bxx		Item 5 (Sample 3 I_{14})														0bxx		Item 6 (Sample 3 Q_{14})													
⋮		⋮														⋮		⋮													

Example conversion, given the big-endian bytes 0x0018FFFE received:

- Split into two data items (i = 0x0018, q = 0xFFFE)
- Parse signed two's complement (i = 24, q = -2)
- Compute fractional value if needed: $i/2^{13}$ and $q/2^{13}$

{I_{14}} Data Payload Format

6. Each I data is a signed two's-complement 14-bit sample with signed extended into 16-bit. Thus, each component is an integer ranging from -8192 to +8191 (or $\pm 2^{13}$).
7. The first I sample is in the upper 16-bit of each data word follows by the second I sample in the lower 16-bit, as seen in Table 34.

Table 34: {I_{14}} Data Payload Arrangement with Upper 2-bit Signed Extended to {I_{16}}

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0bxx		Item 1 (Sample 1 I_{14})														0bxx		Item 2 (Sample 2 I_{14})													
0bxx		Item 3 (Sample 3 I_{14})														0bxx		Item 4 (Sample 4 I_{14})													
0bxx		Item 5 (Sample 5 I_{14})														0bxx		Item 6 (Sample 6 I_{14})													
⋮		⋮														⋮		⋮													

Same conversion example as {$I_{14}Q_{14}$}.

{I_{24}} Data Payload Format

8. Each data word is one I-component as seen in Table 35.

9. Each I data is a signed two's-complement 24-bit sample with signed extended into 32-bit. Thus, each component is an integer ranging from -8388608 to +8388607 (or $\pm 2^{23}$).

Table 35: $\{I_{24}\}$ Data Payload Arrangement with Upper 8-bit Signed Extended to $\{I_{32}\}$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0bxxxxxxxx								Item 1 (Sample 1 I_{24})																							
0bxxxxxxxx								Item 2 (Sample 2 I_{24})																							
0bxxxxxxxx								Item 3 (Sample 3 I_{24})																							
⋮								⋮																							

Examples conversion:

- Given the big-endian bytes 0x0018FFFE, then I_{24} = 0x18FFFE.
- Given the big-endian bytes 0xFF800034, then I_{24} = 0x800034 (or -8388556).
- Compute fractional value if needed: $i/2^{23}$.

Trailer Word Format

Table 36: Trailer Word Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Enables												State and Event Indicators												E	Associated Context Packet Count						
		DV	RL			SI	OR	SL					DV	RL			SI	OR	SL												

State and Event Indicators and the associated Enable bits are positions as indicated in Table 37.

1. For each Indicator bit in the State and Event Indicators field, there is a corresponding Enable bit at the same position in the Enables field.
2. When an Enable bit is set to 1, the corresponding indicator functions as shown in Table 37. Otherwise, the corresponding indicator shall not be considered valid.
3. Unused bits in the Enables field and the Indicators field is set to 0.
4. The E field is set to 0 to specify the Associated Context Packet Count field as undefined.
5. The Associated Context Packet Count field is unused and is set to 0.

Table 37: Trailer Indicator and Enable Bits

Enable Bit Position	Indicator Bit Position	Indicator Name
30	18	Valid Data Indicator
29	17	Reference Lock Indicator
26	14	Spectral Inversion Indicator
25	13	Over-range Indicator
24	12	Sample Loss Indicator

6. The **Valid Data Indicator**, when set to 1, indicates that the data in the packet is valid. When set to zero, it indicates that some condition exists that may invalidate the data.

7. The **Reference Lock Indicator,** when set to 1, indicates all PLLs in the system are locked and stable, and when set to 0, indicates one or all of the PLLs is not locked or unstable. It is very crucial to check this indicator bit.
8. The **Spectral Inversion Indicator**, when set to 1, indicates that the signal conveyed in the data payload has an inverted spectrum with respect to the spectrum of the signal at the system Reference Point (such as the antenna). When processing the data payload, for plotting purpose for instance, follow the suggested solution in Table 38 to properly display the spectrum.

Note: The Spectral Inversion Indicator does not apply to R57x0 models as the I/Q output connectors are not available. Information is listed here for general knowledge only.

9. The **Over-range Indicator** is set to 1 if any data value in the packet has reached full scale at the input of the digitizer.
10. The **Sample Loss Indicator**, when set to 1, indicates that data overflow has occurred **before** the current captured VRT packet, **not within** that current packet. In other words, there are data dropped between the previous packet and the current packet which has the sample loss indicator. The data between those two packets are not continuous and contiguous.

Table 38 lists the conditions in which an indicator would signal an abnormal state and the suggested resolutions.

Table 38: Conditions Causing Abnormal Indicator State and Suggested Resolution

Indicator Name	Abnormal State	Conditions	Suggested Resolution
Valid Data	0	1) One or more PLLs failed to lock. 2) In HDR mode, the NB ADC's filter has not settled.	1) Try to reset the frequency or restart the RTSA. If the condition persists, contact ThinkRF's Support. 2) Discard the data packet and re-acquire the data. This condition is unlikely as the settle time is within hundreds of nanoseconds.
Reference Lock	0	One or more PLLs failed to lock.	Same as 1) above
Spectral Inversion	1	Spectral inversion occurs when the frequency of the local oscillator exceeds that of the RF input signal being processed. At some signal frequency ranges input to the R5500, the IF output spectrum is inverted. Figure 10 illustrates an example.	When this indicator is 1, either swap each {I,Q} data point when both {I,Q} components are available or invert the output data bins of the computed spectral power when have {I} only data.
Over-range	1	The over-range threshold is the absolute full-scale of I or Q data. For WB ADC, the over-range threshold is at Vpeak = 1.0 V; and for NB ADC, Vpeak = 1.6 V.	- Enable the :ATTentuator if it is not yet on. - If :ATTentuator is already on, reduce the input level or the gain settings.

Indicator Name	Abnormal State	Conditions	Suggested Resolution
Sample Loss	1	This condition occurs only in Stream mode when the internal buffer is full.	Use a decimation value such that the transfer rate matches that of the capture rate.

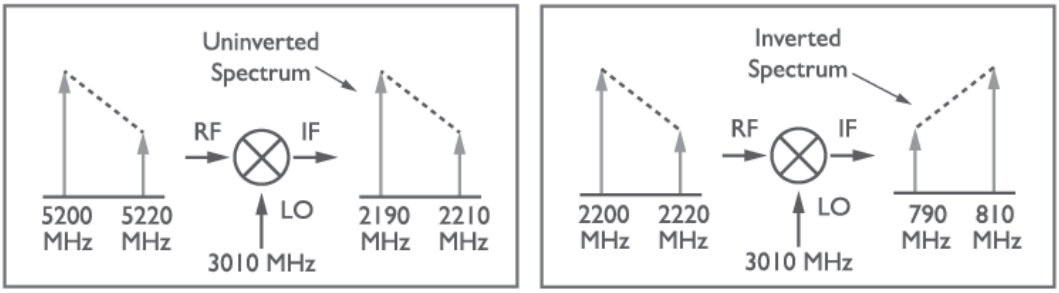


Figure 10: An Example Illustrating Uninverted and Inverted Spectrums

SCPI Command Set

This section is a SCPI reference guide for controlling the ThinkRF R57x0 Real Time Spectrum Analyzer. The R57x0 supports the Standard Commands for Programmable Instruments (SCPI) standard version 1999.0 as described in the following sections. SCPI lends itself to a command line interface and scripting, is supported by the major instrument vendors and provides a high level of familiarity for instrument users.

Note: ThinkRF's version of SCPI does not provide commands for network connection. The R57x0 receives SCPI commands and sends query responds over port 37001. See Appendix A: Connecting to RTSA for more details.

SCPI Language Overview

In the early 1990s, a group of instrument manufacturers developed Standard Commands for Programmable Instrumentation (SCPI) for controlling programmable instruments via a communication link, such as RS232, USB, LAN, etc. SCPI specifies the command structure and syntax using ASCII characters to provide some basic standardization and consistency to the control commands. SCPI commands, hence, lend themselves to communications with equipment via command line interface, scripting and/or programming languages such as C/C++, MATLAB®, Python, etc.

The SCPI language is based on a hierarchical or tree structure as illustrated in Figure 11 an example command set. The top level of the tree is the root node, which is followed by one or more lower-level nodes.

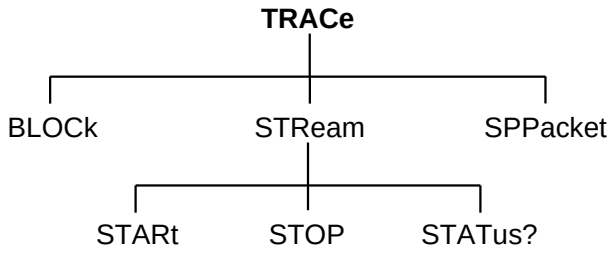


Figure 11: SCPI Language Hierarchical or Tree Structure Example

SCPI defines a measurement function block that is directly applicable to the ThinkRF RTSA. The measurement function converts a physical signal into an internal data form that is available for formatting into bus data. It may perform the additional tasks of signal conditioning and post-conversion calculation. The measurement function box is subdivided into three distinct parts: INPut, SENSe, and CALCulate as seen in Figure 12.

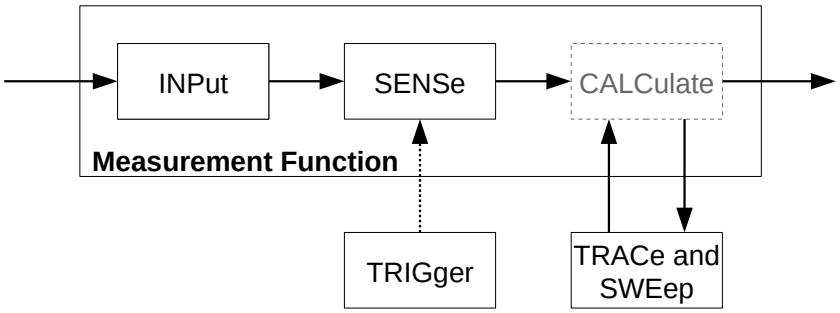


Figure 12: SCPI Measurement Function Block

Refer to the Appendix C: SCPI Command Syntax section for the general SCPI command syntax format and usage details.

IEEE Mandated SCPI Commands

These commands control and query the communication event/error and status registers as defined in the Appendix D: SCPI Status and Event Registers section. They are mandated by the IEEE.

Caution: The mandated IEEE SCPI commands are not affected by *RST command.

*CLS

The Clear Status (CLS) command clears the Status Byte register (STB), the standard Event Status register (ESR), the standard Questionable status register, the standard Operation Status register, and the error/event queue.

Syntax	*CLS
Parameter/Response	None

*ESE/*ESE?

*ESE command enables bits in the ESE register. The decimal integer value entered is the binary equivalent of the desired 8-bit mask. Bits enabled in ESE and set in ESR register will result in the Standard Event Status Summary bit (bit 5) in the STB register being set. This then allows the reading of ESR by using the *ESR? query command to determine the cause.

*ESE? query returns the decimal sum of the enabled bits in the ESE register. The decimal sum is the binary equivalent of the 8-bit mask.

See Figure 13 for the ESE/ESR register bits mapping.

Syntax	*ESE <integer> *ESE?
Parameter/Response	0 - 255
I/O Data Type	<integer>

*ESR?

Query the standard Event Status Register (ESR), which returns the decimal sum of the bits set in the ESR. The decimal sum is the binary equivalent of the 8-bit mask. Any specific bit in the ESR will only appear set if and only if its event has occurred and the corresponding bit in the ESE is also enabled.

See Figure 13 for the ESR register bits mapping.

Caution: This is a destructive read. Once queried, the register is cleared.

Syntax	*ESR?
Parameter	None
Query Response	<integer>
Description	Refer to the Appendix D: SCPI Status and Event Registers section for the ESR register bit definition

*IDN?

Returns the R57x0's identification information string.

Note: The model string returned will not include the option code. To find out which options a model has, use :SYSTem:OPTions? command.

Syntax	*IDN?
Parameter	None
Query Response	"<Manufacturer>,<Model>,<Serial number>,<Firmware version>"
Output Data Type	<string> with comma separated information

*OPC/*OPC?

*OPC (Operation Complete) command sets to confirm bit 0 in the ESR to 1 when all commands received before *OPC or *OPC? have been completed.

*OPC? returns the ASCII character 1 in the Standard Event register indicating completion of all pending operations. The query also stops any new commands from being processed until the current processing is complete.

Syntax	*OPC *OPC?
Parameter	None
Query Response	1

*RST

Resets the R57x0 to its default settings. This includes stopping any running capture mode and trigger mode, and also performs :SYSTem:FLUSh.

*RST does not affect the registers or queues associated with the IEEE mandated commands. Each non-IEEE mandated command description in this reference shows the *RST value when affected.

Caution: If using *RST while in DD mode, after a *RST command, the hardware of the DD path with take ~2 seconds to settle. Hence, do not do any data analysis during this settling time.

Syntax	*RST
Parameter/Response	None

*SRE/*SRE?

The *SRE (Service Request Enable) command enables bits in the SRE register. The decimal integer value entered is the binary equivalent of the desired 8-bit mask to be enabled. Bits enabled in this register allow accessing the equivalent bits status in the STB register.

*SRE? query returns the decimal sum of the enabled bits in the SRE register. The decimal sum is the binary equivalent of the 8-bit mask.

See Figure 13 for the SRE/STB register bits mapping.

Syntax	*SRE <integer> *SRE?
Parameter/Response	0 - 255
I/O Data Type	<integer>

*STB?

*STB? (Status Byte) query returns the decimal sum of the bits set in the STB register without erasing its content. The decimal sum is the binary equivalent of the 8-bit mask. Any specific bit in the STB will only appear set if and only if its event has occurred and the corresponding bit in the SRE is also enabled.

See Figure 13 for the ESE/ESR register bits mapping and the Status Byte Register (SBR) section of the Appendix D for the bits definition.

Syntax	*STB?
Parameter	None
Query Response	<integer>

*TST?

*TST? (self-test) query initiates the device's internal self-test and returns one of the following results:

- 0 - all tests passed.
- 1 - one or more tests failed.

Syntax	*TST?
Parameter	None
Query Response	0 | 1
Output Data Type	<boolean>

*WAI

*WAI (Wait-to-Continue) command suspends the execution of any further commands or queries until all operations for pending commands are completed.

Syntax	*WAI
Parameter/Response	None

SYSTem Commands

These commands control and query the communication event and status registers as defined in the Appendix D: SCPI Status and Event Registers. They are the minimal :SYSTem sets required in all SCPI instruments.

:SYSTem:ABORt

This command will cause the R57x0 to stop the data capturing, whether in the manual trace block capture, triggering or sweeping mode. The R57x0 will be put into the manual mode; in other words, process such as streaming, trigger and sweep will be stopped. The capturing process does not wait until the end of a packet to stop, it will stop immediately upon receiving the command.

Syntax	:SYSTem:ABORt
Parameter/Response	None
*RST State	N/A
Example	`:SYST:ABORT`

:SYSTem:CAPTure:MODE?

This command returns what the current RTSA data capture mode is (i.e. sweeping, streaming or block mode).

When stream or sweep mode is stopped, block mode will resume.

Syntax	:SYSTem:CAPTure:MODE?
Parameter	None
Query Response	BLOCK | STREAMING | SWEEPING
Output Data Type	<character>
*RST State	BLOCK
Example	`:SYST:CAPTURE:MODE?`

:SYSTem:COMMunicate:HISLip:SESSion?

When connected over HiSLIP, this command returns the HiSLIP Session ID, which is used to establish the associated data connection. See Connection Using HiSLIP section of Appendix A for more information.

Syntax	:SYSTem:COMMunicate:HISLip:SESSion?
Parameter	None
Query Response	16-bit integer
Output Data Type	<integer>
*RST State	N/A
Examples	`:SYST:COMM:HISL:SESS?`

Note: Connection types other than HiSLIP do not have a Session ID. In these cases, the response is undefined.

:SYSTem:COMMunicate:LAN:APPLy

This command will apply the changes to the LAN settings and the embedded system will automatically reconfigure the Ethernet to put in effect the new LAN setting. Once the LAN settings are applied, they are not affected by power-on, :STATus:PRESET, or *RST.

Caution: When changing from DHCP to STATIC mode, this command should to be sent only when all the required LAN settings have been set using the appropriate :SYSTem:COMMunicate:LAN commands.

Syntax	:SYSTem:COMMunicate:LAN:APPLy
Parameter/Response	None
*RST State	N/A
Examples	`:SYST:COMM:LAN:APPLY`

:SYSTem:COMMunicate:LAN:CONFigure

The set command will store the new LAN configuration type to be applied to the RTSA. This command does not take effect until :SYSTem:COMMunicate:LAN:APPLy is sent (please refer to the Caution note of the :APPLy command). Once the option is applied, it is not affected by power-on, :STATus:PRESET, or *RST.

The query will return the option set or that of the actual current configuration if one is not set. The CURRENT query will return what is currently and actually used by the RTSA's LAN interface.

Note: *RST command cannot be used to set the box to its manufacturing default state of DHCP. To set the box back to DHCP from a working STATIC mode, use this command or the web-browser as mentioned in the RTSA User's Guide.

Syntax	SYSTem:COMMunicate:LAN:CONFigure DHCP | STATIC SYSTem:COMMunicate:LAN:CONFigure? [CURRENT]
Parameter	Input: DHCP | STATIC Query: [CURRENT]
Query Response	DHCP | STATIC
I/O Data Type	<character>
*RST State	N/A
Examples	`:SYST:COMM:LAN:CONF DHCP` `:SYST:COMM:LAN:CONF? CURRENT`

:SYSTem:COMMunicate:LAN:DNS

The set command will store the new LAN DNS server address(es) to be applied to the RTSA. This command does not take effect until :SYSTem:COMMunicate:LAN:APPLy is sent (please refer to the Caution note of the :APPLy command). Once the setting is applied, it is not affected by power-on, :STATus:PRESET, or *RST.

The query will return the LAN DNS address(es) set or that of the actual current configuration if one is not issued. The CURRENT query will return what is currently and actually used by the RTSA's LAN interface.

Syntax	SYSTem:COMMunicate:LAN:DNS <main DNS>[,alternative DNS] SYSTem:COMMunicate:LAN:DNS? [CURRENT]
Parameter	Input: D.D.D.D[,D.D.D.D] where D = 0 – 255 Query: [CURRENT]
Query Response	D.D.D.D[,D.D.D.D]
I/O Data Type	Comma separated IPs <string>
*RST State	N/A

Examples	`SYSTEM:COMMUNICATE:LAN:DNS 208.67.110.0`
	`SYST:COMM:LAN:DNS 208.67.110.0,208.67.100.10`
	`SYSTEM:COMMUNICATE:LAN:DNS?`
	`SYST:COMM:LAN:DNS? CURRENT`

:SYSTem:COMMunicate:LAN:GATEway

The set command will store the new LAN gateway to be applied to the RTSA. This command does not take effect until :SYSTem:COMMunicate:LAN:APPLy is sent (please refer to the Caution note of the :APPLy command). Once the setting is applied, it is not affected by power-on, :STATus:PRESET, or *RST.

The query will return the gateway address set or that of the actual current configuration if one is not issued. The CURRENT query will return what is currently and actually used by the RTSA's LAN interface.

Syntax	SYSTem:COMMunicate:LAN:GATEway <IPv4 address> SYSTem:COMMunicate:LAN:GATEway? [CURRENT]
Parameter	Input: D.D.D.D where D = 0 – 255 Query: [CURRENT]
Query Response	D.D.D.D
I/O Data Type	<string>

*RST State	N/A

Examples	`SYST:COMM:LAN:GATEWAY 102.101.0.13`
	`SYSTEM:COMMUNICATE:LAN:GATEWAY?`
	`SYST:COMM:LAN:GATE? CURRENT`

:SYSTem:COMMunicate:LAN:IP

The set command will store the new LAN IP to be applied to the RTSA. This command does not take effect until :SYSTem:COMMunicate:LAN:APPLy is sent (please refer to the Caution note of the :APPLy command). Once the setting is applied, it is not affected by power-on, :STATus:PRESET, or *RST.

The query will return the IP address set or that of the actual current configuration if one is not issued. The CURRENT query will return what is currently and actually used by the RTSA's LAN interface.

Syntax	SYSTem:COMMunicate:LAN:IP <IPv4 address> SYSTem:COMMunicate:LAN:IP? [CURRENT]
Parameter	Input: D.D.D.D where D = 0 – 255 Query: [CURRENT]
Query Response	D.D.D.D
I/O Data Type	<string>
*RST State	N/A

Examples	`SYST:COMM:LAN:IP 101.125.1.16`
	`SYSTEM:COMM:LAN:IP?`
	`SYST:COMM:LAN:IP? CURRENT`

:SYSTem:COMMunicate:LAN:MTU

The set command will store the new LAN MTU (maximum transfer unit, in bytes) to be applied to the RTSA. This command does not take effect until :SYSTem:COMMunicate:LAN:APPLy is sent (please refer to the Caution note of the :APPLy command). Once the setting is applied, it is not affected by power-on, :STATus:PRESET, or *RST. The factory default value is 1500 bytes.

The query will return the MTU set or that of the actual current configuration if one is not issued. The CURRENT query will return what is currently and actually used by the RTSA's LAN interface.

Note: Use this command to change the maximum network packet or frame size in a transaction; however, changing the MTU value could affect the transfer rate or causing lost of network packets if the packets have an MTU value larger than that permitted by your network. Check with your network administrator before changing this value.

Syntax	SYSTem:COMMunicate:LAN:MTU <valid MTU value in bytes> SYSTem:COMMunicate:LAN:IP? [CURRENT]
Parameter	Input: 576 – 1500 bytes Query: [CURRENT]
Input Data Type	<integer> | <character>

Query Response	<integer>
*RST State	N/A

Examples	`SYST:COMM:LAN:MTU 1400`
	`SYSTEM:COMM:LAN:MTU?`
	`SYST:COMM:LAN:MTU? CURRENT`

:SYSTem:COMMunicate:LAN:NETMask

The set command will store the new LAN netmask address to be applied to the RTSA. This command does not take effect until :SYSTem:COMMunicate:LAN:APPLy is sent (please refer to the Caution note of the :APPLy command). Once the setting is applied, it is not affected by power-on, :STATus:PRESET, or *RST.

The query will return the mask address set or that of the actual current configuration if one is not issued. The CURRENT query will return what is currently and actually used by the RTSA's LAN interface.

Syntax	SYSTem:COMMunicate:LAN:NETMask <address> SYSTem:COMMunicate:LAN:NETMask? [CURRENT]
Parameter	Input: D.D.D.D where D = 0 – 255 Query: [CURRENT]
Query Response	D.D.D.D
I/O Data Type	<string>
*RST State	N/A

Examples	`SYST:COMM:LAN:NETMASK 255.255.255.0`
	`SYSTEM:COMMUNICATE:LAN:NETM?`
	`SYST:COMM:LAN:NETM? CURRENT`

:SYSTem:COMMunicate:NTP

This command sets or queries the NTP configuration server address(es) of up to three IP addresses. When set, this command will accept server **IP** address(es) in the form of D.D.D.D only, comma separated. However, the query response will return the IP address(es) set through this command or domain/IP address(es) set through R57x0's web administrative page.

Syntax	SYSTem:COMMunicate:NTP <address> SYSTem:COMMunicate:NTP?
Parameter	<IP Address D.D.D.D>, where D = 0 – 255
Query Response	<web/IP addr>[,<web/IP addr>[,<web/IP addr>]]
I/O Data Type	Comma separated IPs <string>
*RST State	N/A
Examples	`SYSTEM:COMMUNICATE:NTP 132.163.96.4` `SYST:COMM:NTP?`

Note: This feature could also be set through R57x0's administrative web page, with up to 3 domain/IP addresses can be entered.

:SYSTem:ERRor[:NEXT]?

This query returns the oldest uncleared error code and message from the SCPI error/event queue. When there are no error messages, the query returns 0,"No error". *RST does not affect the error queue.

Note: It is recommended to do this query command after each non-query command is sent to ensure that the non-query command is executed without error. Since each error message is queued into a buffer, if multiple commands have been sent follow by this query command, it would be uncleared which command has resulted in which error.

Syntax	:SYSTem:ERRor[:NEXT]?
Parameter	None
Query Response	<error code>,<description>
Output Data Type	<NR1>,<string>
Description	Refer to the Appendix E: SCPI Error Codes Used section
*RST State	N/A
Example	:SYST:ERR?

:SYSTem:ERRor:ALL?

This query returns all the uncleared error codes and messages from the SCPI error/event queue. If there are no error messages, the query returns 0,"No error". *RST does not affect the error queue.

Syntax	:SYSTem:ERRor:ALL?
Parameter	None
Query Response	<error code>,<description>{,<error code>,<description>}
Output Data Type	<NR1>,<string>{,<NR1>,<string>}
Description	Refer to the Appendix E: SCPI Error Codes Used section
*RST State	N/A
Example	:SYST:ERR:ALL?

:SYSTem:ERRor:CODE[:NEXT]?

This query is similar to :SYSTem:ERRor[:NEXT]? but returns only the error code from the SCPI error/event queue. When there are no errors, the query returns 0. *RST does not affect the error queue.

Note: Similarly, it is recommended to do this query command (or :SYSTem:ERRor[:NEXT]?) after each non-query command is sent to ensure that the non-query command is executed without error.

Syntax	:SYSTem:ERRor:CODE[:NEXT]?
Parameter	None
Query Response	Error code

Output Data Type	<NR1>
Description	Refer to the Appendix E: SCPI Error Codes Used section
*RST State	N/A
Example	:SYST:ERR:CODE?

:SYSTem:ERRor:CODE:ALL?

This query is similar to :SYSTem:ERRor:ALL? but returns error codes only from the SCPI error/event queue. If there are no errors, the query returns 0. *RST does not affect the error queue.

Syntax	:SYSTem:ERRor:CODE:ALL?
Parameter	None
Query Response	<error code>{,<error code>}
Output Data Type	Comma separated <NR1>
Description	Refer to the Appendix E: SCPI Error Codes Used section
*RST State	N/A
Example	:SYST:ERR:CODE:ALL?

:SYSTem:ERRor:COUNt?

This query returns the number of errors/events in the error/event queue.

Syntax	:SYSTem:ERRor:COUNt?
Parameter	None
Query Response	Count value
Output Data Type	<integer>
*RST State	N/A
Example	:SYST:ERR:COUNT?

:SYSTem:FLUSh

This command clears the R57x0's internal data storage buffer of any data that is waiting to be sent.

IMPORTANT Note: It is highly recommended that the flush command (after issuing :SYSTem:ABORt) should be used before start of any data capture or when switching between different capture modes to clear up the remnants of data in the RTSA.

Caution: Issuing :SYSTem:FLUSh any time during streaming or sweeping mode will cause the stream or sweep capture to stop (abort) and switch automatically to block capture mode.

Note: Flush command only handles the RTSA's internal buffer storage. The host application should ensure that the socket buffer is also cleared up of any potential data in

the socket buffer. This can be done by calling the receive socket (non-blocking) until no data is returned. With Streaming or Sweeping, the start ID in a VRT extension packet marks the beginning of packets belonging to the new stream or sweep. This helps to distinct old packets from new packets.

Syntax	:SYSTem:FLUSh
Parameter/Response	None
*RST State	N/A
Examples	`:SYST:FLUSH`

:SYSTem:LOCK:HAVE?

This query returns the current lock state of the specified task.

Syntax	:SYSTem:LOCK:HAVE? ACQuisition
Parameter	ACQuisition
Input Data Type	<character>
Query Response	1 | 0 1 – Have the lock 0 – Does not have the lock
Output Data Type	<boolean>
*RST State	N/A
Examples	`:SYST:LOCK:HAVE? ACQ`

:SYSTem:LOCK:REQuest?

This query attempts to attain the lock on the R57x0 for a specific task, such as data acquisition. The query returns 1 when lock is successful or 0 if it fails.

Attaining a lock is equivalent to having the sole ownership for that task. This prevents multiple connected applications from doing the same task that would result in an erroneous operation or feedback from the R57x0. The R57x0's system lock ownership works in the following manner:

- The first application to connect to R57x0 will automatically have the lock. The next application will need to perform this query request to attain the lock.
- When there is only one application connected (or the last one remaining), that application will automatically have the lock.
- The last application that requested successfully has the lock until another application requests it.

Any application that doesn't have the specific lock will not be able to perform that task.

Note: When a TCP/IP socket connection is not exited properly, that socket might continue to exist in the R57x0 server for a few minutes. This could affect a situation when only one application is used to connect to the R57x0 as re-connection by that application might not get the lock. This application would then need to request the lock.

Note: HiSLIP defines a more comprehensive instrument locking mechanism that locks out SCPI operations for sessions that do not have the instrument lock. This effectively prevents other HiSLIP clients from grabbing the data stream.

For HiSLIP connections, the HiSLIP lock request should immediately be followed by :SYST:LOCK:REQ? ACQ to ensure that the application regains the data stream.

A mixture of HiSLIP and non-HiSLIP connections will cause issues as the non-HiSLIP connections are unaware of HiSLIP instrument locks.

Syntax	:SYSTem:LOCK:REQuest? ACQuisition
Parameter	ACQuisition
Input Data Type	<character>
Query Response	1 | 0 1 – Successfully locked 0 – Failed to lock
Output Data Type	<boolean>
*RST State	N/A
Example	`:SYST:LOCK:REQ? ACQ`

:SYSTem:OPTions?

This command queries the hardware option(s) or features that a particular RTSA model supported. The response string contains comma separated 3-digit values to represent the options. See Table 39 for the translated list of available codes (when available).

Syntax	:SYSTem:OPTions?
Parameter	None
Query Response	<xxx>{,<xxx>}
Output Data Type	Comma separated 3-digit value (ex: 000, 001, 002)
*RST State	None
Example	`:SYST:OPT?`

Table 39: RTSA Option Codes and the Corresponding Description

Option Code	Description	Related SCPI Command
000	No Special Option/Feature	

:SYSTem:SYNC:MASTer

This command sets the RTSA unit to be the master or slave for a synchronization trigger system with multiple units, in which **only one unit** can be the master.

The master sends a sync-word or pulse (set through :TRIGger:TYPE or :SWEep:ENTRy:TRIGger:TYPE) via its GPIO to that of the slaves to indicate the beginning of a capture. The master RTSA itself will has an internal loop-back of the synchronization signal it sent out.

Syntax	:SYSTem:SYNC:MASTer <boolean>

	:SYSTem:SYNC:MASTer?
Parameter	ON | OFF | 1 | 0
Query Response	0 | 1
I/O Data Type	<boolean>
*RST State	0
Examples	`:SYSTem:SYNC:MASTER ON` `:SYSTem:SYNC:MAST?`

:SYSTem:SYNC:WAIT

This command sets the delay time in nanoseconds that an RTSA system must wait after receiving the satisfying trigger signal and before performing data capture. The delay time should be a multiple of 8 nsec as the RTSA system runs with a 125MHz clock.

Syntax	:SYSTem:SYNC:WAIT <integer> :SYSTem:SYNC:WAIT?
Parameter/Response	0 – 4294967295 (or 2^{32}-1)
I/O Data Type	<integer>
*RST State	0
Examples	`:SYSTem:SYNC:WAIT 120` `:SYSTem:SYNC:WAIT?`

:SYSTem:VERSion?

This query returns the SCPI version number that the instrument software complies with.

Syntax	:SYSTem:VERSion?
Parameter	None
Query Response	YYYY.V
Output Data Type	<NR2>
Example	:SYST:VERS

:SYSTem:DATE

This command sets or queries the current date of the R57x0. When the date is set, the change is applied to the real time clock (RTC) of the R57x0 system, and the :SYSTem:TIME:SYNC field is changed to DISable automatically. The date returned is representative of the current time mode that is UTC.

This command is not affected by a power-on, factory reset, or *RST command.

Syntax	:SYSTem:DATE <integer>,<integer>,<integer> :SYSTem:DATE?
Parameters/Response	<year>,<month>,<date>
I/O Data Type	Comma separated <integers>
Allowable Values	Year: YYYY - requires a four digit integer, 1900 - 9999 Month: 1 - 12 Date: 1 - 31

*RST State	N/A
Examples	`:SYST:DATE 2017,12,2` `:SYSTEM:DATE?`

:SYSTem:TIME

This command sets or queries the current time of the R57x0. When the time is set, the change is applied to the RTC of the R57x0 system, and the :SYSTem:TIME:SYNC field is changed to DISable automatically. *The time returned is representative of the current time mode that is UTC.*

This command is not affected by a power-on, factory reset, or *RST command.

Syntax	:SYSTem:TIME <integer>,<integer>,<integer>[,<integer>] :SYSTem:TIME?
Parameters	<hour>,<minute>,<second>[,<millisecond>]
Query Response	<hour>,<minute>,<second>[,<millisecond>] in UTC
I/O Data Type	Comma separated <integers>
Allowable Values	Hour: 0 - 23 Minute: 0 - 59 Second: 0 – 59 Millisecond: 0 - 999
*RST State	N/A
Examples	`:SYST:TIME 10,30,15` `:SYSTEM:TIME?`

:SYSTem:TIME:ADJust

This command adjusts the system time relative to its current time.

Further information will be provided in a future revision of this document.

Syntax	:SYSTem:TIME:ADJust <integer> [unit] :SYSTem:TIME:ADJust?
Parameters	<second or sub-second> [unit]
Allowable Values	0 – 4294967295 (or 2^{32} - 1)
Query Response	<integer>
Default I/O unit	ns
*RST State	0
Examples	`:SYST:TIME:ADJUST 10 ns` `:SYSTEM:TIME:ADJUST?`

:SYSTem:TIME:SYNC

This command selects the time synchronization source for R57x0 and the query returns the source selected. Choosing NTP (Network Time Protocol) as the synchronization source will impact the system real time clock (RTC), causing it to update either at a continuous interval or one time only. When :SYSTem:DATE and/or :SYSTem:TIME commands are used to change the time, the source will automatically be changed to DISable.

*RST does not affect this command. At factory install, the synchronization is defaulted to disabled.

Syntax	:SYSTem:TIME:SYNC DISable | NTP,{ONCE | CONTinuous} :SYSTem:TIME:SYNC?
Parameter	DISable | NTP,{ONCE | CONTinuous}
Query Response	DISabled | NTP,{ONCE | CONTinuous}
I/O Data Type	<character> | Comma separated <characters>
*RST State	DISable

Examples	`:SYST:TIME:SYNC NTP,ONCE`
	`:SYST:TIME:SYNC DISABLE`
	`:SYSTEM:TIME:SYNC?`

STATus Commands

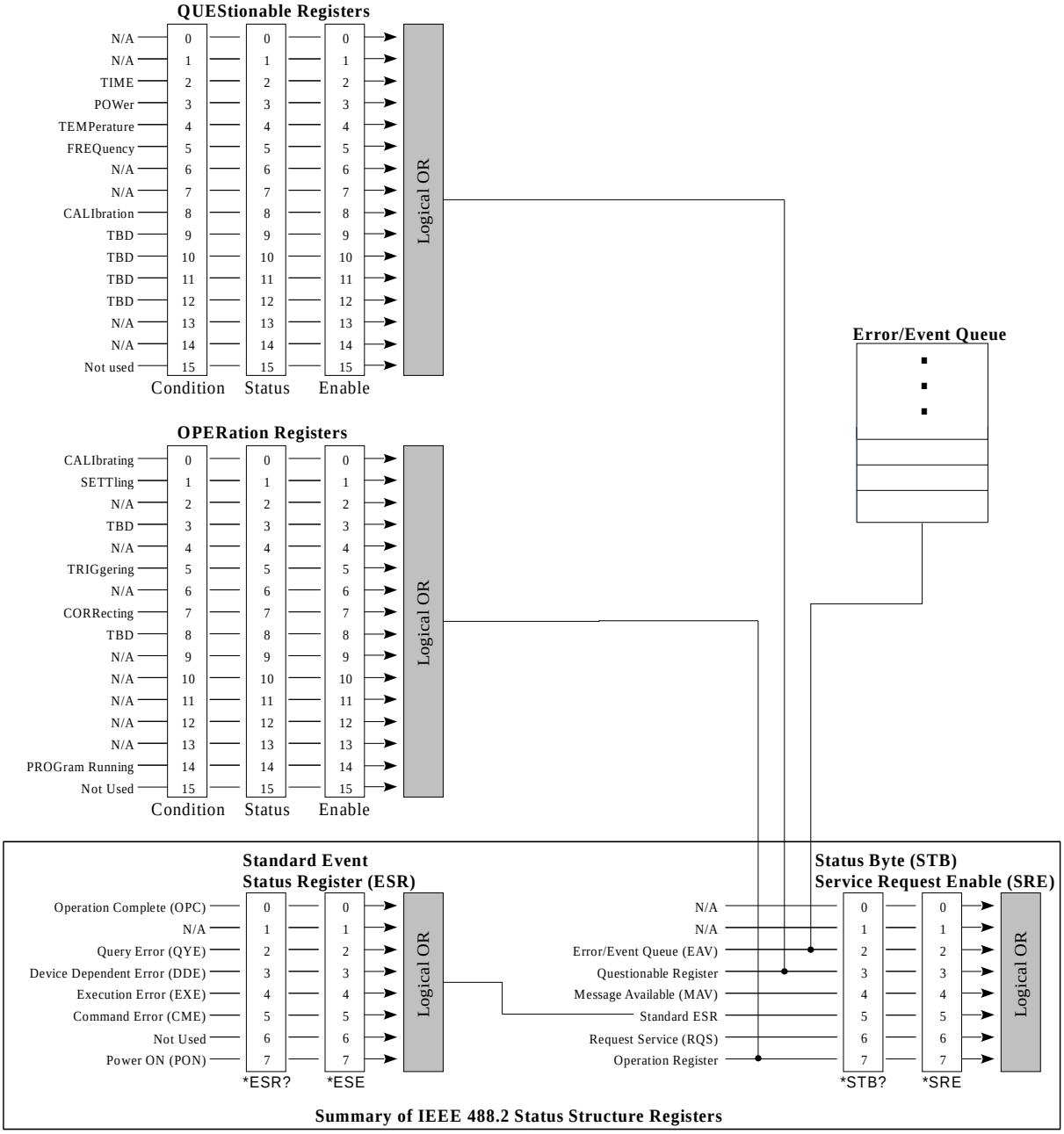


Figure 13: Status Reporting Structure with Status & Enable Registers

The STATus commands control the SCPI-defined status-reporting structures as illustrated in Figure 13.

SCPI defines the QUEStionable, OPERation, Instrument SUMmary and INSTrument registers in addition to those in IEEE 488.2. These registers conform to the IEEE 488.2 specification and each may be comprised of a condition register, an event register, an enable register, and negative and positive transition filters.

SCPI also defines an IEEE 488.2 queue for status. The queue provides a human readable record of instrument events. The application programmer may individually enable events into the queue. :STATus:PRESET enables errors and disables all other events. If the summary of the queue is reported, it shall be reported in bit 2 of the status byte register. A subset of error/event numbers is defined by SCPI.

:STATus:OPERation[:EVENt]?

This command queries the standard Operation Status Register (OSR, page 98) for any operation event. The query returns the decimal sum of the bits set in the OSR. The decimal sum is the binary equivalent of the 16-bit mask. The last bit is unused.

A bit set in the OSR when an enabled transition on the condition is detected (refer to Appendix D: SCPI Status and Event Registers). Transitions are enabled using the Operation Transition Registers (see :STATus:OPERation:NTRansition and :STATus:OPERation:PTRansition).

Caution: This query clears all bits in the register to 0. However, new events may occur concurrently and are guaranteed not to be missed.

See Figure 13 for the Operation Status register bits mapping.

Syntax	:STATus:OPERation[:EVENt]?
Parameter	None
Query Response	0 – 32767 (2^{15}-1)
Output Data Type	<integer>
*RST State	None
Example	`:STAT:OPER?`

:STATus:OPERation:CONDition?

This command queries the standard Operation Condition Register (OCR) for any operation condition. The query returns the decimal sum of the bits set in the OCR. The decimal sum is the binary equivalent of the 16-bit mask. The last bit is unused. The content of the OCR remains unchanged after it is read.

The data in this register is continuously updated to reflect the most current conditions.

See Figure 13 for the Operation Condition Register bits mapping.

Syntax	:STATus:OPERation:CONDition?
Parameter	None
Query Response	0 – 32767 (2^{15}-1)
Output Data Type	<integer>
*RST State	None
Example	`:STAT:OPER:COND?`

:STATus:OPERation:ENABle

This command enables or queries bits in the Operation Enable Register (OER). The decimal integer value entered is the binary equivalent of the desired 16-bit mask to be enabled. Bits enabled in this register allow the equivalent bit status in the OSR (see page 98) to propagate up through the Service Request (SRQ) chain.

Bits enabled in the OER and set in the OSR will result in the Standard Operation Status Summary bit (bit 7) in the STB register being set. See Figure 13.

Syntax	:STATus:OPERation:ENABle <integer> :STATus:OPERation:ENABle?
Parameter/Response	0 – 32767 (2^{15}-1)
I/O Data Type	<integer>
*RST State	0
Examples	`:STAT:OPER:ENAB 256` `:STAT:OPER:ENAB?`

:STATus:OPERation:NTRansition

This command enables or queries bits in the Operation Negative Transition Register (ONTR). The decimal integer value entered is the binary equivalent of the desired 16-bit mask to be enabled. Bits enabled in ONTR will allow a negative transition in the corresponding condition to set the bit in the OSR (see page 98).

Syntax	:STATus:OPERation:NTRansition <integer> :STATus:OPERation:NTRansition?
Parameter/Response	0 – 32767 (2^{15}-1)
I/O Data Type	<integer>
*RST State	0
Examples	`:STAT:OPER:NTR 256` `:STAT:OPER:NTR?`

:STATus:OPERation:PTRansition

This command enables or queries bits in the Operation Positive Transition Register (OPTR). The decimal integer value entered is the binary equivalent of the desired 16-bit mask to be enabled. Bits enabled in OPTR will allow a positive transition in the corresponding condition to set the bit in the OSR (see page 98).

Syntax	:STATus:OPERation:PTRansition <integer> :STATus:OPERation:PTRansition?
Parameter/Response	0 – 32767 (2^{15}-1)
I/O Data Type	<integer>
*RST State	0
Examples	`:STAT:OPER:PTR 256` `:STAT:OPER:PTR?`

:STATus:PRESET

This command presets the R5500 (similar to *RST), and OSE and QSE to zero.

Syntax	:STATus:PRESET
Parameter/Response	None

:STATus:QUEStionable[:EVENt]?

This command queries the standard Questionable Status Register (QSR, page 98) for any event. The query returns the decimal sum of the bits set in the QSR. The decimal sum is the binary equivalent of the 16-bit mask. The last bit is unused.

A bit set in the QSR when an enabled transition on the condition is detected (refer to Appendix D: SCPI Status and Event Registers). Transitions are enabled using the Questionable Transition Registers (see :STATus:QUEStionable:NTRansition and :STATus:QUEStionable:PTRansition).

Caution: This query clears all bits in the register to 0. However, new events may occur concurrently and are guaranteed not to be missed.

See 13 for the Questionable Status register bits mapping.

Syntax	:STATus:QUEStionable[:EVENt]?
Parameter	None
Query Response	0 – 32767 (2^{15}-1)
Output Data Type	<integer>
*RST State	None
Example	`:STAT:QUES?`

:STATus:QUEStionable:CONDition?

This command queries the standard Questionable Condition Register (QCR) for any questionable condition. The query returns the decimal sum of the bits set in the QCR. The decimal sum is the binary equivalent of the 16-bit mask. The last bit is unused. The content of the QCR remains unchanged after it is read. The data in this register is continuously updated to reflect the most current conditions.

See Figure 13 for the QCR bits mapping.

Syntax	:STATus:QUEStionable:CONDition?
Parameter	None
Query Response	0 – 32767 (2^{15}-1)
Output Data Type	<integer>
*RST State	None
Example	`:STAT:QUES:COND?`

:STATus:QUEStionable:ENABle

This command enables bits in the Questionable Enable Register (QER). The decimal integer value entered is the binary equivalent of the desired 16-bit mask to be enabled. Bits enabled in this register allow the equivalent bit status in the QSR to propagate up through the Service Request (SRQ) chain.

Bits enabled in QER and set in QSR will result in the Standard Questionable Status Summary bit (bit 3) in the STB register being set. See Figure 13.

Syntax	:STATus:QUEStionable:ENABle <integer> :STATus:QUEStionable:ENABle?
Parameter/Response	0 – 32767 (2^{15}-1)
I/O Data Type	<integer>
*RST State	0
Examples	`:STAT:QUES:ENAB 256` `:STAT:QUES:ENAB?`

:STATus:QUEStionable:NTRansition

This command enables or queries bits in the Questionable Negative Transition Register (QNTR). The decimal integer value entered is the binary equivalent of the desired 16-bit mask to be enabled. Bits enabled in QNTR will allow a negative transition in the corresponding condition to set the bit in the QSR (see page 98).

Syntax	:STATus:QUEStionable:NTRansition <integer> :STATus:QUEStionable:NTRansition?
Parameter/Response	0 – 32767 (2^{15}-1)
I/O Data Type	<integer>
*RST State	0
Examples	`:STAT:QUES:NTR 256` `:STAT:QUES:NTR?`

:STATus:QUEStionable:PTRansition

This command enables or queries bits in the Operation Positive Transition Register (QPTR). The decimal integer value entered is the binary equivalent of the desired 16-bit mask to be enabled. Bits enabled in QPTR will allow a positive transition in the corresponding condition to set the bit in the QSR (see page 98).

Syntax	:STATus:QUEStionable:PTRansition <integer> :STATus:QUEStionable:PTRansition?
Parameter/Response	0 – 32767 (2^{15}-1)
I/O Data Type	<integer>
*RST State	0
Examples	`:STAT:QUES:PTR 256` `:STAT:QUES:PTR?`

:STATus:TEMPerature?

This command queries the RTSA's internal temperature provided by one or more temperature sensors. The response field varies depending on how many sensors are available in an RTSA model. The R57x0 model, for instance, returns comma separated values for the sensors at the RF, Mixer and Digital sections.

Syntax	:STATus:TEMPerature?
Parameter	None
Query Response	For R57x0: <RF>,<Mixer>,<Digital> temperatures
Output Data Type	Comma separated <NRf>
Unit	degrees Celsius
*RST State	None
Example	:STATUS:TEMP?

INPut Commands

:INPut:ATTenuator

This command sets or queries the fix attenuation of the R57x0's RFE.

Note: This command **applies to R57x0-408 and their variants only**. It is not applicable for -418 and -427 and their variants, see :INPut:ATTenuator:VARiable command instead.

Syntax	:INPut:ATTenuator <integer> :INPut:ATTenuator?
Parameter	0 | 10 | 20 | 30 [dB]
Input Data Type	<integer>, optional <character> unit
Query Response	0 | 10 | 20 | 30
Output Data Type	<integer>
*RST State	30
Examples	`:INP:ATT 20` `:INPUT:ATT?`

:INPut:ATTenuator:VARiable

This command sets or queries the variable attenuation of the R57x0's RFE.

Notes:
- This command **applies to R57x0-418, -427, and their variants only**. For R57x0-408 and their variants, see :INPut:ATTenuator command instead.
- *Recommend setting the attenuation to 0 dB for best performance*

Syntax	:INPut:ATTenuator:VARiable <integer [dB]> :INPut:ATTenuator?
Parameter	0 | 10 | 20 | 30 [dB]
Input Data Type	<integer>, optional <character> unit
Query Response	0 | 10 | 20 | 30
Output Data Type	<integer>
*RST State	30
Examples	`:INP:ATT:VAR 0` `:INPUT:ATT:VAR?`

:INPut:GAIN

This command sets or queries the input gain stage for an RTSA. The number of gain stages is dependent on the models as listed below. Any out of range index will result in an Execution Error response (see Error and Event Queue section). Contact ThinkRF's Support for further details and the gain ranges of each stage.

Syntax	:INPut:GAIN <Index> <boolean> :INPut:GAIN? <Index>
Parameter	<Index> <ON | OFF | 1 | 0>
Input Data Type	<integer> <boolean>
Allowable Values	**Index:** Varying depending on the product model *- R57x0-408 & its variant: No controllable gain stage* *- R57x0-418, 427 & their variants: 1, 2. See Table 40 for the gain settings' performance*
Query Response	1 | 0
Output Data Type	<boolean>
*RST State	1 for all available stages

Examples	`:INPUT:GAIN 2 ON`
	`:INP:GAIN? 1`
	`:INP:GAIN 1 0`

Note: The reference level context information (see page 32) is **only valid** when all the gain stages are enabled for R57x0-418, 427 and their variants.

Table 40: Performance of The Gain Settings of R57x0-418, 427 and Their Variants

Stage 1	Stage 2	Gain Level	Performance
0 (OFF)	0 (OFF)	Low	High Saturation Level
0 (OFF)	1 (ON)	--	Not Optimal*
1 (ON)	0 (OFF)	Medium	Good DANL and Saturation Level
1 (ON)	1 (ON)	High	Best DANL (Reference Level context data provided)

* This setting is not recommended.

:INPut:GAIN:HDR

This command sets or queries variable NB IF gain of the HDR signal path, and thus, the command is to be used with HDR input mode only (i.e. set :INPut:MODE HDR).

Note: The HDR gain is a gain block at the receiver back-end. It will not, therefore, impact the signal-to-noise ratio (SNR). The HDR gain is used in particular for optimizing the third order intercept point (IP3) by determining the most linear spot of the cumulative amplification I/O characteristics in the receiver chain.

Since the IP3 measurement relies on the relative signal-to-3rd order product ratio and not on the absolute signal value nor the SNR, the R57x0 product is not calibrated for the different HDR gain levels. *The only HDR gain value that is calibrated and optimized is the default 25 dB gain.*

Syntax	:INPut:GAIN:HDR <NR1 [unit]> :INPut:GAIN:HDR? [MAX | MIN]
Parameter	-10 – 34 [unit]
Input Data Type	<NR1> [<character>]
Query Response	-10 – 34
Output Data Type	<NR1>
Default I/O unit	dB
*RST State	25

Examples	`:INPUT:GAIN:HDR -5`
	`:INP:GAIN:HDR 20 dB`
	`:INP:GAIN:HDR?`

:INPut:MODE

This command sets or queries the R57x0's RFE mode of operation.

Notes: The different RFE modes (in combination with any DSP applied) affect the VRT IF data packing method due to the different output data type width. For example, SH mode is I_{14} data format, but with decimation, it would change to $I_{14}Q_{14}$ data output. The VRT's Stream ID would identify the data format accordingly. See Table 2: Radio RFE Modes and DSP Data Output Formats and VRT's IF Data Packet Class for more details.

- It is also important to see Table 2 for the IBW of each mode and the related notes.

Syntax	:INPut:MODE ZIF | DD | HDR | SH | SHN :INPut:MODE?
Parameter/Response	ZIF | DD | HDR | SH | SHN
I/O Data Type	<character>
*RST State	*Product version dependent*[1] Also see the Caution note of the *RST command if DD mode is used
Examples	`:INP:MODE SH` `:INPUT:MODE?`

[1] The RFE mode availability is product dependent. Hence, the *RST state and the initial power-up default would be different depending on the product version.

SOURce Commands

:SOURce:REFerence:PLL

This command selects and queries the 10MHz PLL reference clock source, whether it be via the internal free-run clock source, the embedded GNSS module's clock, or through the external SMA connector.

To see which timing reference source used to discipline the 10 MHz GNSS reference oscillator, with or without a GNSS fix, refer to :GNSS:REFerence? query command.

Caution: When the external 10MHz reference is used, its reference level **must** be between **-10dBm** and **0dBm**. Exceeding the level of 0dBm will result in permanent damage to the internal clock circuit. Additionally, the 10MHz reference must be powered down prior to powering down the R57x0.

Note: :GNSS[:ENABle] must be set to ON before selecting the GNSS option for this command; otherwise, an error of "-221 Settings Conflict" will be returned.

Syntax	:SOURce:REFerence:PLL INT | EXT | GNSS :SOURce:REFerence:PLL?
Parameter/Response	INT | EXT | GNSS
I/O Data Type	<character>
*RST State	GNSS
Examples	`:SOURCE:REF:PLL INT` `:SOUR:REF:PLL?`

:SOURce:REFerence:PPS

This command selects and queries the PPS source for synchronizing the NTPD (NTP daemon) and the VRT timestamp system, whether it be via the external GPIO source, or through the GNSS's module.

Notes:

- :GNSS[:ENABle] must be set to ON before selecting the GNSS option for this command; otherwise, an error of "-221 Settings Conflict" will be returned.

- If the GNSS option is already selected, and GNSS module is disabled through :GNSS[:ENABle], the option would remain as GNSS until this command is called to be changed to EXT.

Syntax	:SOURce:REFerence:PPS EXT | GNSS :SOURce:REFerence:PPS?
Parameter/Response	EXT | GNSS
I/O Data Type	<character>
*RST State	GNSS
Examples	`:SOURCE:REF:PPS GNSS` `:SOUR:REF:PPS?`

SENSe Commands

[:SENSe]:DECimation

This command sets or queries the rate of decimation of samples in a trace capture. When the rate is set to 1 (or OFF), no decimation is performed on the trace capture. The decimation range varies depending on the RFE modes as described below.

In HDR mode, the decimation is done directly by the on-board NB ADC. The decimation value supported by this mode is 1, 2 and 4.

In the remaining RFE modes, R57x0 uses DDC to provide 10 levels of decimation of values 4, 8, 16, 32, 64, 128, 256, 512, 1024 (i.e. decimation rate = 2^{level} where level = 2 – 10). The decimation process consists of CIC and FIR filters, each type of filters with its own decimator. The decimator captures one sample at every <integer> number of samples. The filters are arranged in the following manner:

- For the decimation rate of 4, only a FIR filter with a fixed decimation by 4 is used, CIC filter is bypassed.
- For the decimation rates of 8 to 1024, a 4-stage CIC of rate 4 to 512 is applied first for each I and/or Q data. The resulting I and/or Q data pipes are then passed to a FIR filter with a fixed decimation of 2 to arrive at the rate set. For example, for a rate of 16, I and Q data will first pass-through the CIC filters with a decimation rate of 8. The CIC output will be further decimated by 2 by the FIR filter which has a fixed decimation rate of 2.

Note: When in SH/SHN mode and a decimation is on, the R57x0 will automatically be shifted by 35MHz to the zero IF before decimation is applied. This implies **the VRT data output will be {I,Q} for SH/SHN with decimation**.

Syntax	:SENSe:DECimation OFF | <integer> :SENSe:DECimation? [MAX | MIN]

Parameter	OFF | <decimation value>
Input Data Type	<integer> | <character>
Allowable Values	For ZIF, SH, SHN modes: OFF, 1, 4, 8, 16, 32, 64, 128, 256, 512 and 1024 1 Equivalents to decimation off.
Query Response	For ZIF, SH, SHN modes: 1, 4, 8, 16, 32, 64, 128, 256, 512 and 1024
Output Data Type	<integer>
*RST State	1
Examples	`:DEC 16` `:SENSE:DEC OFF`

[:SENSe]:FREQuency:CENTer

This command sets or queries the center frequency of the RTSA with one exception as mentioned in the following note.

Note: For the DD mode, this command does not apply as DD mode is not tune-able (however, [:SENSe]:FREQuency:SHIFt command is still applicable to those modes). To use the frequency range below 50 MHz, use `:INPut:MODE DD` command instead.

The frequency resolution varies depending on the RFE modes of operation. ZIF, SH and SHN signal paths utilize the WB ADC; thus, the frequency tuning resolution is 10Hz. For those receiver modes, the resolution can be down to the nearest 1Hz resolution (±0.23Hz) using [:SENSe]:FREQuency:SHIFt command. While for HDR receiver mode, the frequency resolution is 10Hz without further frequency shifting capability. Refer to RF Receiver Front-End (page 17) for more details.

For example, the system is in ZIF mode, to tune to a frequency of 2441.16MHz require the sending of two commands: [:SENSe]:FREQuency:CENTer 2441.1MHz and [:SENSe]:FREQuency:SHIFt 60kHz. The set values can be verified by querying. If a valid frequency with an inappropriate resolution is set, the frequency value will be rounded down to the nearest valid resolution, no error is set.

In addition, depending on the product models, the allowable range of programmable frequencies varies. Check with your product's data sheet. For example, R57x0-408 has a range of 0.1 to 8GHz, while R57x0-427 has 0.1 to 27GHz.

Syntax	[:SENSe]:FREQuency:CENTer <NRf [unit]> [:SENSe]:FREQuency:CENTer? [MAX | MIN]
Parameters	RF Center frequency [unit] ***Varies depending on the product model***
Input Data Type	<NRf> [<character>]
Query Response	<integer>
Default I/O Unit	Hz
*RST State	2400000000

Examples	`:FREQ:CENTER 2441.5 MHz`
	`SENSE:FREQ:CENT 200000000`
	`:FREQ:CENT 2.01 GHZ`
	`SENSe:FREQ:CENTer?`

[:SENSe]:FREQuency:IF?

This command queries the IF frequencies that are used for the current input mode and center frequency.

This command works in all R57x0 models but the number and significance of the IF frequencies will vary depending on the model and configured options (see :SYSTem:OPTions? command). The IF index can be specified either as a positive number (1 to N) where 1 indicates the first IF mixing stage after the input from the front end, or as a negative number (-1 to -N) where -1 indicates the last IF before the digitizer input. If the input index is beyond the number of available IFs, :SYSTem:ERRor[:NEXT]? will return a -222,“Data out of range” response.

Syntax	[:SENSe]:FREQuency:IF? <non-zero integer>
Parameters	IF index
Input Data Type	Non-zero <integer>
Allowable Values	*Varies depending on RFE input mode, frequency, model and options*
Query Response	IF frequency
Output Data Type	<integer>
Default Output Unit	Hz
*RST State	N/A
Examples	`SENSE:FREQ:IF? -1`

[:SENSe]:FREQuency:LOSCillator?

This command queries the frequency of the local oscillator (LO) 1, 2 or 3 in corresponding to current the RTSA's center frequency set.

Syntax	[:SENSe]:FREQuency:LOSCillator? <1 | 2 | 3>
Parameter	None
Query Response	<integer> 0 := LO Off
*RST State	1
Example	`:FREQ:LOSC? 2`

[:SENSe]:FREQuency:SHIFt

This command sets or queries the frequency shift value. A negative shift value corresponds to a left shifting.

This command is also used in additional to [:SENSe]:FREQuency:CENTer to fine tune the RTSA down to 1Hz resolution.

Note: Frequency shift mode is not available for some RFE modes of operation. Also, when enabled, it would affect the data output format of some RFE modes. See Table 2: Radio RFE Modes and DSP Data Output Formats (page 17).

Syntax	[:SENSe]:FREQuency:SHIFt <NRf [unit]> [:SENSe]:FREQuency:SHIFt? [MAX | MIN]
Parameters	Frequency [unit]
Input Data Type	<NRf> [<character>]
Allowable Values	-62.5 – 62.5 MHz
Query Response	Shifted frequency
Output Data Type	<integer>
Default I/O Unit	Hz
*RST State	0

Examples	`:FREQ:SHIF -10.5 MHz`
	`SENSE:FREQ:SHIFT 20000000.0`
	`SENSe:FREQ:SHIFT?`
	`FREQ:SHIFT? MAX`

[:SENSe]:LOCK:REFerence?

This command queries the lock status of the PLL reference clock in the digital card.

Syntax	[:SENSe]:LOCK:REFerence?
Parameter	None
Query Response	0 | 1 1 Reference PLL is locked 0 Reference PLL is not locked
Output Data Type	<boolean>
*RST State	N/A
Example	`LOCK:REF?`

[:SENSe]:LOCK:RF?

This command queries the lock status of the RF VCO (Voltage Control Oscillator) in the RFE.

Syntax	[:SENSe]:LOCK:RF?
Parameter	None
Query Response	0 | 1 1 RF VCO is locked 0 RF VCO is not locked
Output Data Type	<boolean>
*RST State	N/A
Example	`LOCK:RF?`

GNSS Commands

The following set of commands are used for configuring the GNSS module as well as query its related information. Other GNSS related configurations could be found in SOURce Commands section.

Note: If GNSS is not enable, all :GNSS commands, except for :GNSS[:ENABle], will not response to set or query commands. A “-200,"Execution error"” message will be returned through :SYSTem:ERRor[:NEXT]? query.

:GNSS[:ENABle]

This command turns on or off, or queries the state of the GNSS module. Turning the module on would result in GNSS data being sent back from the RTSA as VRT context packets. The GNSS VRT packets are sent one every second. See Formatted GPS Geolocation section for more information.

Note: When turned on from the off state, the GNSS module will subject to Time To First Fix (TTFF) or Time to Subsequent Fix (TTSF), which is a measure of the time required for a GNSS receiver after switching on to acquire sufficient satellite signals and navigation data, and calculate an accurate position solution.

The module is defaulted to ON at powered up.

Syntax	:GNSS[:ENABle] ON | OFF | 1 | 0 :GNSS[:ENABle]?
Parameter	ON | OFF | 1 | 0
Response	0 | 1 1 GNSS module is on 0 GNSS module is off
I/O Data Type	<boolean>
*RST State	1
Examples	`:GNSS ON` `:GNSS:ENABLE?`

:GNSS:ADELay

The ADELay command allows bi-directional shifting of the 1PPS output in relation to the UTC 1PPS reference in one nanosecond steps. This allows antenna cable delay compensation, as well as retarding or advancing the 1PPS pulse arbitrarily. The delay is in nanoseconds and limited to 16-bit integer range or -32768 to 32767.

Note: This command is not affected by the *RST. The default factory value is 50 ns. The RTSA will retain the user’s last enter value through power cycles.

Syntax	:GNSS:ADELay <16-bit Integer> :GNSS:ADELay?

Parameter/Response	-32768 – 32767 (in nanoseconds)
I/O Data Type	<NR1>
*RST State	N/A
Example	`:GNSS:ADEL 200` `:GNSS:ADELAY?`
Output Example	`200`

:GNSS:CONStellation

This command sets the satellite constellation(s) to be tracked by the RTSA's GNSS module or queries which satelite constellation(s) is set. Users could set up to two constellations. When two options are used, they could be sent in any order; however, when query, the device will return the two in this order: GPS, BEIDOU, GLONASS.

Note: This command is not affected by the *RST. The default factory options are "GPS,GLONASS". The RTSA will retain the user's last enter value through power cycles.

Syntax	:GNSS:CONStellation <option 1>[,option 2] :GNSS:CONStellation?
Parameter	GPS, BEIDOU, GLONASS
Query Response	<option 1>[,option 2]
I/O Data Type	Comma separated <characters> if more than one option is used
*RST State	NONE

Examples	`:GNSS:CONS GPS,BEIDOU`
	`:GNSS:CONSTELLATION GPS`
	`:GNSS:CONS?`

:GNSS:POSition?

This command returns the last known GNSS position in degrees latitude, degrees longitude and altitude in meters, such as:

45.421500,75.697200,70.000000

In the absence of GNSS fix, the command returns invalid data, corresponding to all "1" values in the GPS VRT context packet definition, or:

512.000000,512.000000,67108863.968750

Syntax	:GNSS:POSition?
Query Response	latitude (degrees),longitude (degrees),altitude (meters)
Output Data Type	Comma separated <NR2>
*RST State	N/A
Example	`:GNSS:POS?`
Output Example	`45.421500,75.697200,70.000000`

:GNSS:REFerence?

This command returns the timing reference source used to discipline the 10 MHz GNSS reference oscillator. A value of "GNSS" is returned if a GNSS fix is obtained and is being used as the disciplining source for the GNSS reference oscillator. A value of "INT" is returned in the absence of a GNSS fix, in which case the 10 MHz GNSS reference oscillator is in a holdover state until a GNSS fix is obtained.

The GNSS reference oscillator is a valid reference source regardless of GNSS fix status.

Note: If GNSS is selected as the PLL reference source, the REF LED indicator flashes green in the presence of GNSS fix or flashes yellow otherwise.

The absence of GNSS fix is also reflected in bit 9 of the IEEE488 Questionable Status Register (QSR), with a 1 indicating absence of a GNSS fix.

Syntax	:GNSS:REFerence?
Parameter	None
Query Response	GNSS | INT
Output Data Type	<character>
*RST State	GNSS
Examples	`:GNSS:REF?`

TRIGger Commands

Note: Trigger is not supported with HDR and DD input modes.

:TRIGger:TYPE

This command sets or queries the type of trigger event. Setting the :TRIGger:TYPE to NONE is equivalent to disabling the trigger execution, while setting to any other type will enable the trigger engine.

The LEVel trigger type is condition by the start and stop frequencies range and the amplitude level. See the :TRIGger:LEVel command.

The PULSe and WORD trigger types belong to the external synchronization trigger through a GPIO port (see External Triggering, page 22). *This external trigger type uses the TRIG IN pin of the GPIO port.*

The PPS trigger type is similar to the PULSE type with the pulse occurs at every second and hence, the capture. Therefore, it is important that the total data capture size should be less than one second. *This external trigger type uses the PPS pin of the GPIO port.*

Syntax	:TRIGger:TYPE LEVel | PERiodic | PPS | PULSe | WORD | NONE :TRIGger:TYPE?
Parameter/Response	LEVEL | PERIODIC | PPS | PULSE | WORD | NONE
I/O Data Type	<character>
*RST State	NONE
Examples	`:TRIG:TYPE LEVEL` `:TRIG:TYPE?`

:TRIGger:LEVel

This command sets or queries the frequency range and amplitude of a frequency domain level trigger. If the sampled signal amplitude exceeds the defined trigger level at any single sample within the defined frequency range then the trigger will occur and the associated IQ data will be stored.

The frequency range encompasses all FFT bins of which their center frequencies are within the range defined by START and STOP. The defined START and STOP frequencies may exceed, but only affect, the range defined by the IBW (with consider of the DDC decimation) centered around the [:SENSe]:FREQuency:CENTer value.
The threshold error is ±3 dBm or less when the trigger level is set

- at or below the maximum level mentioned in Table 41 and
- ~15 dBm or higher above the noise floor.

When the level is set higher and further away from the maximum level or within ~20 dBm of and closer to the noise floor, the threshold error increases. Recommend adjusting the attenuation level (using :INPut:ATTenuator:VARiable or :INPut:ATTenuator) accordingly if desiring a higher threshold level (as shown in Table 41), or lower noise floor level.

Note: The information provided in Table 41 applies to firmware version 1.1.0 or higher for R57x0. Recommend to update your device's firmware if the version is older.

Table 41: Maximum Threshold Level Where +/-3 dBm Error or Less Still Hold For A Given Attenuation Level

Attenuation (dB)	Maximum Threshold (dBm)
0	-15
10	-5
20	0
30	5

Refer to the Frequency Domain Triggering section for more information.

Syntax	:TRIGger:LEVel <NRf [unit]>,<NRf [unit]>,<NR1 [unit]> :TRIGger:LEVel?
Parameters/Response	<start>,<stop>,<level>

Input Data Type	Comma separated values with: Frequency: <NRf> [<character>] Level: <NRf> [<character>]
Allowable Values	Frequency: See [:SENSe]:FREQuency:CENTer Levels: Dependent on the attenuation setting. See Table 41.
Output Data Type	<integer>,<integer>,<NRf>
Default I/O Units	Hz,Hz,dBm
*RST State	N/A (Trigger is off)

Examples	`:TRIG:LEVEL 2000 MHZ,2100 MHZ,-70 DBM`
	`:TRIG:LEVEL 15000000,15050000,-50`
	`:TRIG:LEVEL?`

:TRIGger:PERiodic

Further information will be provided in a future revision of this document.

TRACe Commands

A "trace capture" consists of a set of continuous data samples, ranging from 128 samples to a maximum determined by the R57x0 version (see :TRACe:BLOCk:PACKets and :TRACe:SPPacket). Each data word is 32-bit wide, arranged differently depending on the :INPut:MODE and see VRT's Data Payload Format, page 38.

ThinkRF's R57x0 data packet returned through a network is complied with the industry standard VRT protocol. Therefore, every data packet returned is encapsulated with a VRT header and a VRT trailer. In addition, the VRT packet format sets a limit on the maximum number of samples per packet. Refer to the “Receiver Context Class” subsection of the VITA-49 Radio Transport Protocol section for further details on the VRT packet organization.

To do a single **block** capture of continuous data, the total number of samples captured is determined by the number of samples per packet (:TRACe:SPPacket) and the number of packets per block (:TRACe:BLOCk:PACKets). When the block data capture command (:TRACe:BLOCk:DATA?) is issued, the R57x0 will capture and store the total number of samples into a buffer. Hence, the samples within a single block capture is continuous from one packet to the other, but not necessary between successive block capture commands issued.

In **streaming** mode, the number of samples per packet (:TRACe:SPPacket) must be set to determine the size of each packet coming back. The samples from one packet to another will be continuous until the sample loss indicator (aka overflow indicator) is detected within the trailer of the data packet. When this indicator is high in the current VRT packet, it indicates that data overflow occurs **after** the current captured packet, not within the packet. In other words, the samples of the immediate packet following after the current packet that has the sample loss indicator bit high are not continuous from those of the current packet.

Note: The :DECimation command can be used to slow down the capture rate, thus, effectively lowers the rate of discontinuity between packets to provide contiguous data stream of data.

The R57x0 can stored up to 32 MSa of ZIF or 64 MSa of SH continuous data.

:TRACe:BLOCk:DATA?

This command will start the single block capture and the return of all trace packets set by :TRACe:BLOCk:PACKets command, with each packet of the size set through :TRACe:SPPacket command. The data within a single block capture trace is continuous from one packet to the other, but not necessary between successive block capture commands issued.

Syntax	:TRACe:BLOCk:DATA?
Parameter	None
Query Response	Control port 37001: empty string Data port 37000: Hexadecimal bytes
Output Data Type	<NRr>
*RST State	N/A
Examples	`:TRACE:BLOCK:DATA?`

Note: The status of the query will be return through the control port 37001 as usual, however the data will be returned through the data port 37000. Once the :TRACe:BLOCk:DATA? command is issued, a block of SPP * PACKets of data will be returned. In other words, :TRACe:BLOCk:DATA? needs to be sent only once to get SPP * PACKets block of data.

The returned data in each VRT packet is presented in continuous hexadecimal chunk, as shown here:

```
Response <NRr> ::= <VRT header bytes>{<data payload bytes>}
                   [<4 bytes VRT trailer>]
```

Further description on the VRT data output formats can be found in the VRT's IF Data Packet Class section, page 37.

:TRACe:BLOCk:PACKets

This command sets or queries the total number of packets set in the RTSA. The maximum is limited by the storage capacity of a R57x0 and the samples per packet (SPP) size set through :TRACe:SPPacket. Therefore, when :TRACe:BLOCk:PACKets? MAX query command is sent, the returned value will vary depending on the SPP value of an RTSA and the data output format. For example, the R57x0 has 128 MBytes storage capacity, if SPP is 32768 with $I_{14}Q_{14}$ output format, then the maximum packet size is 1023 (or 128 MB / (4 bytes-per-sample * (32768 + 6))). If I_{14} is the output format, then the maximum is 2047 (or 128 MB * / (2 bytes-per-sample * (32768 + 6))).

In single block capture mode, this command is used in conjunction with the :TRACe:SPPacket command to set the total number of samples to capture. In other words, the data from one packet to the next within a single block capture mode is continuous.

Syntax	:TRACe:BLOCk:PACKets <integer> :TRACe:BLOCk:PACKets? [MAX | MIN]

Parameter	Input: <Packet value> Query: [MAX | MIN]
Input Data Type	<integer> | <character>
Allowable Values	1 – (RTSA's maximum storage storage capacity ÷ (# bytes-per-sample * (SPP value + 6 Header and trailer words)))
Query Response	<integer>
*RST State	1
Examples	`:TRACE:BLOC:PACK 100` `:TRACE:BLOCK:PACK?`

:TRACe:SPPacket

This command sets or queries the number of Samples Per Packet (SPPacket). In block capture mode, it is used in conjunction with the :TRACe:BLOCk:PACKets command to set the total number of (continuous and contiguous) samples to capture.

The upper bound of the SPP is limited by the VRT's 16-bit Packet Size field less the VRT's headers and any optional fields (see IF Data Packet Class for more details). The 16-bit Packet Size defines the total number of 32-bit **words** in each packet, not **samples** which could have different bits per sample. However, the total samples must be a multiple of 32 due to the use of burst transfer method of the capture engine. The maximum SPP is, therefore, simplified to 65504 or (2^{16} – 32) for all data format.

The lower bound of the SPP is limited by the capture engine's minimal transfer requirement of 256 samples. Table 42 summarizes the SPP boundary sizes and the required multiple values for different data output format.

Table 42: Max, Min, and Required Multiples for SPP and Samples-per-word for Different Data Output Format

Format	Samples-per-*word*	Min SPP Size	Max SPP Size	Required Multiples
{$I_{14}Q_{14}$}	1	256	65504	32
{I_{14}}	2			
{I_{24}}	1			

Syntax	:TRACe:SPPacket <integer> :TRACe:SPPacket? [MAX | MIN]
Parameter	Input: Number of samples Query: [MAX | MIN]
Input Data Type	<integer> | <character>
Allowable Values	256 – 65504, *must be a multiple of 32*
Query Response	<integer>
*RST State	1024
Examples	`:TRACE:SPP 4096` `:TRAC:SPP?`

:TRACe:STReam:STARt

This command begins the execution of the real time stream capture. It will also initiate data capturing. Data packets will be streamed (or pushed) from the R57x0 whenever data is available.

Through the sending of a VRT Extension Context Packet carrying the ID value, the use of an ID in this command is to indicate the beginning of new data packets belonging to a new stream start. Even though the start ID value is optional, a VRT Extension Context Packet with the New Stream Start ID (page 37) value will **always** be sent out after this command is received and before data packets of the new stream become available. When no ID value is provided, the default ID value 0 is returned in the Context Packet.

Note: Once :TRACe:STReam:STARt is issued, the RTSA will not accept any setting changes. Changes can be sent after :TRACe:STReam:STOP command is issued.

Syntax	:TRACe:STReam:STARt [ID]
Parameter	Stream ID value
Input Data Type	<integer>
Query Response	None
*RST State	0 (Stream stopped)
Examples	`:TRAC:STREAM:START 1` `:TRACE:STR:START`

:TRACe:STReam:STOP

This command stops the stream capture. After receiving the command, the RTSA system will stop when the current capturing VRT packet is completed with the required samples (as opposed to :SYSTem:ABORt).

Note: After this command is issued, :SYSTem:FLUSh command should be issued as well as to clear up any data remained in the internal memory.

Syntax	:TRACe:STReam:STOP
Parameter/Response	None
*RST State	N/A (Stream stopped)
Examples	`:TRACE:STREAM:STOP` `:TRAC:STREAM:STOP`

SWEep Commands

Note: Currently, only one single sweep list is supported. Thus, some description on list in this section might not apply. For example, the string identifier is not needed yet, neither is list editing as there is only one list. The entries, however, can be configured as described.

A sweep control setup consists of defining one or more sweep lists and one or more entries for each list. The sweep execution is controlled by issuing the commands (such as start, stop or resume) listed under :LIST.

A sweep list can be thought of as being similar to a spreadsheet or table where the columns define the different specific capture engine configurations (such as :ATTenuator, :FREQuency and :DECimation), and the rows as sweep entries with each consisting of a sweep frequency or range and its associated capture engine configurations.

A :SWEep:LIST is created and identified using a unique string identifier set by the user. A list may be edited, deleted and/or executed using the :SWEep:LIST command set. Each list is executed indefinitely or a finite number of time as determined by the :ITERations command.

More information will be provided in the future revision of this document for multiple lists handling.

The :SWEep:ENTRy commands provide the ability to define the capture engine configurations for each sweep entry including the equivalent of :INPut, :SENSe and :TRIGger commands. There may be any number of entries in a sweep list for up to 500. Sweep entries are identified by an index number and may be inserted, edited and/or deleted like rows in a table or spreadsheet. A sweep entry is created by using either :NEW or :COPY and :SAVE command. The entry will not be part of a list until :SAVE is issued.

If trigger is defined for an entry, captured data is returned only if a trigger event occurred. Otherwise, when the :DWEll time is reached, the trigger is aborted and the next sweep entry will be executed.

During sweeping, the RTSA internal buffer might become full, at which point the sweep engine will pause. The engine will resume sweeping once ***the buffer*** is freed up.

The engine will stop when the iterations have been reached or either a :SYSTem:ABORt or :SWEep:LIST:STOP command has been issued.

Notes:

- Unlike with [:SENSe]:FREQuency:CENTer, the center frequency command of a sweep entry can take a frequency range and the step size as the parameters.

- Unlike :TRACe:BLOCk capture, sweep mode data packets, whether VRT context or digitized data, are “streamed” (similar to :TRACe:STReam). As soon as :SWEep:LIST:STARt command is issued, this will initiate also the data capturing and data packets will be “pushed” from the R57x0 when available.

- When sweep is stopped, the RTSA will retain the settings of the last performed sweep entry when :STOP command is received and executed. Any non-sweep commands can then be operated on the RTSA. When the :SWEep is resumed (:STARt), the settings as per the sweep entries are executed.

- When the RTSA is sweeping, any non-sweep commands sent will resulted in an error and are not executed. The sweep will not be affected and keep on running. However, sweep related settings can still be changed while sweep is running.

:SWEep:LIST:ITERations

This command sets or queries the number of times the sweep list is repeated.

Syntax	:SWEep:LIST:ITERations <integer> :SWEep:LIST:ITERations?
Parameter/Response	0 – 4294967295 (or 2^{32}-1) 0 := infinity
I/O Data Type	<integer>
*RST State	0
Examples	`:SWEEP:LIST:ITER 10` `:SWE:LIST:ITER?`

:SWEep:LIST:STARt

This command begins the execution of the current sweep list from the first entry.

This command will also initiate data capturing. Data packets will be streamed (or pushed) from the R57x0 whenever it is available.

Through the sending of a VRT Extension Context Packet carrying the ID value, the use of an ID in this command is to indicate the beginning of new data packets belonging to a new sweep start. Even though the start ID value is optional, a VRT Extension Context Packet with the New Sweep Start ID (page 37) value will **always** be sent out after this command is received and before data packets of the new sweep become available. When no ID value is provided, the default ID value 0 is returned in the Context Packet.

Syntax	:SWEep:LIST:STARt [ID]
Parameter	<List ID>
Input Data Type	<integer>
Query Response	None
*RST State	0 (Sweep stopped)
Examples	`:SWEEP:LIST:STAR` `:SWE:LIST:START`

:SWEep:LIST:STATus?

This query returns the current status of the sweep engine.

Syntax	:SWEep:LIST:STATus?
Parameter	None
Query Response	RUNNING | STOPPED
Output Data Type	<character>
*RST State	STOPPED
Examples	`:SWEEP:LIST:STATUS?` `:SWE:LIST:STAT?`

:SWEep:LIST:STOP

This command stops the sweeping and stores the entry index where it is stopped. The RTSA retains the settings of the last performed sweep entry when :STOP command is executed.

Syntax	:SWEep:LIST:STOP
Parameter/Response	None
*RST State	N/A (Sweep stopped)
Examples	`:SWEEP:LIST:STOP` `:SWE:LIST:STOP`

Note: This command should be issued to clear the R57x0's data buffer of any data that has not been sent from the R57x0 prior to setting up the next capturing process.

:SWEep:ENTRy:COPY

This commands will copy and populate all the capture engine configurations under :SWEep:ENTRy with values from the sweep entry of the specified index. No new entry is created until :SWEep:ENTRy:SAVE command is issued and any changes will not affect the existing entry.

Syntax	:SWEep:ENTRy:COPY <integer>
Parameter	Sweep entry integer index
Input Data Type	<integer>
Allowable Values	If :SWEep:ENTRy:COUNt? returns non-zero, 1 to :COUNt? value If :SWEep:ENTRy:COUNt? returns zero, an execution error is returned
Query Response	None
*RST State	N/A
Examples	`:SWEEP:ENTR:COPY` `:SWE:ENTRY:COPY`

:SWEep:ENTRy:COUNt?

This query command returns the number of entries available in a list.

Syntax	:SWEep:ENTRy:COUNt?
Parameter	None
Query Response	<integer>
*RST State	N/A
Examples	`:SWEEP:ENTR:COUNT?`

:SWEep:ENTRy:DELETE

This commands delete one or all the entries. When an entry is deleted, the following indexes if existed will be reduced by one accordingly, just as rows in a spreadsheet.

Syntax	:SWEep:ENTRy:DELETE <integer> | ALL
Parameter	<Entry index value> | ALL
Input Data Type	<integer> | <character>
Allowable Values	1 to :SWEep:ENTRy:COUNt? query value
*RST State	N/A
Examples	`:SWEEP:ENTR:DELETE 5` `:SWE:ENTR:DELETE ALL`

:SWEep:ENTRy:NEW

This commands will populate all the capture engine configurations under :SWEep:ENTRy with default values. No new entry is created until :SWEep:ENTRy:SAVE command is issued.

Syntax	:SWEep:ENTRy:NEW
Parameter/Response	None
*RST State	N/A
Examples	`:SWEEP:ENTRY:NEW`

:SWEep:ENTRy:READ?

This query command returns the current configuration settings of a sweep entry.

Syntax	:SWEep:ENTRy:READ? <integer>
Parameter	[Entry index value]
Input Data Type	<integer>
Allowable Values	1 to :SWEep:ENTRy:COUNt? query value
Query Response	<integer>,<{<integer> | <character>}> :== <RFE mode>,<freq start>,<freq stop>,<freq step>,<freq shift>, <decimation>,<attenuator>,<IF gain>,<HDR gain>,<SPPacket>, <packets>,<dwell:second>,<dwell:microsecond>, <trigger type: NONE | PULSe | WORD | <LEVel,freq start,freq stop, amplitude> | PPS>
Output Data Type	Comma separated <integer> and <character> values
*RST State	N/A
Examples	`:SWEEP:ENTR:READ? 5` `:SWE:ENTR:READ? 1`

:SWEep:ENTRy:SAVE

This command saves a new entry into the current editing list with all the current capture engine configurations under :SWEep:ENTRy. The saving is done by inserting either the new entry **before** the specified index value or to the end of the list when no index value is given.

When saved, a new entry is given an index value. Index value starts from 1. When an index value is specified along with the :SAVE command, the new entry will take the index of that value and all other following indexes will be incremented by one accordingly, just

as rows in a spreadsheet. Otherwise, the new index will be one up from the index of the last sweep entry in the list.

When there are no existing entries and an index value other than 1 is specified, an error will be returned. Similarly for non-existing index location except if the index value is equal to the value returned by :SWEep:ENTRy:COUNt? plus one.

Syntax	:SWEep:ENTRy:SAVE [integer]
Parameter	[Entry index value]
Input Data Type	[<integer>]
Allowable Values	:SWEep:ENTRy:COUNt? query value + 1
*RST State	N/A
Examples	`:SWEEP:ENTR:SAVE` `:SWE:ENTR:SAVE 5`

:SWEep:ENTRy:ATTenuator

Refers to the :INPut:ATTenuator section (page 64) for the definition of this command.

Examples	`:SWEEP:ENTRY:ATTENUATOR 20` `:SWEEP:ENTR:ATT?`

:SWEep:ENTRy:ATTenuator:VARiable

Refers to the :INPut:ATTenuator:VARiable section (page 64) for the definition of this command.

Examples	`:SWEEP:ENTRY:ATT:VAR 10` `:SWEEP:ENTR:ATT:VAR?`

:SWEep:ENTRy:DECimation

Refers to the [:SENSe]:DECimation section (page 68) for the definition of this command.

Examples	`:SWEEP:ENTR:DEC 16` `:SWEEP:ENTRY:DEC?`

:SWEep:ENTRy:FREQuency:CENTer

This command or query defines the center frequency or a range of center frequencies to sweep. When a range is provided, the sweep will step through the center frequencies with the value provided by :SWEep:ENTRy:FREQuency:STEP.

Syntax	:SWEep:ENTRy:FREQuency:CENTer <NRf [unit]>[,<NRf [unit]>] :SWEep:ENTRy:FREQuency:CENTer?
Parameter	<start freq [unit]>[,<stop freq [unit]>]
Input Data Type	<NRf> [<character>] | Comma separated <NRf> [<character>]
Allowable Values	**Varies depending on the product model**
Query Response	<start freq>,<stop freq>
Output data Type	<integer>,<integer>
Default I/O Units	Hz
*RST State	2400000000,2480000000

Examples	`:SWEEP:ENTRY:FREQ:CENT 0,10 GHZ`
	`:SWE:ENTRY:FREQ:CENT 2400 MHZ,6 GHZ`
	`:SWE:ENTR:FREQ:CENT 2400000000`
	`:SWEEP:ENTRY:FREQ:CENTER?`

:SWEep:ENTRy:FREQuency:STEP

This command or query defines the frequency step size for the sweep center frequency range specified by :SWEep:ENTRy:FREQuency:CENTer command. If a range is not given, the step size is ignored.

Syntax	:SWEep:ENTRy:FREQuency:STEP <NRf [unit]> :SWEep:ENTRy:FREQuency:STEP?
Parameter	<freq [unit]>
Input Data Type	<NRf> [<character>]
Allowable Values	0 – Maximum frequency of the R57x0 model used
Query Response	<integer>
Default I/O Units	Hz
*RST State	100000000

Examples	`:SWEEP:ENTRY:FREQ:STEP 10.5 MHZ`
	`:SWE:ENTRY:FREQ:STEP 4000 KHZ`
	`:SWEEP:ENTR:FREQ:STEP 10000000`
	`:SWEEP:ENTR:FREQ:STEP?`

:SWEep:ENTRy:FREQuency:SHIFt

Refers to the [:SENSe]:FREQuency:SHIFt section (page 70) for the definition of this command.

Examples	`:SWEEP:ENTR:FREQ:SHIFT 25 MHZ` `:SWEEP:ENTRY:FREQ:SHIF?`

:SWEep:ENTRy:GAIN:HDR

Refers to the :INPut:GAIN:HDR section (page 66) for the definition of this command.

Examples	`:SWEEP:ENTR:GAIN:HDR -10` `:SWEEP:ENTRY:GAIN:HDR?`

:SWEep:ENTRy:MODE

Refers to the :INPut:MODE section (page 66) for the definition of this command.

Examples
```
:SWEEP:ENTRY:MODE ZIF
:SWE:ENTR:MODE?
```

:SWEep:ENTRy:DWELl

This command or query defines the maximum amount of time to wait for the trigger of a sweep entry to occur, after which the trigger is aborted and the next sweep entry, if existed, will run. However, when the required amount of data has been captured before the dwell time has been reached, the sweep engine will move onto the next entry.

Note that, the default dwell time is 0 second, 0 microsecond. This is equivalent to an infinite dwell time. In this case, the sweep engine will move on as soon as the current data capture amount has been met (as explained in the previous paragraph).

When the trigger type is NONE, dwell time is ignored.

Syntax :SWEep:ENTRy:DWELl <integer>[,<integer>]
:SWEep:ENTRy:DWELl?

Parameter <second>[,<microsecond>]

Allowable Values 0 – 4294967295 (or $2^{32} – 1$)
0,0 := infinity

Query Response <second>,<microseconds>

I/O Data Type <integer> | Comma separated <integers>

*RST State 0,0 (infinite dwell time)

Examples
```
:SWEEP:ENTR:DWEL 5,30
:SWEEP:ENTR:DWELL 2
:SWEEP:ENTR:DWELL?
```

:SWEep:ENTRy:PPBlock

This command (where PPBlock is defined as Packets per block) has the same functionality as the :TRACe:BLOCk:PACKets command since at each sweep frequency step of an entry, a block of data can be captured.

Refers to the :TRACe:BLOCk:PACKets section (page 77) for the definition of this command.

Examples
```
:SWEEP:ENTR:PPB 10
:SWEEP:ENTRY:PPB?
```

:SWEep:ENTRy:SPPacket

Refers to the :TRACe:SPPacket section (page 78) for the definition of this command.

Examples
```
:SWEEP:ENTR:SPP 16384
:SWEEP:ENTRY:SPP?
```

:SWEep:ENTRy:TRIGger:LEVel

Refers to the :TRIGger:LEVel section (page 75) for the definition of this command.

Examples
```
:SWEEP:ENTR:TRIG:LEV 2400 MHZ,2900 MHZ,-60
:SWEEP:ENTRY:TRIGGER:LEVEL?
```

:SWEep:ENTRy:TRIGger:TYPE

Refers to the :TRIGger:TYPE section (page 74) for the definition of this command.

Note: For PPS (as well as PULSE) trigger type, it is important that the total data capture size should take into account this PPS (or pulse range) time frame, the sweep step tuning time, and the decimation rate. For safe measure, it should be less than 500 ms for PPS type. A large capture size could cause missing the pulse.

Examples
```
:SWEEP:ENTR:TRIG:TYPE LEVEL
:SWEEP:ENTRY:TRIG:TYPE?
```

Appendix A: Connecting to RTSA

ThinkRF RTSA supports two different methods of device connection, which will be explained in the following sections.

Caution pertaining to multi-users:

ThinkRF RTSA allows multiple applications to connect to a unit simultaneously or concurrently, however, each connection method to be mentioned in this Appendix will behave differently.

Simple 2-port TCP/IP Connection method does not support independent sessions. *Therefore, the actions of one user may over-write those of another.* If multiple applications are connecting to the unit, it is advised that only one of those is controlling the unit at any time.

Connection Using HiSLIP method, on the other hand, provides both exclusive and share locks. Refers to “Locking Mechanism” section of the IVI HiSLIP document for more information. However, the HiSLIP locking mechanism is only effective when all simultaneous connections are of HiSLIP type. If any other connection methods are used, HiSLIP locking mechanism does not apply and the same caution mentioned above applied.

Simple 2-port TCP/IP Connection

Note: We recommend using the HiSLIP connection method described in the next section as the standard HiSLIP method provides many features not available with this simple connection, particularly lock handling during multi-user access.

ThinkRF RTSAs are network ready devices conveying control commands and data using TCP/IP protocol. Each RTSA receives SCPI commands and sends query responds over port 37001, and sends VRT context and data packets over port 37000, as illustrated in Figure 14.

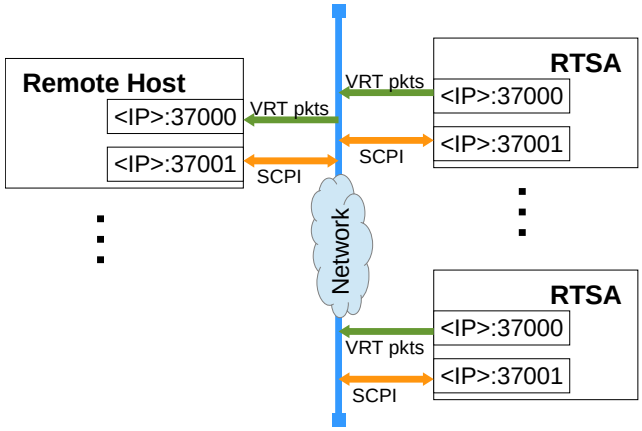


Figure 14: 2-port TCP/IP connection to RTSA

An RTSA, when powered up, will have a dynamic or preassigned static IP address, which when bind with a port will form a network socket. To successfully establish a connection to an RTSA, **both <IP>:37001** and **<IP>:37000** sockets must be created one right after the other **in the order as mentioned**.

In addition, refer to the "Connecting to the R57x0" of the *R57x0 User Guide* for more information on how to connect to R57x0 and to determine its IP address.

Connection Using HiSLIP

Note: The HiSLIP connection method is highly recommended over the other connection methods given its many beneficial features as listed below that the others lack.

High-Speed LAN Instrument Protocol (HiSLIP) is an industry standard developed by the Interchangeable Virtual Instruments (IVI) Foundation (www.ivifoundation.org). It is designed as a modern emulation of the IEEE-488 instrumentation bus standard and provides more sophisticated capabilities to instruments, including:

- instrument locking (shared and exclusive locks),
- service request from the instrument,
- multiple sessions, even from the same client.

Refer to the HiSLIP documentation, IVI-6.1 (www.ivifoundation.org/specifications/default.aspx), for further details.

ThinkRF RTSA acts as a HiSLIP server, listening on a TCP port 4880. Two TCP connections to the same port are established in a single HiSLIP connection using the initialization sequence described in the standard. The two connections are linked together with a common Session ID, with one connection serves as a synchronous channel and the other an asynchronous channel.

The synchronous channel primarily carries the command-response SCPI channel and all communication is controlled from the client (controller). The asynchronous channel is truly bidirectional, allowing either the client (controller) or the server (instrument) to signal each other at any time. This capability avoids unnecessary polling, allowing event driven applications to be developed.

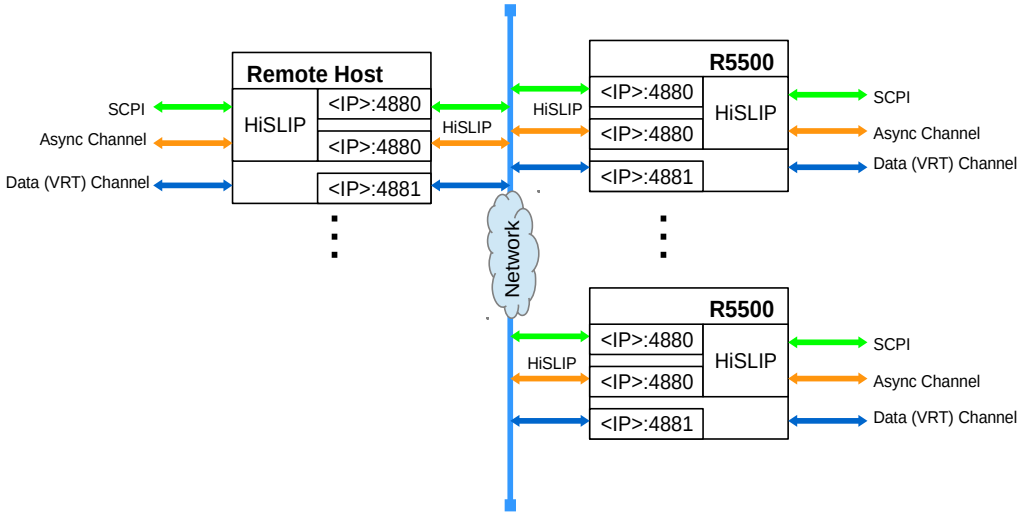


Figure 15: HiSLIP and TCP connections to RTSA

HiSLIP port 4880 only addresses the control channel of the R57x0, similar to the role that TCP socket 37001 plays in the 2-port TCP/IP connection mentioned in the previous section. A third TCP connection at port 4881 to RTSA is required to carry the VRT data stream. The Session ID used to associate the synchronous and asynchronous channels during connection establishment is also used outside of HiSLIP to further associate the data channel to the HiSLIP connection.

The following steps summarize how a VRT stream connection is associated with a HiSLIP connection:

1. The host establishes a HiSLIP connection with the instrument on port 4880.
2. The host establishes a TCP connection with the instrument on port 4881.
3. The host queries the HiSLIP connection's Session ID via SCPI using the :SYSTem:COMMunicate:HISLip:SESSion? command.
4. The host sends a message to the instrument with the HiSLIP Session ID. This method is similar to how HiSLIP associates the asynchronous channel to the synchronous channel.
5. The instrument searches the Session ID among its established connections. The instruments acknowledges the association if the Session ID is found.

The message sent over the data connection is derived from HiSLIP. For reference, the following table describes the HiSLIP message format. All values are in network order (big-endian).

Table 43: HiSLIP Message Header Format

Field	Octets	Field Offset
Prologue (ASCII “HS”)	2	0
Message Type	1	2
Control Code	1	3
Message Parameter	4	4
Payload Length	8	8
Data (optional)	Payload Length	16

This message pair used to establish the data channel is modeled after the HiSLIP *AsyncInitialize* and *AsyncInitializeResponse* messages used to establish the asynchronous channel on port 4880.

Table 44: ThinkRF Vendor Specific Message Type Value Definitions

Designation	Channel	Numerical Value
ThinkRFDataInitialize	Data	128
ThinkRFDataInitializeResponse	Data	129

Table 45: ThinkRF Data Channel Initialization Transaction

Step	Initiator	Message Content	Action
1	Client	Opens the data TCP connection	Client does an active TCP open on port 4881.
2	Client	<ThinkRFDataInitialize><0>	The client sends the SessionID of

Step	Initiator	Message Content	Action
		<SessionID><0>	the HiSLIP connection to associate this data channel with it.
3	Server	<ThinkRFDataInitializeResponse><0><SessionID><0> (successful)	Server acknowledges and echoes back the Session ID.
3	Server	<*ThinkRFDataInitializeResponse*><0><0x80000000><0> (unsuccessful)	Server did not find the Session ID. It sends back an error code (MSB = 1).
4		Once the transaction completes successfully, the TCP connection is ready to transmit the VRT stream from the server to the client.	

Once the data connection is associated, it is free to carry the VRT stream.

Appendix B: Protocols for Discovering RTSA

This section explains the two different protocols for discovering any RTSA devices available on the same local network as the host computer(s). These protocols can**not** be used to find any RTSAs on a different network.

Discovery Using mDNS/DNS-SD

The LAN eXtensions for Instrumentation (LXI) Consortium (www.lxistandard.org) has standardized the use of multicast DNS (mDNS) and DNS-based service discovery (DNS-SD) protocols as the discovery protocols for network-connected instruments. These protocols, commonly referred to as Zero Configuration Networking or Zeroconf, were originally developed by Apple Computer for discovering local network services such as printers but have been expanded to support any network service.

Conforming with the industry standard, R57x0 supports device discovery using the mDNS/DNS-SD protocols. Industry standard tools and applications supporting these protocols are now able to discover the R57x0 with the said firmware version. Refer to the following resources for more information about mDNS/DNS-SD protocols:

- http://www.lxistandard.org/Specifications/Specifications.aspx
- https://tools.ietf.org/html/rfc6762 for mDNS
- https://tools.ietf.org/html/rfc6763 for DNS-SD

Discovery Using Broadcast UDP

ThinkRF also provides a simple broadcast UDP protocol for discovering RTSA devices. The remote host computer would first send out a UDP message of broadcast type to port 18331. The message contains a query request code followed by query discovery version in big-endian order as follows:

<request code><discovery version>

where each field is:

Name	Data Type	Length	Required Value
<request code>	32-bit unsigned integer	1	0x93315555
<discovery version>	32-bit unsigned integer	1	2

The discovery version is used to determine how to parse the response message. Note that the <> bracket is for clarity of the explanation purpose only, not to be included in the message.

An RTSA with the discovery version 2 would respond with the following data:

<response code><discovery version><RTSA model><RTSA S/N><firmware version>

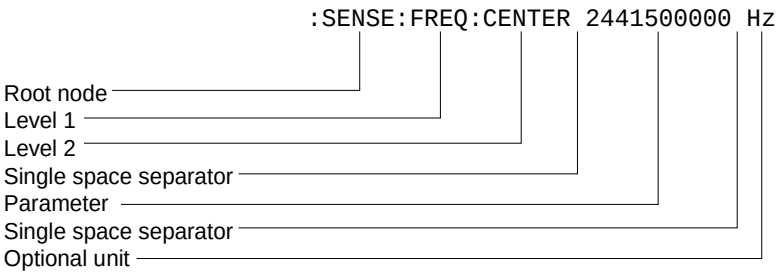
where each field is:

Name	Data Type	Length	Response Value
<response code>	32-bit unsigned integer	1	0x93316666
<discovery version>	32-bit unsigned integer	1	2
<RTSA model>	ASCII character, null-padded	16	RXXXX-XXX (ex: R5700-427)
<RTSA S/N>	ASCII character, null-padded	16	XXXXXX-XXX (ex: 120600-020)
<firmware version>	ASCII character, null-padded	20	vX.X.X (ex: v1.0.0)

The IP address of an RTSA can be retrieved from the responding socket. The RTSA may be identified by matching the serial number (S/N) in the response message with the S/N on the label of the RTSA.

Appendix C: SCPI Command Syntax

Each SCPI command consists of a root node, one or more lower level nodes, follow by applicable parameters and separators:



Entering Commands

SCPI commands have both a long and short version, such as :SOURCE and :SOUR. The SCPI interface responds to either version, but will not respond to variations of either version. The interface does not differentiate between upper-case and lower-case letters but only the long or short form of a command.

An example correct and incorrect SCPI entry format for :SOURce command:

	Command Entry		
Correct Entry	:SOURCE :SOUR	:SOURce :sour	:source
Incorrect Entry	:SOU	:SOURC	:sourc

Note: At the end of each SCPI command string, whether a single command or multiple commands separated by semicolons ";", *a new line-feed or carriage return is required.* Example in C: ":FREQ:CENTER 2400 MHZ\n" or "FREQ:CENT 2400 MHZ;INP:ATT 0\n".

Notation

Notation	Description
:	Links command keywords together
;	Separates multiple commands entered together on a single program message
single space	Uses to separate a parameter from a command or unit from a parameter
,	Uses to separate multiples parameters of a command
[]	Uses to optionally enclose zero or more parameters
{.} or {.}*	The enclosed item maybe included zero or more times
{.}+	The enclosed items occurs one or more times
{.|.|.}	One and only one of the two or more enclosed items separated by | maybe included
<>	Uses to enclose *required* parameter descriptions
?	Indicates query command, use where applicable
|	Indicates “or” and is used to separate alternative parameter options
::=	Means “is defined as”

Parameter types

This section defines different SCPI parameter data type.

Parameter Type	Description
<boolean>	ON | OFF | 1 | 0 Boolean parameters are always returned as 1 or 0 in NR1 format by query commands
<integer> <int>	Unsigned integer of NR1 format Ex: 1 or 3432
<NR1>	Signed integer without a decimal point (implied radix point) Ex: -25 or 0
<NR2>	Signed number with an explicit radix point Ex: -1.234 or 1.0 or 0.0
<NR3>	Scaled explicit decimal point numeric value with and an exponent Ex: 2.73e+2 or 2.351e2
<NRf>	<NR1>|<NR2>|<NR3>
<NRr>	Non-decimal numeric value such as hexadecimal, octal or binary
<char> <character>	Character program data Ex: MAXimum or MEDium
<string>	ASCII string surrounded by single or double quotes Ex: “This is an example”

Default Units

Parameter	Default Unit
frequency	Hz
time	s or ns where applicable
voltage	V
absolute amplitude	dBm
relative amplitude	dB

Units other than the default may be specified. If units are not specified then the default units apply. Note the following examples, which are all equivalent.

Example	`:FREQ:CENTer 2441.5 MHz`
is equivalent to	`:FREQ:CENTer 2441500000`
is equivalent to	`:FREQ:CENTer 2441500000 Hz`
is equivalent to	`:FREQ:CENTer 2441500 kHz`
is equivalent to	`:FREQ:CENTer 2441.5e6`

Appendix D: SCPI Status and Event Registers

The RTSA's SCPI interface has a status and event reporting system that enables the user to handle device events. The interface conforms to IEEE Std 488.2-1987 and SCPI 1999.0. This section discusses these status registers, status register enable masks, event queues and event handling.

Status Byte Register (SBR)

The SBR is used to determine the specific nature of the event or condition. It is read by issuing a *STB? command. The contents of the SBR are clear by issuing either a *STB? or *CLS command.

Bit	Name	Description
0	*not used*	This bit is not used and always 0.
1	*not used*	This bit is not used and always 0.
2	Error / Event Available (EAV)	This bit is set if there are any unread error or event in the System Error queue. It is read using the :SYSTem:ERRor[:NEXT]? command.
3	Questionable Register Summary	Summary of the Questionable Status register
4	Message Available (MAV)	This bit is set if there is any unread data in the Output queue.
5	Standard Event Status Bit (ESB)	This bit is set if there is any unread or non-cleared data in the Standard Event Status register.
6	Master Summary Status	This bit is set when any of the other bits are set.
7	Operation Register Summary	Summary of the Operation Status register

Standard Event Status Register (ESR)

The ESR is used to determine the nature of the status and error conditions. It is read by issuing a *ESR? command. The contents of the ESR are cleared by issuing either a *ESR? or *CLS command.

Bits in the ESR will cause a Service Request only when the corresponding bits in the Standard Events Status Enable Register are set.

Bit	Name	Description
0	Operation Complete (OPC)	Set to indicate that all pending operations are complete and R5500 is ready to accept another command, or that query results are available.
1	Request Control (RQC)	This bit is not used and always 0.
2	Query Error (QYE)	Set to indicate that a query has been made for which no response is available. Query errors have SCPI error codes from -499 to -400.

Bit	Name	Description
3	Device Dependent Error (DDE)	Set to indicate that a device-dependent error has occurred. Device-dependent errors have SCPI error codes from -399 to -300.
4	Execution Error (E)	Set to indicate that a parameter exceeds its allowed range. Execution errors have SCPI error codes from -299 to -200.
5	Command Error (CME)	Set to indicate that a command error has occurred. Command errors have SCPI error codes from -199 to -100.
6	*not used*	This bit is always 0.
7	Power ON (PON)	Set once upon power-up.

Operational Status Register (OSR)

The OSR is a 16-bit register that is used to determine the state of operation. It is read by issuing a :STATus:QUEStionable[:EVENt]? command.

Bit	Name	Description
0	CALibrating	This bit is currently not used and always 0.
1	SETTling	This bit is set when the device is tuning or is otherwise not yet ready to capture data.
2	RANGing	This bit is currently not used and always 0.
3	SWEeping	This bit is currently not used and always 0.
4	MEASuring	This bit is currently not used and always 0.
5	Waiting for TRIG	This bit indicates that the device is armed and waiting for a trigger event.
6	Waiting for ARM	This bit indicates that the device is configured for triggering but has not been armed.
7	CORRecting	This bit is currently not used and always 0.
8	Data Available	This bit indicates that new data is available to be read. Note that this bit may toggle momentarily so transition detection should be used.
9-12	*not used*	These bits are not used and always 0.
13	INSTrument summary	This bit is currently not used and always 0.
14	PROGram running	This bit is currently not used and always 0.
15	*not used*	This bit is not used and always 0.

Questionable Status Register (QSR)

The QSR is a 16-bit register that is used to indicate conditions that may cause the measurement results to be of questionable quality. It is read by issuing a :STATus:QUEStionable[:EVENt]? command.

Bit	Name	Description
0	VOLTage	This bit is currently not used and always 0.
1	CURRent	This bit is currently not used and always 0.
2	TIME	This bit is currently not used and always 0.
3	POWer	This bit is currently not used and always 0.
4	TEMPerature	This bit is currently not used and always 0.
5	FREQuency	This bit is currently not used and always 0.
6	PHASe	This bit is currently not used and always 0.
7	MODulation	This bit is currently not used and always 0.
8	CALibration	This bit is currently not used and always 0.
9	POSition	This bit is used by GNSS-equipped RTSA products to indicate GNSS out-of-lock.
10-12	*not used*	These bits are not used and always 0.
13	INSTrument summary	This bit is currently not used and always 0.
14	Command Warning	This bit is currently not used and always 0.
15	*not used*	This bit is not used and always 0.

Output Queue

The R57x0 has an Output FIFO Queue that is structured as a FIFO and holds the response messages to queries. The SBR's MAV bit is set when there are messages in the queue. The unread results of a previous command are cleared from the queue when a new command or query is received.

Error and Event Queue

The R57x0 has an Error and Event FIFO Queue that holds up to 16 errors and events. It is queried using the :SYSTem:ERRor[:NEXT]? command. The *CLS command clears all entries from the queue.

Appendix E: SCPI Error Codes Used

Code	Message	Description
0	No error	
Command error, range [-199, -100]		
-144	Character data too long	The character data contained more than 12 characters.
-171	Invalid expression	The command syntax was incorrect.
Execution error, range [-299, -200]		
-200	Execution error	A generic execution error for which more specific information is not available.
-210	Trigger error	
-220	No matched module	The specific operation is not installed.
-221	Settings conflict	Indicates that a legal program data element was parsed but could not be executed due to the current device state
-222	Data out of range	A parameter was of the proper type but outside of the defined range for the specific command.
-223	Too much data	A parameter was received that contained more data than the device could handle.
-224	Illegal parameter value	A parameter was received that is NOT allowed for the particular command.
-230	Data corrupt or stale	Possibly invalid data; new reading started but not completed since last access.
-240	Hardware error	Indicates that a legal program command or query could not be executed because of a hardware problem in the device.
-241	Hardware missing	Indicates that a legal program command or query could not be executed because of missing device hardware. For example, an option is not installed.
Device specific error, range [-399, -300]		
-310	System error	
-321	Out of memory	An internal operation needed more memory than that was available.
-330	Self test failed	
-340	Calibration failed	
-350	Query overflow	The SCPI remote interface error queue overflowed.
Query error, range [-499, -400]		
-410	Query INTERRUPTED	A condition causing an INTERRUPTED query error occurred.
ThinkRF's RTSA Specific, range [-999, -900]		
-901	No data	Read trace command issued while there is no data available.
-911	Need firmware upgrade	The current firmware needs upgrading.
-912	Invalid option license	The option could not be installed because of invalid license.

Appendix F: SCPI Commands Quick Reference

This section summarizes the SCPI commands available for interfacing with R57x0. The commands are listed alphabetically based on the main node, then sub-nodes, so on. The sub-nodes are grouped and listed alphabetically based on functionality.

See Appendix C's Notation section for details on notations used in the Parameter column.

The Release column indicates from which **firmware** release version that the commands are available. **Grayed-out** commands are not yet implemented.

Keyword	Parameter	Description	Release
IEEE Mandated		*Page 44*	
*CLS		Clear all status registers	v1.0
*ESE	<integer>	Event Status Enable register	v1.0
*ESE?		Query ESE register	v1.0
*ESR?		Query Event Status Register	v1.0
*IDN?		Query device identification	v1.0
*OPC		Operation Complete	v1.0
*OPC?		Query OC	v1.0
*RST		Reset to factory default	v1.0
*SRE	<integer>	Service Request Enable bits	v1.0
*SRE?		Query SRE register	v1.0
*STB?		Query Status Byte register	v1.0
*TST?		Query self-test status	v1.0
*WAI		Wait-to-Continue	v1.0
:GNSS		*Page 72*	
[:Enable]	ON | OFF | 1 | 0	Enables the status of the GNSS module	v1.0
[:Enable]?			
:ADELay	-32767 – 32798	Sets the GNSS's antenna cable delay compensation, in nsec	v1.1.0
:ADELay?			
:CONStellation	GPS | BEIDOU | GLONASS	Sets one or two satellite constellations to be tracked	v1.1.0
:CONStellation?			
:POSition?		Queries the last known GNSS position in degrees latitude, degrees longitude and altitude in meters	v1.0
:REFerence?		Queries which timing reference source used to discipline the 10 MHz GNSS reference oscillator	v1.0
:INPut		*Page 64*	
:ATTenuator	0 | 10 | 20 | 30	Set the fix attenuation for R57x0-408 models and their variants	v1.0
:ATTenuator?			
:VARiable	0 | 10 | 20 | 30	Set the variable attenuation for R57x0-418 and -427 models and their variants	v1.0

Keyword	Parameter	Description	Release
:VARiable?			
:GAIN	<index> <ON | OFF | 1 | 0>	Set an input gain stage to be on or off. The index range is model dependent	v1.0
:GAIN?	<index>		
:HDR		Set gain level for the narrow-band ADC of the HDR signal path	v1.0
:HDR?	[MAX | MIN]		
:MODE	ZIF | DD | HDR | SH | SHN	Select the receiver mode of operation. See the complete command description section for special notes.	v1.0
:MODE?			
[:SENSe]		*Page 68*	
:DECimation	OFF | <integer>	Set the decimation rate as an exponent of 2 (i.e. rate = 2^{level} where level = 0, 1, 2 - 10)	v1.0
:DECimation?	[MAX | MIN]		
:FREQuency			
:CENTer	<NRf [unit]>	Set the center frequency of the RFE	v1.0
:CENTer?	[MAX | MIN]		
:IF?	<non-zero integer>	Query the IF frequencies that are used for the current input mode and center frequency	v1.0
:LOSCillator?	<1 | 2 | 3>	Get the frequency of the external LO 1, 2, or 3 in corresponding to current the RTSA's center frequency	v1.0
:SHIFt	<NRf [unit]>	Set the frequency shift value (not available for HDR mode)	v1.0
:SHIFt?	[MAX | MIN]		v1.0
:LOCK			
:REFerence?		Query the lock status of the PLL reference clock	v1.0
:RF?		Query the lock status of the RFE's RF VCO	v1.0
:SOURce		*Page 67*	
:REFerence			
:PLL	INT | EXT | GNSS	Select the 10MHz reference clock source	v1.0
:PLL?			
:PPS	EXT | GNSS	Select the PPS synchronization source	v1.0
:PPS?			
:STATus		*Page 59*	
:OPERation			
[:EVENt]?		Return the standard Operation Status Register (OSR) for any event	v1.0
:CONDition?		Queries the Operation Condition Register for any operation event	v1.0
:ENABle	<integer>	Enables bits in the Operation Enable Register	v1.0
:ENABle?			
:NTRansition		Enables bits in the Operation Negative Transition Register	v1.0
:NTRansition?			
:QTRansition		Enables bits in the Operation Positive Transition Register	v1.0

Keyword	Parameter	Description	Release
:QTRansition?			
:PRESET		Presets the R57x0 (similar to *RST)	v1.0
:QUEStionable			
[:EVENt]?		Return the standard Questionable Status Register (QSR) for any event	v1.0
:CONDition?		Return the Questionable Condition Register for any operation event	v1.0
:ENABle	<integer>	Enable bits in the Questionable Enable Register	v1.0
:ENABle?			
:NTRansition		Enable bits in the Questionable Negative Transition Register	v1.0
:NTRansition?			
:QTRansition		Enable bits in the Questionable Positive Transition Register	v1.0
:QTRansition?			
:TEMPerature?		Return the R57x0's internal ambient temperature	v1.0
:SWEep		*Page 79*	
:LIST			
:ITERations	<integer>	Define the number of times the list is repeated during execution	v1.0
:ITERations?			
:STARt	[integer]	Begin execution of the current sweep list from the first entry	v1.0
:STATus?			
:STOP		Stop execution of the current sweep list	v1.0
:ENTRy			
:COPY	<integer>	Copie the settings of an existing sweep entry into the current settings for quick editing	v1.0
:COUNt?		Get the number of entries available in the list	v1.0
:DELETE	<integer> | ALL	Delete a specified entry or all entries	v1.0
:NEW		Set the sweep entry's capture configuration settings to default values	v1.0
:READ?	<integer>	Get the settings of an existing sweep entry	v1.0
:SAVE	[integer]	Save the current editing entry to the end of the list or before the specified ID location in the list when the integer value is given	v1.0
			v1.0
:ATTenuator	As defined in :INPut:ATTenuator, page 64		v1.0
:VAR[?]	As defined in :INPut:ATTenuator:VARiable, page 64		v1.0
:DECimation	As defined in [:SENSe]:DECimation, page 68		v1.0
:DECimation?			
:FREQuency			
:CENTer	<NRf [unit]>[,<NRf [unit]>] ::= <start freq>[,<stop freq>]	Set the center frequency or a range of center frequencies that are stepped by the value defined by :SWEep:ENTRy:FREQuency:STEP	v1.0
:CENTer?			
:SHIFt	As defined in [:SENSe]:FREQuency:SHIFt, page 70		v1.0
:SHIFt?			v1.0

Keyword	Parameter	Description	Release
:STEP	<NRf [unit]>	Set the amount of frequency that the center frequency is stepped by	v1.0
:STEP?			
:GAIN			
:HDR	As defined in :INPut:GAIN:HDR, page 66		v1.0
:HDR?			
:MODE	As defined in :INPut:MODE, page 66		v1.0
:MODE?			
:DWELl	<integer>[,<integer>] ::= <sec>[,<microsec>]	Set the maximum amount of time to wait for the trigger of a sweep entry to occur, after which the trigger is aborted and the next sweep entry if existed will run. When the trigger type is NONE, dwell time is ignored. Default 0.0 sec.	v1.0
:DWELl?			
:PPBlock	Same as :TRACe:BLOCk:PACKets, page 77		v1.0
:PPBlock?			
:SPPacket	As defined in :TRACe:SPPacket, page 78		v1.0
:SPPacket?			
:TRIGger			
:LEVel	As defined in :TRIGger:LEVel, page 75		v1.0
:LEVel?			
:TYPE	As defined in :TRIGger:TYPE, page 74		v1.0
:TYPE?			
:SYSTem		*Page 47*	
:ABORt		Abort the current data capturing process and puts the RTSA system into a normal manual mode (i.e. sweep, trigger, and streaming will be aborted)	v1.0
:CAPTure			
:MODE?		Get the current capture mode of the RTSA (i.e. sweeping, streaming or block mode)	v1.0
:COMMunicate			
:HISLip			
:SESSion?		Return the HiSLIP connection Session ID	v1.0
:LAN			
:APPLy		Apply the new RTSA's LAN settings from the commands above, which will then take effect. This command should be applied only once all the required LAN settings have been set.	v1.0
:CONFigure	DHCP | STATIC	Set the RTSA's LAN to use DHCP or STATIC configuration type	v1.0
:CONFigure?	[CURRENT]		v1.0
:DNS	<main DNS>[,alt DNS]	Set the RTSA's LAN DNS address(es)	v1.0
:DNS?	[CURRENT]		
:GATEway	<IPv4 address>	Set the RTSA's LAN Gateway address	v1.0
:GATEway?	[CURRENT]		
:IP	<IPv4 address>	Set the new IPv4 address for the RTSA's LAN	v1.0
:IP?	[CURRENT]		
:MTU	<MTU value>	Set the MTU value between 256 and 1500, inclusive	v1.0

Keyword	Parameter	Description	Release
:MTU?	[CURRENT]		
:NETMask	<IPv4 address>	Set the RTSA's LAN netmask address	v1.0
:NETMask?	[CURRENT]		
:NTP	<IPv4 address>	Set an NTP IP serve address for time updating	
:ERRor			v1.0
[:NEXT]?		Return the SCPI error/event	v1.0
:ALL?		Return all the error codes and messages	v1.0
:CODE			
[:NEXT]?		Return the SCPI error/event code only	v1.0
:ALL?		Return all the error codes only	v1.0
:COUNt?		Returns the number of errors	v1.0
:FLUSh		Clear the R57x0's internal data storage buffer of any remaining old data that has not been transferred out of the RTSA.	v1.0
:LOCK			
:HAVE?	ACQuisition	Return the current lock state of the task specified	v1.0
:REQuest?	ACQuisition	Request the R57x0 to provide a lock on a specific task such that only the application that has the lock can perform the task.	v1.0
:OPTions?		Returns comma separated 3-digit values to represent the hardware option(s) or features available with a particular RTSA model.	v1.0
:SYNC			
:MASTer	ON | OFF | 1 | 0	Set an RTSA unit to be the master or slave for a synchronization trigger system with multiple units. Affects :TRIG:TYPE PULSe or WORD.	v1.0
:MASTer?			
:WAIT	<integer>	Set the delay time in nanoseconds that the system must wait after receiving the trigger signal before performing data capture.	v1.0
:WAIT?			
:VERSion?		Return the SCPI compliance version	v1.0
:DATE	<integer>,<integer>,<integer> ::= <year>,<month>,<date>	Set the date	v1.0
:DATE?		Get the date in the box	v1.0
:TIME	<integer>,<integer>,<integer>[,<integer>] | <char> ::= <hr>,<min>,<s>[,<ms>]	Set the time	v1.0
:TIME?		Returns <hr>,<min>,<s> in UTC time	v1.0
:ADJust	<integer>	Adjust the system time relative to its current time	
:SYNC	DISable | NTP,{ONCE | CONTinuous}	Select the synchronization source and mode	v1.0
:SYNC?			v1.0
:STATus?		Return the status of the time synchronization	
:TRACe		*Page 76*	
:BLOCk			
:DATA?		Initiate the sending of the IQ data captured	v1.0

Keyword	Parameter	Description	Release
:PACKets	<integer>	Set the number of IQ data packets to be captured per block (a block = :PACKets * SPP)	v1.0
:PACKets?	[MAX | MIN]		
:SPPacket	<integer>	Define the number of IQ samples per VRT packet, and must be a multiple of 16	v1.0
:SPPacket?	[MAX | MIN]		
:STReam			
:STARt	[integer]	Initiate the capture, storage and streaming of IQ data	v1.0
:STOP		Stop streaming	v1.0
:TRIGger		*Page 74*	
:LEVel	<NRf [unit]>,<NRf[unit]>,<NRf [unit]> ::= <start>,<stop>,<level>	Set the frequency range and amplitude of a frequency domain level trigger	v1.0
:LEVel?			
:PERiodic	<integer [unit]>	Set the time period of a periodic trigger	TBD
:PERiodic?			
:TYPE	LEVel | PERiodic | PPS | PULSe | WORD | NONE	Set or disables the trigger type	v1.0
:TYPE?			

R55x0 vs. R57x0 List of Changes

This section provides a list of changes in the R57x0's Programmer's Guide *as compare to* that of the R55x0.

Sections	R5700/R5750
Functional Overview	Updated the Architecture and RF Receiver Front-end sections to reflect the new capabilities with new hardware changes (see Figure 1 and Figure 2)
SCPI Commands	
:GNSS	Added new :GNSS subsystem and command(s) to GNSS Commands section
:OUTput	Removed :OUTput subsystem as it is not available on the R57x0 products, and hence no CONNector and HIF options
[:SENSe]	Removed [:SENSe]:FREQuency:INVersion? as no IQ OUT feature
:SOURce	Added new GNSS option for + :SOURce:REFerence:PLL + :SOURce:REFerence:PPS
:SYSTem	Removed WBIQ option mentioning from :SYSTem:OPTions? and across the document as this option is not available with the R57x0 products
VRT Protocol	- Added section Formatted GPS Geolocation section for GNSS VRT context packets

References

1. “Standard Commands for Programmable Instruments (SCPI)”, SCPI Consortium, May 1999, version 1999.0, http://www.spiconsortium.org
2. "VITA Radio Transport (VRT) Draft Standard" VITA-49.0 – 2007, VITA Standard Organization, 31 October 2007, Draft 0.21, http://www.vita.com/
3. "IEEE Standard Codes, Formats, Protocols, and Common Commands", ANSI/IEEE Standard 488.2-1992, http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&isnumber=5581&arnumber=213762&punumber=2839

Document Revision History

This section summarizes document revision history.

Document Version[1]	Release Date	Revisions and Notes
v1.0.0	Aug 20, 2018	First release of this document, basing on R5500's Programmer's Guide. See R55x0 vs. R57x0 List of Changes for a summary of main changes.
v1.1.0	June 1, 2019	- Changed R5700 to R57x0 to refer to R5700 and R5750 products - Corrected & added new information to Table 2 Added: - New GNSS SCPI commands are added: :GNSS:ADELay and :GNSS:CONStellation - A Caution note in the *RST command with regard to DD mode - To :TRIGger:LEVel section, improved performance information in Table 41 Updated: - Simple 2-port TCP/IP Connection section to include the importance of port connection order, 37001 follows by 37000 - Corrected/made consistent the SCPI command's specification - In Trailer Word Format section, Grayed-out Over-range Indicator as not yet available - Removed [:SENSe]:FREQuency:INVersion? as no IQ OUT feature

[1] Document Version is not the same as the firmware Release Version as mentioned in Appendix F: SCPI Commands Quick Reference.