

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Keysight / Agilent E6401A* at our website: ***Click HERE***

Agilent Technologies

Agilent Technologies E6501A/E6502A/E6503A VXI Receivers

User's Guide

Agilent Technologies
Part Number: E6500-90015
Printed in USA
February 2000

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Notice

The information contained in this document is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Agilent Technologies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Agilent Technologies assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Agilent Technologies.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another language without prior written consent of Agilent Technologies.

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c)(1) and (c)(2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Agilent Technologies
1400 Fountaingrove Parkway
Santa Rosa, CA 95403-1799, U.S.A.

© Copyright Agilent Technologies 2000

Windows NT® and Windows 95® are registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Warranty

Certification

Agilent Technologies certifies that this product met its published specifications at the time of shipment from the factory. Agilent Technologies further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology (NIST, formerly NBS), to the extent allowed by the Institute's calibration facility, and to the calibration facilities of other International Standards Organization members.

Warranty

This Agilent Technologies system product is warranted against defects in materials and workmanship for a period corresponding to the individual warranty periods of its component products. Instruments are warranted for a period of three years. During the warranty period, Agilent Technologies will, at its option, either repair or replace products that prove to be defective.

Warranty service for products installed by Agilent Technologies and certain other products designated by Agilent Technologies will be performed at Buyer's facility at no charge within Agilent Technologies service travel areas. Outside Agilent Technologies service travel areas, warranty service will be performed at Buyer's facility only upon Agilent Technologies's prior agreement and Buyer shall pay Agilent Technologies's round trip travel expenses. In all other areas, products must be returned to a service facility designated by Agilent Technologies.

For products returned to Agilent Technologies for warranty service, Buyer shall prepay shipping charges to Agilent Technologies and Agilent Technologies shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to Agilent Technologies from another country.

Agilent Technologies warrants that its software and firmware designated by Agilent Technologies for use with an instrument will execute its programming instructions when properly installed on that instrument. Agilent Technologies does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

LIMITATION OF WARRANTY. The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. AGILENT TECHNOLOGIES SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES. THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. AGILENT TECHNOLOGIES SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

Assistance

Product maintenance agreements and other customer assistance agreements are available for Agilent Technologies products.

For assistance, call your local Agilent Technologies Sales and Service Office (refer to "Service and Support" on page xii).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Service and Support

Any adjustment, maintenance, or repair of this product must be performed by qualified personnel. Contact your customer engineer through your local Agilent Technologies Service Center. You can find support information on the web at http://www.tmo.hp.com/tmo/datasheets/English/index.html.

If you do not have access to the Internet, one of these Agilent Technologies centers can direct you to your nearest Agilent Technologies representative:

United States:	Agilent Technologies Test and Measurement Call Center PO Box 4026 Englewood, CO 80155-4026 (800) 452 4844 (toll-free in US)
Canada:	Agilent Technologies Canada Ltd. 5150 Spectrum Way Mississauga, Ontario L4W 5G1 (905) 206 4725
Europe:	Agilent Technologies European Marketing Centre Postbox 999 1180 AZ Amstelveen The Netherlands (31 20) 547 9900
Japan:	Agilent Technologies Ltd. Measurement Assistance Center 9-1, Takakura-Cho, Hachioji-Shi Tokyo 192, Japan (81) 426 56 7832 (81) 426 56 7840 (FAX)
Latin America:	Agilent Technologies Latin American Region Headquarters 5200 Blue Lagoon Drive, 9th Floor Miami, Florida 33126, U.S.A. (305) 267 4245, (305) 267-4220 (305) 267 4288 (FAX)
Australia/New Zealand:	Agilent Technologies Australia Ltd. 31-41 Joseph Street Blackburn, Victoria 3130 Australia 1 800 629 485 (Australia) 0800 738 378 (New Zealand) (61 3) 9210 5489 (FAX)
Asia-Pacific:	Agilent Technologies Asia Pacific Ltd. 17-21/F Shell Tower, Times Square 1 Matheson Street, Causeway Bay Hong Kong (852) 2599 7777 (852) 2506 9285 (FAX)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Safety and Regulatory Information

Review this product and related documentation to familiarize yourself with safety markings and instructions before you operate the instrument. This product has been designed and tested in accordance with international standards.

WARNING

The WARNING notice denotes a hazard. It calls attention to a procedure, practice, or the like, that, if not correctly performed or adhered to, could result in personal injury. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

CAUTION

The **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like, which, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

Instrument Markings

	When you see this symbol on your instrument, you should refer to the instrument's instruction manual for important information.
	This symbol indicates hazardous voltages.
	The laser radiation symbol is marked on products that have a laser output.
	This symbol indicates that the instrument requires alternating current (ac) input.
	The CE mark is a registered trademark of the European Community. If it is accompanied by a year, it indicates the year the design was proven.
	The CSA mark is a registered trademark of the Canadian Standards Association.
1SM1-A	This text indicates that the instrument is an Industrial Scientific and Medical Group 1 Class A product (CISPR 11, Clause 4).
	This symbol indicates that the power line switch is ON.
	This symbol indicates that the power line switch is OFF or in STANDBY position.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Safety Earth Ground



This is a Safety Class I product (provided with a protective earthing terminal). An uninterruptible safety earth ground must be provided from the main power source to the product input wiring terminals, power cord, or supplied power cord set. Whenever it is likely that the protection has been impaired, the product must be made inoperative and secured against any unintended operation.

Before Applying Power

Verify that the product is configured to match the available main power source as described in the input power configuration instructions in this manual. If this product is to be powered by autotransformer, make sure the common terminal is connected to the neutral (grounded) side of the ac power supply.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Typeface Conventions

Italics

- Used to emphasize important information:
  Use this software *only* with the Agilent Technologies xxxxxX system.
- Used for the title of a publication:
  Refer to the *Agilent Technologies xxxxxX System-Level User's Guide*.
- Used to indicate a variable:
  Type LOAD BIN *filename*.

Instrument Display

- Used to show on-screen prompts and messages that you will see on the display of an instrument:
  The Agilent Technologies xxxxxX will display the message **CALI SAVED**.

[Keycap]

- Used for labeled keys on the front panel of an instrument or on a computer keyboard:
  Press **[Return]**.

{Softkey}

- Used for simulated keys that appear on an instrument display:
  Press ***{Prior Menu}***.

User Entry

- Used to indicate text that you will enter using the computer keyboard; text shown in this typeface must be typed *exactly* as printed:
  Type LOAD PARMFILE
- Used for examples of programming code:
  #endif // ifndef NO_CLASS

Path Name

- Used for a subdirectory name or file path:
  Edit the file *usr/local/bin/sample.txt*

Computer Display

- Used to show messages, prompts, and window labels that appear on a computer monitor:
  The **Edit Parameters** window will appear on the screen.
- Used for menus, lists, dialog boxes, and button boxes on a computer monitor from which you make selections using the mouse or keyboard:
  Double-click **EXIT** to quit the program.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

What You'll Find in This Manual...

Chapter 1

- **Product Description and Configurations**

Chapter 1 introduces the receiver systems, shows the receiver options that extend the standard functionality, describes how to inspect the receiver, describes the front-panel features, illustrates standard receiver configurations, and contains a list of the accessories supplied.

Chapter 2

- **Getting Started**

Chapter 2 provides information about electrostatic discharge, how to set the logical address switches of each module, how to install the modules into a mainframe, describes the PC or UNIX workstation system requirements, how to install the software, how to start the virtual front panel, and how to check operation.

Chapter 3

- **Using the Receiver**

Chapter 3 provides an overview of the receiver functionality, shows how to use the driver software, and shows how to use the virtual front panel.

Chapter 4

- **Theory of Operation**

Chapter 4 contains in-depth theory of operation including analog gain, autoranging, and dynamic range optimization.

Chapter 5

- **Specifications**

Chapter 5 contains the receiver specifications.

Chapter 6

- **Command Reference**

Chapter 6 provides a listing of all driver software commands used to control the receiver operation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Contents

Contents . . . . . Contentsi
Notice ix
Restricted Rights Legend ix
Warranty x
Certification x
Warranty x
Assistance xi
Service and Support xii
Safety and Regulatory Information xiii
Safety Earth Ground xiv
Before Applying Power xiv
Typeface Conventions xv
What You'll Find in This Manual... xvi

1. Product Description and Configurations

Introducing the E6501A/E6502A/E6503A
VXI Receivers 1-2
E6501A VXI Receiver 1-2
E6502A VXI Receiver 1-2
E6503A VXI Receiver 1-3
Receiver Options 1-4
Initial Inspection 1-5
Front-Panel Features 1-6
Standard Receiver Configurations 1-22
E650XA Mainframe Options 1-22
Accessories Supplied 1-29

2. Getting Started

Electrostatic Discharge Information 2-2
Preparation for Use 2-4
Procedure 2-5
Local Bus Compatibility 2-8
Installing the Receiver 2-9
Cabling the Receiver 2-10
10 MHz Reference 2-10
Installing the MXI Controller Cable 2-10
Configuring a Multiple Mainframe System 2-10
PC or UNIX Workstation System Requirements 2-13
Installing the Software 2-14
Configuring the VXI Bus Timeout 2-14
Starting the Virtual Front Panel 2-15
Checking Operation 2-19
Procedure 2-19

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3. Using the Receiver

E650XA VXI Receiver Overview 3-2
Core Receiver Capabilities 3-2
Monitor Mode 3-2
Search Mode 3-4
Multiple Demodulations 3-5
Digital IF (DDC) Bandwidths 3-6
Gain Control and Dynamic Range Optimization 3-7
Analog Outputs 3-7
Digital Outputs 3-9
Receiver Capabilities By Configuration 3-11
Using the Virtual Front Panel 3-12
File Menu 3-12
Settings Menu 3-12
General Setup Dialog Box 3-13
Layout Menu 3-14
Mezzanine Menu 3-14
Open Menu 3-16
New Spectral Display 3-16
Shortcut Menu in Spectral Display 3-17
16 MHz Stare 3-18
Mezz. 1 Controls 3-18
Audio Controls (Mezz 1) 3-20
RSSI for Mezzanine 1 3-20
Search Controls for Mezzanine 1 3-20
Search Display for Mezzanine 1 3-22
IF Channel Controls 3-22
Tuner Controls 3-23
BFO Control 3-24
Window Menu 3-24
Using the Driver Software 3-25
Programmer's Block Diagrams 3-25
Mezzanine Data Select Modes 3-25
Command Group Numbers 3-25
Maximum FFT Length 3-26
Maximum Number of FFT Processes 3-26
DSP Considerations 3-27
Serial Loading 3-27
DSP Loading 3-27
Driver Revision 3-28
Default Receiver Settings 3-28
Opening and Closing an Instrument Session 3-28
Return Values 3-31
Pointers to Memory Addresses 3-31
Receiver Programming Examples 3-31
To set up a search process 3-31
To set up an FFT measurement 3-33

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To change tuner frequency 3-34
To change the IF bandpass filter setting 3-34
To set the IF gain 3-35
To set tuner input attenuation 3-36
To set search mode resolution bandwidth 3-36
To set span in monitor mode 3-36
To set mezzanine data select mode 3-37
To turn the tuner 10 MHz reference off 3-37
To turn the IF processor 10 MHz reference on 3-37
To activate automatic frequency control 3-38
To lock autoranging 3-38
To set up dynamic range optimization 3-38
To set up a channelized power measurement 3-39
To set up and start a monitor process 3-40
To set up demodulation, turn on an audio channel, and set squelch . 3-41
Multi-Threading Considerations 3-42
Synchronizing Multiple IF Processors and Capturing Data 3-46
Hardware Configuration 3-46
Software Configuration 3-47
Software Trigger 3-47
Hardware Trigger 3-47
Distributing Clocks 3-48
Synchronizing the DDCs 3-48
Arming the DSP 3-49
Locking Autorange 3-50
Captured Data Format 3-50
Sending Indefinite Samples 3-56
Data Collection Programming Examples 3-57
DMA Block Size Considerations 3-59
Common Functions for Collection 3-60
Scenario 1: Capture N Samples of Digital I/Q Data Across VXI Bus 3-63
Scenario 2: Capture N Samples of Digital I/Q Data From VXI Bus Using a Trigger 3-68
Scenario 3: Capture Digital I/Q Data Indefinitely Across VXI Bus . 3-75
Scenario 4: Capture Digital I/Q Data Indefinitely From VXI Bus Using a Trigger 3-81
Scenario 5: Stream Digital I/Q Data Indefinitely to the Link Port Using a Trigger 3-88
Scenario 6: Stream Digital I/Q Data Indefinitely to the Link Port . . 3-94
Scenario 7: Stream N Samples of Digital I/Q Data to the Link Port . 3-99
Scenario 8: Stream N Samples of Digital I/Q Data to the Link Port Using Multiple Triggers 3-103

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 9: Stream N Samples of I/Q Data to the Link Port Using a Trigger 3-109
Scenario 10: Stream ADC Data Indefinitely to the Link Port 3-115
Scenario 11: Stream N Samples of ADC Data to the Link Port 3-120
Scenario 12: Stream N Samples of ADC Data to the Link Port Using a Single Trigger 3-124
Scenario 13: Stream N Samples of ADC Data to the Link Port Using Multiple Triggers 3-129
Scenario 14: Capture N Samples of Full Rate ADC Data Across the VXI Bus 3-134
Scenario 15: Capture N Samples of ADC Data From VXI Bus Using a Trigger 3-139

4. Theory of Operation

E6501A/E6502A/E6503A VXI Receiver Description 4-2
E6401A 20 to 1000 MHz Downconverter Operation 4-4
Functions 4-4
Description 4-4
Inputs and Outputs 4-5
E6401A 4-5
E6402A Local Oscillator Operation 4-6
Functions 4-6
Description 4-6
2nd Local Oscillator 4-6
10 MHz Reference 4-6
First Local Oscillator 4-7
E6402A Option 002 Module 4-7
Inputs and Outputs 4-8
E6402A 4-8
E6402A Option 002 4-8
E6403A 1000 to 3000 MHz Block Downconverter Operation 4-9
Functions 4-9
Description 4-9
Inputs and Outputs 4-10
E6404A IF Processor Operation 4-11
Functions 4-11
Description 4-11
Mezzanine Board Description 4-11
Overview of Automatic Gain Control (AGC) 4-13
Autoranging Operation 4-13
Autoranging Benefits 4-14
Autoranging Routine Attack and Decay Time 4-14
30 kHz and 700 kHz Analog Filters 4-14
Processing Gain 4-15
Dynamic Range Optimization 4-15
Dynamic Range Optimization Attack and Decay Time 4-16

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Interrelationship Between Autoranging, DRO, Correction RAM, and DSP 4-16
Dynamic Range Optimization in Search Mode 4-17
FFT-Based Measurements 4-18
FFT Background 4-18
FFT Properties 4-18
FFT Process 4-19
Stepped FFT Measurements 4-19
Windowing 4-20
Purpose for Windowing 4-20
Window Implementation 4-20
Window Characteristics 4-21
Window as Resolution Bandwidth Filter 4-21
FFT Resolution Bandwidth Range (Search Mode) 4-21
FFT Resolution Bandwidths <5 kHz (Search Mode) 4-22
FFT Resolution Bandwidths for DDC IF Pan Windows 4-22
Improved Sensitivity Using FFTs 4-22
Inputs and Outputs 4-23

5. Specifications

Frequency-Related Specifications 5-2
Amplitude-Related Specifications 5-4
IF (Intermediate Frequency) Processing 5-7
Physical Characteristics 5-14
General Information 5-17
Environmental Information 5-19
VXI Information 5-20
Regulatory Information 5-21

6. Programming Command Reference

Overview 6-2
Driver Architecture 6-3
Capability Classes 6-3
Measure Capability Class Functions 6-4
Route Class Functions 6-5
Application Functions 6-5
Return Values 6-6
Special Values 6-9
Command Lists 6-10
Pointers to Memory Addresses 6-10

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1 Product Description and Configurations

In This Chapter

- Introducing the E6501A, E6502A, E6503A VXI Receivers
- Receiver Options
- Initial Inspection
- Front-Panel Features
- Standard Receiver Configurations
- Accessories Supplied

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Introducing the E6501A/E6502A/E6503A VXI Receivers

Each standard E6501A/E6502A/E6503A VXI receiver is a combination of E6500A VXI tuner modules and an E6404A IF processor module. The modules comprising each receiver model are listed below.

NOTE

Refer to Figure 4-3 for a block diagram showing the mezzanine board, digital downconverters (DDCs), and digital signal processor (DSP).

E6501A VXI Receiver

The E6501A XVI receiver consists of:

- One E6401A 20–1000 MHz downconverter module
- One E6402A local oscillator module
- One E6403A 1000–3000 MHz block downconverter (optional)
- One E6404A IF processor module contains:
  - One IF channel
  - One mezzanine with a digital downconverter and a digital signal processor

E6502A VXI Receiver

The E6502A XVI receiver consists of:

- Two E6401A 20–1000 MHz downconverter modules
- Two E6402A local oscillator modules
- Two E6403A 1000–3000 MHz block downconverters (optional)
- One E6404A IF processor module contains:
  - Two IF channels (Option 040)
  - Two mezzanines, each with a digital downconverter and a digital signal processor (Option 031)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6503A VXI Receiver

The E6503A XVI receiver consists of:

- Two E6401A 20–1000 MHz downconverter modules
- One E6402A Option 002 dual channel local oscillator module
- Two E6403A 1000–3000 MHz block downconverters (optional)
- One E6404A IF processor module contains:
  - Two IF channels (Option 040)
  - One mezzanine with two digital downconverters and a digital signal processor (Option 022)

The E650XA VXI Series receivers, consisting of three to seven C-size modules, are implemented in the VXI platform to provide the flexibility needed to address different system requirements. Agilent Technologies supplies the hardware and the driver software. A simple virtual front panel program (no source code is provided) is also supplied for Windows NT® and Windows 95®. This program can be used to demonstrate some of the receiver's capabilities, and also be used to verify equipment operation.

The E650XA Series receivers can be used for signal searching, signal collection (demodulation of AM, FM, USB, LSB, ISB, PM, and CW signals), and direction finding applications using multiple channels for signals of interest.

The E6500A tuner provides high dynamic range downconversion to a fixed bandwidth, 21.4 MHz intermediate frequency (IF) output. The input frequency range of 20 to 1000 MHz can be extended to 3000 MHz by adding an E6403A 1000–3000 MHz block downconverter module (Option 003).

The E6404A IF processor (IFP) module digitizes the IF output of the tuner, provides analog filtering and gain control, performs analog-to-digital conversion (ADC), digital filtering, digital signal processing, demodulation and fast Fourier transforms (FFTs), and digital-to-analog audio outputs. Also, digital data outputs provide I/Q data from the digital down converters (DDCs) and raw A/D data.

Controlling the receivers requires an MXI controller card in slot zero of the VXI mainframe, and software. Driver software for Windows NT® and Windows 95® is included with this product, enabling the user to control the receiver with high-level commands. Also required is the HP I/O Libraries (VXI plug and play drivers). Note that the driver software must be converted for use on a UNIX workstation. Driver source code is included for this purpose.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Receiver Options

Table 1-1 shows options that add functionality to the standard receivers.

Table 1-1 E6501A/E6502A/E6503A Receiver Options

Option	Description	E6501A[11]	E6502A[12]	E6503A[13,14]
003	Increases frequency coverage to 3 GHz	x	x[1]	x[1]
006	Adds a 6 slot E1421B VXI mainframe[2]	x	x	x
013	Adds a 13 slot E1401B VXI mainframe[2]	x	x	x
022	Adds 1 DDC[3,4] to mezzanine #1	x	x	
025	Adds 4 DDCs[3,4] to mezzanine #1	x	x	x[5]
026	Adds 4 MBytes RAM[6] to mezzanine #1	x	x	x
031	Adds a second mezzanine[7,8]	x		x
032	Adds 1 DDC[9] to mezzanine #2	x	x	x
035	Adds 4 DDCs[9] to mezzanine #2	x	x	x
036	Adds a second mezzanine and 4 MBytes RAM[10]	x	x	x
0B1	Adds extra user's guide	x	x	x

1. Adds two E6403A 3 GHz block downconverter VXI modules to the receiver.
2. Receiver specifications are guaranteed only in VXI mainframes.
3. Digital downconverters are used for digital filtering and for channelization of multiple signals for demodulation by the DSP.
4. Option 022 and 025 may not be ordered together. Choose one or the other. A maximum quantity of ONE of these options is allowed.
5. E6503A Option 025 adds three DDCs to mezzanine #1 (not four), since two reside on the standard configuration.
6. 4 MB of static RAM useful for capture and delay data memory applications. May be ordered with Option 022 or 025 (choose one), if desired.
7. Second mezzanine includes one DDC and one DSP. E6502A includes the second mezzanine in its standard configuration.
8. Option 031 and 036 may not be ordered together. Choose only one.
9. Option 032 and 035 MUST be ordered with either Option 031 or 036 (choose one). Option 031 or 036 are necessary to provide the second mezzanine on which the Option 032 or 035 DDCs must reside. Option 032 and 035 may not be ordered together. Choose one or the other. A maximum quantity of ONE of these options (032 or 035) is allowed.
10. 4 MB of static RAM useful for capture and delay data memory applications. Option 036 may not be ordered with the E6501A Option 031 or E6503A Option 031. Choose only one. Since the E6502A includes a second mezzanine, the E6502A Option 036 simply adds 4 MB of RAM.
11. E6501A allows a second mezzanine to be added (Option 031) but not a second IF channel since only one tuner front-end is included.
12. Includes E6404A IF processor with module Options 031 (mezzanine #2) and 040 (IF Ch. 2).
13. Includes E6404A IF processor with module Options 022 (DDC #2) and 040 (IF Ch. 2).
14. To configure more than two RF channels, call your Agilent Technologies sales representative for available custom configurations.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Initial Inspection

Inspect the shipping container for damage. If the shipping container or cushioning material is damaged, keep it until you have verified that the contents are complete and you have tested the module(s) mechanically and electrically.

Table 1-2 through Table 1-9 contain a listing of the accessories shipped with each module. If the contents are incomplete or if the modules do not pass the "Checking Operation" procedure found at the end of Chapter 2, notify the nearest Agilent Technologies Service and Support office. A listing of Agilent Technologies Service and Support offices is located at the front of this manual. If the shipping container is damaged or the cushioning material shows signs of stress, also notify the carrier. Keep the shipping materials for inspection by the carrier. The Agilent Technologies office will arrange for repair or replacement without waiting for a claim settlement.

If the shipping materials are in good condition, retain them for possible future use. You may wish to ship the module(s) to another location or return it to Agilent Technologies for service.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Front-Panel Features

In this section, the front-panel features will be described for each of the modules used to create the three standard receiver configurations described in this manual. Each of the three configurations are comprised of three to seven modules. An illustration of each receiver system including module position is included. The modules are:

- E6401A 20–1000 MHz downconverter
- E6402A local oscillator
- E6402A Option 002 local oscillator with dual LO outputs
- E6403A 1000–3000 MHz block downconverter
- E6404A IF processor
- E6404A Option 031 (mezzanine #2) and Option 040 (IF Ch. #2) IF processor
- E6404A Option 022 (DDC #2 on mezzanine #1) and Option 040 (IF Ch. #2) IF processor

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6401A 20–1000 MHz Downconverter

Access LED flashes whenever the module is being accessed via the VXI backplane.

21.4 MHz IF Output is the tuner output signal port that connects to the E6404A IF In port.

2nd LO Input port connects to the E6402A 2nd LO Output port. The signal frequency is approximately 1200 MHz.

1st LO Input port connects to the E6402A 1st LO Output port. The signal frequency ranges from approximately 1241.4 to 2221.4 MHz.

Block Downconv Input port connects to the E6403A Block Downconv Output port that ranges, in bands, from approximately 250 to 900 MHz (used for Option 003 configuration).

20–1000 MHz Input is the receiver input port. When using an Option 003 configuration to extend the frequency coverage range to 3 GHz, this port is connected to the E6403A 20–1000 MHz Output port.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

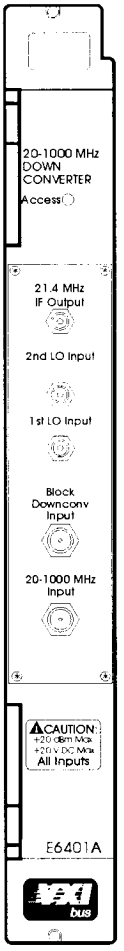


Figure 1-1 E6401A 20–1000 MHz Downconverter Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6402A Local Oscillator

Access LED flashes whenever the module is being accessed via the VXI backplane.

Unlock LED lights when the 1st LO or 2nd LO is unlocked.

Ref TTL Out is the reference 10 MHz TTL-level output signal that connects to a slot 0 MXI controller. The TTL output provides a common reference for the VXI backplane. This allows the 10 MHz reference on the VXI backplane to be locked to the receiver reference.

Ext Ref In is an input port for an external reference 10 MHz signal.

Ref Out provides a reference 10 MHz output signal. This port typically connects to the E6404A IF processor Ref In port. This ensures the ADC sample clock is locked to the system reference.

3rd LO Out is typically used for a tuner system that requires conversion to baseband frequencies.

BD LO Output provides a 1200 MHz (approximately) signal to the E6403A BD LO Input port (when the E6403A Option 003 is present).

2nd LO Output port provides a 1200 MHz (approximately) signal that connects to the E6401A 2nd LO Input port.

1st LO Output port provides a 1241.4 to 2221.4 MHz (approximately) signal that connects to the E6401A 1st LO Input port.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

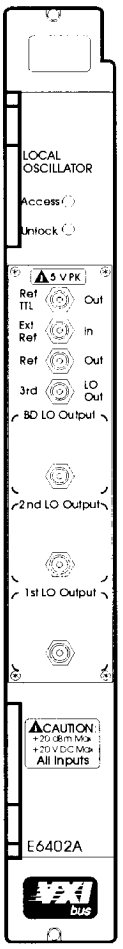


Figure 1-2 E6402A Local Oscillator Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6402A Option 002 Local Oscillator with Dual Outputs

Access LED flashes whenever the module is being accessed via the VXI backplane.

Unlock LED lights when the 1st LO or 2nd LO is unlocked.

Ref TTL Out is the reference 10 MHz TTL-level output signal that connects to a slot 0 MXI controller. The TTL output provides a common reference for the VXI backplane. This allows the 10 MHz reference on the VXI backplane to be locked to the receiver reference.

Ext Ref In is an input port for an external reference 10 MHz signal.

Ref Out provides a reference 10 MHz output signal. This port typically connects to the E6404A IF processor Ref In port. This ensures the ADC sample clock is locked to the system reference.

3rd LO Out is typically used for a tuner system that requires conversion to baseband frequencies.

BD LO Output ports (2) each provide a 1200 MHz (approximately) signal to the E6403A BD LO Input port (when the E6403A Option 003 is present).

2nd LO Output ports (2) each provide a 1200 MHz (approximately) signal that connects to the E6401A 2nd LO Input port.

1st LO Output ports (2) each provide a 1241.4 to 2221.4 MHz (approximately) signal that connects to the E6401A 1st LO Input port.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

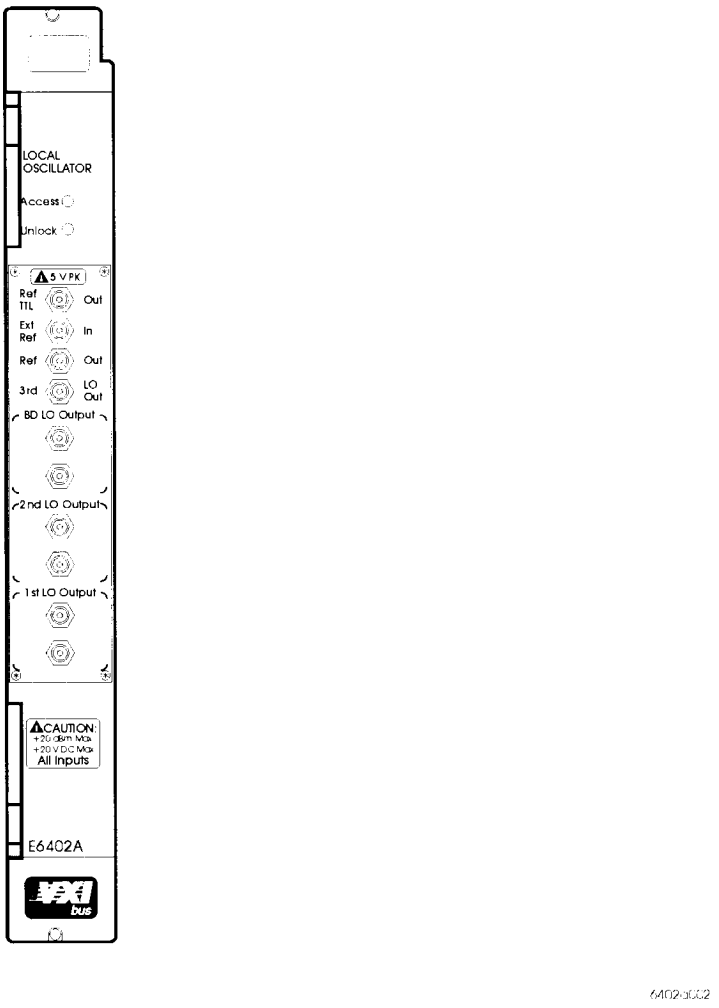


Figure 1-3 E6402A Option 002 Local Oscillator with Dual Outputs Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6403A 1000–3000 MHz Block Downconverter

Access LED flashes whenever the module is being accessed via the VXI backplane.

BD LO Input port connects to the E6402A BD LO Output port. This signal is approximately 1200 MHz.

Block Downconv Output port connects to the E6401A Block Downconv Input port that ranges, in bands, from approximately 250 to 900 MHz.

20–1000 MHz Output port connects to the E6401A 20–1000 MHz Input port.

20–3000 MHz Input is the receiver input port when using an Option 003 configuration to extend the frequency coverage range to 3 GHz.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

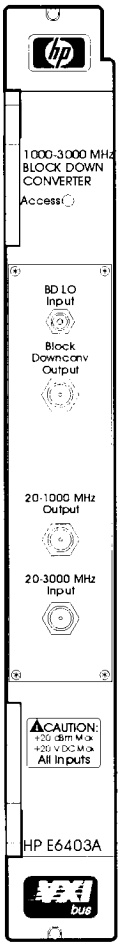


Figure 1-4 E6403A 1000–3000 MHz Block Downconverter Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6404A IF Processor

Access LED flashes whenever the module is being accessed via the VXI backplane.

Ch 1 IF In is the IF input port that connects to the 21.4 MHz IF Output port of the E6401A 20-1000 MHz downconverter.

Ref In port connects to the E6402A Ref Out port.

Link Port 1
Link Port 2 are the digital data output ports that are capable of providing I/Q data from the DDCs and raw A/D data. The data in each link port is sampled at 28.5333 MSa/s. Raw A/D data requires both link ports.

The eight pin connector provides six signals plus two grounds (4 bits of data + clock + ack => 6 signals).

Audio/Trigger is the analog audio output port. The standard E6404A IF processor used in the E6501A receiver is capable of one audio output signal. By installing Option 025, the E6404A IF processor is capable of five audio output signals. The audio output also has driver selectable FM de-emphasis.

This port also provides trigger input or output. Trigger input is utilized to initiate data output flow. Trigger output is currently not used. The trigger can be an active low or active high.

The trigger output is a 50 ohm output. The trigger input is a Schmitt trigger input with a minimum pulse width of 105 ns.

The trigger input allows you to trigger multiple IF processors in a mainframe simultaneously by using a single trigger signal input to the master IF processor. This signal is buffered into the VXI backplane and distributed to all slave IF processors. This function is useful for triggering synchronous data collection, or synchronous operation of the DDCs.

Ref Out port provides a 10 MHz reference output signal.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

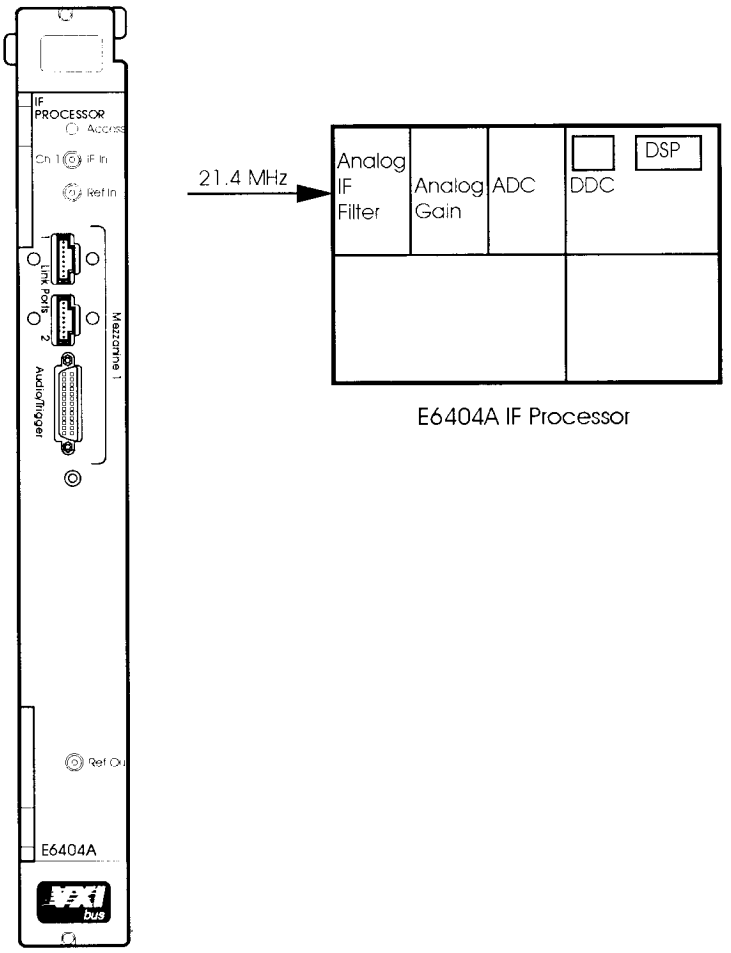


Figure 1-5 E6404A IF Processor Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6404A Option 031, 040 IF Processor

Access

LED flashes whenever the module is being accessed via the VXI backplane.

Ch 1 IF In

is the channel 1 IF input port that connects to the 21.4 MHz IF Output port of one of the E6401A 20-1000 MHz downconverters.

Ref In

port connects to the E6402A Ref Out port.

Link Port 1
Link Port 2
(IF Channel 1)

are the digital data output ports that are capable of providing I/Q data from the DDCs and raw A/D data. The data in each link port is sampled at 28.5333 MSa/s. Raw A/D data requires both link ports.

The eight pin connector provides six signals plus two grounds (4 bits of data + clock + ack => 6 signals).

Audio/Trigger
(IF Channel 1)

is the analog audio output port. The E6404A Option 031 and Option 040 IF processor used in the E6502A receiver is capable of one audio output signal for IF Channel 1 and one audio output signal for IF Channel 2. By installing Option 025, the E6404A IF processor is capable of five audio output signals for channel 1. The audio output also has driver selectable FM de-emphasis.

This port also provides trigger input or output. Trigger input is utilized to initiate data output flow. Trigger output is currently not used. The trigger can be an active low or active high.

The trigger output is a 50 ohm output. The trigger input is a Schmitt trigger input with a minimum pulse width of 105 ns.

The trigger input allows you to trigger multiple IF processors in a mainframe simultaneously by using a single trigger signal input to the master IF processor. This signal is buffered into the VXI backplane and distributed to all slave IF processors. This function is useful for triggering synchronous data collection, or synchronous operation of the DDCs.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Link Port 1
Link Port 2
(IF Channel 2)

are the digital data output ports that are capable of providing I/Q data from the DDCs and raw A/D data. The data in each link port is sampled at 28.5333 MSa/s. Raw A/D data requires both link ports.

The eight pin connector provides six signals plus two grounds (4 bits of data + clock + ack => 6 signals).

Audio/Trigger
(IF Channel 2)

is the analog audio output port. The E6404A Option 031 and Option 040 IF processor used in the E6502A receiver is capable of one audio output signal for IF Channel 1 and one audio output signal for IF Channel 2. By installing Option 035, the E6404A IF processor is capable of five audio output signals for channel 2. The audio output also has driver selectable FM de-emphasis.

This port also provides trigger input or output. Trigger input is utilized to initiate data output flow. Trigger output is currently not used. The trigger can be an active low or active high.

The trigger output is a 50 ohm output. The trigger input is a Schmitt trigger input with a minimum pulse width of 105 ns.

The trigger input allows you to trigger multiple IF processors in a mainframe simultaneously by using a single trigger signal input to the master IF processor. This signal is buffered into the VXI backplane and distributed to all slave IF processors. This function is useful for triggering synchronous data collection, or synchronous operation of the DDCs.

Ref Out

port provides a 10 MHz reference output signal.

Ch 2 IF In

is the channel 2 IF input port that connects to the 21.4 MHz IF Output port of one of the E6401A 20-1000 MHz downconverters.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

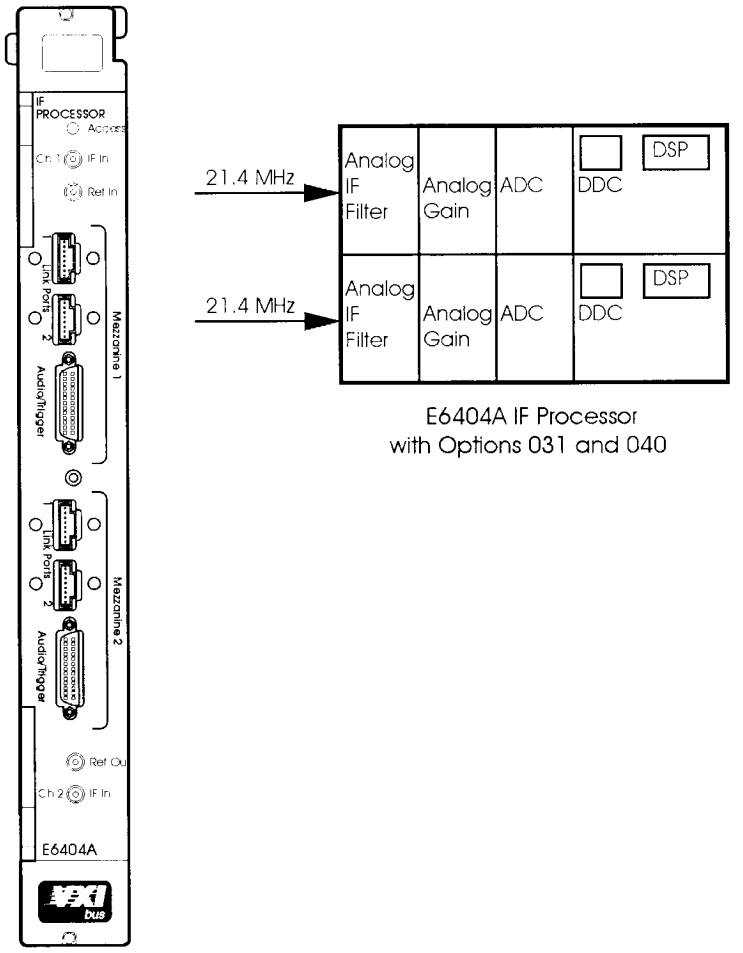


Figure 1-6 E6404A Option 031, 040 IF Processor Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6404A Option 022, 040 IF Processor

Access — LED flashes whenever the module is being accessed via the VXI backplane.

Ch 1 IF In — is the channel 1 IF input port that connects to the 21.4 MHz IF Output port of one of the E6401A 20-1000 MHz downconverters.

Ref In — port connects to the E6402A Ref Out port.

Link Port 1
Link Port 2
(IF Channel 1) — are the digital data output ports that are capable of providing I/Q data from the DDCs and raw A/D data. The data in each link port is sampled at 28.5333 MSa/s. Raw A/D data requires both link ports.

The eight pin connector provides six signals plus two grounds (4 bits of data + clock + ack => 6 signals).

Audio/Trigger
(IF Channel 1) — is the analog audio output port. The E6404A Option 022 and Option 040 IF processor used in the E6503A receiver is capable of two audio output signals for IF Channel 1. By installing Option 025, the E6404A IF processor is capable of five audio output signals. The audio output also has driver selectable FM de-emphasis.

This port also provides trigger input or output. Trigger input is utilized to initiate data output flow. Trigger output is currently not used. The trigger can be an active low or active high.

The trigger output is a 50 ohm output. The trigger input is a Schmitt trigger input with a minimum pulse width of 105 ns.

The trigger input allows you to trigger multiple IF processors in a mainframe simultaneously by using a single trigger signal input to the master IF processor. This signal is buffered into the VXI backplane and distributed to all slave IF processors. This function is useful for triggering synchronous data collection, or synchronous operation of the DDCs.

Ref Out — port provides a 10 MHz reference output signal.

Ch 2 IF In — is the channel 2 IF input port that connects to the 21.4 MHz IF Output port of one of the E6401A 20-1000 MHz downconverters.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

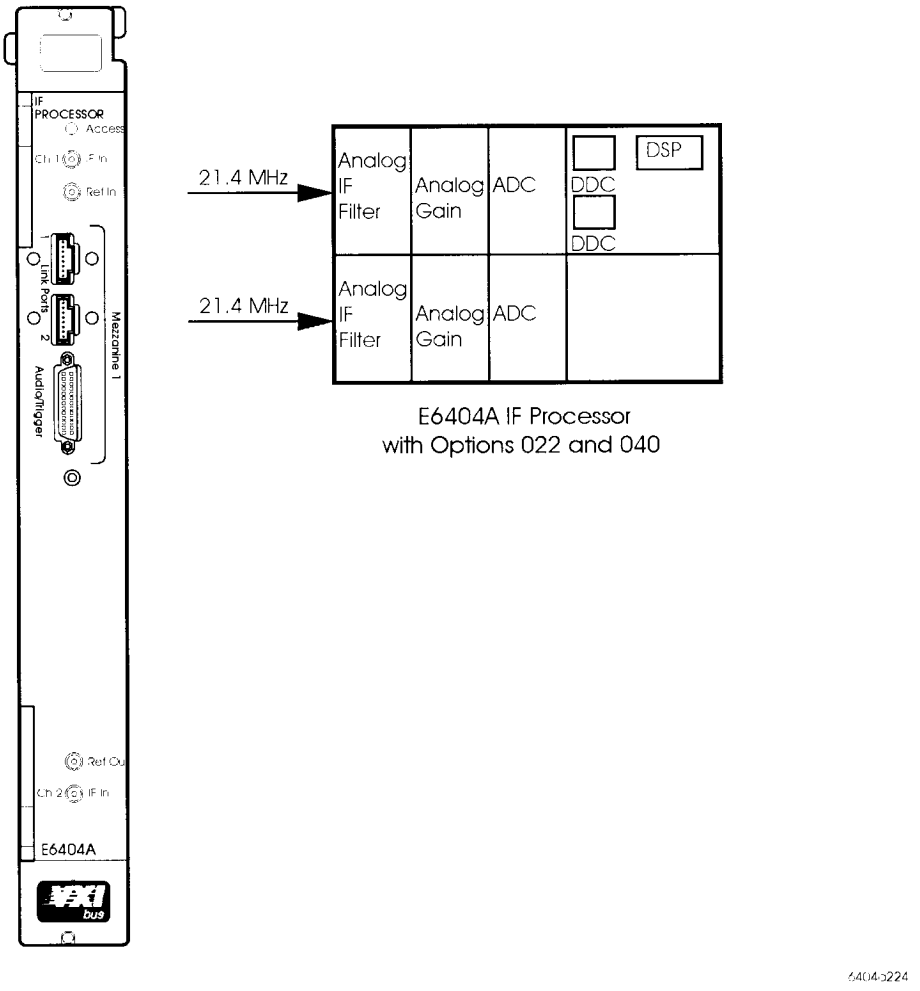


Figure 1-7 E6404A Option 022, 040 IF Processor Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Standard Receiver Configurations

In this section, a front-panel illustration and brief description is given for each of the three standard receivers described in this manual. Each front-panel illustration depicts module position and receiver cabling configuration.

E650XA Mainframe Options

- E650XA Option 006 (6 slot mainframe)
- E650XA Option 013 (13 slot mainframe)

Note

- Systems are *not* shipped cabled. To account for the large number of potential receiver configurations, it is possible to have extra cables when finished configuring your system.
- E6500A mainframe Options 006 and 013 may be used with any of the standard receiver configurations. Note that the E6502A Option 003 configuration must use the Option 013 mainframe because it requires seven slots.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6501A Configuration

The E6501A receiver consists of three single-slot C-size VXI modules:

- E6402A local oscillator
- E6401A 20–1000 MHz downconverter
- E6404A IF processor

If you are upgrading from an E6500A VXI tuner to a receiver configuration, and you wish to use the Ref TTL Out signal and connect it to the EXT CLK connector on the MXI controller card, note the following:

- You must have an E6402A LO module with a serial prefix number of 3822 or greater.
- You must set up the MXI controller card to use an external clock signal.

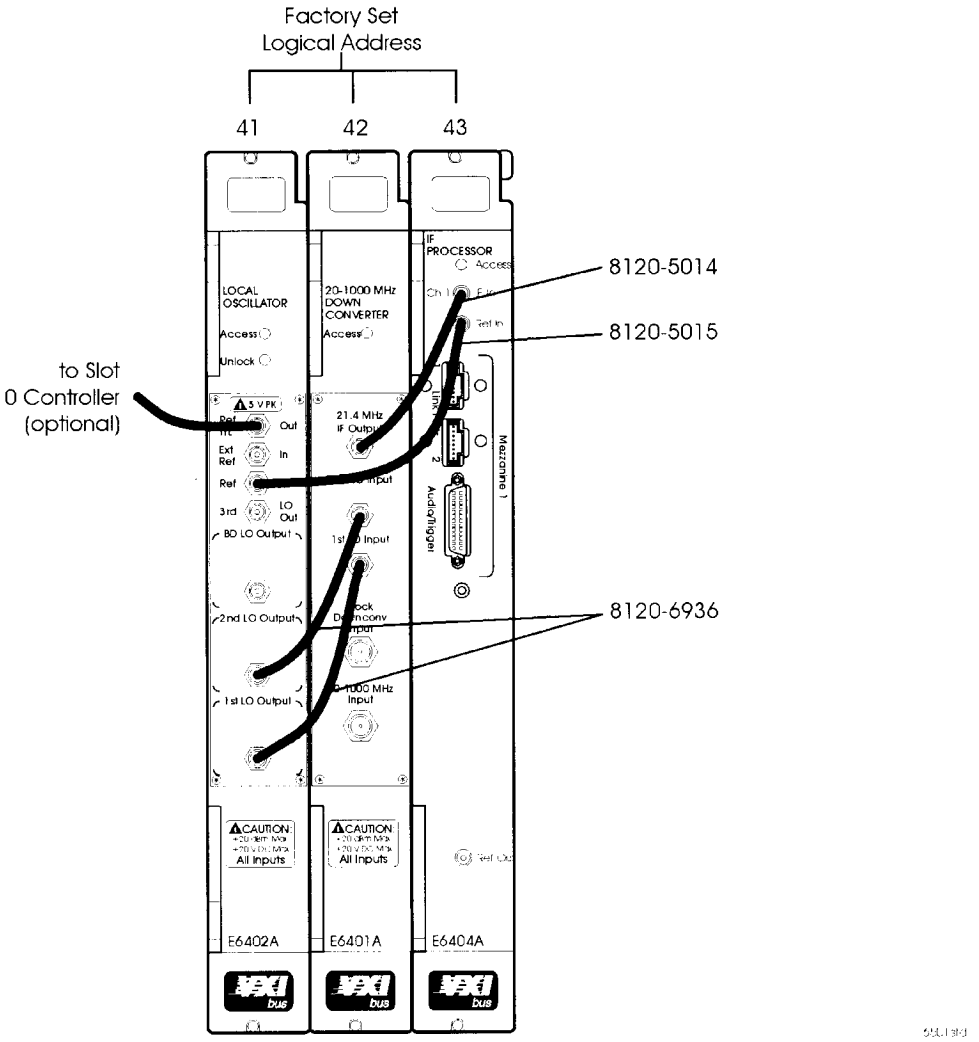


Figure 1-8 E6501A Receiver Configuration

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6501A Option 003 Configuration

The E6501A Option 003 receiver consists of four single-slot C-size VXI modules:

- E6402A local oscillator
- E6401A 20–1000 MHz downconverter
- E6403A 1000–3000 MHz block downconverter
- E6404A IF processor

If you are upgrading from an E6500A VXI tuner to a receiver configuration, and you wish to use the Ref TTL Out signal and connect it to the EXT CLK connector on the MXI controller card, note the following:

- You must have an E6402A LO module with a serial prefix number of 3822 or greater.
- You must set up the MXI controller card to use an external clock signal.

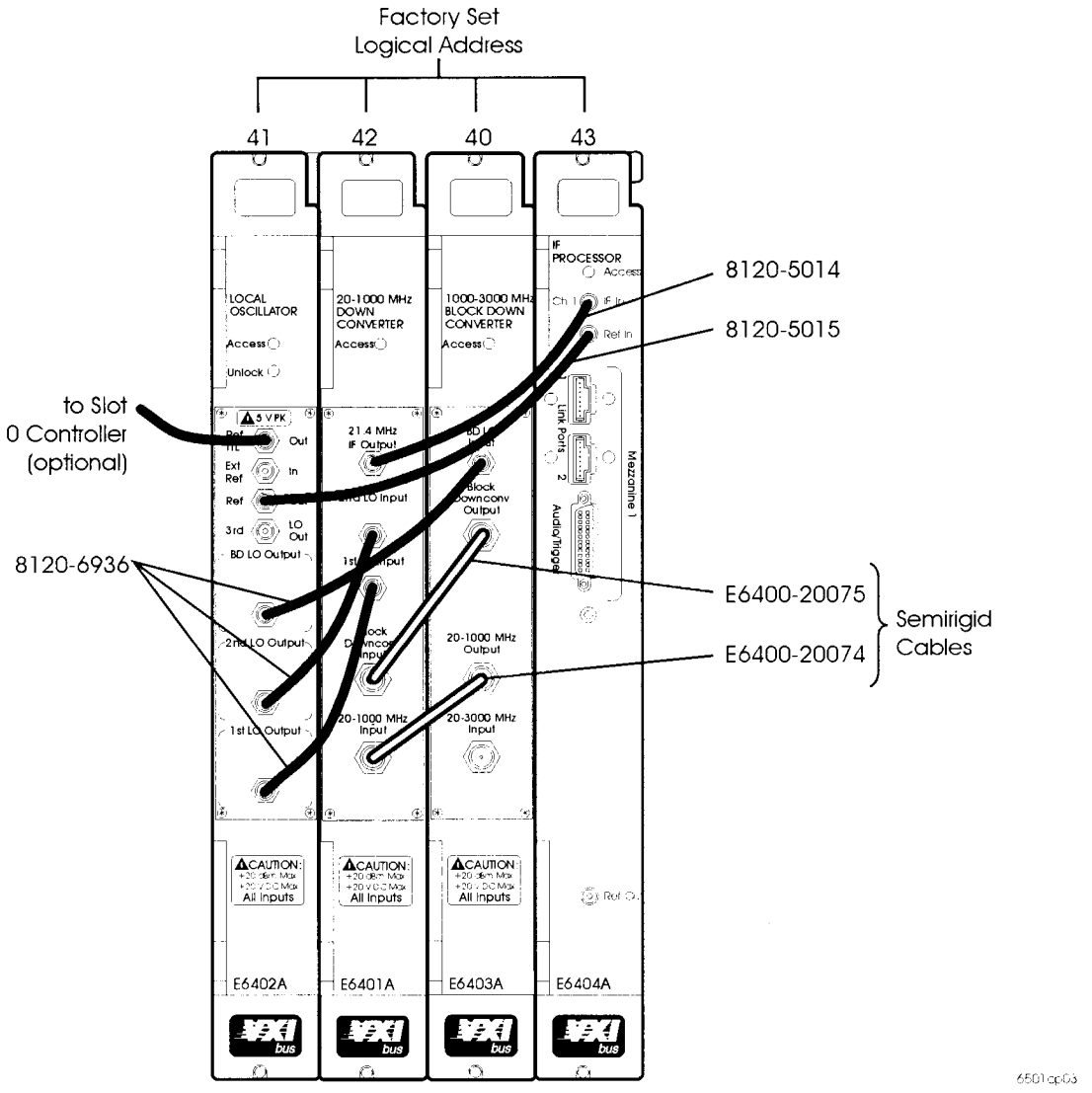


Figure 1-9 E6501A Option 003 Receiver Configuration

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6502A Configuration

The E6502A dual channel receiver consists of five single-slot C-size VXI modules:

- E6402A local oscillator (2)
- E6401A 20–1000 MHz downconverter (2)
- E6404A IF processor with Option 031 (mezzanine #2) and Option 040 (IF channel #2)

If you are upgrading from an E6500A VXI tuner to a receiver configuration, and you wish to use the Ref TTL Out signal and connect it to the EXT CLK connector on the MXI controller card, note the following:

- You must have an E6402A LO module with a serial prefix number of 3822 or greater.
- You must set up the MXI controller card to use an external clock signal.

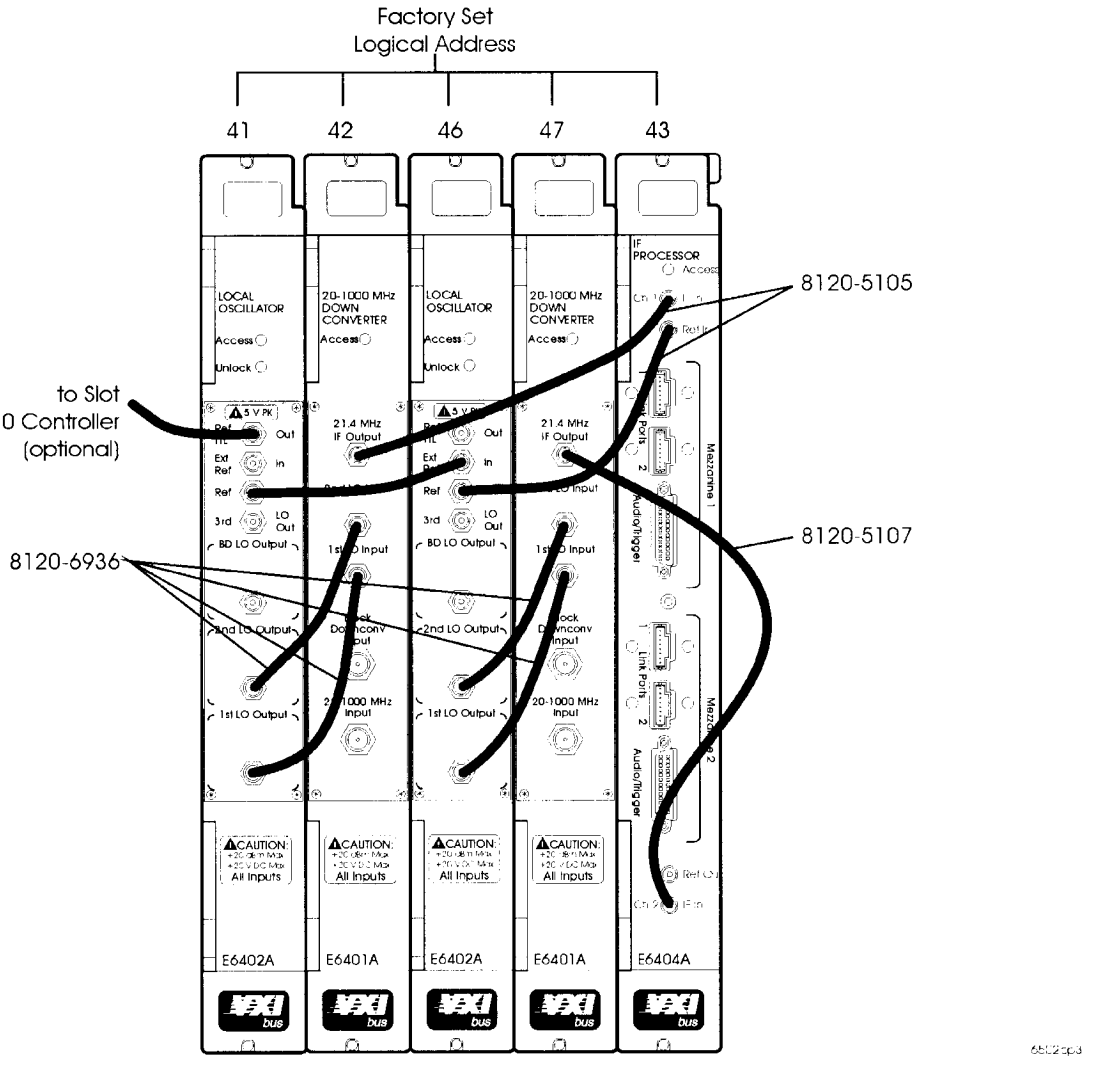


Figure 1-10 E6502A Dual Channel Receiver Configuration (independent LOs)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6502A Option 003 Configuration

The E6502A Option 003 dual channel receiver consists of seven single-slot C-size VXI modules:

- E6402A local oscillator (2)
- E6401A 20–1000 MHz downconverter (2)
- E6403A 1000–3000 MHz block downconverter (2)
- E6404A IF processor with Option 031 (mezzanine #2) and Option 040 (IF channel #2)

If you are upgrading from an E6500A VXI tuner to a receiver configuration, and you wish to use the Ref TTL Out signal and connect it to the EXT CLK connector on the MXI controller card, note the following:

- You must have an E6402A LO module with a serial prefix number of 3822 or greater.
- You must set up the MXI controller card to use an external clock signal.

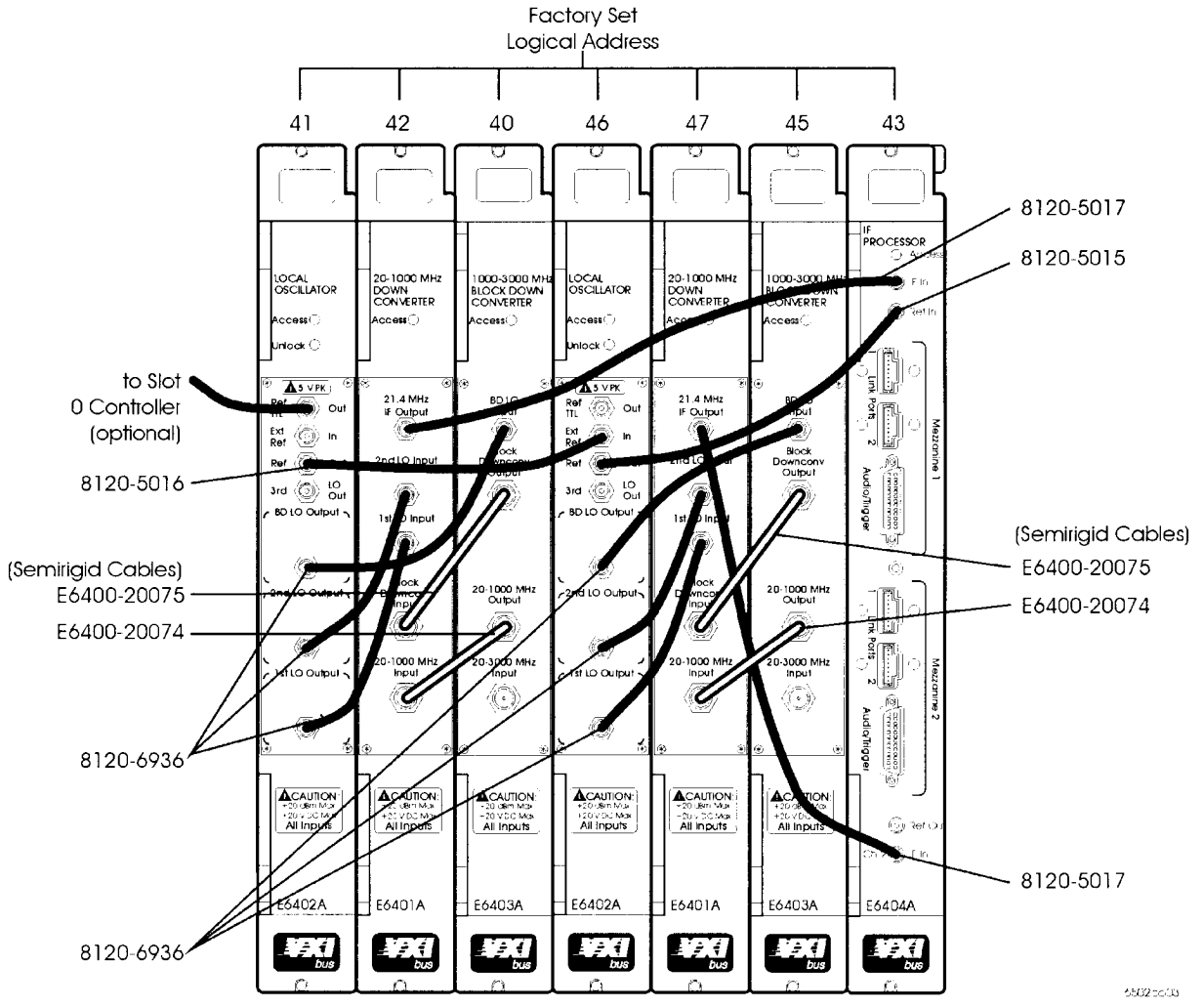


Figure 1-11 E6502A Option 003 Dual Channel Receiver Configuration (independent LOs)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6503A Configuration

The E6503A dual channel receiver consists of four single-slot C-size VXI modules:

- E6402A Option 002 local oscillator
- E6401A 20–1000 MHz downconverter (2)
- E6404A IF processor with Option 022 (DDC #2 on mezzanine #1) and Option 040 (IF channel #2)

If you are upgrading from an E6500A VXI tuner to a receiver configuration, and you wish to use the Ref TTL Out signal and connect it to the EXT CLK connector on the MXI controller card, note the following:

- You must have an E6402A LO module with a serial prefix number of 3822 or greater.
- You must set up the MXI controller card to use an external clock signal.

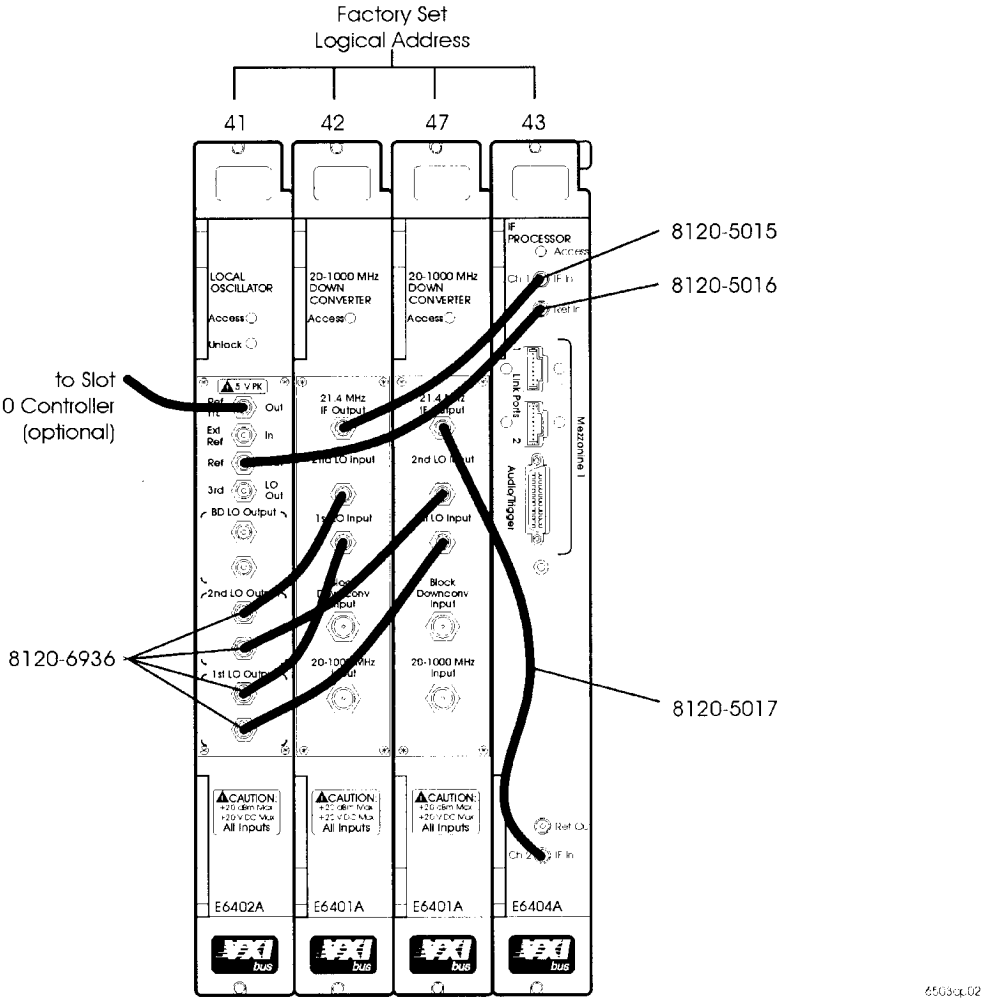


Figure 1-12 E6503A Dual Channel Receiver Configuration (shared LOs)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6503A Option 003 Configuration

The E6503A Option 003 dual channel receiver consists of six single-slot C-size VXI modules:

- E6402A Option 002 local oscillator
- E6401A 20–1000 MHz downconverter (2)
- E6403A 1000–3000 MHz block downconverter (2)
- E6404A IF processor with Option 022 (DDC #2 on mezzanine #1) and Option 040 (IF channel #2)

If you are upgrading from an E6500A VXI tuner to a receiver configuration, and you wish to use the Ref TTL Out signal and connect it to the EXT CLK connector on the MXI controller card, note the following:

- You must have an E6402A LO module with a serial prefix number of 3822 or greater.
- You must set up the MXI controller card to use an external clock signal.

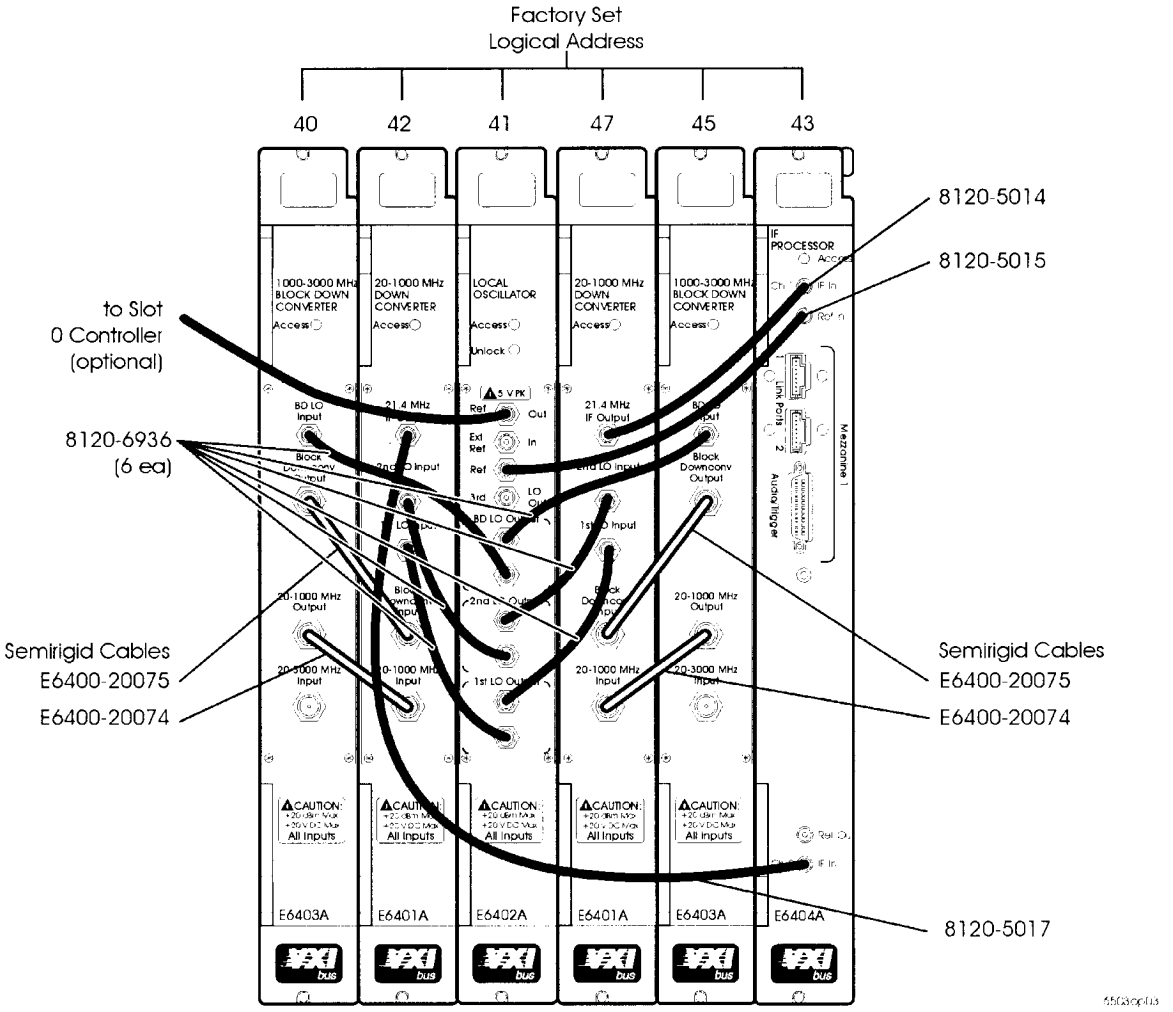


Figure 1-13 E6503A Option 003 Dual Channel Receiver Configuration (shared LOs)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Accessories Supplied

Table 1-2 E6401A 20–1000 MHz Downconverter Accessories

Description	Quantity	Part Number
Coaxial cable, SMC (f)/SMC (f), 120 cm	2	8120-6936

Table 1-3 E6401A Option 001 20–1000 MHz Downconverter Accessories

Description	Quantity	Part Number
Coaxial cable, SMC (f)/SMC (f), 120 cm	2	8120-6936
Coaxial cable, SMB (f)/SMB (f), 120 cm	1	8120-5015

Table 1-4 E6402A Local Oscillator Accessories

Description	Quantity	Part Number
Coaxial cable[1],SMB (f)/SMB (f), 205 cm	1	8120-5017

1. This cable connects the Ref TTL Out connector to the slot 0 controller. This connection is optional.

Table 1-5 E6402A Option 002 Local Oscillator Accessories

Description	Quantity	Part Number
Coaxial cable[1], SMB (f)/SMB (f), 205 cm	1	8120-5017

1. This cable connects the Ref TTL Out connector to the slot 0 controller. This connection is optional.

Table 1-6 E6403A 1000–3000 MHz Block Downconverter Accessories

Description	Quantity	Part Number
Coaxial cable, SMC (f)/SMC (f), 120 cm	1	8120-6936
Semirigid cable, SMA (m)/SMA (m)	1	E6400-20074
Semirigid cable, SMA (m)/SMA (m)	1	E6400-20075
Semirigid cable, SMA (m)/SMA (m)	1	E6400-20088
Semirigid cable,SMA (m)/SMA (m)	1	E6400-20089

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 1-7 E6404A IF Processor Accessories

Description	Quantity	Part Number
Coaxial cable, SMB (f)/SMB (f), 100 cm	1	8120-5014
Coaxial cable, SMB (f)/SMB (f), 120 cm	1	8120-5015
Coaxial cable, SMB (f)/SMB (f), 160 cm	1	8120-5016
Coaxial cable, SMB (f)/SMB (f), 205 cm	1	8120-5017
User's Guide	1	E6500-90006
Receiver Driver Software	1	E6500-10003

Table 1-8 E6404A Option 031 and 040 IF Processor Accessories

Description	Quantity	Part Number
Coaxial cable, SMB (f)/SMB (f), 100 cm	1	8120-5014
Coaxial cable, SMB (f)/SMB (f), 120 cm	1	8120-5015
Coaxial cable, SMB (f)/SMB (f), 160 cm	1	8120-5016
Coaxial cable, SMB (f)/SMB (f), 205 cm	1	8120-5017
User's Guide	1	E6500-90006
Receiver Driver Software	1	E6500-10003

Table 1-9 E6404A Option 022 and 040 IF Processor Accessories

Description	Quantity	Part Number
Coaxial cable, SMB (f)/SMB (f), 100 cm	1	8120-5014
Coaxial cable, SMB (f)/SMB (f), 120 cm	1	8120-5015
Coaxial cable, SMB (f)/SMB (f), 160 cm	1	8120-5016
Coaxial cable, SMB (f)/SMB (f), 205 cm	1	8120-5017
User's Guide	1	E6500-90006
Receiver Driver Software	1	E6500-10003

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2 Getting Started

In This Chapter

- Electrostatic Discharge
- Preparation for Use
- Checking Operation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Electrostatic Discharge Information

Electrostatic discharge (ESD) can damage or destroy electronic components. All work on electronic assemblies should be performed at a static-safe work station. The following figure shows an example of a static-safe work station using two types of ESD protection:

- Conductive table-mat and wrist-strap combination.
- Conductive floor-mat and heel-strap combination.

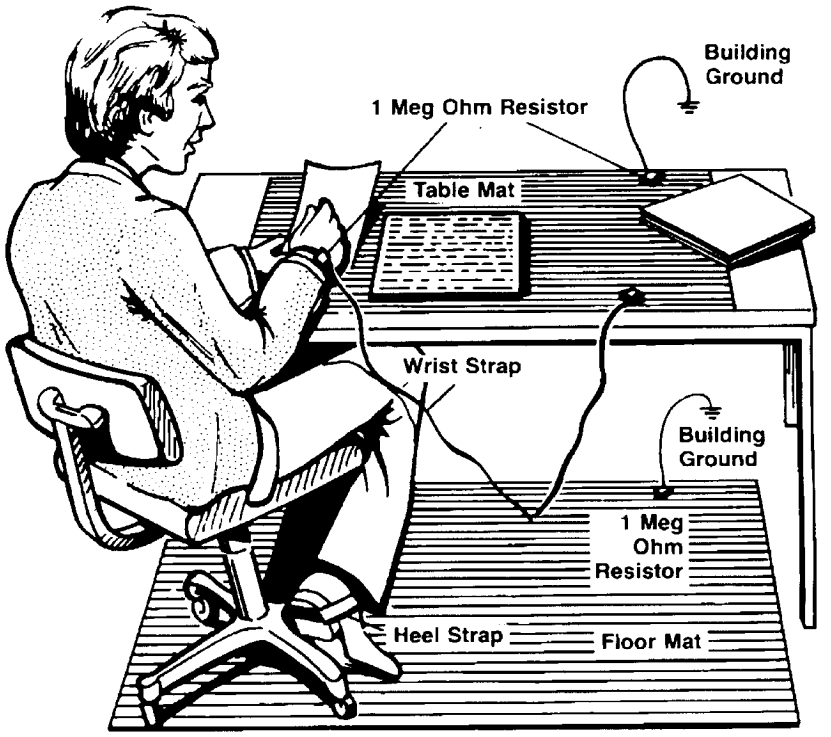


Figure 2-1 Example of a Static-Safe Work Station

Both types, when used together, provide a significant level of ESD protection. Of the two, only the table-mat and wrist-strap combination provides adequate ESD protection when used alone.

To ensure user safety, the static-safe accessories must provide at least 1 MΩ of isolation from ground. Refer to Table 2-1 for information on ordering static-safe accessories.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

WARNING **These techniques for a static-safe work station should not be used when working on circuitry with a voltage potential greater than 500 volts.**

Table 2-1 Static-Safe Accessories

Part Number	Description
9300-0797	3M static control mat 0.6 m × 1.2 m (2 ft × 4 ft) and 4.6 cm (15 ft) ground wire. (The wrist-strap and wrist-strap cord are not included. They must be ordered separately.)
9300-0980	Wrist-strap cord 1.5 m (5 ft).
9300-1383	Wrist-strap, color black, stainless steel, without cord, has four adjustable links and a 7 mm post-type connection.
9300-1169	ESD heel-strap (reusable 6 to 12 months).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Preparation for Use

This procedure describes how to configure and install C-size VXI modules into a C-size VXIbus mainframe by using the following steps:

1. Setting the logical address switches
2. Installing the receiver
3. Cabling the receiver

WARNING

SHOCK HAZARD. Only service-trained personnel who are aware of the hazards involved should install, remove, or configure the system. Before you perform any procedures in this guide, disconnect ac power and field wiring from the mainframe.

To avoid electrical shock, always cover unused slots with the faceplate panels that came with the mainframe.

CAUTION

- STATIC ELECTRICITY. Static electricity is a major cause of component failure. To prevent damage to the electrical components in the mainframe and plug-in modules, observe anti-static techniques whenever handling a module.
- It is your responsibility to ensure that adequate cooling is supplied to all modules installed in the mainframe. Section B.7.2.4 of the VXIbus Specification (Revision 1.3) discusses module cooling requirements. Section B.7.3.5 discusses mainframe cooling requirements.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Setting the Logical Address Switches

Each module is shipped with a factory-set logical address as shown in the following table. Each module installed in a mainframe must have a unique logical address. If there is more than one module with the same factory-set logical address (for example, more than one module of the same model), you must change the logical address of each additional module. This can be accomplished by incrementing the factory-set logical address number by five for each additional module of the same model.

Table 2-2 Logical Address Settings

Model	Factory-Set Logical Address	Secondary Address Setting
E6401A	42	47
E6402A	41	46
E6402A Option 002	41	46
E6403A	40	45
E6404A	43	48
E6404A Option 031, 040	43	48
E6404A Option 022, 040	43	48

Procedure

Follow the steps and refer to Table 2-2 to set each module's logical address:

1. Locate the logical address switch on each module.
2. Set the module's logical address using the following guidelines:
   a. Use the factory-set logical address. If you have modules with the same logical address, change the address of one or more of those modules until all modules have different logical addresses.
   b. If multiple mainframes are connected via MXIbus, make sure the logical addresses of the modules in the mainframe are within the logical address window for that mainframe.
   c. Valid logical addresses are 1 through 255. Most Agilent Technologies modules are statically configured modules, meaning you have to physically set the address on a switch. A dynamically configured module address is set programmatically by the resource manager. To dynamically configure a module that supports dynamic configuration, its logical address must be set to 255. Note, however, that if a statically configured module is set to 255, the resource manager will not dynamically configure any module.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

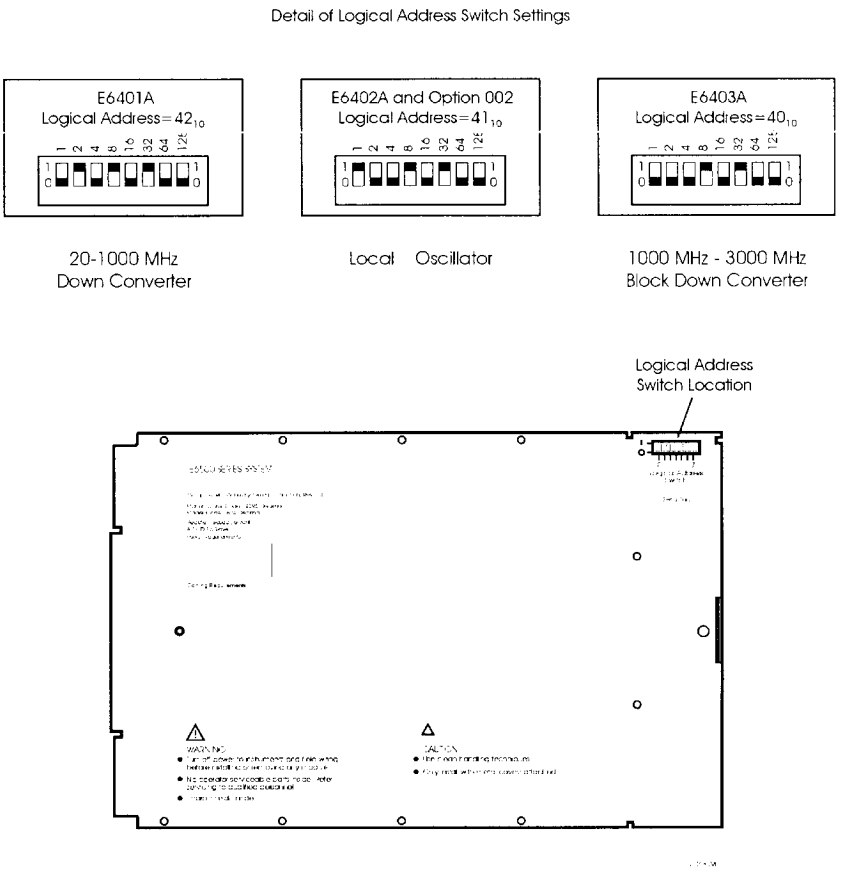


Figure 2-2 E6401A/E6402A/E6403A Side Cover with Logical Address Switch Location and Address Switch Configurations for each Model of Module

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

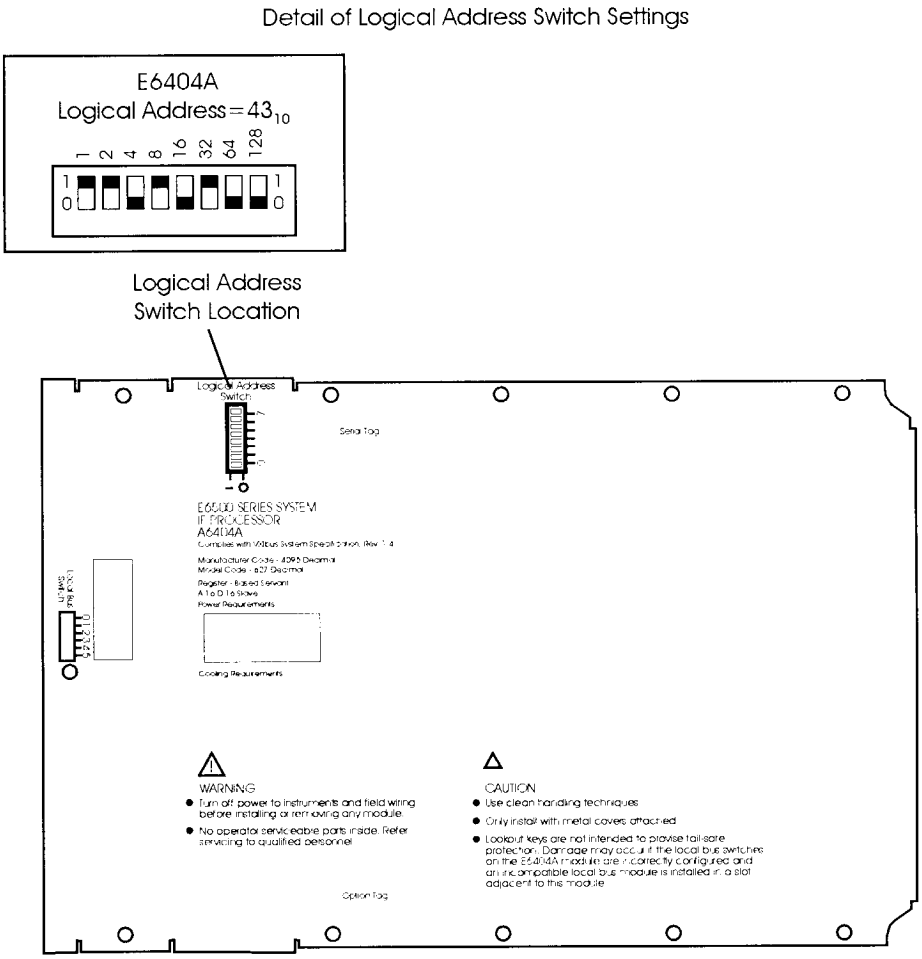


Figure 2-3 E6404A Side Cover with Logical Address Switch Location and Address Switch Configuration

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Local Bus Compatibility

The E6404A is shipped with three top logo bases. Two of these bases prevent incompatible modules from being placed into an adjacent slot to the left or right of the keyed module. The top logo bases (also called lockout keys) are fitted at the top of the module. Refer to Figure 2-4 for the keys that are shipped with the E6404A.

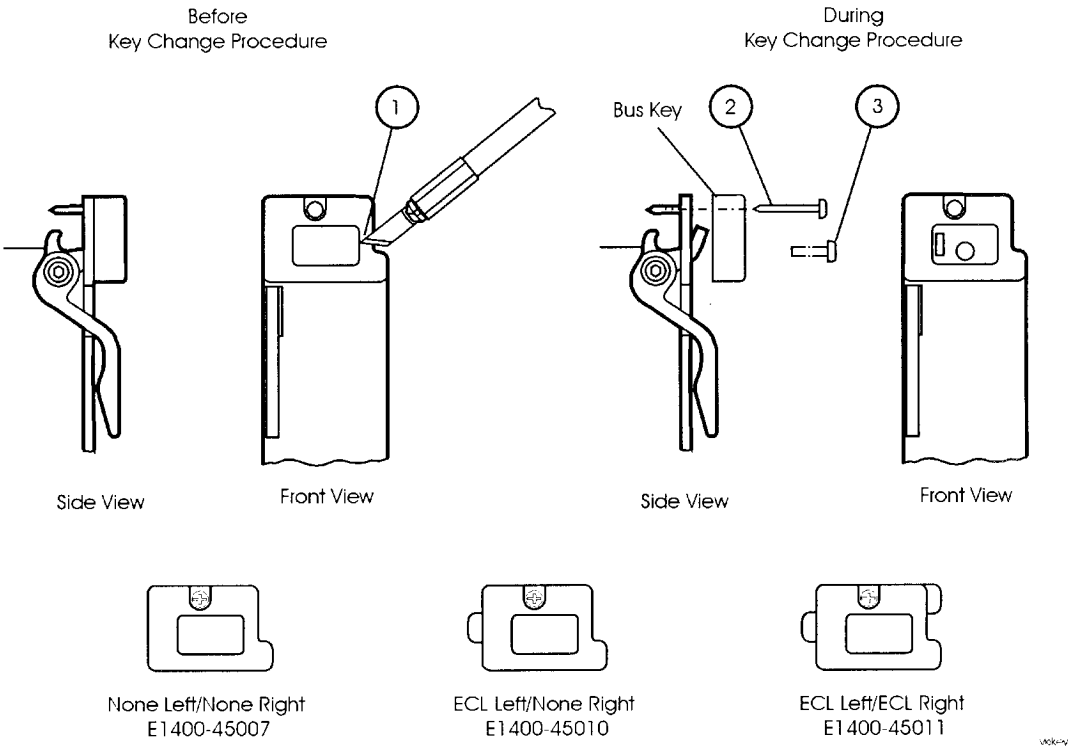


Figure 2-4 Top Logo Base

The local bus links modules together in a type of "daisy chain." When configuring a series of modules for local bus operation, you must determine whether a module is ECL or TTL compatible. The E6404A is ECL compatible. Compatible Agilent Technologies modules are keyed and allowed to be configured side-by-side to form the local bus chain. Incompatible modules cannot be placed side-by-side into the VXI mainframe for local bus operation.

Note

The local bus described here is not the same as the local bus used by Agilent Technologies Lake Stevens Division for their data flow.

The local bus switch is shipped with all switches set to 0, which means that the local bus is disconnected to the left and to the right in the chain. Refer to Figure 2-3 for the location of the local bus switch. Refer also to Figure 3-24 for a diagram of the switches and various configuration settings.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 2-3 describes the terms located next to the local bus switch on the side panel of the E6404A.

Table 2-3 Local Bus Switch Terminology

Term	Description
Sync Tx	ECL to the right; disconnected to the left.
Buffered Sync Rx[1]	Transmitting ECL to the left and right.
Unbuffered Sync Rx	Transmitting ECL to the left and right.
Terminal Synch Rx	ECL to the left; disconnected to the right.
Sole Module	Transmitting between the two mezzanines. Disconnected to the left and to the right.
All switches set to 0	Disconnected to the left and to the right.

1. Buffered Sync is recommended approximately every fourth IFP. This provides greater fan-out from the ECL source. This is an approximate recommendation, because the loading is also a function of the mainframe. Buffered Sync will add a 1 to 2 nanosecond time delay.

Installing the Receiver

The modules can be installed in any slot except 0. The modules should be installed in the order shown for that system configuration so the cables can be correctly connected between the modules.

Use the following steps to install the receiver:

1. Set up the VXI mainframe. See the installation guide for your Agilent Technologies C-size VXIbus mainframe. If the mainframe is turned on, turn it off by pressing the button in the lower-left front corner.
2. Select the correct slot for each module. See the systems illustrations in the "Standard Receiver Configurations" section in Chapter 1.
3. With the extractor handles lifted, insert each module into the mainframe by aligning the top and bottom of the module with the card guides inside the mainframe. Slowly push the module straight into the slot until it seats in the backplane connectors. When installing modules in the E1421B mainframe, the "top" of a module will be on the left when it is installed horizontally.
4. Secure the modules to the mainframe by pushing the extractor handles in until they lock in place.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Cabling the Receiver

Refer to the "Standard Receiver Configurations" section in Chapter 1 for cabling configuration illustrations.

10 MHz Reference

We recommend that:

- any external reference used for the E650XA have good phase noise and stability
- the slot 0 controller use the E6402A Ref TTL out signal to ensure optimum performance of the overall system

Installing the MXI Controller Cable

The MXI controller cable must be installed in a specific orientation for proper operation of the receiver/PC. Locate the connector closest to the label on the cable that reads:

CONNECT THIS END TO
DEVICE CLOSEST TO
MXIBUS CONTROLLER
IN THIS DAISY CHAIN

Connect this end to the PC.

Configuring a Multiple Mainframe System

Perform the following procedure to ensure proper communication between mainframes for a multiple mainframe configuration.

Note

Switch settings in the MXIbus card will also need to be changed for multiple mainframe operation. Refer to the National Instruments MXIbus User's Guide for information.

1. Click the **Start** button in the taskbar at the bottom of the screen.
2. Move the mouse pointer over the **Programs** command. The **Programs** menu appears.
3. Move the mouse pointer over the **Ni-vxi** command. The **Ni-vxi** menu appears.
4. Click the **T&M Explorer** item. Refer to Figure 2-5. The dialog box shown in Figure 2-6 appears.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

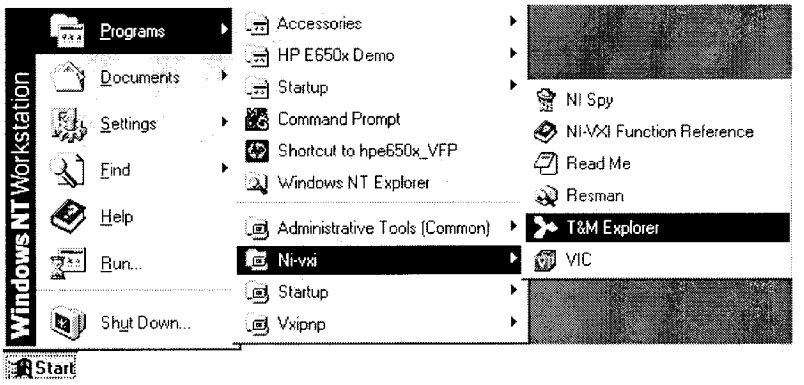


Figure 2-5 Starting the T&M Explorer

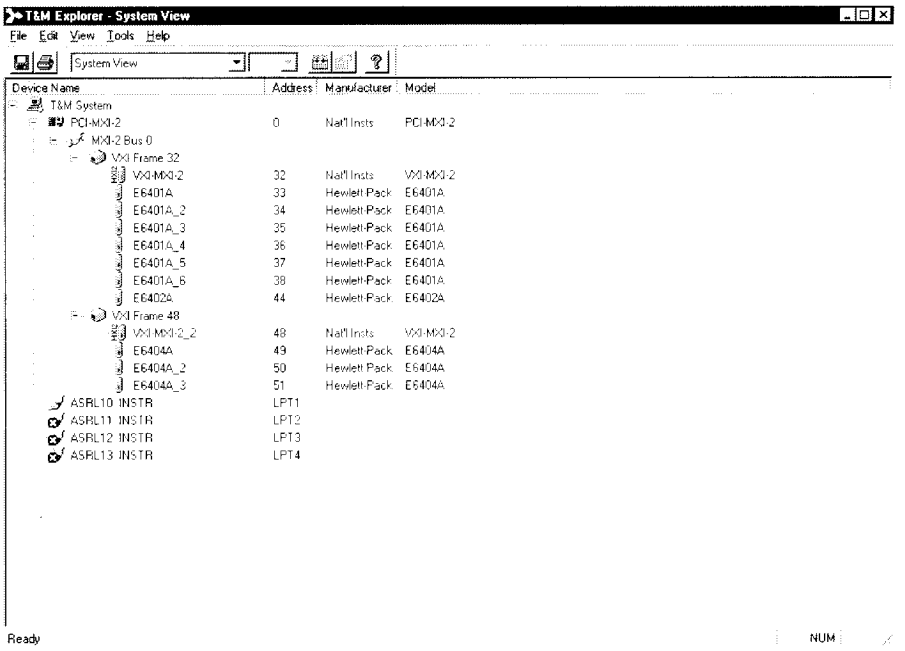


Figure 2-6 T&M Explorer

5. Double-click the **VXI-MXI-xx** item. The **VXI-MXI-xx Properties** dialog box appears as shown in Figure 2-7.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

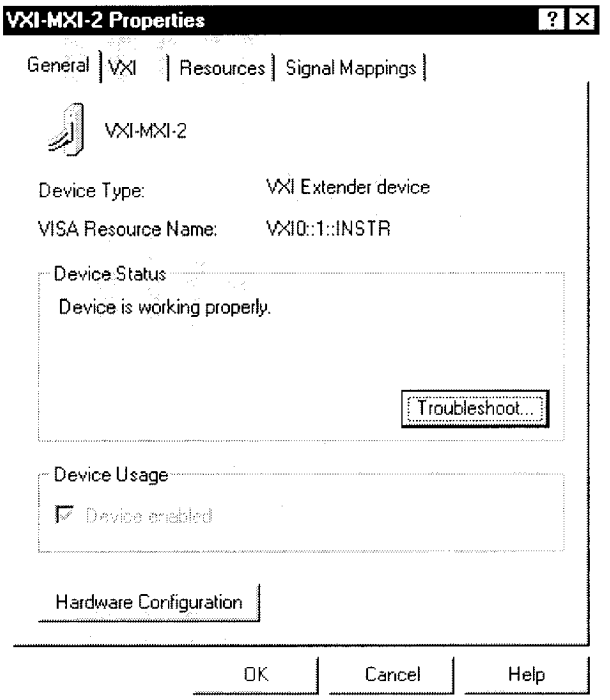


Figure 2-7 VXI-MXI-xx Properties

6. Click the **Hardware Configuration** button. The **Settings for VXI-MXI-xx** dialog box appears as shown in Figure 2-8.

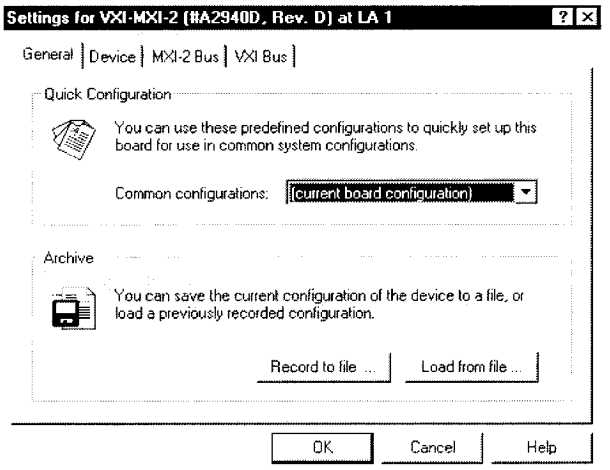


Figure 2-8 Settings for VXI-MXI-xx

7. Click the **Device** tab. Refer to Figure 2-9.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

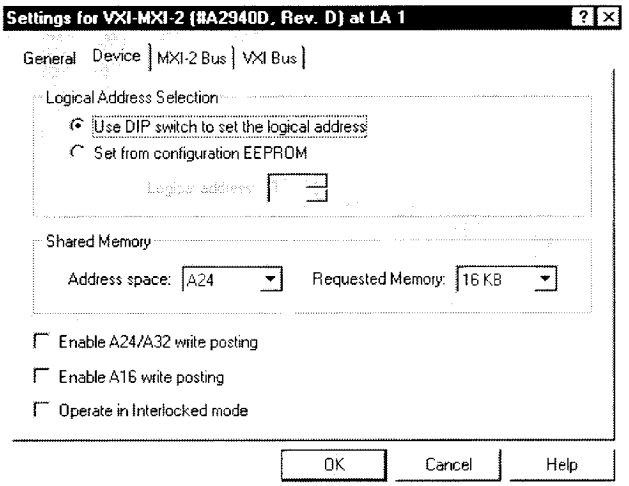


Figure 2-9 Device Tab

8. Select **Operate in Interlocked mode.**
9. Click the **OK** button in the **Device** tab.
10. Click the **OK** button in the **Settings for VXI-MXI-xx** dialog box.
11. Repeat step 5 through 10 for the other mainframe(s).
12. Close the **T&M Explorer** dialog box.

PC or UNIX Workstation System Requirements

Table 2-4 shows the minimum hardware and software requirements for a PC or UNIX workstation.

Table 2-4 System Requirements

	PC	UNIX Workstation
computer	pentium	HP 9000 Series 700 computer
operating system	Windows NT 3.5 or greater Windows 95 or greater	HP-UX 9 or greater
RAM	16 MB or greater	
interface card	PCI/VXI interface card	E1489 MXI interface card or equivalent

The driver software is VXI Plug&Play compatible. Source code is provided for the driver software. In order to use a UNIX workstation, you must convert the driver software.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Installing the Software

This section shows how to install the driver and virtual front panel demonstration software files onto a PC. The driver software does not run on a UNIX workstation.

Table 2-5 List of Installation Files

Component	File
Header Files	C:\Program Files\Hewlett-Packard\E650x\Include\hpe650x.h
Header Files	C:\Program Files\Hewlett-Packard\E650x\Include\hpe650x.hlp
Library Files	C:\Program Files\Hewlett-Packard\E650x\lib\hpe650x.lib
Library Files	C:\Program Files\Hewlett-Packard\E650x\lib\hpe650x.exp
Program Files	C:\Program Files\Hewlett-Packard\E650x\vfp\Demo.exe
DLL Files	C:\Windows\System32\hpe650x.dll

1. If you are running Windows NT®, log in as administrator, or as a user with administrator privileges.
2. Insert the E650XA CD into the CD-ROM drive.
3. Click the **Start** button in the taskbar at the bottom of the screen.
4. Move the mouse pointer over the **Settings** command. The **Settings** menu appears.
5. Click the **Control Panel** folder. The **Control Panel** appears.
6. Double-click the **Add/Remove Programs** icon. The **Add/Remove Programs Properties** dialog box appears.
7. In the **Install/Uninstall** tab, click the **Install** button. The **Install Program From Floppy Disk or CD-ROM** wizard appears.
8. Click the **Next** button. The install wizard will guide you through installation.

Configuring the VXI Bus Timeout

The VXI bus timeout defaults to 100 µs. In order to ensure that the driver software returns data and result codes for all situations, set the VXI bus timeout to 500 µs.

Note

The following procedure applies to the PC configuration and not UNIX.

1. Click the **Start** button in the taskbar at the bottom of the screen.
2. Move the mouse pointer over the **Programs** command. The **Programs** menu appears.
3. Move the mouse pointer over the **Ni-vxi** command. The **Ni-vxi** menu appears.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4. Click the **T&M Explorer** item. Refer to Figure 2-5. The dialog box shown in Figure 2-6 appears.
5. Double-click the **VXI-MXI-xx** item. Refer to Figure 2-7.
6. Click the **Hardware Configuration** button. Refer to Figure 2-8.
7. Click the **VXI** tab in the properties dialog box.
8. Change the bus timeout (BTO) parameter to 500 µs.
9. Click the **OK** button in the **VXI** tab.
10. Click the **OK** button in the **Settings for VXI-MXI-xx** dialog box.
11. Close the **T&M Explorer** dialog box.

Starting the Virtual Front Panel

The following procedure shows how to start the virtual front panel software on a PC running Windows NT® or Windows 95®. This procedure assumes that the receiver modules are installed, and that power to the VXI mainframe is turned on.

Note

The virtual front panel software does not run on a UNIX workstation. No source code is provided for the virtual front panel software. The virtual front panel is provided as a means to demonstrate some of the capabilities of the receiver hardware. It is not intended as a general purpose user interface. It is provided to help users get started before implementing their own software using the driver provided.

1. Click the **Start** button in the taskbar at the bottom of the screen.
2. Move the mouse pointer over the **Programs** command. The **Programs** menu appears.
3. Move the mouse pointer over the **Ni-vxi** command. The **Ni-vxi** menu appears.
4. Click the **Resman** item. Refer to Figure 2-10. The dialog box shown in Figure 2-11 appears. The Resman program must be run once after the E650XA or any VXI equipment has been turned on.
5. Click the **Close** button.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

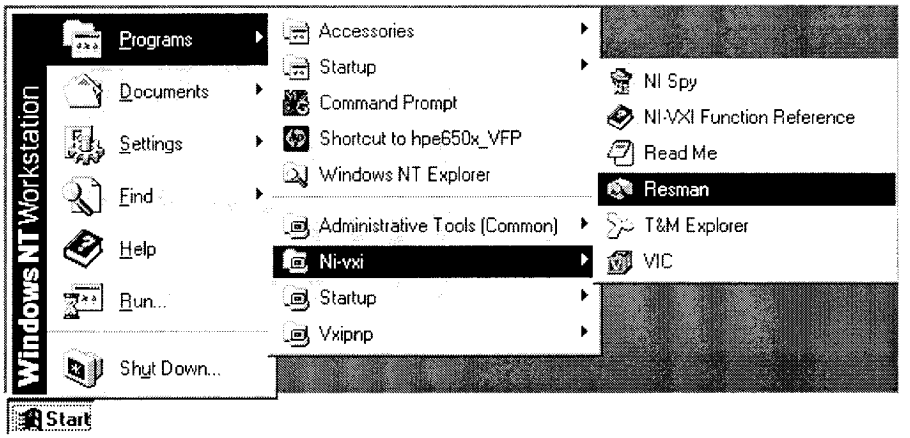


Figure 2-10 Running the Resource Manager

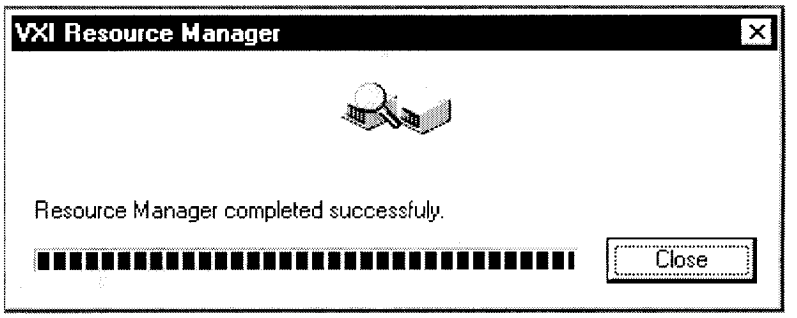


Figure 2-11 VXI Resource Manager

6. Click the **Start** button in the taskbar at the bottom of the screen.
7. Move the mouse pointer over the **Programs** command. The **Programs** menu appears.
8. Move the mouse pointer over the **HPE650X Demo Program** command. The **HPE650X Demo Program** menu appears.
9. Click the **Demo Virtual Front Panel** item.

   If your receiver is an E6501A, make sure the selections in your setup dialog box match those shown in Figure 2-12.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

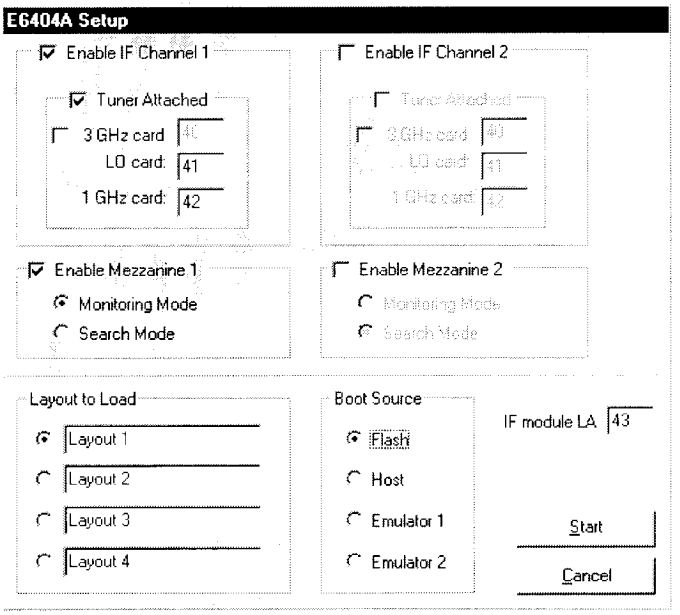


Figure 2-12 Setup for the E6501A

If your receiver is an E6502A, make sure the selections in your setup dialog box match those shown in Figure 2-13.

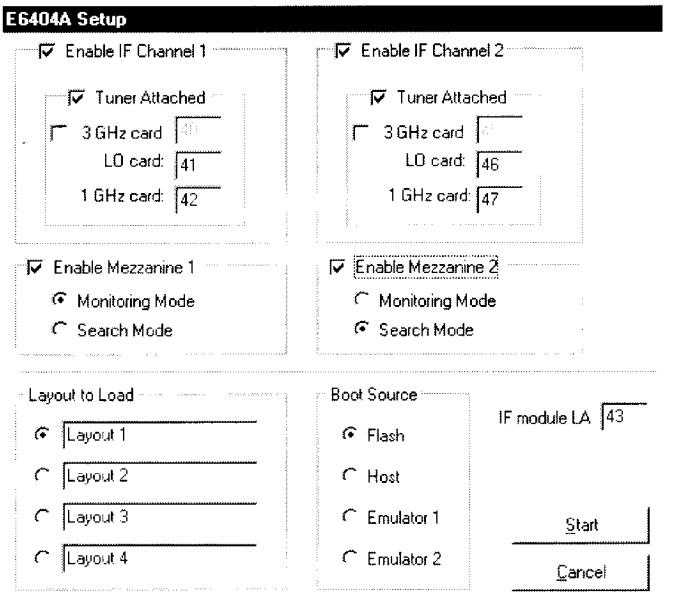


Figure 2-13 Setup for the E6502A

If your receiver is an E6503A, make sure the selections in your setup dialog box match those shown in Figure 2-14.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

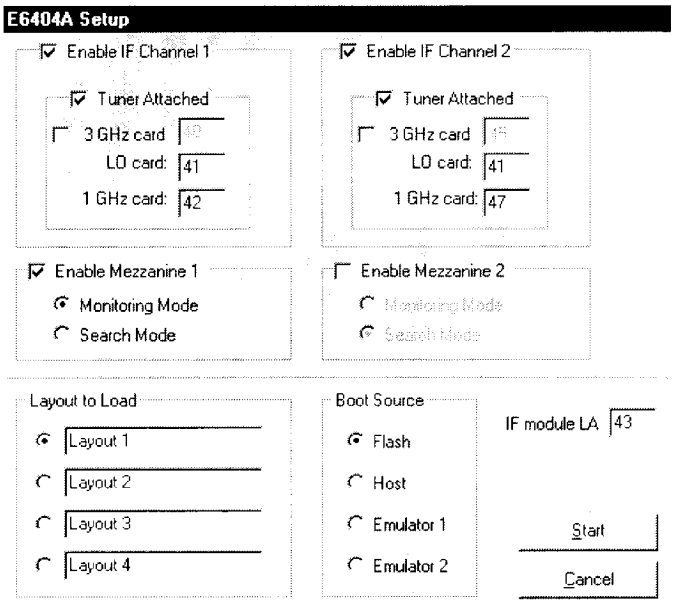


Figure 2-14 Setup for the E6503A

10. Click the **Start** button to start the virtual front panel.

Note

Certain combinations of computers and VXI interface cards may cause the virtual front panel not to function properly at initial start up. This condition may require different BIOS settings. The BIOS settings that have been found to correct this problem are as follows:

PnP operating system [no]
Advanced BIOS-PCI Busmastering - ALL SLOTS set to [Enabled]

Note

If the virtual front panel does not fit within the computer monitor, the Display Properties under the Control Panel must be reset. For example, for a 17-inch monitor, change the font size to Small Fonts, Desktop Area to 1024 by 768, and Refresh Frequency to 70 Hz.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Checking Operation

This procedure applies only to receiver-level configurations. There is *no* procedure for verifying individual module operation. This procedure assumes a single IF channel with one DDC in the IF processor configuration. The same procedure can be used to test dual channel systems with multiple DDCs installed.

To verify the correct operation of the receiver configurations, you will need the following test equipment and adapters:

Table 2-6 Test Equipment Needed

Equipment and Accessories	Critical Specification for Test Equipment	Recommended Model
Signal source	Frequency range: 20–1000 MHz or to 3000 MHz (if Option 003 is present) Amplitude accuracy: 1.5 dB	8648C
Power meter	Power range: -30 to 0 dBm Power accuracy: 0.1 dB	437B
Power sensor	Power range: -30 to +20 dBm	8482A
Adapter	Type-N (f) to BNC (f)	1250-1474
Adapter	Type-N (m) to BNC (f)	1250-0780
Adapter	BNC (f) to SMA (m)	1250-1200
Cable	BNC (m) to (m)	10501A

Procedure

1. Turn on the VXI mainframe.
2. Set the signal source power level to -20 dBm (CW) and the frequency to 30 MHz.
3. Connect the BNC cable to the signal source and measure the output power with a power meter. Adjust for an output power reading of -20 dBm on the power meter.
4. Connect the BNC cable that is connected to the signal source to the E6401A 20–1000 MHz Input port or, if option 003 is present, to the E6403A 20–3000 MHz Input port.
5. Start the virtual front panel as described in "Starting the Virtual Front Panel" in this chapter.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6. Click the **Open** pulldown command, then click the **RSSI for Mezzanine 1** item. The RSSI for Mezzanine 1 screen is displayed as shown in Figure 2-15. Each column corresponds to a DDC on the mezzanine.
7. Ensure that Monitoring Mode is selected in the Mezzanine1 pulldown menu. The RSSI feature does not function in Search mode.
8. Click the button below one of the columns in the **RSSI for Mezzanine 1** screen to read the signal level in dBm.
9. Click the **Open** pulldown command, then click the **Tuner Controls** item.

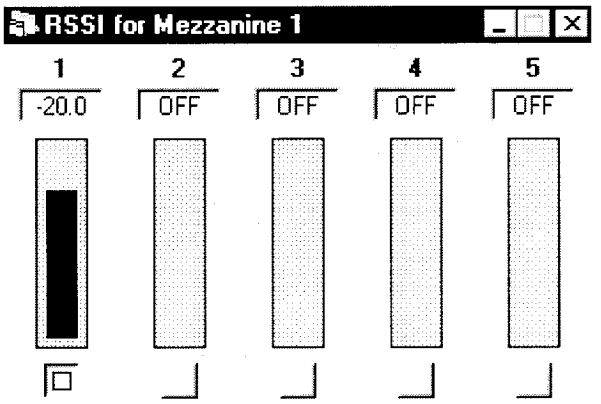


Figure 2-15 RSSI for Mezzanine 1

10. In the **Tuner Controls** screen, change the **Tuner Freq.** to read **30** (MHz).
11. Verify that the amplitude is -20 dBm +/- 3 dB.
12. Set the signal source to the next frequency listed in Table 2-7.
13. Repeat steps 10 through 12 for each frequency listed in Table 2-7.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 2-7 Frequency Points Measured for Operator's Check Procedure

System Type	Preselector Band	Frequency (MHz)
without Option 003	1	30
	2	50
	3	70
	4	100
	5	130
	6	180
	7	300
	8	400
	9	600
	10	800
with Option 003	11	1100
	12	1500
	13	2000
	14	2800

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3 Using the Receiver

In This Chapter

- E650XA VXI Receiver Overview
- Using the Virtual Front Panel (VFP)
- Using the Driver Software
- Synchronizing Multiple IF Processors and Capturing Data

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E650XA VXI Receiver Overview

This section describes the core and unique receiver capabilities. First, the "Core Receiver Capabilities" section describes capabilities common to all E650XA receivers. Then, the "Receiver Capabilities By Configuration" section shows a side-by-side comparison of the capabilities of the various receiver configurations.

Core Receiver Capabilities

The E650XA core receiver hardware consists of the E6402A local oscillator, the E6401A 20–1000 MHz downconverter, and the E6404A IF processor. All receivers have a frequency range from 20 MHz to 1 GHz. With the E6403A Option 003 block downconverter added, the frequency range is extended from 20 MHz to 3 GHz.

Note

The software driver supports tuning the receiver down to 2 MHz. However, specifications, typicals, and characteristics do not apply below 20 MHz.

The receiver can operate in one of two modes: search or monitor. The E6502A dual-channel receiver is capable of running one channel in search mode, the other channel in monitor mode, or both channels in the same mode.

Monitor Mode

Monitor mode can be thought of as the software equivalent of a receiver. That is, those tasks which would ordinarily be performed with a receiver are performed in monitoring mode. For example, multiple demodulations, link port connectivity, data capturing (to either the link port or optional SRAM), RSSI measurements, and FFTs.

In monitor mode, the receiver is tuned to an 8 MHz spectrum. (Note that the E6502A dual-channel receiver with independent LOs can monitor a 16 MHz spectrum.) The analysis of the spectrum is performed without re-tuning the tuner thus keeping a continuous IF data flow to the IF processor.

One of three anti-aliasing filters may be selected: 8 MHz, 700 kHz, and 30 kHz. Depending upon the application, any one of the three filters may be used in monitoring mode. (This mode is also referred to as "stare" mode because the tuner is fixed-tuned allowing the user to stare at a continuous spectrum for a high probability of intercept.)

The anti-aliasing filters are centered about the tuned frequency. The filter width determines how far from the tuned frequency the digital downconverters (DDCs) may be set; that is, using a tuned frequency of

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

101 MHz and a filter of 700 kHz, the DDC may be tuned from 100.650 MHz to 101.350 MHz.

The IF channel data may be routed in several ways: IF channel 1 may be routed to each mezzanine/DDCs (mezzanine data select mode 1); IF channel 2 (optional) may be routed to all mezzanine/DDCs (mezzanine data select mode 2); a combination of IF channel 1 and 2 data routed to each mezzanine (mezzanine data select mode 3); IF channel 1 data may be routed to mezzanine 1 and its associated DDCs and IF channel 2 (optional) routed to mezzanine 2 and its associated DDCs (mezzanine data select mode 4).

One distinct advantage of monitoring mode is the use of the DDCs as digital drop receivers. Depending upon the options installed in the receiver, up to 10 digital drop receivers may be operating simultaneously. There are some limitations of the digital drop receiver concept. Specifically, the DDCs have a finite allowable bandwidth.

The DDC bandwidths are referred to as digital IF bandwidths. There are 36 selectable bandwidths. When the maximum digital IF bandwidth is selected (462 kHz), only one DDC per mezzanine can be used as a digital drop receiver. As the digital IF bandwidth is reduced, more DDCs can be used as digital drop receivers. Digital IF bandwidths below approximately 30 kHz allow all installed DDCs to be used as digital drop receivers. Refer to Table 3-1.

Another function that may be performed in monitor mode is FFTs. There are two general data streams on which FFTs may be performed: DDC data and full span data. Full span data is all of the ADC data for a given filter.

Figure 3-1 shows the receiver in the monitoring mode. The uppermost FFT display shows the full span spectral display (8 MHz in this case). The two spectral displays beneath the full span display are two DDCs that are tuned to two separate frequencies: 127 MHz for the left display, and 130 MHz for the right display. Note that the DDC frequency is shown in two places: in the Mezz 1 Controls window (DDC Freq), and in the lower-left hand corner of the spectral display windows. Also note that the span setting for each of the DDC displays is the same and is listed in the Mezz 1 Controls window (IF bandwidth) and in the lower-right hand corner of the spectral display windows.

This screen capture shows the DDCs being tuned to separate frequencies independently within the 8 MHz stare window.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

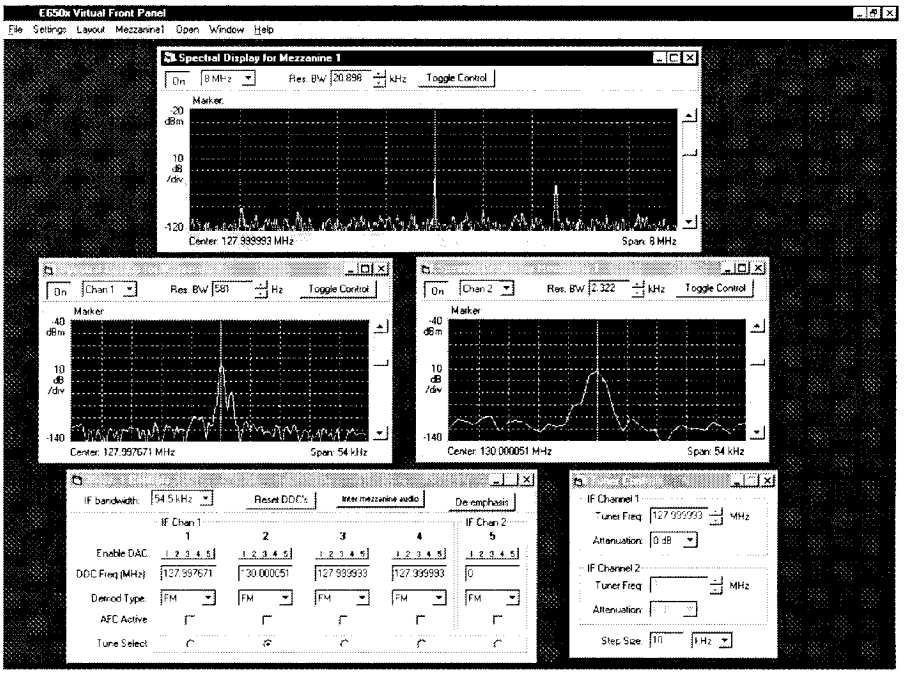


Figure 3-1 Monitor Mode and Multiple Spectral Displays in the VFP

Search Mode

In search mode, the receiver scans across its frequency range in 8 MHz steps. While the receiver dwells at each 8 MHz spectrum, the digital signal processor (DSP) performs an FFT. In this mode, the DDCs are not used and all of the data is sent directly to the DSP to perform an FFT on the entire span of data.

Larger FFTs are performed in search mode. As a result, better signal resolution is attained. Similarly, the code used to tune the tuners is optimized for the fastest operation.

The DSP is essentially making block mode measurements in search mode. No real-time measurements are performed; however, all of the autoranging and dynamic range benefits are realized.

Figure 3-2 shows a 20 MHz to 1000 MHz search window in the virtual front panel (VFP). Note that the spectrum is actually "stitched" together in 8 MHz increments.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

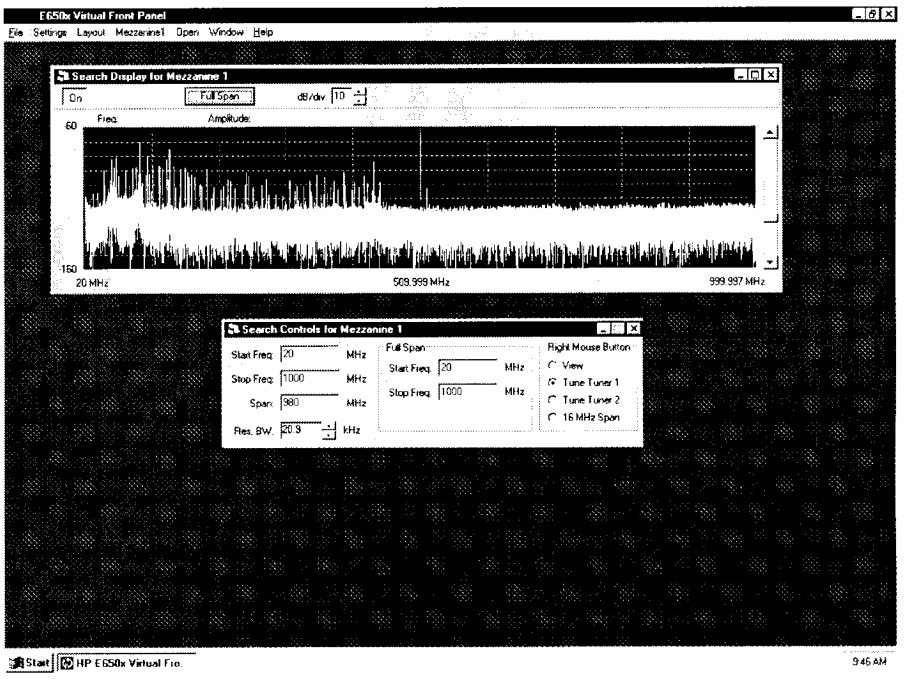


Figure 3-2 Search Mode Window in VFP

Multiple Demodulations

Multiple demodulations of AM, FM, USB, LSB, PM, and CW signals within the 8 MHz monitor (stare) spectrum are possible. Though the standard receivers are installed with one DDC per mezzanine, up to 5 total can be installed per mezzanine.

Each DDC provides the capability of signal channelization and tuning to a signal within the 8 MHz spectrum. The DSP performs the actual demodulation. Therefore, with 10 DDCs installed in a receiver, up to 10 simultaneous demodulations can be performed.

The number of possible demodulations is dependent upon the number of DDCs installed, and the DDC bandwidth. Table 3-1 shows the maximum number of demodulations allowed versus the digital IF bandwidth (DDC bandwidth). The information in Table 3-1 assumes five DDCs are installed per mezzanine.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-1 Demodulations Versus Digital IF (DDC) Bandwidth

Number of DDCs	Bandwidth (Maximum)
1	462 kHz
2	187 kHz
3	83 kHz
4	54 kHz
5	34 kHz

Digital IF (DDC) Bandwidths

The digital IF bandwidths, or DDC bandwidths, are controlled by the DDC(s) on the mezzanine assembly. Receivers with more than one DDC on a mezzanine are all set to the same digital IF bandwidth. The E6502A is a dual receiver with two mezzanines. This configuration allows the digital IF bandwidth to be different between the two mezzanines.

The digital IF bandwidths are equivalent to span width in monitor mode processes as shown in Figure 3-1.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-2 shows the digital IF (DDC) bandwidths available.

Table 3-2 Digital (DDC) IF Bandwidths

(0)	247 Hz	(13)	29 kHz	(26)	187 kHz
(1)	493 Hz	(14)	34 kHz	(27)	201 kHz
(2)	740 Hz	(15)	44 kHz	(28)	218 kHz
(3)	1 kHz	(16)	54 kHz	(29)	238 kHz
(4)	2.4 kHz	(17)	64 kHz	(30)	262 kHz
(5)	3 kHz	(18)	74 kHz	(31)	291 kHz
(6)	5 kHz	(19)	83 kHz	(32)	327 kHz
(7)	6.25 kHz	(20)	93 kHz	(33)	374 kHz
(8)	10 kHz	(21)	109 kHz	(34)	436 kHz
(9)	12.5 kHz	(22)	123 kHz	(35)	462 kHz
(10)	15 kHz	(23)	138 kHz		
(11)	20 kHz	(24)	154 kHz		
(12)	25 kHz	(25)	171 kHz		

The number shown in parenthesis indicates the value passed to the E650XA driver to set the corresponding bandwidth.

Gain Control and Dynamic Range Optimization

Each receiver has analog autoranging gain control from +12 dBm to –48 dBm in 2 dB steps. The receivers automatically control gain in each band depending on the signal level and thus allow the monitoring of both high- and low-level signals across the entire search band. In addition, the fast response time ensures that the digitizer (ADC) is protected from overload by high-level signals.

Analog Outputs

A special multi-pin connector (Audio/Trigger) on the E6404A IF processor provides demodulated analog audio outputs. The output connector can be used to connect to amplified speakers, headphones, and analog audio recorders. In addition, options can be ordered to extend the number of simultaneous analog audio output signals to 10. By connecting an external cable from this connector to an audio breakout box (E3245A), ten headphones or ten speakers can be connected for monitoring of up to ten audio signals. Note that the maximum audio bandwidth of the audio signal is 15 kHz. Refer to Figure 3-3.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

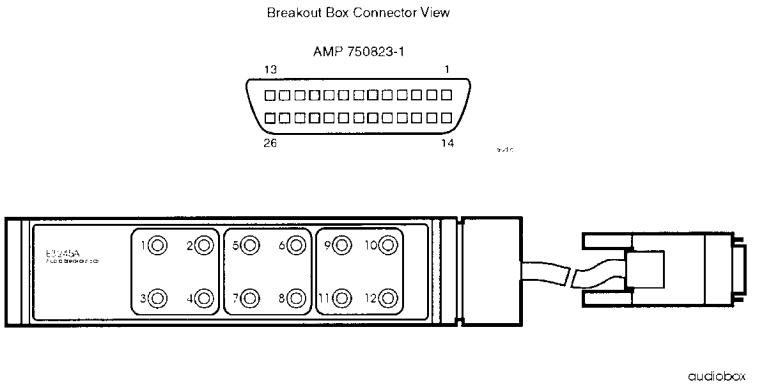


Figure 3-3 Audio/Trigger Connector Pinouts and Audio Breakout Box

Note

The Audio/Trigger connector shown in Figure 3-3 shows the rear view of the connector.

Table 3-3 shows the pin numbers for the Audio/Trigger connector and corresponding Audio Breakout Box jack connections.

The Audio/Trigger connector on the front panel of the E6404A IF processor is part number AMP 750823-1. To configure your own cable, order AMP 750833-1 cable connector and 750850-3 backshell kit.

Table 3-3 Audio/Trigger Connector Pinouts

VFP Channel Number	Audio Breakout Box Jack	Audio/Trigger Connector Signal Pin Number	Audio/Trigger Connector Return Pin Number
1	1	15	14
2	2	3	4
3	3	2	1
4	4	16	17
5	5	19	18
6	6	7	8
7	7	6	5
8	8	20	21
9	9	23	22
10	10	11	12
Input Trigger	11	10	9

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Digital Outputs

The receivers are capable of various digital data outputs from the Link Port connectors on the E6404A IF processor. These ports provide full rate digitized data from the analog-to-digital converter (ADC), or I and Q data from the DDC for use by an external DSP device or digital recorder (to PC, VXI, or VME).

The Link Port connector on the front panel of the E6404A IF processor is part number AMP 1-104074-0. The mating connector is part number AMP 487550-5. The eight pin Link Port connector is compatible with the Analog Devices 2106X family of DSP devices. Link port cables are available from Transtech Parallel Systems to connect to external DSP devices (part number TTC27-x, where x = length of cable in cm).

Note

Only 3 meter (or shorter) cables can be used with the receiver.

Table 3-4 shows the Link Port pin numbers and corresponding signals. Figure 3-4 shows the Link Port connectors and pin number orientation on the E6404A IF processor.

Table 3-4 Sharc Link Port Pinouts

Pin Number	Signal
1	CLK
2	ACK
3	GND
4	DAT (0)
5	DAT (1)
6	GND
7	DAT (2)
8	DAT (3)

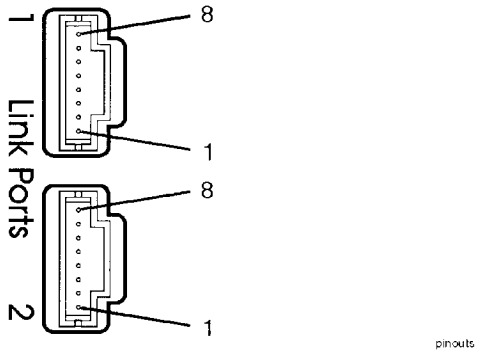


Figure 3-4 Link Port Pin Orientation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-5 Link Port Digital Data Output

Digital Data (Link Ports)	Parameters
full rate data	28.533 Msamples/sec (using 2 link ports) 57.066 MBytes/sec (16 bit samples)
digital I/Q data	247 Hz to 462 kHz DDC bandwidth (up to 10 signals)

Figure 3-5 shows the output data format for full rate data.

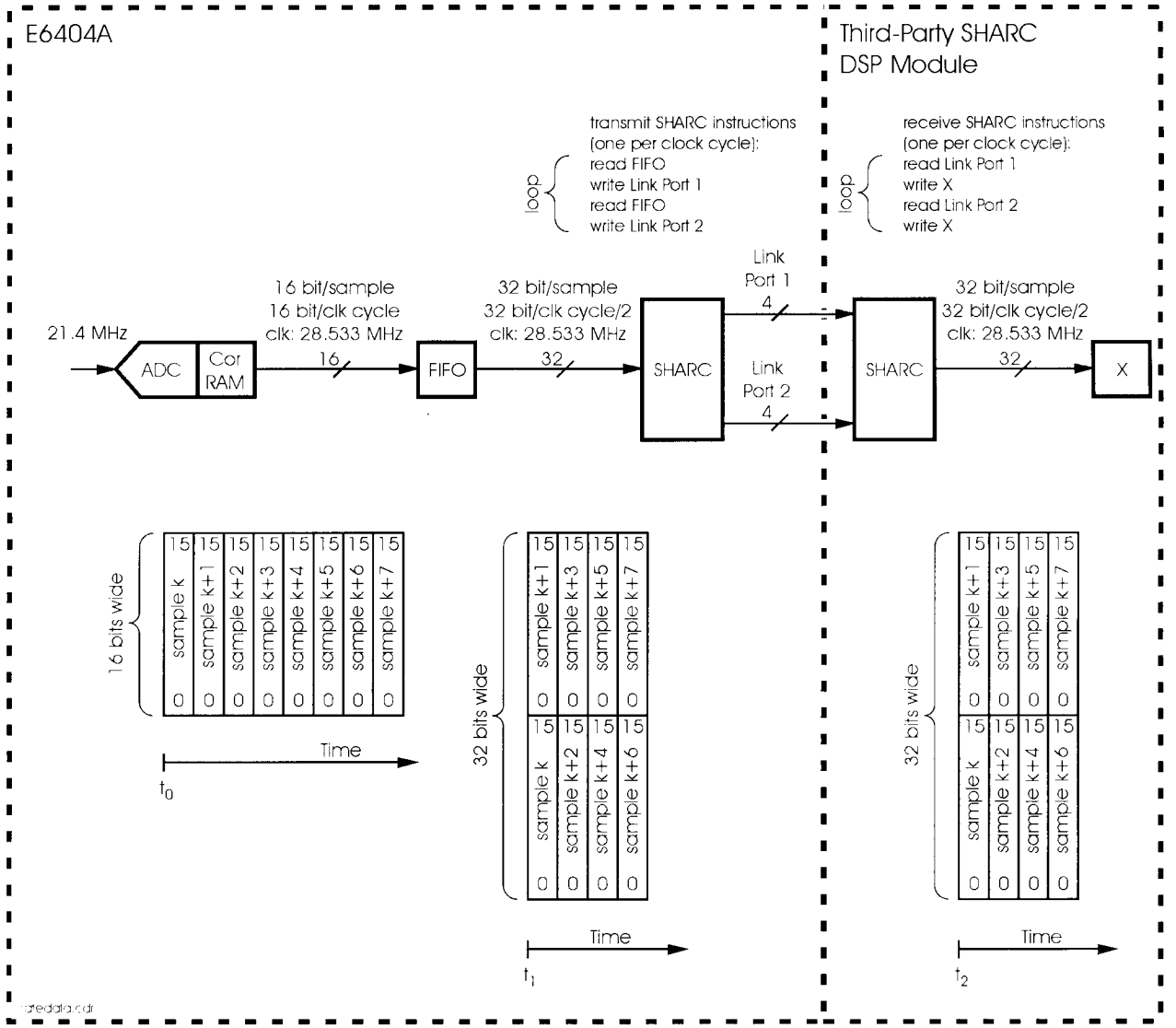


Figure 3-5 Full Rate Data Sharc Link Output Data Format

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Receiver Capabilities By Configuration

Table 3-6 shows the capabilities that vary between the three standard receiver configurations. Refer also to Table 1-1 for options that extend standard receiver functionality.

Table 3-6 E650XA VXI Receiver Capabilities

	E6501A	E6502A	E6503A
RF input	single	dual	dual
analog audio output channels	1	2	2
link ports	2	4	2
IF channels	1	2	2
simultaneous search and monitor		x	
direction finding applications			x
maximum monitor (stare) bandwidth	8 MHz	16 MHz[1]	8 MHz
maximum DDCs in standard configuration without adding extra mezzanines.	5	10	5
maximum optional DDC configuration	10	10	10

1. The 16 MHz of "stare" bandwidth is achieved by including two 8 MHz FFTs, side-by-side, in one 16 MHz window. This is *not* intended as a 16 MHz continuous bandwidth capable of looking at signals with modulation bandwidths greater than 8 MHz.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using the Virtual Front Panel

This section provides descriptions of the functions found in the virtual front panel (VFP) graphical user interface (GUI). The VFP enables users to interactively change settings and display results from measurements such as FFTs.

Note

The VFP is intended to demonstrate some of the receiver hardware capabilities. It is provided to show the user a simple example of how they might design their own interface. The VFP is not intended to be a full featured software solution.

File Menu

The File menu performs three functions as shown in Figure 3-6.

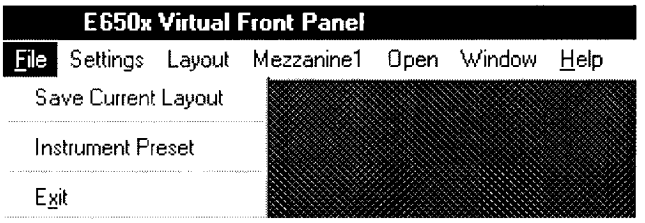


Figure 3-6 File Menu

Save Current Layout — You can open and arrange displays in the main VFP window according to your needs, then save the layout with this function. Note that up to four different layouts can be saved by using the Layout menu.

Instrument Preset — Selecting Instrument Preset will configure the receiver to the preset state.

Exit — Exits the VFP GUI.

Settings Menu

The Settings menu is used to set up various receiver functions in the General Setup dialog box. The Settings menu is shown in Figure 3-7.



Figure 3-7 Settings Menu

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

General Setup Dialog Box

Figure 3-8 shows the General Setup dialog box.

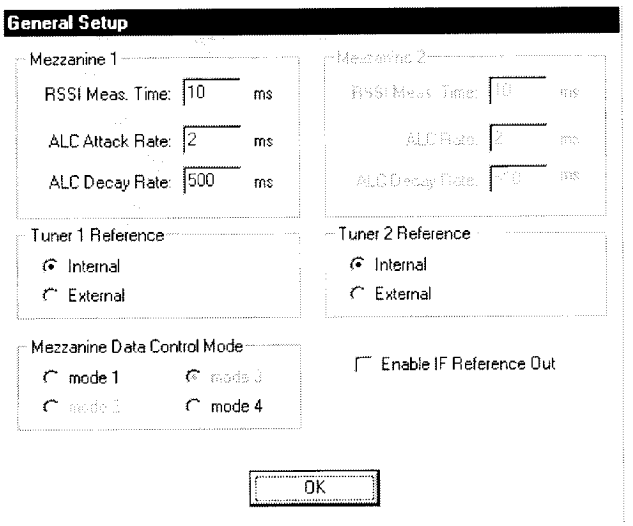


Figure 3-8 General Setup Dialog Box

RSSI Meas. Time: This function sets the amount of received signal strength indication (RSSI) measurement time.

ALC Attack Rate: The ALC attack rate controls the attack time of the ALC for the audio output. The value is in milliseconds (ms).

ALC Decay Rate: The ALC decay rate controls the decay time of the ALC for the audio output. The value is in milliseconds (ms).

Internal: Selects the use of the internal 10 MHz reference of the tuner.

External: Deselects the internal 10 MHz reference of the tuner.

mode 1
mode 2
mode 3
mode 4 Selects the desired data routing through the IF processor. Refer to the "Mezzanine Data Select Modes" section in this chapter.

Note Mode 4 must be used for simultaneous search and monitoring (E6502A) or for 16 MHz stare.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Enable IF Reference Out Enables the 10 MHz reference output in the IF processor.

Layout Menu

The Layout menu is used to select between four user-defined layouts. The Layout menu is shown in Figure 3-9.

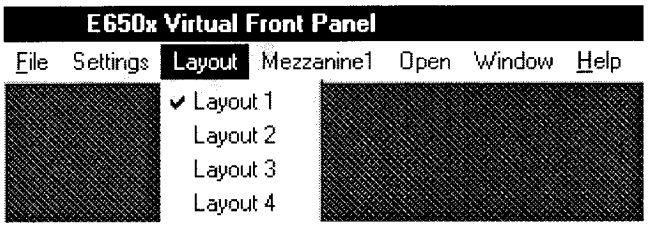


Figure 3-9 Layout Menu

Layout 1
Layout 2
Layout 3
Layout 4 This function allows you to save and select one of four different display layouts.

To define a layout, select the layout number in the Layout menu, open and arrange displays as needed, then save the layout using the Save Current Layout command in the File menu.

Mezzanine Menu

The Mezzanine menu is used to select between monitoring and search modes. A pre-defined layout for monitoring mode is shown in Figure 3-10.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

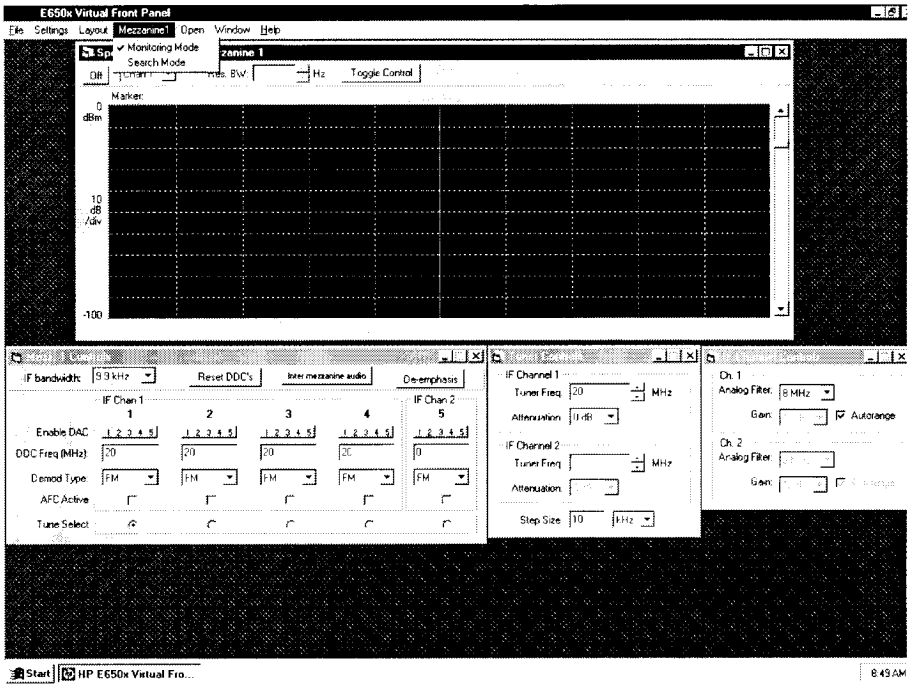


Figure 3-10 Mezzanine Menu with Monitoring Mode Selected

A pre-defined layout for search mode is shown in Figure 3-11.

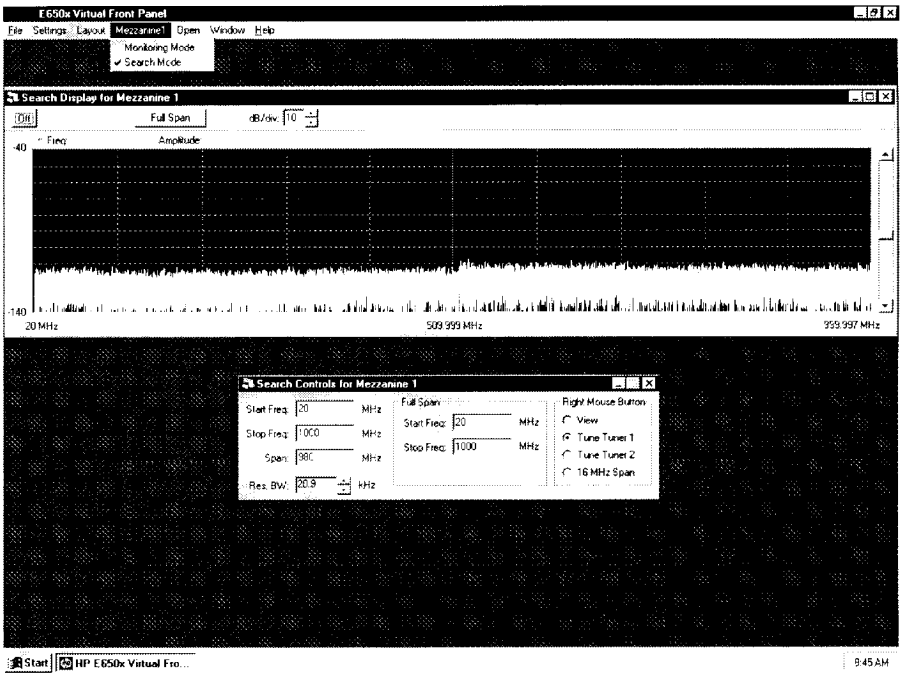


Figure 3-11 Mezzanine Menu with Search Mode Selected

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Open Menu

The Open menu is the main access to most functions. The Open menu is shown in Figure 3-12.

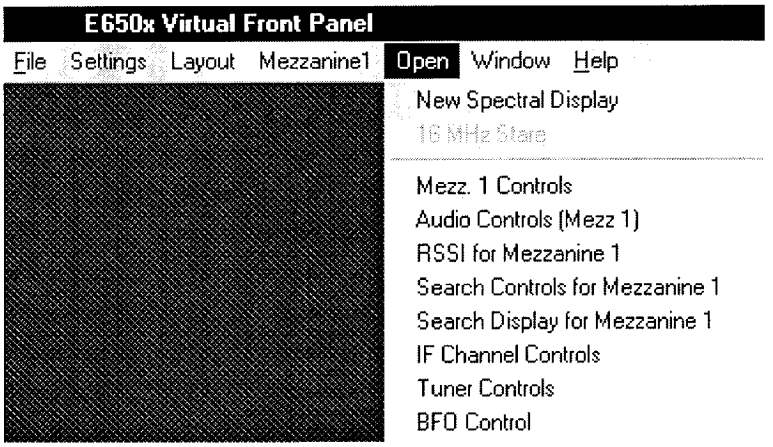


Figure 3-12 Open Menu

New Spectral Display

Figure 3-13 shows the New Spectral Display. Multiple spectral displays can be opened. These displays are only used in monitoring mode.

Spectral displays, or IF pan windows, allow you to channelize signals. The IF pan windows are also referred to as digital drop receivers (DDR).

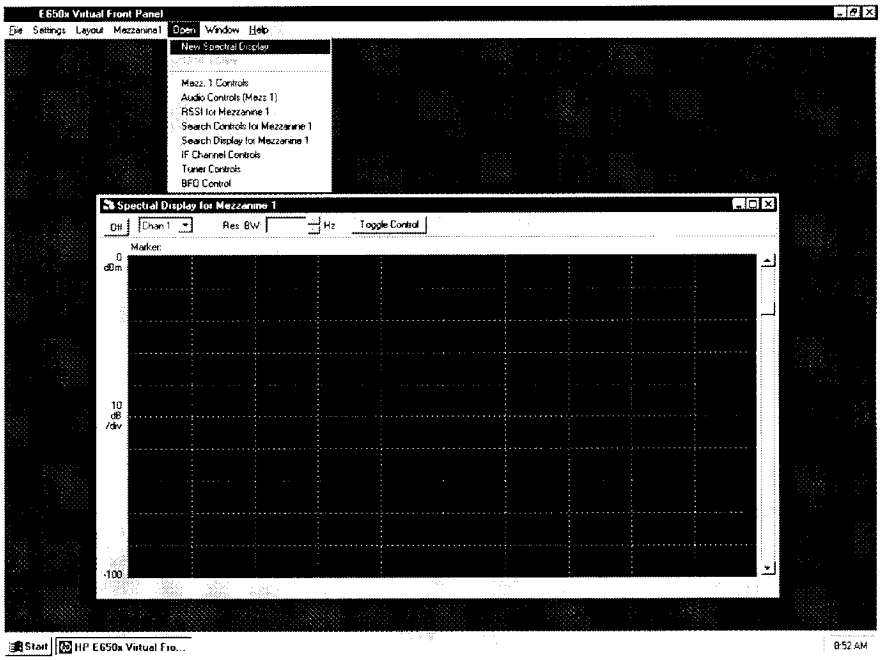


Figure 3-13 New Spectral Display

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Off
On Turns the spectral display on and off.

Chan This drop-down menu allows you to select a specific channel or select the 8 MHz monitor (stare) mode where you monitor an entire 8 MHz spectrum. This function works in conjunction with the functions found in the Mezz Controls dialog box. Note that the digital IF bandwidth setting in the Mezz Controls dialog box determines the number of possible channels (DDCs). In general, the wider the bandwidth, the fewer available channels (DDCs). Refer to Table 3-1 for the limitations.

Res. BW
Averages
dB/div The Res. BW function allows you to change the resolution bandwidth.

The Averages function corresponds to the number of trace averages to smooth the trace displayed.

The dB/div function changes the dB per vertical division from 1 dB to 100 dB in 5 dB steps.

Toggle Control Toggles between the Res. BW, Averages, and dB/div functions.

Shortcut Menu in Spectral Display

By clicking the right mouse button when the mouse pointer is in a spectral display window, a shortcut menu appears as shown in Figure 3-14.

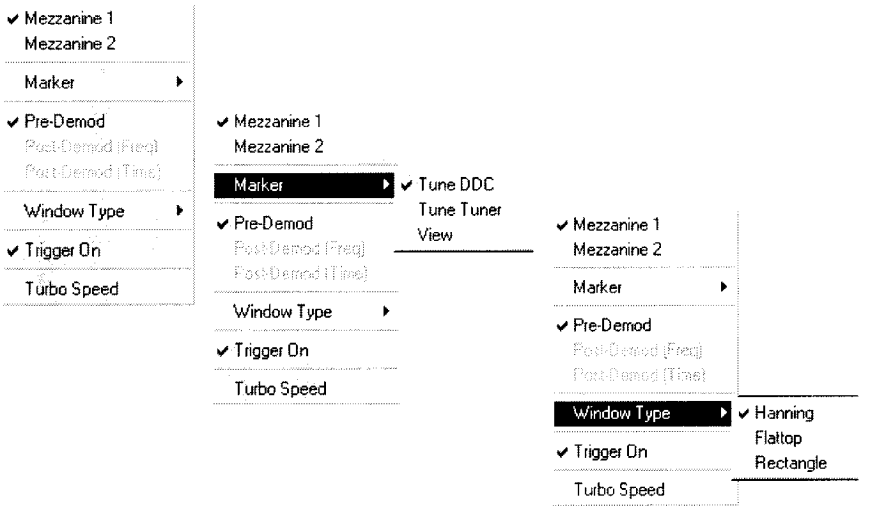


Figure 3-14 Spectral Display Shortcut Menu

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Mezzanine 1
Mezzanine 2 Selects either mezzanine 1 or 2.

Marker Clicking the left mouse button inside a spectral display causes a marker to appear. The marker will display frequency and amplitude information. One of three functions may be selected.

With the Tune DDC function selected, the DDC frequency will track with the marker frequency.

With the Tune Tuner function selected, the tuner frequency will track with the marker frequency.

The View function simply allows viewing of the marker and does not adjust frequency.

Window Type This function selects Hanning, Flatop, or Rectangle windowing. Refer to Chapter 4 for an explanation of windowing and its purpose.

Turbo Speed Increases the display refresh rate.

16 MHz Stare

This function is only available in the E6502A dual channel receiver with independent LOs. The 16 MHz Stare function can monitor, or stare, at a 16 MHz spectrum, rather than an 8 MHz spectrum as in the other receivers.

Note

The IF module must be set to mode (3 or 4) for this window to work with 16 MHz of "stare" bandwidth.

Mezz. 1 Controls

The mezzanine control panel allows you to control the DDCs and their bandwidth. Note that the DDC bandwidth is not the same as the analog bandwidth. The DDC bandwidth corresponds to the span in the IF pan windows. Figure 3-15 shows the mezzanine control panel.

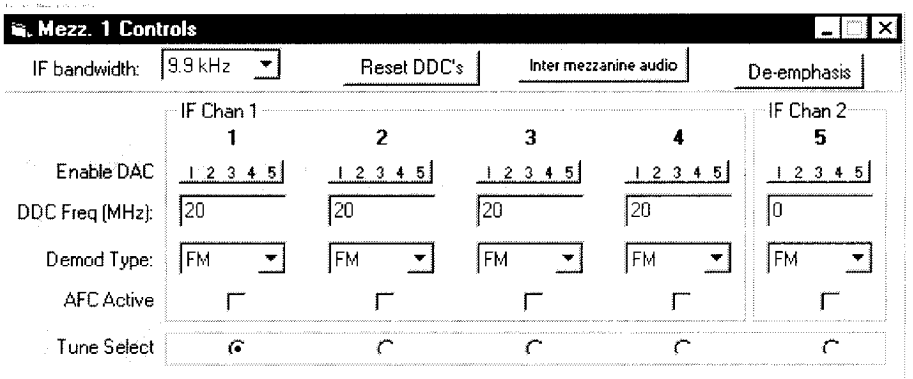


Figure 3-15 Mezz. 1 Controls

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IF bandwidth The drop-down menu allows you to select the digital IF bandwidth for the DDCs on a given mezzanine. Note that the DDCs on any given mezzanine must all be set to the same bandwidth, but they can each be set to a different frequency.

Reset DDCs Forces the DDCs to a known state.

Inter mezzanine audio Routes audio signals from one mezzanine to the other in a dual mezzanine configuration. This allows the use of the audio breakout box for 10 simultaneous audio outputs.

Note

If you are in Mezz. 1 Controls and you click Inter mezzanine audio, you will send all the DAC outputs out the mezzanine 2 audio connector and vice versa.

De-emphasis Improves FM sensitivity.

Enable DAC This function corresponds to the audio output located on the front panel of the E6404A IF processor. This allows you to listen to the demodulated audio for a specified DDC.

DDC Freq (MHz): This function allows you to tune any given DDC to a frequency within the analog bandwidth.

Demod Type: This function selects the type of demodulation on each DDC.

Note that you cannot simultaneously perform multiple demodulation types on any given DDC. For example, you cannot select AM demodulation on DDC 1 on IF channel 1, and FM demodulation on DDC 1 on IF channel 1.

AFC Active Enables/disables automatic frequency control.

Tune Select This radio button corresponds to which DDC to tune (frequency). If the marker in the spectral display is configured to Tune DDC, then the DDC frequency will track the marker frequency.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Audio Controls (Mezz 1)

This control corresponds to the audio output located on the front panel of the E6404A IF processor. The control panel shown in Figure 3-16 is used to adjust the audio volume level controlled by the DAC that was selected with the Enable DAC function for the DDC in the Mezz. 1 Controls panel, as well as enable and set the level of squelch.

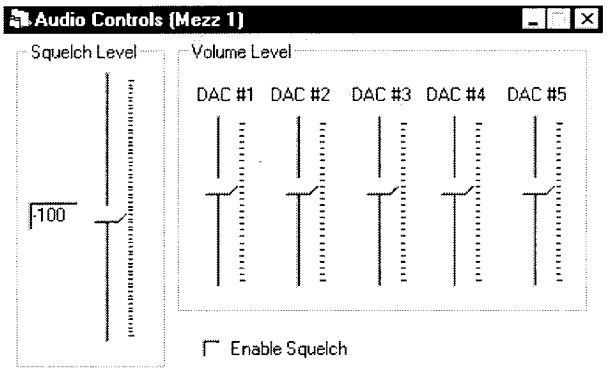


Figure 3-16 Audio Controls

RSSI for Mezzanine 1

The RSSI (Received Signal Strength Indicator) window automatically computes the channelized power in a given DDC and graphically displays the results. The bold numbers above each column shown in Figure 3-16 correspond to the DDC numbers in the mezzanine. Activate an RSSI measurement for a given DDC by clicking the corresponding box shown below each column.

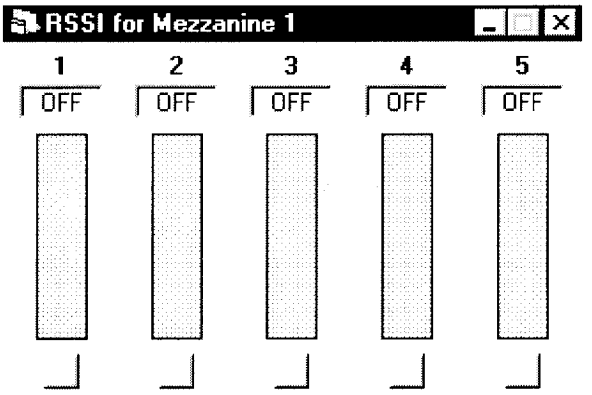


Figure 3-17 RSSI Controls

Search Controls for Mezzanine 1

The search control panel is used to control various functions associated with the search process. Figure 3-18 shows the search control panel.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

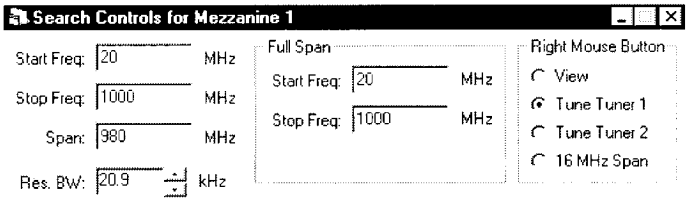


Figure 3-18 Search Controls for Mezzanine 1

Note

The software driver supports tuning the receiver down to 2 MHz. However, specifications, typicals, and characteristics do not apply below 20 MHz.

Start Freq: Sets the start frequency of the search display window. The start frequency must be *greater than or equal to* 2 MHz.

Stop Freq: Sets the stop frequency of the search display window.

Span: Sets the span of the search display window.

Res. BW: Sets the resolution bandwidth of the search display window.

Note

Since there is a finite number of displayable points, it is possible to have a smaller actual resolution bandwidth than display bandwidth.

Full Span Start Freq: The search display window will default to this setting of start frequency when you click on the Full Span button.

Full Span Stop Freq: The search display window will default to this setting of stop frequency when you click on the Full Span button.

View This function allows you to view the frequency and amplitude of the marker when you click the right mouse button inside the search display window.

Tune IF Chan 1 This function causes the tuner's frequency to track the frequency of the marker when you click the right mouse button inside the search display window.

16 MHz Span Allows 16 MHz spans in monitor (stare) mode (E6502A only).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Search Display for Mezzanine 1

The search display shows a spectral display of either the entire frequency span of the receiver, or a span defined by the user in the Search Controls panel. The results of data are taken in 8 MHz steps and displayed as one continuous spectrum from the specified start frequency to the specified stop frequency. Figure 3-19 is an example of a search display.

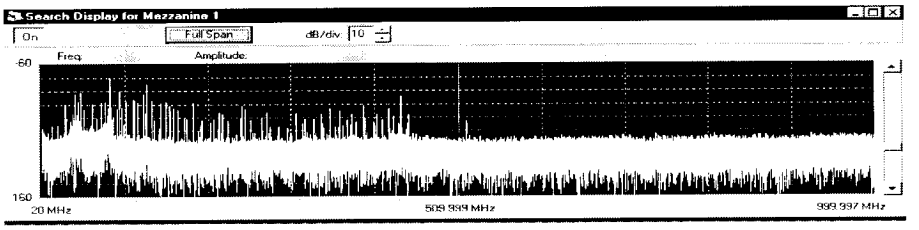


Figure 3-19 Search Display for Mezzanine 1

Off
On Turns the search display on and off.

Full Span The Full Span button sets the start and stop frequency settings to those defined in the Search Controls panel under Full Span.

dB/div The dB/div function changes the dB per vertical division from 1 dB to 20 dB.

IF Channel Controls

The IF channel controls allow you to select the desired analog filter and gain setting for each IF channel. Figure 3-20 shows the IF channel controls.

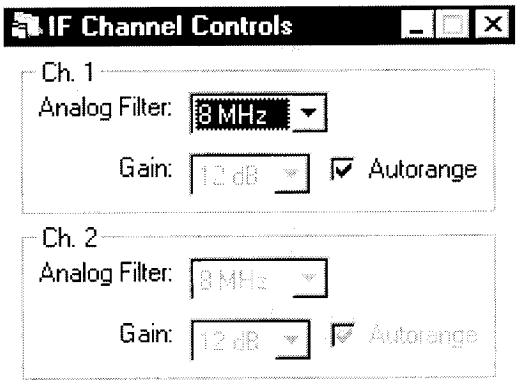


Figure 3-20 IF Channel Controls

Analog Filter This function selects the 8 MHz, 700 kHz, or 30 kHz analog bandpass filters in the IF processor.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Gain With the Autorange deselected, the analog gain can be set from 0 dB to 60 dB in 2 dB steps. The analog gain amplifiers are in the IF processor.

Autorange Enables/disables autoranging. Autoranging gain control allows the receiver to automatically control gain according to the signal level. In this way, both high- and low-level signals can be monitored across the entire search band.

Tuner Controls

The tuner controls allow you to tune the frequency and set input attenuation for each IF channel. Each IF channel may control one tuner. It is possible to have one tuner attached to two IF channels, but not two tuners attached to one IF channel. Figure 3-21 shows the tuner controls.

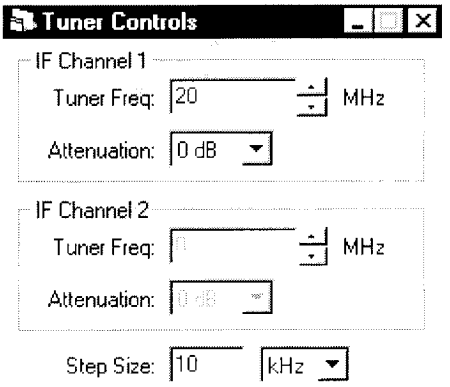


Figure 3-21 Tuner Controls

Tuner Freq: Sets the tuner frequency in steps defined using the Step Size function.

Attenuation Sets the input attenuation from 0 dB to 30 dB in 10 dB steps.

Step Size Sets the frequency step size for the Tune Freq function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

BFO Control

The BFO (beat frequency oscillator) control allows you to introduce a signal at a known frequency into the detected signal. This function is used extensively in suppressed carrier forms of demodulation. Figure 3-22 shows the BFO control.

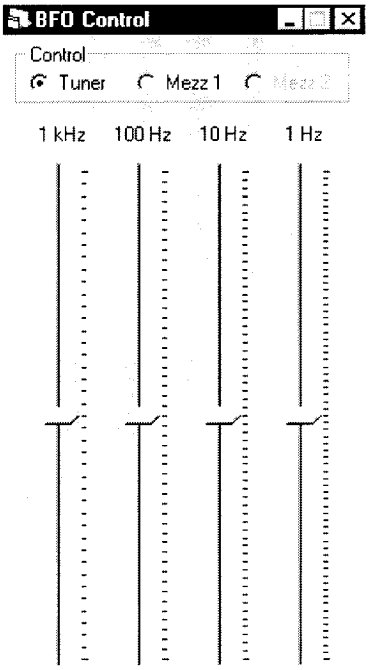


Figure 3-22 BFO Control

Window Menu

The Window Menu shows a list of the currently open windows. This function allows you to select a window and place it on top for viewing. The Window menu is shown in Figure 3-23.

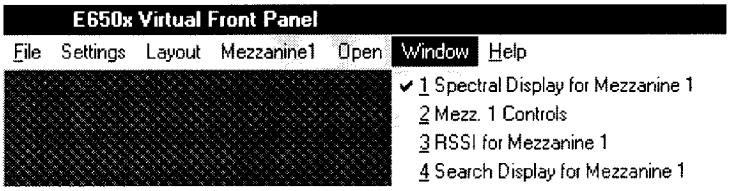


Figure 3-23 Window Menu

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using the Driver Software

Programmer's Block Diagrams

The programmer's block diagrams can be found at the end of this chapter. These diagrams are intended as an aid when programming the receiver. They show the mezzanine data select modes for routing data through the IF processor, as well as the main sections of the IF processor. The main sections of the IF processor are grouped by number. The commands in chapter 6 are mapped to these group numbers to help identify the section of the IF processor that a command is controlling.

Note

The driver software refers to IF channel 1 as IF channel 0, and IF channel 2 as IF channel 1. This is also the logic shown on the programmer's block diagrams.

Mezzanine Data Select Modes

There are four modes for routing the IF channel data. Mode 1 is used when there are two mezzanines and both receive data from IF channel 1. The E6501A receiver has one IF channel and one mezzanine. With option 031, a second mezzanine is added.

Mode 2 is used when there are two mezzanines and both receive data from IF channel 2. Since the E6501A receiver does not have an IF channel 2, mode 2 only applies to the E6502A and E6503A receivers.

Mode 3 is the crossover mode. This mode is used when there are two mezzanines and utilizes up to five DDCs on each mezzanine. Mode 3 takes IF channel 1 data and routes it to the first four DDCs on mezzanine 1, and then to the fifth DDC on mezzanine 2. Similarly, IF channel 2 routes data to the first four DDCs on mezzanine 2, and then to the fifth DDC on mezzanine 1. This mode is used primarily by the E6503A for direction finding applications. In this case, IF channel 1 and IF channel 2 data is routed to mezzanine 1, which has two DDCs (standard E6503A configuration).

Mode 4 takes the data from IF channel 1 and routes it to mezzanine 1 (up to five DDCs). This mode then takes the data from IF channel 2 and routes it to mezzanine 2 (up to five DDCs).

Command Group Numbers

The commands are grouped by number: group 0, group 1, group 2, group 3, group 4, or group 5. Refer to the programmer's block diagrams and the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

following table for an explanation of command group numbers and what they control.

Table 3-7 Group Command Numbers

Group Number	Description
0	Group 0 commands control the overall receiver system.
1	Group 1 commands control the analog filters in the E6404A IF processor.
2	Group 2 commands control the ADC(s) in the E6404A IF processor.
3	Group 3 commands control the DDC(s) in the E6404A IF processor.
4	Group 4 commands control the DSP(s) in the E6404A IF processor.
5	Group 5 commands control the link port data and the audio data in the E6404A IF processor.

Maximum FFT Length

The driver software supports a total FFT length of 4096 points for monitoring processes (monitor and IF pan displays). That is, a total FFT length for all monitor and IF pan displays for a given mezzanine cannot exceed 4096 points.

With the DDCs bypassed in search mode, the driver software automatically sets the FFT length.

Because the receiver performs FFTs, it has discrete resolution bandwidth settings. Therefore, the lengths of the transform are restricted to powers of two.

Maximum Number of FFT Processes

FFT processes refers to any process where FFT data is returned. For example, search and monitor windows in the VFP each constitute one process. Each IF pan display in the VFP constitutes one process. The maximum number of FFT processes that can be run is four. However, another constraint is the maximum combined FFT length of 4096 points for all FFT processes. The driver software will not allow the sum of the FFT processes to exceed a total FFT length of 4096 points, even if fewer than four processes are currently running.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DSP Considerations

There are two loading considerations: serial loading and DSP loading.

Serial Loading

Serial loading is a condition where the period of the frame synchronization clock is too small to transmit the downconverted data to the DSP. The amount of information sent by each of the DDCs is determined by the decimation rate.

To ensure that serial loading does not occur, the DDCs will not introduce too much data (essentially by restricting bandwidth combinations). Refer to Table 3-8 to help understand how serial loading is managed in the receiver.

Table 3-8 Number of DDCs Versus Maximum Bandwidth

Number of DDCs	Maximum Bandwidth (kHz)
1	462
2	187
3	83
4	54
5	34

The information shown in Table 3-8 is relative to data rates and is not related to the number and types of processes running.

DSP Loading

Unlike serial loading, DSP loading is more subtle and is somewhat undetectable by the driver and processor. This form of loading is detectable by the user when performing various tasks such as listening to demodulated audio.

DSP loading is a function of the intensity of the process being run. There are several processes that the processor is capable of performing. These processes along with their relative DSP loading weight are shown in Table 3-9.

Examples of DSP loading are distorted or no audio, or distorted spectral displays.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-9 DSP Processes Versus DSP Loading Weight

Heavy	Medium	Light
SSB/ISB	FM	CW
PM	AM	
full rate data capture	link port I/Q data capture	

When using the receiver, these factors should be considered when deciding how to use the receiver in a given application.

Driver Revision

The *hpe650x_revision_query* command will return the revision of the software driver and the firmware revision of the receiver.

Default Receiver Settings

Table 3-10 is a list of the default receiver settings whenever power is cycled or the hpe650x_**reset** command is executed.

Table 3-10 Default Receiver Settings

Receiver Function	Default Setting
IF bandpass filter	8 MHz
mezzanine data select mode	1
IF gain	manual/0 dB gain
IF bandwidth	10 kHz
tuner 10 MHz reference	on

Opening and Closing an Instrument Session

The commands shown in Table 3-11 are required for opening and closing an instrument session.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-11 Commands for Opening and Closing an Instrument Session

Command	Description
result= hpe650x_**init** (rsrcName, idQuery, resetInstr, pInstrumentID);	This command establishes communication with the IF processor. ***rsrcName****:* ensures that the logical address of the IF processor is correct. This value is typically "VXI0 : : 43 : : INSTR". ***idQuery****:* determines if the option string will be read by the driver. A value of VI_TRUE forces the driver to thread the option string and enable error checking based on the results. ***resetInstr****:* determines if the DSP will be reset after the code is loaded. A value of VI_TRUE will reset the DSP. ***pInstrumentID:*** is the pointer to the address of memory allocated for the instrument ID. All subsequent commands to the receiver must be referenced by this instrument ID.
result= hpe650x_**initIFChannel** (InstrumentID, IFchan, tuner_attached, InitialRFFrequency, LO_LogicalAddr, OneGHz_LogicalAddr, ThreeGHz_LogicalAddr);	This command initializes the IF channel, establishes communication with the tuner section, and sets logical addresses of the LO and downconverter(s). ***InstrumentID***: pointer to the address of memory allocated for the instrument ID. ***IFchan:*** defines the IF channel to use. A value of 0 corresponds to IF channel 1; a value of 1 corresponds to IF channel 2. ***tuner_attached:*** confirms whether the IF channel is controlling a tuner. A value of VI_TRUE indicates that the IF channel is controlling a tuner. ***InitialRFFrequency:*** Sets the initial RF frequency. ***LO_LogicalAddr:*** Sets the logical address of the LO. The value set at the factory is 41. ***OneGHz_LogicalAddr:*** Sets the logical address of the 1 GHz downconverter. The value set at the factory is 42. ***ThreeGHz_LogicalAddr:*** Sets the logical address of the 3 GHz option 003 downconverter. The value set at the factory is 40. If your system does not have a 3 GHz option, this parameter should be set to zero.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-11 Commands for Opening and Closing an Instrument Session

Command	Description
result= hpe650x_**setMezzanineDataSelectMode** (!nstrumentID, mode);	This command selects the mezzanine data select mode for data routing through the IF processor. ***InstrumentID***: pointer to the address of memory allocated for the instrument ID. ***mode:*** selects mode 1, 2, 3, or 4 for data routing through the IF processor. Refer to "Mezzanine Data Select Modes" for more information.
User's controlling program	
result= hpe650x_**close** (InstrumentID);	This command terminates the software connection to the receiver and de-allocates resources associated with the receiver. ***InstrumentID***: pointer to the address of memory allocated for the instrument ID.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Return Values

The E650X receiver functions return a defined value depending on the outcome of the process. For example, if the function was successful, 0 is returned. If the function failed, an error in the range of 0xFEC0 0000 to 0xFEC0 FFFF is returned. In other words, a zero corresponds to the successful execution of the function, while any negative value corresponds to an error condition.

Warnings have values >0 (any positive value). A warning indicates a non-fatal problem (function succeeded) during the execution of the function process.

Refer to Chapter 6 for a list of return values.

Pointers to Memory Addresses

The success, error, and warning values are the only values returned. Retrieving actual data, such as IF attenuator setting, tuner frequency, digital IF bandwidth, etc., is accomplished by using pointers. A pointer is a parameter that points to the address of allocated memory where the data is stored. These parameters have the letter "P" somewhere in their data type name. For example, ViPSession, ViPInt32, and ViPReal64 indicate that their corresponding parameters are pointers to the location where actual data is stored. Most commands starting with "get" have at least one pointer to the location of data.

Receiver Programming Examples

To set up a search process

The following procedure is the minimum set of calls required to start a search process for IF channel 1. Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

The user must write a program for retrieving the search data using the hpe650x_**getSearchTraceLength** and hpe650x_**getSearchTrace** commands.

```
result= hpe650x_init("VXI0 : : 43 : : INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setSearchMode(InstrumentID, 0);
result= hpe650x_setSearchResolutionBWParameters(InstrumentID, 0, 0,35,4000);
result= hpe650x_setSearchOutputTraceLength(InstrumentID, 0, 4000);
result= hpe650x_startSearch(InstrumentID, 0);
        {
        //To Do: Put your program code here to retrieve and process data.
        }
result= hpe650x_stopSearch(InstrumentID, 0);
result= hpe650x_close(InstrumentID);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note

A complete, fully functional C^{++} search application, with source code and a graphical user interface, is available on the Internet at:

http://wwwdb.tm.agilent.com/cgi-bin/DSP/tmoArsenal.cgi?FP=productArsenal&Area=Datasheet&Template=overview&Tid=HPE6501A&Language=English&model_no=HP+E6501A

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To set up an FFT measurement

The following procedure is an example of calls to set up an FFT measurement for IF channel 1. Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

The user must write a program for retrieving the FFT data using the hpe650x_**getTraceLength** and hpe650x_**getFFTTrace** commands.

```
result= hpe650x_init("VXI0 : : 43 : : INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result= hpe650x_setAnalogFilter(InstrumentID, 0, 0);
result= hpe650x_activateAutoranging(InstrumentID, 0);
result= hpe650x_setFFTDDCNumber(InstrumentID, 0, 0, 4);
result= hpe650x_setFFTLength(InstrumentID, 0, 0, 4096);
result= hpe650x_setFFTAverages(InstrumentID, 0, 0, 10);
result= hpe650x_setFFTWindowType(InstrumentID, 0, 0, 0);
result= hpe650x_setReturnAllFFTData(InstrumentID, 0, 0, 0);
result= hpe650x_setTunerFrequency(InstrumentID, 0, 20000000);
result= hpe650x_setDDCFrequency(InstrumentID, 0, 0, 20000000);
result= hpe650x_startFFT(InstrumentID, 0, 0);
        {
        //To Do: Put your program code here to retrieve and process data.
        }
result= hpe650x_stopFFT(InstrumentID, 0, 0);
result= hpe650x_close(InstrumentID);
```

Note

A complete, fully functional C++ search application, with source code and a graphical user interface, is available on the Internet at:

http://wwwdb.tm.agilent.com/cgi-bin/DSP/tmoArsenal.cgi?FP=productArsenal&Area=Datasheet&Template=overview&Tid=HPE6501A&Language=English&model_no=HP+E6501A

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To change tuner frequency

The following procedure is an example of how to change the tuner frequency for the IF channel 1 tuner from the initial 20 MHz specified in the hpe650x_**initIFChannel** command to 40 MHz. Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

```
result = hpe650x_init("VXI0::43::INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);

.
.
.

result= hpe650x_setTunerFrequency(InstrumentID, 0, 40000000);

.
.
.

result= hpe650x_close(InstrumentID);
```

To change the IF bandpass filter setting

The following procedure is an example of how to change the IF bandpass filter for IF channel 1 from the default value of 8 MHz to 700 kHz. Note that the index to the array of valid IF bandwidths is as follows:

0 = 30 kHz
1 = 700 kHz
2 = 8 MHz

Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

```
result = hpe650x_init("VXI0::43::INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result= hpe650x_setAnalogFilter(InstrumentID, 0, 1);

.
.
.

result = hpe650x_close(InstrumentID);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To set the IF gain

The following procedure is an example of how to change the IF gain for IF channel 1 from autoranging to a fixed gain. Refer to Table 3-12 for the index number (in parenthesis) that corresponds to the desired input range setting.

Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

Note

The dBm numbers in Table 3-12 correspond to the maximum recommended input level for a given gain setting. IF gain will be maximized at index 60 (-48 dBm), while IF attenuation will be maximized at index 0 (12 dBm). Due to input amplifier distortion, applying power levels greater than 0 dBm into the IFP is not recommended.

Table 3-12 Index to Input Range Settings

(0)	12 dBm	(16)	-4 dBm	(32)	-20 dBm	(48)	-36 dBm
(2)	10 dBm	(18)	-6 dBm	(34)	-22 dBm	(50)	-38 dBm
(4)	8 dBm	(20)	-8 dBm	(36)	-24 dBm	(52)	-40 dBm
(6)	6 dBm	(22)	-10 dBm	(38)	-26 dBm	(54)	-42 dBm
(8)	4 dBm	(24)	-12 dBm	(40)	-28 dBm	(56)	-44 dBm
(10)	2 dBm	(26)	-14 dBm	(42)	-30 dBm	(58)	-46 dBm
(12)	0 dBm	(28)	-16 dBm	(44)	-32 dBm	(60)	-48 dBm
(14)	-2 dBm	(30)	-18 dBm	(46)	-34 dBm		

```
result= hpe650x_init("VXI0 : : 43 : : INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result  hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result= hpe650x_setIFGain(InstrumentID, 0,10);

.
.
.

result= hpe650x_close(InstrumentID);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To set tuner input attenuation

The following procedure is an example of how to change the tuner input attenuation for IF channel 1 to 30 dB. Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

```
result= hpe650x_init("VXI0 : : 43 : : INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result = hpe650x_setTunerAttenuation(InstrumentID, 0, 30);

.
.
.

result= hpe650x_close(InstrumentID);
```

To set search mode resolution bandwidth

The following procedure is an example of how to change the search mode resolution bandwidth for IF channel 1 to 462 kHz.

Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

```
result = hpe650x_init("VXI0 : : 43 : : INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result= hpe650x_setSearchResolutionBWParameters(InstrumentID, 0, 0,35,8000);

.
.
.

result= hpe650x_close(InstrumentID);
```

To set span in monitor mode

The following procedure is an example of how to change the monitor mode span for IF channel 1 to 25 kHz. The span is equal to the digital (DDC) IF bandwidth setting (247 Hz to 462 kHz). Refer to Table 3-2 for the bandwidth index numbers.

Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

```
result= hpe650x_init("VXI0 : : 43 : : INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result = hpe650x_setDigitalIFBandwidth(InstrumentID, 0, 12);

.
.
.

result= hpe650x_close(InstrumentID);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To set mezzanine data select mode

The following procedure is an example of how to change the mezzanine data select mode so that data is routed to all DDCs. The default is mode 1.

```
result= hpe650x_init("VXI0::43::INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result= hpe650x_setMezzanineDataSelectMode(InstrumentID, 4);
.
.
.
result= hpe650x_close(InstrumentID);
```

To turn the tuner 10 MHz reference off

The following procedure is an example of how to turn off the 10 MHz reference for the IF channel 1 tuner. The default is on. A value of 0 indicates that an external source will be used.

Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

```
result= hpe650x_init("VXI0::43::INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result= hpe650x_selectTuner10MHzReference(InstrumentID, 0, 0);
.
.
.
result= hpe650x_close(InstrumentID);
```

To turn the IF processor 10 MHz reference on

The following procedure is an example of how to turn on the 10 MHz reference in the IF processor. The default is off.

Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

```
result= hpe650x_init("VXI0::43::INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result= hpe650x_setIF10MHzReferenceOut(InstrumentID, enable);
.
.
.
result= hpe650x_close(InstrumentID);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To activate automatic frequency control

The following procedure is an example of how to activate automatic frequency control for DDC 0 on the mezzanine in IF channel 1.

Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

```
result= hpe650x_init("VXI0 : : 43 : : INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result= hpe650x_activateAFC(InstrumentID, 0, 0);

.
.
.

result= hpe650x_close(InstrumentID);
```

To lock autoranging

The following procedure is an example of how to lock autoranging for IF channel 1. This function is useful when performing any process where gain changes are not desirable.

Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

```
result= hpe650x_init("VXI0 : : 43 : : INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result= hpe650x_setAutorangeLock(InstrumentID, 0, VI_TRUE, VI_FALSE,5);

.
.
.

result= hpe650x_close(InstrumentID);
```

To set up dynamic range optimization

The following procedure is an example of how to set up dynamic range optimization (DRO) for IF channel 1.

DRO works in conjunction with the autoranging gain to optimize the receiver's dynamic range. Autoranging maintains the optimum level at the ADC, while DRO maintains the optimum level at the DDC input. The DRO uses a window comparator to monitor the peak composite signal in the analog passband. To avoid responding to momentary fluctuations in signal amplitudes, the attack and decay commands are used. When the peak signal level increases above the upper threshold of the window and remains there for a time equal to the DRO attack time setting, the DRO immediately re-optimizes the correction RAM. Conversely, when the peak signal level decreases below the lower threshold of the window and remains there for a time equal to the DRO decay time setting, the DRO re-optimizes the correction RAM.

Refer to Chapter 4 for a complete explanation of the attack and decay response times of the DRO.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The response times that can be set are 500 µsec to 1 sec, which are set by passing in units between 2 and 2000. A value of 2000 equals 1 sec (approximately). One time unit is approximately 500 µsec.

Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

```
result= hpe650x_init("VXI0::43::INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result= hpe650x_setDROAttackTime(InstrumentID, 0, 1);
result= hpe650x_setDRODecayTime(InstrumentID, 0, 1000);

.
.
.

result= hpe650x_close(InstrumentID);
```

To set up a channelized power measurement

The following procedure is an example of how to set up a channelized power measurement using the received signal strength indication (RSSI) function for DDC 1 in IF channel 1.

The user must write a program for retrieving the RSSI value using the hpe650x_**getRSSIvalue** command.

Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

```
result= hpe650x_init("VXI0::43::INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result= hpe650x_setRSSIMeasTime(InstrumentID, 0, 10);
result= hpe650x_setDDCFrequency(InstrumentID, 0, 0, 1, 20000000);
result= hpe650x_startRSSI(InstrumentID, 0, 0);
        {
        //To Do: Put your program code here to retrieve and process data.
        }
result= hpe650x_stopRSSI(InstrumentID, 0, 0);
result= hpe650x_close(InstrumentID);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To set up and start a monitor process

The following procedure is an example of how to set up a monitor process for IF channel 1 with a center frequency of 100 MHz and a span of 8 MHz (full span). Note that the full span setting is a special case. Full span is selected by setting the DDCnum parameter in the hpe650x_**setFFTDDCNumber** command to 5 for DDC number 5.

The user must write a program for retrieving the FFT data using the hpe650x_**getTraceLength** and hpe650x_**getFFTTrace** commands.

Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

```
result= hpe650x_init("VXI0 : : 43 : : INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 20000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result= hpe650x_setFFTDDCNumber(InstrumentID, 0, 0, 5);
result= hpe650x_setFFTPostDemod(InstrumentID, 0, 0, 0);
result= hpe650x_setFFTLength(InstrumentID, 0, 0, 4096);
result= hpe650x_setFFTAverages(InstrumentID, 0, 0, 1);
result= hpe650x_setFFTWindowType(InstrumentID, 0, 0, 0);
result= hpe650x_setReturnAllFFTData(InstrumentID, 0, 0, 0);
result= hpe650x_setTunerFrequency(InstrumentID, 0, 100000000);
result= hpe650x_setDDCFrequency(InstrumentID, 0, 0, 100000000);
result= hpe650x_startFFT(InstrumentID, 0, 0);
        {
        //To Do: Put your program code here to retrieve and process data.
        }
result= hpe650x_stopFFT(InstrumentID, 0, 0);
result= hpe650x_close(InstrumentID);
```

Note

A complete, fully functional C++ search application, with source code and a graphical user interface, is available on the Internet at:

http://wwwdb.tm.agilent.com/cgi-bin/DSP/tmoArsenal.cgi?FP=productArsenal&Area=Datasheet&Template=overview&Tid=HPE6501A&Language=English&model_no=HP+E6501A

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To set up demodulation, turn on an audio channel, and set squelch

The following procedure is an example of how to set up FM demodulation on a 100 MHz signal for DDC 1 and DAC 1 in IF channel 1, and listen to the signal.

Mezzanine data select mode 1 is the default mode. Include the hpe650x_**setMezzanineDataSelectMode** command if you want to change to a mezzanine data select mode other than mode 1.

```
result= hpe650x_init("VXI0::43::INSTR", VI_TRUE, VI_TRUE, &InstrumentID);
result= hpe650x_initIFChannel(InstrumentID, 0, VI_TRUE, 100000000, 41, 42, 40);
result= hpe650x_setMonitoringMode(InstrumentID, 0);
result= hpe650x_setDemodType(InstrumentID, 0, 0, 0);
result= hpe650x_setDigitalIFBandwidth(InstrumentID, 0, 36);
result= hpe650x_turnOnAudioChannel(InstrumentID, 0, 0, 0);
result= hpe650x_setVolumeLevel(InstrumentID, 0, 0, 50);
result= hpe650x_setSquelchLevel(InstrumentID, 0, 20);
result= hpe650x_setSquelchState(InstrumentID, 0, VI_TRUE);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Multi-Threading Considerations

This example shows a code sample of a search based application that is implemented using multi-threading. It is important to note how thread blocking is implemented in this example to illustrate how to avoid issues of updating parameters while a search session is active and attempting to send and receive data simultaneously via the VISA library.

The code is taken from a Windows NT® and Microsoft® Foundations Class (MFC) based sample application.

The constructor below establishes a hpe650x session, and sets some of the initial search parameters.

```
CSearchSampleView::CSearchSampleView()
        : CFormView(CSearchSampleView::IDD)
{
        ViStatus result;

        //{{AFX_DATA_INIT(CSearchSampleView)
        m_start = _T("800.000");
        m_stop = _T("900.000");
        m_resbw =  T(" 20.898");
        //}}AFX_DATA_INIT


        mf_start    = 800E6;  // Check out the cellular band
        mf_stop     = 900E6;

        mb_havetrace = false;

        mb_lock     = false;
        mb_running  = false;
        mb_restart  = false;

        mi_resbw      5;  // Res BW set using an index to an enumerated list

        // Start the session with a bunch of defaulted parameters...

        result = hpe650x_init( IFProcessorADDR, 1, 1, &mysession);
        if( IsErr( result)) DoErrorMessage( result);
        result = hpe650x_initIFChannel( mysession, 0, 1, 20000000, LOADDR,
                ONEGHZADDR, THREEGHZADDR);
        if( IsErr( result)) DoErrorMessage( result);

        result   hpe650x_setSearchMode( mysession, MEZZANINE);
        if( IsErr( result)) DoErrorMessage( result);

        result = hpe650x_activateAutoranging( mysession, IFCHANNEL);
        result = hpe650x_setSearchType( mysession, MEZZANINE, DSPDECIMATION);
        result = hpe650x_setSearchOutputTraceLength( mysession,MEZZANINE,
                        SEARCHLENGTH);
        result = hpe650x_setSearchResolutionBW( mysession, MEZZANINE,
                        mi_resbw);
        result = hpe650x_setSearchSpan( mysession, MEZZANINE, mf_start,
                        mf_stop);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This UI service event starts the worker-thread to begin searching.

```
void CSearchSampleView::OnGo()
{
        if( !mb_running)
        {
                mb_running = true;
                AfxBeginThread(ProcessSearchData, this);
        }
}
// The following function is called as a worker-thread to provide
// search data services.
//
// The lpParam passed in is a reference to the CSearchSampleView object to it
// can refer to member variables

UINT ProcessSearchData( LPVOID lpParam)
{
        CSearchSampleView *theView = (CSearchSampleView*) lpParam;

        ViStatus result;

        ViReal64 amplitudes[ 8192], trash[ 8192];

        // Set the process lock on
        theView->mb_lock = true;

        // Fire up the search
        result = hpe650x_startSearch( theView->mysession, MEZZANINE);

        Sleep( 500); // Let the DSP get going before burdening it with a call for data
        do
        {
                        result = hpe650x_getSearchTrace( theView->mysession,
                                        MEZZANINE, amplitudes, trash);

                // Draw only if successful results..
                if( !theView->IsErr( result))
                                theView->DrawSearchTrace( SEARCHLENGTH, amplitudes);
                Sleep( TRACEDELAY); // Give up a few clock cycles
        } while( theView->mb_running); // Did someone want us to quit?
        result = hpe650x_stopSearch( theView->mysession, MEZZANINE);

        // Release the process lock
        theView->mb_lock = false;
        return 0;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This UI service event shows changing resolution bandwidth based on the user's interaction with a control. Notice that before actually changing the value on the hardware, the search worker-thread is blocked, then restarted once the value is changed.

```
void CSearchSampleView::OnDeltaposSpinresbw(NMHDR* pNMHDR, LRESULT* pResult)
{
    ViReal64   bandwidth;
    char       buffer[ 12];

    NM_UPDOWN* pNMUpDown = (NM_UPDOWN*)pNMHDR;

    int dir = pNMUpDown->iDelta;
    int resbw = mi_resbw - dir;

    // Make sure we stay within the possible enumerated resolution bandwidths.
    if( resbw < 0   resbw > 35)
        resbw = mi_resbw;

    mi_resbw = resbw;
    BlockWorkerThread();

    ViStatus result = hpe650x_setSearchResolutionBW( mysession, MEZZANINE,
                mi_resbw);
    result = hpe650x_getSearchResolutionBW( mysession, MEZZANINE, &bandwidth);

    RestartWorkerThread();

    sprintf( buffer, "%7.3f", (float) bandwidth / 1E3);  // Show in kHz
    SetDlgItemText( IDC_RESBW, (LPCTSTR) buffer);

    *pResult = 0;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Here is an example of one possible implementation of thread service handling for search.

```
void CSearchSampleView::BlockWorkerThread()
{
        if( mb_running)
        {
                mb_running = false:
                mb_restart = true:
                while( mb_lock)
                        Sleep( 1):
        }
}
void CSearchSampleView::RestartWorkerThread()
{
        if( mb_restart)
        {
                mb_lock    = false:
                mb_restart = false:
                mb_running = true:
                AfxBeginThread(ProcessSearchData, this):
        }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Synchronizing Multiple IF Processors and Capturing Data

Data synchronization may be broken into two large categories: hardware configuration and software configuration. The synchronization process is closely coupled to the capture process. That is, usually synchronization will be performed before trying to capture data. For the modules to participate in a synchronous group, they must be configured to do so. There are two places that the modules must be configured: the hardware switches located on the VXI modules, and the software.

Note

During normal receiver operation, synchronization is not needed. This functionality is provided only for those users that require a coherent multi-channel configuration.

Hardware Configuration

To synchronize multiple IF processors, the switches located on the IF processor module must be set to allow the incoming trigger to be propagated. Refer to Figure 3-24. Also, refer to "Local Bus Compatibility" in Chapter 2.

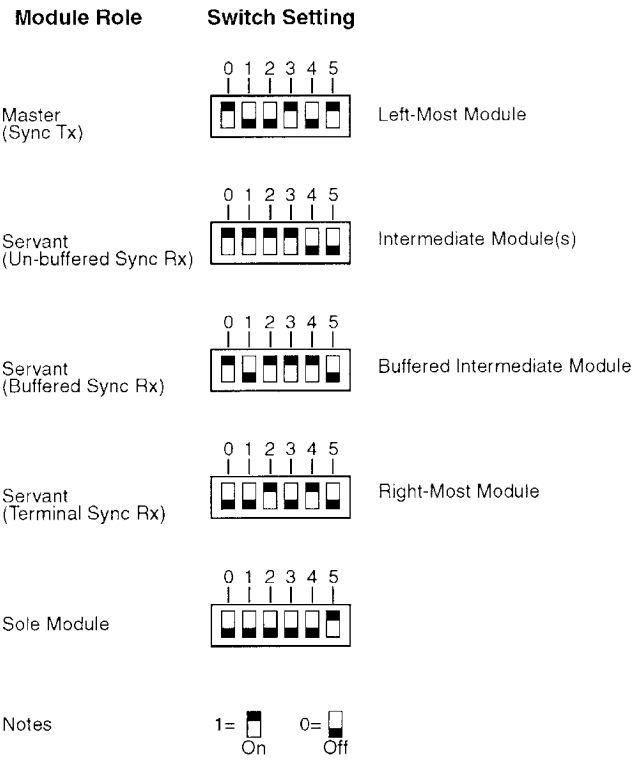


Figure 3-24 Hardware Configuration

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The switches are located on the side of the module closest to and in between the VXI bus connectors (Local Bus Switch).

The master IF processor must be the left-most module in the configuration. Note that the tuner modules are not shown because their location is not critical in the configuration. However, all IF modules*must* be installed in the VXI mainframe as a contiguous group. In addition, any VXI modules installed adjacent to IF processors configured for local bus operation must be configured for *no local bus* operation.

Note the different switch setting between the servant modules that are simply passing the trigger, and the final servant module in the chain.

Also, note that for a single module to be synchronized (that is, both mezzanines have their DDCs synchronized), the switches must be set as shown in the "Sole Module" configuration in Figure 3-24.

Software Configuration

Once the hardware is configured correctly, the first step to configuring the software is establishing a trigger. The trigger source may be either a hardware trigger or software trigger.

Note

Data capture requires software *and* hardware triggering as described below.

Software Trigger

Software triggering is defined as being able to synchronize events using only the equipment shipped with the system. An external system is not required to accomplish these tasks.

For software triggering, only synchronizing the DDCs is supported. Currently, there is no way to perform synchronous data collection or to have the DSPs from more than one mezzanine participate in a synchronous event using only software.

Hardware Trigger

This form of synchronization consists of having an external system provide a trigger to the master IF processor.

Note

The controlling computer does not have a way of knowing precisely when the trigger occurs. Therefore, the system designer *must* take into account a mechanism for knowing when the trigger happened. If commands are sent to any modules in the group, the driver cannot determine if sending the command will invalidate the capture setup.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Getting synchronous data occurs in three steps. First, the master's clock needs to be distributed to each servant module; next, the DDCs need to be synchronized, and finally, the data collection process needs to be started/stopped. The discussion that follows assumes the modules have been configured properly.

Distributing Clocks

Distributing one module's clock (the master IF) is accomplished by first designating the module as the master (issuing the hpe650x_**setMasterIFClock** command), then telling the module to set its clock destination to the VXI back-plane (issuing the hpe650x_**selectBackplaneFs** command).

Once the master module has been specified, and its clock has been distributed to the back-plane, the servant modules need to be specified (issuing the hpe650x_**setSlaveIF** command), telling each module to set its source to the VXI back-plane (hpe650x_**selectBackplaneFs** command), and finally issuing a system reset command (hpe650x_**hardSystemReset** command).

Note

The importance of issuing the specified commands in the sequence they appear cannot be overemphasized. If the commands are *not* sent in the order specified, system stability cannot be guaranteed.

Note

The hpe650x_**hardSystemReset** and hpe650x_**reset** commands are *not* the same and should not be substituted for one another. After issuing the hpe650x_**hardSystemReset** command on an IF processor, that module must be completely re-initialized

Synchronizing the DDCs

For a given configuration, the IF processors need to have some designations. That is, one IF processor per group needs to be designated as the master, while the remaining modules need to be designated servants. In this way, the command flow will be different to each IF processor depending on whether the module has been designated the master or servant.

In terms of synchronizing the DDCs, the reason a master needs to be selected is primarily for hardware triggering of the DDCs. As for the software triggering scenario, the order in which the commands are sent is not critical. However, for consistency, the master should be issued the first command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

When hardware triggering the DDCs, once the command has been issued to prepare to synchronize the last servant, then the next step in the process is to generate a trigger. It is important to realize that the E650X receiver does not have a way to detect when the trigger has occurred. When issuing a "hardware prepare to synchronize" trigger to the modules, once the final module in the group has been issued the command, the user/programmer is responsible for signaling the system that the E650X is ready for a trigger to be asserted.

Once the DDCs are synchronous, either by establishing a hardware trigger, or issuing a software trigger, it is imperative that no commands that will cause the DDCs to reset be sent to the IF processor. It is the programmer's responsibility to ensure that no commands are sent to the system that will interrupt the data stream. Examples of commands that will interrupt the data stream are the hpe650x_**setDDCFrequency** and hpe650x_**setTunerFrequency**.

From a procedural standpoint, the best solution is to set up the data paths, tune the RF section, adjust the IF processors as necessary, then synchronize the DDCs.

If hardware synchronization is chosen, and the user/programmer selects not to have the DDCs synchronous any longer (or decides to abort the process because of an invalid equipment setting), no command needs to be sent to the system. That is, the user may operate the equipment as before.

Arming the DSP

The methodology for arming the DSPs for data collection is similar to synchronizing the DDCs. Arming the DSP for data collection is really a two stage process. In the first stage, the user/programmer needs to designate what the DSP will do when the trigger arrives. This can be specified by setting the "capture" commands with the suspended value equal to true. That is, the capture commands are set up as before. However, the suspended value should be set to true. Setting the suspended value to true informs the DSP that the capture process should be done when the trigger arrives.

Currently, the way to arm the DSP is through a hardware trigger fed into the front panel connector. If the user wishes to be able to programmatically trigger the DSP, it may be possible to use the VXI Trigger Out connector on the slot zero controller.

The DSP does not have to have the DDCs synchronized before it is armed. However, if the user wishes to collect synchronous data, then they must first synchronize the DDCs.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note

Arming the DSP is a system level command. This means that when the command to arm the DSPs is sent, then no other commands should be sent to the IF processor. From the driver's perspective, not every command communicates with the IF processor. However, to be safe, no commands should be sent to the driver once the arm DSP commands have been issued, unless the user wishes to abort the process.

Locking Autorange

In capture/triggering scenarios, it is desirable to prevent the IF processors from autoranging. More importantly, it is desirable to have each of the IF processors utilizing the same correction values. This can be accomplished by turning off the autorange on all modules, gather the correction values from the master module, and setting each of the servant modules to use the same correction values.

Captured Data Format

Whether being retrieved from SRAM (over the VXI bus) or from the link ports, the data returning from the receiver is interleaved. The interleaving schemes are described in this section.

Correcting Captured Data

In general, the programmer must remember to appropriately interpret the results. It is important to note that the full scale of the ADC ranges from minus 1 volt to plus 1 volt, for a peak-to-peak range of 2 volts. Also note that attenuation and gain have been introduced by autoranging or dynamic range optimization.

To convert captured data to instantaneous input voltage, captured data must be multiplied by the result of hpe650x_**getDataCorrectionValue**. When using this command, autoranging must be locked throughout the time during which data is taken and the correction value is queried. Otherwise, an erroneous data correction value may be returned because of range changes between the time the data is captured and the time the correction value is queried.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

I/Q Data From the Link Ports

When I/Q data is output from the link port, the data is in signed 16-bit integer format (Int16) in which two words are arranged as shown in Table 3-13. The return data is a 32-bit long word, with 16 bits of I data in the upper word and 16 bits of Q data in the lower word.

Table 3-13 I/Q Data Output from Link Ports

Bits 31 through 16	Bits 15 through 0
DDC1 I[0]	DDC1 Q[0]
DDC2 I[0]	DDC2 Q[0]
DDC3 I[0]	DDC3 Q[0]
DDC4 I[0]	DDC4 Q[0]
DDC5 I[0]	DDC5 Q[0]
DDC1 I[1]	DDC1 Q[1]
DDC2 I[1]	DDC2 Q[1]
DDC3 I[1]	DDC3 Q[1]
DDC4 I[1]	DDC4 Q[1]
DDC5 I[1]	DDC5 Q[1]
...	...
DDC1 I[n-1]	DDC1 Q[n-1]
DDC2 I[n-1]	DDC2 Q[n-1]
DDC3 I[n-1]	DDC3 Q[n-1]
DDC4 I[n-1]	DDC4 Q[n-1]
DDC5 I[n-1]	DDC5 Q[n-1]

For example, if DDC1, 2, and 4 are activated, and I/Q data is being captured and output to the link port, then there will be 3 $\times n$ 32-bit words output from the link port for that mezzanine. The first long word received is for DDC1, the next long word is for DDC 2, the next for DDC 4.

I/Q Data From SRAM over VXI

When I/Q data is stored in SRAM, it must be retrieved via the VXI interface. The driver separates the I and Q data into two separate type Int16 arrays (one for I and one for Q) and then returns the data. This interleaving format is described in Table 3-14. This is accomplished by using the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

hpe650x_**getCaptureDigitalIQData** command. For an example, see the Scenario 1 sample code.

Table 3-14 I/Q Data Output from SRAM via VXI

From I Data Parameter	From Q Data Parameter
DDC1 I[0]	DDC1 Q[0]
DDC2 I[0]	DDC2 Q[0]
DDC3 I[0]	DDC3 Q[0]
DDC4 I[0]	DDC4 Q[0]
DDC5 I[0]	DDC5 Q[0]
DDC1 I[1]	DDC1 Q[1]
DDC2 I[1]	DDC2 Q[1]
DDC3 I[1]	DDC3 Q[1]
DDC4 I[1]	DDC4 Q[1]
DDC5 I[1]	DDC5 Q[1]
...	...
DDC1 I[n-1]	DDC1 Q[n-1]
DDC2 I[n-1]	DDC2 Q[n-1]
DDC3 I[n-1]	DDC3 Q[n-1]
DDC4 I[n-1]	DDC4 Q[n-1]
DDC5 I[n-1]	DDC5 Q[n-1]

ADC Data (Full Span) from the Link Ports

These samples are in offset binary format (unsigned 16-bit integers, UInt16). To restore a sample to its original 2's complement (signed integer, Int16) format, 32k (32 × 1024) must be subtracted from each sample.

The link port ADC data is sent as a total of four 16-bit words of ADC data returned in two 32-bit words. Note that both link ports (four bits each) are required to interface the E6404A and third party SHARC DSP chips. The data are arranged as shown in Table 3-15. Also note that ADC data contains a large scale dither signal, if it has not been disabled.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

See Figure 3-5 to understand the architecture associated with full rate transfers.

Table 3-15 Full Rate ADC Data Output from Link Ports

Bits 31 through 16	Bits 15 through 0
ADC sample [1]	ADC sample [0]
ADC sample [3]	ADC sample [2]
ADC sample [5]	ADC sample [4]
ADC sample [7]	ADC sample [6]
...	...
ADC sample [n-3]	ADC sample [n-4]
ADC sample [n-1]	ADC sample [n-2]

ADC Data (Full Span) from SRAM ADC Data

When I/Q data is stored in SRAM, it must be retrieved via the VXI interface using the hpe650x_**getCaptureFullRateADCData** command. This data is retrieved from SRAM by the driver as a vector of signed 16-bit integer values (Int16). Also note that ADC data contains a large scale dither signal, if it has not been disabled. The format is shown in Table 3-16. For an example, see the Scenario 14 sample code.

Table 3-16 Full Rate ADC Data Output from SRAM via VXI

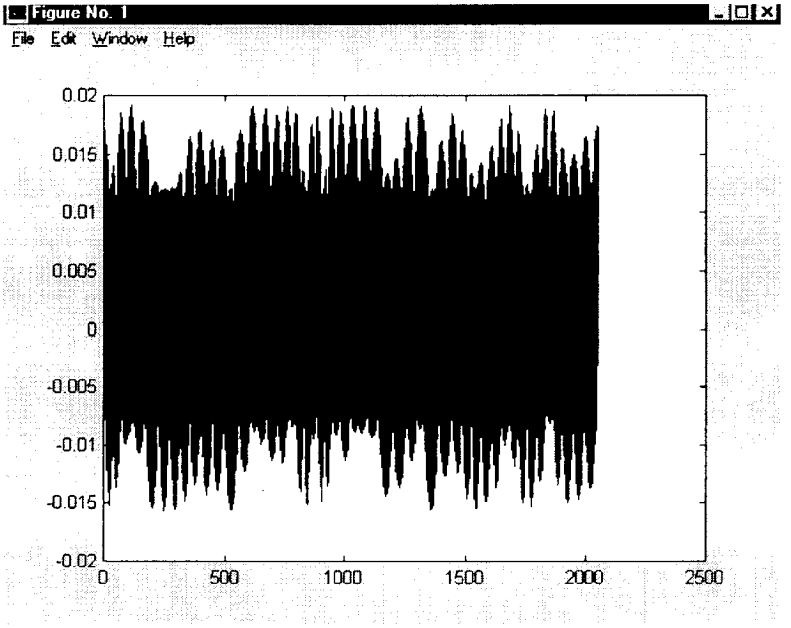
Data Parameter
ADC sample [0]
ADC sample [1]
ADC sample [2]
...
ADC sample [n-2]
ADC sample [n-1]

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MatLab Example

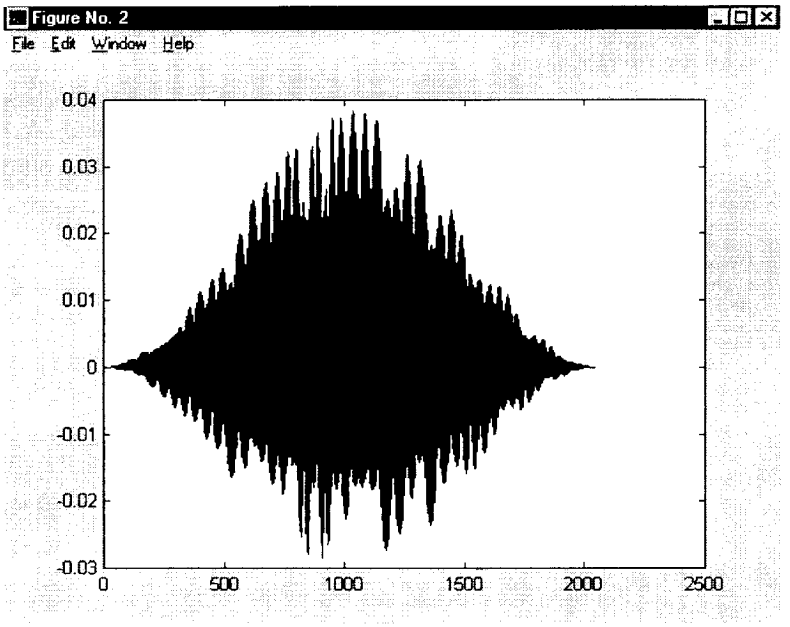
Some sample Matlab code for producing an FFT spectrum of ADC data is shown below, followed by typical displays of the waveforms.

```
fid=fopen('siggen2.txtdata','r') %note that siggen2.txtdata had the correction factor applied
test=fscanf(fid'%g', [1,248]);
siglen=length(test);
figure(1)
plot(test) %time domain
figure(2)
test=2*(test.*hanning(siglen)'); % correction factor for single sinusoid in middle of hanning window is 2
plot(test) %time domain with hanning window
figure(3)
plot(20*log10(abs(fft(test)/siglen)))
max(20*log10(abs(fft(test)/siglen)))+3 % since two tones need to ad 3 dB to get input signal peak value
```

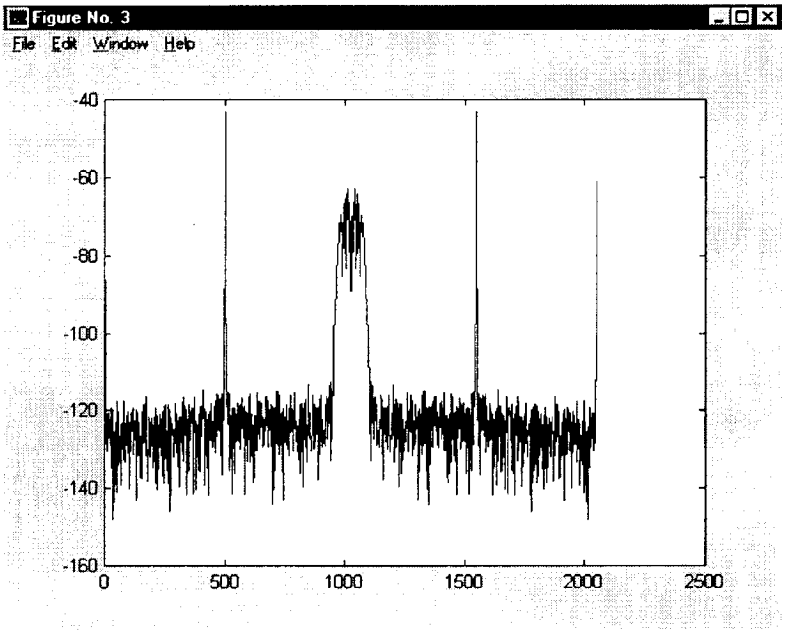


ADC data in time domain for "figure (1)" as viewed by Matlab program

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



ADC data in time domain with Hanning window for "figure (2)"



ADC data in frequency domain with Hanning window and all correction factors for "figure (3)"

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Sending Indefinite Samples

Capturing indefinite samples is a special case that offers additional flexibility to the user. However, sending indefinite samples has different meaning depending on the context (full rate data, or I/Q and whether or not the data is first captured into SRAM).

In general, sending indefinite samples when capturing I/Q data will have different meanings depending on if the data is stored in SRAM or not. If the data is to be captured in SRAM (not streamed out the link ports), then sending indefinite samples will cause the DSP to continually write over the SRAM values in a FIFO fashion.

If the data is sent to the link port (not stored in SRAM), then sending indefinite samples will cause the capture process to run until a "stop capture" command is sent.

In all cases, the data is retrieved from the DSP in 32 bit, unsigned integer format (ViUInt32).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Data Collection Programming Examples

This section provides programming examples for data collection processes. In addition, this section provides a methodology for finding the data collection programming example that is specific to your task. Currently, there are 15 possible data collection scenarios, as shown in Figure 3-25.

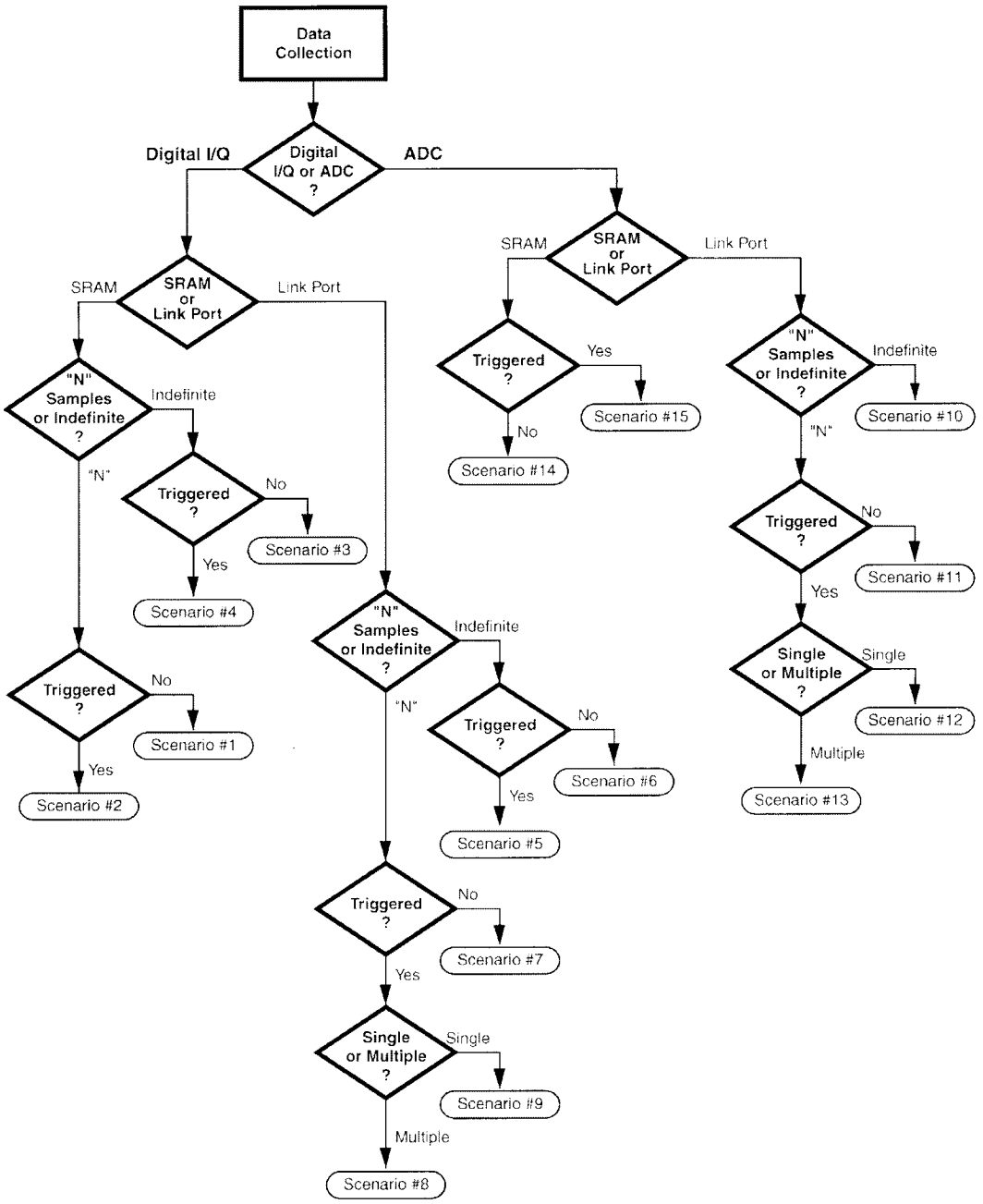


Figure 3-25 Data Collection Scenario Flow Chart

First, find the scenario number that corresponds to your specific task using the flow chart in Figure 3-25. Then, refer to the corresponding programming example in this section. A tabular version of Figure 3-25 is shown in the Table 3-17.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-17 Data Collection Scenarios

Capture Scenario Number	Data Format	SRAM Capture	Data Output Path	Number of Samples	Triggered or Freerun	Triggered - Multiple or Single
1	I/Q	Yes	VXI Bus	N	Freerun	
2	I/Q	Yes	VXI Bus	N	Triggered	Single
3	I/Q	Yes	VXI Bus	INDEF	Freerun	
4	I/Q	Yes	VXI Bus	INDEF	Triggered	Single
7	I/Q	No	Link port	N	Freerun	
9	I/Q	No	Link port	N	Triggered	Single
8	I/Q	No	Link port	N	Triggered	Multiple
6	I/Q	No	Link port	INDEF	Freerun	
5	I/Q	No	Link port	INDEF	Triggered	Single
14	ADC	Yes	VXI Bus	N	Freerun	
15	ADC	Yes	VXI Bus	N	Triggered	Single
11	ADC	No	Link port	N	Freerun	
12	ADC	No	Link port	N	Triggered	Single
13	ADC	No	Link port	N	Triggered	Multiple
10	ADC	No	Link port	INDEF	Triggered	Single

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DMA Block Size Considerations

When using scenarios 5, 6, 7, 8, and 9, which transmit digital I/Q data using the top link port connector on the IF processor, the recommended and tested data transfer process is the DMA method. Care must be taken to ensure that the firmware running on the receiving device (for instance, the DSP card) uses the same DMA block size as the E650XA receiver. For example, if you wish to receive 10000 digital I/Q samples from 2 DDCs at 30 kHz BW, the DMA block size is 144; therefore, (144 X 2 channels) = 288 samples transmitted per output packet from the link port. Further, 10000 / 288 = 69 with a remainder of 64; therefore, to complete the entire transfer, a total of 69 packet transfers of size 288 plus one final packet transfer of size 128 would be required. Refer to Table 3-18 for the DMA block size values.

Table 3-18 DMA Block Size Per Active Channel

DMA Block Size	DDC Bandwidth	DMA Block Size	DDC Bandwidth
2	250 Hz	240	75 kHz
3	500 Hz	246	85 kHz
4	750 Hz	276	95 kHz
6	1 kHz	306	110 kHz
14	2.4 kHz	335	126 kHz
16	3 kHz	372	139 kHz
30	5 kHz	372	156 kHz
34	6.25 kHz	372	177 kHz
60	10 kHz	376	189 kHz
68	12.5 kHz	416	204 kHz
80	15 kHz	459	221 kHz
108	20 kHz	500	241 kHz
135	25 kHz	550	265 kHz
144	30 kHz	600	295 kHz
126	35 kHz	644	332 kHz
162	45 kHz	672	380 kHz
175	54 kHz	702	442 kHz
231	66 kHz	738	468 kHz

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Common Functions for Collection

The code in the **Commonex.c** file shown below defines functions that are common to most of the fifteen capture scenarios found in this section. When examining the code for a particular capture scenario, refer back to commonex.c as necessary.

```
#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include <time.h>
#define MSEC_PER_CLOCK_TICK (float)(1000.0 / CLOCKS_PER_SEC)
void mSleep(ViInt32 milli_sec);
/* configures master if channel for synchronized autorange operation */
ViStatus ConfigureMasterChannel( ViSession s, ViInt32 IFChnl, ViPInt32 RAMpg, ViPInt32 Gp, ViPInt32 Gsp, ViPInt32 sysG)
{
    ViStatus result;
    ViInt32  att; /* This is a throw away parameter for sync */
    /* this function locks both fast HW autorange and disables DRO */
    if( VI_SUCCESS != (result = hpe650x_setAutorangeLock( s, IFChnl, VI_TRUE, 0, VI_FALSE)))
        return result;
    /* this function reads the autorange state of the master channel */
    if( VI_SUCCESS != (result = hpe650x_getAutorangeState( s, IFChnl, Gp, Gsp, sysG)))
        return result;
    /* this function reads the correction ram page number of the master channel */
    if( VI_SUCCESS != (result = hpe650x_getIFAttenuator( s, IFChnl, &att, RAMpg)))
        return result;
    /* !!!! NOTE: to undo synch the user should call hpe650xsetAutorangeLock with 'lock' = VI_FALSE. 'lock' is the third parameter in the function */
    return VI_SUCCESS;
}
/* configures slave if channel for synchronized autorange operation */
ViStatus ConfigureSlaveChannel( ViSession s, ViInt32 IFChnl, ViInt32 RAMpg, ViInt32 Gp, ViInt32 Gsp, ViInt32 sysG)
{
    ViStatus result;
    /* this function disables DRO while the autorange state is being configured */
    if( VI_SUCCESS != (result = hpe650x_stopDynamicRangeOptimization( s, IFChnl)))
        return result;
    /* this function sets the autorange state of the slave channel */
    if( VI_SUCCESS != (result = hpe650x_setAutorangeState( s, IFChnl, Gp, Gsp, sysG)))
        return result;
    /* this function locks both fast HW autorange and disables DRO */
    if( VI_SUCCESS != (result = hpe650x_setAutorangeLock( s, IFChnl, VI_TRUE, RAMpg, VI_TRUE)))
        return result;
    /* !!!! NOTE: to undo synch the user should call hpe650xsetAutorangeLock with 'lock'    VI_FALSE. 'lock' is the        third parameter in the function */
    return VI_SUCCESS;
}
ViStatus ConfigureMasterIFP( ViSession s)
{
    ViStatus result;
    /* this function instructs the master if channel to send it's clock to the VXI backplane */
    if( VI_SUCCESS != (result = hpe650x_setMasterIFClock( s)))
        return result;
    /* this function instructs the master if channel to accept the VXI backplane clock */
    if( VI_SUCCESS != (result = hpe650x_selectBackplaneFs( s)))
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        return result;
    /* We must do a hard reset to reboot the DSP chips, this time with one master clock. */
    result = hpe650x_hardSystemReset( s);
    /*      It is necessary to close the VXI session after the hardSystemReset, because an initialization will be      performed in the main program, which creates a new
session. If the original session were left open, we risk      exceeding the maximum open VXI sessions.*/
    result = hpe650x_close( s);
    return VI_SUCCESS;
}
/* configures slave if channel for synchronized clock operation */
ViStatus ConfigureSlaveIFP( ViSession s)
{
    ViStatus result;
    /* this function instructs the slave if channel to configure it's clock to accept the VXI backplne clock */
    if( VI_SUCCESS != (result = hpe650x_setSlaveIFClock( s)))
        return result;
    /* this function instructs the slave if channel to accept the VXI backplane clock */
    if( VI_SUCCESS != (result = hpe650x_selectBackplaneFs( s)))
        return result;
    /* this function is a safety net since we have just switched the slave clock source */
    result = hpe650x_hardSystemReset( s);
    /* turn off VCO on slave */
    if( VI_SUCCESS != (result = hpe650x_disableVCO( s)))
        return result;
    /*      It is necessary to close the VXI session after the hardSystemReset, because an initialization will be      performed in the main program, which creates a new
session. If the original session were left open, we risk      exceeding the maximum open VXI sessions.*/
    result = hpe650x_close( s);
    return VI_SUCCESS;
}
/* configures system DDCs for synchronized operation */
ViStatus SynchronizeDDCs( ViInt32 sessions, ViSession s[], ViInt32 mezzperifp)
{
    int     i, j;
    ViStatus result;
    for( i = 0; i < sessions; i++)
        for( j = 0; j < mezzperifp; j++)
        {
            result = hpe650x_prearmDDCsForSynchronization( s[ i], j);
            if( VI_SUCCESS != result)
                return result;
        }
    for( i = 0; i < sessions; i++)
        for( j = 0; j < mezzperifp; j++)
        {
            result = hpe650x_armDDCsForSynchronization( s[ i], j, VI_TRUE);
            if( VI_SUCCESS != result)
            return result;
            mSleep(50);
        }
    return VI_SUCCESS;
}
void mSleep(ViInt32 milli_sec)

{
    clock_t start_time;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    if (milli_sec > 0)
    {
        start_time = clock();
        if (start_time == (clock_t) -1)
            return;
        while ((float)(clock() - start_time) * (float)MSEC_PER_CLOCK_TICK < (float)milli_sec);
    }
}
/*
  When called, 'error_exit' will get the description of the error that was returned and print it to the screen, along with the file name and line number of the procedure call that returned the error code. 'error_exit' will then cleanly close the instrument exit the program.
*/
ViStatus error_exit( ViSession s, ViStatus status_code, int linenum, char srcfile[] )
{
    char buff[256];
    FILE *stream;
    hpe650x_error_message(s, status_code, buff );
    stream = fopen( "errorlog.txt", "w" );
        fprintf( stream, "\nThe Driver call at line %d in file %s\n",linenum,srcfile);
        fprintf( stream, "returned error code %#010x, \"%s\".\n", status_code, buff);
    fclose( stream );
    printf("\nThe Driver call at line %d in file %s\n",linenum,srcfile);
    printf("returned error code %#010x, \"%s\".\n", status_code, buff);
    printf("Exiting ...\n");
    hpe650x_close(s);
    exit((int)status_code);
    return status_code;   /* This is only here to suppress compiler warnings */
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 1: Capture N Samples of Digital I/Q Data Across VXI Bus

Use the following programming example for capturing a specified number of digital I/Q data samples across the VXI bus without the use of a trigger to begin the capture process.

Figure 3-26 shows the main process steps for this scenario.

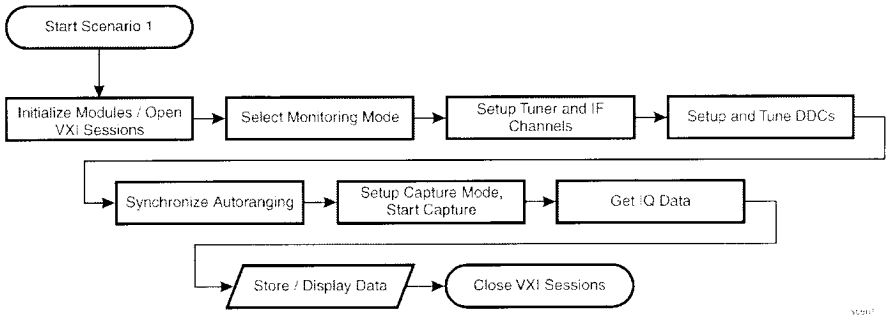


Figure 3-26 Scenario 1: Capture N Samples of Digital I/Q Data Across VXI Bus

```
/*
    Scenario 1:Capture n-samples of digital IQ data and output to
    the VXI bus with no trigger.

    Notes:
    1. The following files must exist in the same directory as the source code.
        "hpe650x.h"
        "commonex.h"
        "commonex.c"
        "visatype.h"
        "vpptype.h"

    2. The file "hpe650x.lib" must be available during linking.
    3. This program requires the installation of SRAM on Mezzanine 1 of the IF Processor whose VXI address is stored in IFPvxiID[0]. SRAM is also required on other
mezzanines in the system if more than one IFP exists or if MEZZperIFP is greater than 1.

    4. The user should read through the code modifying the following, as necessary:
        a. Constants that define the number of IF channels, IF processors and mezzanines
        b. Variables that define VXI addresses
        c. SYNC_AUTORANGE controls whether all IFP autoranging is synchronized.
        d. TunerExists should be set to VI_TRUE if a tuner exists otherwise is should be set to VI_FALSE.
        f. "fcenter" which defines the tuner's center frequency
        g. "ddcfreq" sets the frequency tuning of the DDCs relative to the IF.
        h. "ddcbw" sets the bandwidth of the DDCs. See the user documentation for a table of appropriate values.
        i. "AnalogFilter", "suspend", "format", "collect" and "output" can be set to predefined constants as described below.

    5. The commands hpe650x_init(), hpe650x_initIFChannel() and hpe650x_setMonitoringMode() must be executed before other commands, such as
hpe650x_setTunerFrequency().
    Note: This example has been simplified to operate with no triggers and with only one DDC on one mezzanine. Please see the other scenarios for examples
with multiple DDCs and mezzanines.
    Disclaimer: This code is provided AS IS. It is a sample and unsupported.

*/
#include <stdlib.h>
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS      1
/* Number of IF processors installed */
#define SYSIFPS      1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs. This is an array for which each element contains a VXI address string. This example shows a system    with only one IFP. */
ViRsrc  IFPvxiID[ SYSIFPS] = { "VXI0::43::INSTR"};
/* Extend for IFs.
      There are arrays for which each element contains a VXI address integer. The first dimension spans the system IFPs.     The second dimension spans the IF channels for each IF processor.  This example shows the VXI module addresses for     one 3 GHz tuner.  */
ViInt32 LO_log_addr[ SYSIFPS][ IFCHNLS] = { 41};
ViInt32 OneG_log_addr[ SYSIFPS][ IFCHNLS] = { 42};
ViInt32 ThreeG_log_addr[ SYSIFPS][ IFCHNLS] = { 40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_FALSE
/*      If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/*****************************************************************************
     Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1   0
#define MEZZ1  0
#define MEZZ2  1
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE      1
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING   0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY    3
#define DIGITALIQ          0
#define ADCDATA            1
#define VXIBUS             0
#define PORT               1
#define TRIGGER            1
#define FREERUN            0
#define SRAM               1
#define LINKPORT0
/*****************************************************************************/
/*
  The macro 'rcheck' saves the return status in the variable 'ret'. If the return status indicates an error, 'rcheck'    will call 'error_exit'. The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
*/
#define rcheck(A) ( ((result = A) == VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp],result, __LINE__,__FILE__)) )
int main()
{
    int             i, ifp, mezz, chan;
    ViStatusresult;
    ViSessionsessionID[ SYSIFPS];
    ViInt32     correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
    ViInt32     sample = 2000;          /*      Number of samples or 0 if taking indefinite length. */
    ViInt32     length = 2000;
    ViInt16     IData[ 2000], QData[ 2000];
    ViReal64 finit = 20e6;              /*              This number should be less than 1 GHz.*/
    ViReal64 fcenter = 2600e6;          /*      This number can be any valid frequency for the tuner.*/
    ViInt16     ddcfreq = 0;            /*      See user manual for "hpe650x_setDDCFrequency"*/
    ViInt16     ddcbw = 8;              /*  See user manual for "hpe650x_setDigitalIFBandwidth"*/
    ViInt32     AnalogFilter = IF700KHZ;    /* Other choices are IF30KHZ or IF8MHZ          */
    ViBoolean suspend = FREERUN;        /* Choices are FREERUN or TRIGGER               */
    ViBoolean format = DIGITALIQ;       /* Choices are DIGITALIQ or ADCDATA             */
    ViBoolean collect = SRAM;           /* Choices are SRAM or LINKPORT                 */
    ViBoolean output = VXIBUS;          /* Choices are VXIBUS or PORT                   */


    /* Configure the receiver state, initialize all IFP's and IF's */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
    {
        rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
        for( chan = 0; chan < IFCHNLS; chan++)
            /*      Initialize the IF channels including establishing communication to their addresses. The initial     frequency needs to be below 1 GHz.*/
            rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],     OneG_log_addr[ ifp][ chan],
ThreeG_log_addr[ ifp][ chan]));
    }

    /* Put all IFPs into monitoring mode */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
    /* Set up and tune the IFs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( chan = 0; chan < IFCHNLS; chan++)
        {
            /*      Set the tuner frequency to the value of "fcenter"*/
            rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter ));
            /*      Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
            rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));
            /*      Activate autorange once after the initialization of each mezzanine.*/
            rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
        }
    /* Set up and tune DDCs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            /* The following function can be used for each DDC installed */
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC1, RELATIVE, ddcfreq));
            /*      rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC2, RELATIVE, ddcfreq));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
*/
#define rcheck(A) ( ((result = A) == VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp],result, __LINE__,__FILE__)) )
int main()
{
    int             i, ifp, mezz, chan;
    ViStatusresult;
    ViSessionsessionID[ SYSIFPS];
    ViInt32     correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
    ViInt32     sample = 2000;          /*      Number of samples or 0 if taking indefinite length. */
    ViInt32     length = 2000;
    ViInt16     IData[ 2000], QData[ 2000];
    ViReal64 finit = 20e6;              /*              This number should be less than 1 GHz.*/
    ViReal64 fcenter = 2600e6;          /*      This number can be any valid frequency for the tuner.*/
    ViInt16     ddcfreq = 0;            /*      See user manual for "hpe650x_setDDCFrequency"*/
    ViInt16     ddcbw = 8;              /*  See user manual for "hpe650x_setDigitalIFBandwidth"*/
    ViInt32     AnalogFilter = IF700KHZ;    /* Other choices are IF30KHZ or IF8MHZ          */
    ViBoolean suspend = FREERUN;        /* Choices are FREERUN or TRIGGER               */
    ViBoolean format = DIGITALIQ;       /* Choices are DIGITALIQ or ADCDATA             */
    ViBoolean collect = SRAM;           /* Choices are SRAM or LINKPORT                 */
    ViBoolean output = VXIBUS;          /* Choices are VXIBUS or PORT                   */


    /* Configure the receiver state, initialize all IFP's and IF's */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
    {
        rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
        for( chan = 0; chan < IFCHNLS; chan++)
            /*      Initialize the IF channels including establishing communication to their addresses. The initial     frequency needs to be below 1 GHz.*/
            rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],     OneG_log_addr[ ifp][ chan],
ThreeG_log_addr[ ifp][ chan]));
    }

    /* Put all IFPs into monitoring mode */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
    /* Set up and tune the IFs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( chan = 0; chan < IFCHNLS; chan++)
        {
            /*      Set the tuner frequency to the value of "fcenter"*/
            rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter ));
            /*      Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
            rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));
            /*      Activate autorange once after the initialization of each mezzanine.*/
            rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
        }
    /* Set up and tune DDCs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            /* The following function can be used for each DDC installed */
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC1, RELATIVE, ddcfreq));
            /*      rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC2, RELATIVE, ddcfreq));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    if( VI_SUCCESS == result)
    {
        /*  This writes the I and Q data to the file results1.txt in the same directory as the executable file.  */
        FILE *stream;
        stream = fopen( "mez1ddc1.txt", "w" );
        for( i = 0; i < length; i++ )
            fprintf( stream, "%5d%5d\n", IData[ i], QData[ i]);
        fclose( stream );

        /*This is a print statement to display the data on the monitor.*/
        for( i = 0; i < length; i++)
            printf(" I[ %4d] = %5d   Q[ %4d] = %5d\n", i, IData[ i], i, QData[ i]);

    }
    /* Close all IFPs*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        rcheck( hpe650x_close(sessionID[ ifp]));
    return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 2: Capture N Samples of Digital I/Q Data From VXI Bus Using a Trigger

Use the following programming example for capturing a specified number of digital I/Q data samples across the VXI bus using a trigger to begin the capture process.

Figure 3-27 shows the main process steps for this scenario.

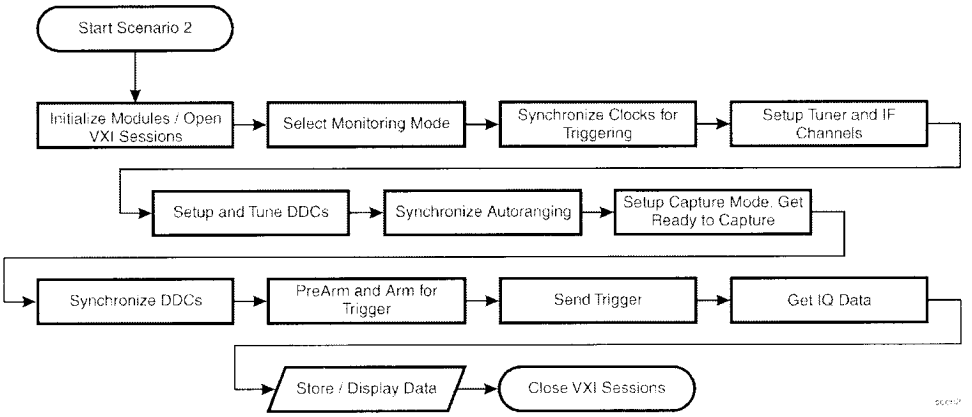


Figure 3-27 Scenario 2: Capture N Samples of Digital I/Q Data From VXI Bus Using a Trigger

```
/*
Scenario 2:Capture n-samples of digital IQ data and output to
the VXI bus with an external trigger.

Notes:

1. The following files must exist in the same directory as the
   source code.
   "hpe650x.h"
   "commonex.h"
   "commonex.c"
   "visatype.h"
   "vpptype.h"

2. The file "hpe650x.lib" must be available during linking.
3. This program requires the installation of SRAM on Mezzanine 1
   of the IF Processor whose VXI address is stored in IFPvxiID[0].
   SRAM is also required on other mezzanines in the system
   if more than one IFP exists or if MEZZperIFP is greater than 1.

4. The user should read through the code modifying the
   following, as necessary:
   a. Constants that define the number of IF channels, IF
      processors and mezzanines
   b. Variables that define VXI addresses
   c. SYNC_AUTORANGE controls whether all IFP autoranging
      is synchronized.
   d. TunerExists should be set to VI_TRUE if a tuner exists
      otherwise is should be set to VI_FALSE.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

e. SYNC_CLOCKS is set to VI_TRUE when coherent measurements across multiple mezzanines and/or multiple IFPs is desired.
f. "fcenter" which defines the tuner's center frequency
g. "ddcfreq" sets the frequency tuning of the DDCs relative to the IF.
h. "ddcbw" sets the bandwidth of the DDCs. See the user documentation for a table of appropriate values.
i. "AnalogFilter", "suspend", "format", "collect" and "output" can be set to predefined constants as described below.

5. The commands hpe650x_init(), hpe650x_initIFChannel() and hpe650x_setMonitoringMode() must be executed before other commands, such as hpe650x_setTunerFrequency().
6. To synchronize DDCs across multiple mezzanines, the third parameter of hpe650x_armDDCsForSynchronization in "commonex.c" must be set to VI_TRUE. In addition, two trigger signals must be sent from a external source to mezzanine 1.
7. IMPORTANT: In this example, the two trigger signals MUST be sent when prompted, and BEFORE ANY SUBSEQUENT COMMANDS, otherwise the program will hang at the hpe650x_getCaptureDigitalIQData() command.

Disclaimer: This code is provided AS IS. It is a sample and unsupported.

```
*/
#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS     1
/* Number of IF processors installed */
#define SYSIFPS     1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs.
      This is an array for which each element contains a VXI address string.
      This example shows a system with only one IFP. */
ViRsrc  IFPvxiID[ SYSIFPS] = { "VXI0::43::INSTR"};
/* Extend for IFs.
      There are arrays for which each element contains a VXI address integer.
      The first dimension spans the system IFPs.  The second dimension spans
      the IF channels for each IF processor.  This example shows the VXI
      module addresses for one 3 GHz tuner.  */
ViInt32 LO_log_addr[ SYSIFPS][ IFCHNLS] = { 41};
ViInt32 OneG_log_addr[ SYSIFPS][ IFCHNLS] = { 42};
ViInt32 ThreeG_log_addr[ SYSIFPS][ IFCHNLS] = { 40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_FALSE
/*      If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/* Use this to switch on synchronized IFP clocks (VI_TRUE).  This MUST be
      done if there is more than one mezzanine on an IFP, even if only one
      mezzanine is used in the measurement.  This is because of how the
      trigger signal is propagated throughout multiple mezzanines.*/
#define SYNC_CLOCKS VI_TRUE
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/*****************************************************************************
    Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1   0
#define MEZZ1  0
#define MEZZ2  1
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE      1
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING   0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY    3
#define DIGITALIQ         0
#define ADCDATA           1
#define VXIBUS            0
#define PORT              1
#define TRIGGER           1
#define FREERUN           0
#define SRAM              1
#define LINKPORT0
/*****************************************************************************/
/*    The macro 'rcheck' saves the return status in the variable 'ret'.
      If the return status indicates an error, 'rcheck' will call 'error_exit'.
      The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.*/
#define rcheck(A) ( ((result = A) == VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp].result.__LINE__,__FILE__)) )
int main()
{
      int                 i, ifp, mezz, chan;
      ViStatusresult;
      ViSessionsessionID[ SYSIFPS];
      ViInt32     correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
      ViInt32     sample    2000;
      ViInt32     length    2000;
      ViInt32     totlength = 4000;
      ViInt16     IData[ 4000], QData[ 4000];
      ViReal64 finit = 20e6;                  /*   This number should be less than 1 GHz.          */
      ViReal64 fcenter = 2600e6;              /*   This number can be any valid frequency for the tuner.  */
      ViInt16     ddcfreq = 0;                /*   See user manual for "hpe650x_setDDCFrequency"   */
      ViInt16     ddcbw = 8;                  /*   See user manual for "hpe650x_setDigitalIFBandwidth"  */

      ViInt32     AnalogFilter = IF700KHZ;        /* Other choices are IF30KHZ or IF8MHZ          */
      ViBoolean suspend = TRIGGER;                /* Choices are FREERUN or TRIGGER               */
      ViBoolean format = DIGITALIQ;               /* Choices are DIGITALIQ or ADCDATA             */
      ViBoolean collect = SRAM;                   /* Choices are SRAM or LINKPORT                 */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    ViBoolean output = VXIBUS;                  /* Choices are VXIBUS or PORT                    */
    /* Configure the receiver state, initialize all IFP's and IF's */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
    {
        rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp])):
        for( chan = 0; chan < IFCHNLS; chan++)
            /*    Initialize the IF channels including establishing communication to their addresses. The initial    frequency needs to be below 1 GHz.*/
            rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],    OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan])):
    }


    /* Put all IFPs into monitoring mode */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz)):
    /*    Synchronize clocks between all mezzanines and IFPs.  The IFP at sessionID[ 0] is always the master in this    example.*/
    if( VI_TRUE == SYNC_CLOCKS)
    {
        /* Instructs the master IFP to send its clock out VXI.  Since the sample clock is also the DSP clock, this    action risks causing the DSPs to hang.  Therefore
all the modulues are also reinitialized to reboot the    DSPs.*/
        rcheck( ConfigureMasterIFP( sessionID[ 0])):
        rcheck( hpe650x_init( IFPvxiID[ IF1], VI_TRUE, VI_TRUE, &sessionID[ IF1])):
        for( chan = 0; chan < IFCHNLS; chan++)
            rcheck( hpe650x_initIFChannel( sessionID[ IF1], chan, TunerExists, finit, LO_log_addr[ IF1][ chan],    OneG_log_addr[ IF1][ chan],
ThreeG_log_addr[ IF1][ chan])):
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ IF1], mezz)):
        /* Now configure all slave IFPs.  The slaves are instructed to accept the master clock.  Finally, the slaves    must be reinitialized to boot the DSPs.*/
        for( ifp = 1; ifp < SYSIFPS; ifp++)
        {
            rcheck( ConfigureSlaveIFP( sessionID[ ifp])):
            rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp])):
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],    OneG_log_addr[ ifp][ chan],
ThreeG_log_addr[ ifp][ chan])):
            for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz)):
        }
    }
    /* Set up and tune the IFs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( chan = 0; chan < IFCHNLS; chan++)
        {
            /*    Set the tuner frequency to the value of "fcenter"*/
            rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter )):
            /*    Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
            rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter)):
            /*    Activate autorange once after the initialization of each mezzanine.*/
            rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan)):
        }


/* Set up and tune DDCs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        {
            /* The following function can be used for each DDC installed */
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC1, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC5, RELATIVE, ddcfreq));
        /*  rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC2, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC3, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC4, RELATIVE, ddcfreq));*/
            /* only need to call this function once per mezzanine */
            rcheck( hpe650x_setDigitalIFBandwidth( sessionID[ifp], mezz, ddcbw));
        }
    /* synchronize autoranging on all modules in system */
    if( VI_TRUE == SYNC_AUTORANGE)
    {
        /* sync autoranging on master channel */
        rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile,  &Gsystem));
        /* synch autoranging on the slave channels */
        if( IFCHNLS > 1)
            rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile,  Gsystem));
    }
/* Set up to do the actual data capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC1, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC5, VI_TRUE ));
        /*  rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC2, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC3, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC4, VI_TRUE ));*/
            mSleep( 50);        /* allow for DDC transients */
            rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
            rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
            rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
            rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
            rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
            rcheck( hpe650x_startCapture( sessionID[ifp], mezz ));
        }
    /*  Synchronize all DDCs*/
rcheck( SynchronizeDDCs( SYSIFPS, sessionID, MEZZperIFP));
    /*  Prearm -- required before arming.  This stops any current activity.*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_prearmDSPForDataCollection( sessionID[ ifp], mezz));



    /* Arm the master DSP.  The master is defined as sessionID[ 0] in this example.*/
    rcheck( hpe650x_armDSPForDataCollection( sessionID[ 0], MEZZ1, VI_FALSE));

    /* Arm the slave DSPs */
    if( MEZZperIFP > 1)
        rcheck( hpe650x_armDSPForDataCollection( sessionID[0], MEZZ2, VI_TRUE ));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
if( SYSIFPS > 1)
    for( ifp = 1; ifp <= SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_armDSPForDataCollection( sessionID[ifp], mezz, VI_TRUE ));

/*    Prompt user for the trigger.*/
printf("Fire the trigger.  Press Enter to continue\n");
getchar();
/* sleep while data is captured */
mSleep( 1000);
/*    The remaining code only accommodates one IFP and two Mezzanines.
It will gather measurements taken from two DDCs.
The data output will be interleaved:

I[ddc1,0] Q[ddc1,0] I[ddc2,0] Q[ddc2,0]
I[ddc1,1] Q[ddc1,1] I[ddc2,1] Q[ddc2,1]
                    ...
I[ddc1,n] Q[ddc1,n] I[ddc2,n] Q[ddc2,n]*/

/* Retrieve data from mezzanine 1 */
do
{
    /* Loop until the DSP finishes giving us the data across the bus.  Notice that we gather
    a quantity of samples defined by "totlength", which should be the total number of DDCs
    enabled for capturing times "length."In this example, we've enabled DDC1 and DDC5,
    each taking 2000 samples, making "totlength" equal to 4000.*/
    result = hpe650x_getCaptureDigitalIQData( sessionID[ 0], MEZZ1, IData, QData, &totlength);
} while ( DATANOTREADY == result);
/*  Output the captured data to the monitor or a file.*/
if( VI_SUCCESS == result)
{
    /*  This writes the I and Q data to the file results1.txt in the same directory as the executable file.  */
    FILE *stream;
    stream = fopen( "mez1ddc1.txt", "w" );
    for( i = 0; i < length; i++)
        fprintf( stream, "%5d%5d\n", IData[ i*2], QData[ i*2]);
    fclose( stream );

    stream = fopen( "mez1ddc5.txt", "w" );
    for( i = 0; i < length; i++)
        fprintf( stream, "%5d%5d\n", IData[ i*2+1], QData[ i*2+1]);
    fclose( stream );

    stream = fopen( "mez1all.txt", "w" );
    for( i = 0; i < totlength; i++)
        fprintf( stream, "%5d%5d\n", IData[ i], QData[ i]);
    fclose( stream );


    /*This is a print statement to display the data on the monitor.*/
    for( i = 0; i < totlength; i++)
        printf(" I[ %4d] = %5d   Q[ %4d] = %5d\n", i, IData[ i], i, QData[ i]);
}

/* Retrieve data from mezzanine 2 */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
do
{
        /* Loop until the DSP finishes giving us the data across the bus.  Notice that we gather
        a quantity of samples defined by "totlength", which should be the total number of DDCs
        enabled for capturing times "length."In this example, we've enabled DDC1 and DDC5,
        each taking 2000 samples, making "totlength" equal to 4000.*/
        result = hpe650x_getCaptureDigitalIQData( sessionID[ 0], MEZZ2, IData, QData, &totlength);
} while ( DATANOTREADY == result);
/*  Output the captured data to the monitor or a file.*/
if( VI_SUCCESS == result)
{
        /*  This writes the I and Q data to the file results1.txt in the same directory as the executable file.  */
        FILE *stream;
        stream = fopen( "mez2ddc1.txt", "w" );
        for( i = 0; i < length; i++)
                fprintf( stream, "%5d%5d\n", IData[ i*2], QData[ i*2]);
        fclose( stream );

        stream    fopen( "mez2ddc5.txt", "w" );
        for( i = 0; i < length; i++ )
                fprintf( stream, "%5d%5d\n", IData[ i*2+1], QData[ i*2+1]);
        fclose( stream );

        stream = fopen( "mez2all.txt", "w" );
        for( i = 0; i < totlength; i++ )
                fprintf( stream, "%5d%5d\n", IData[ i], QData[ i]);
        fclose( stream );
        /*This is a print statement to display the data on the monitor.*/
        for( i = 0; i < totlength; i++)
                printf(" I[ %4d] = %5d   Q[ %4d] = %5d\n", i, IData[ i], i, QData[ i]);
}
/* Close all IFPs*/
for( ifp = 0; ifp < SYSIFPS; ifp++ )
        rcheck( hpe650x_close(sessionID[ ifp]));
return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 3: Capture Digital I/Q Data Indefinitely Across VXI Bus

Use the following programming example for capturing an indefinite number of digital I/Q data samples across the VXI bus without the use of a trigger to begin the capture process.

Figure 3-28 shows the main process steps for this scenario.

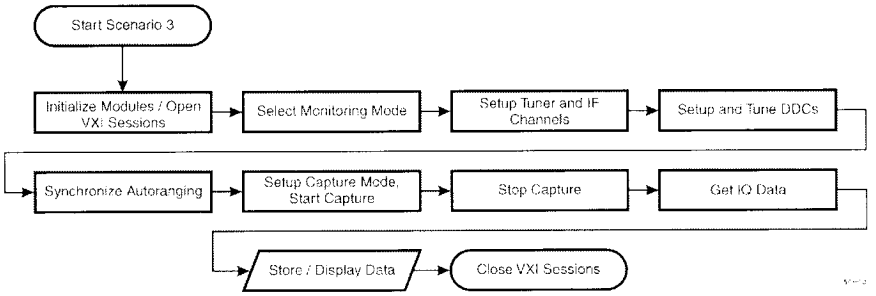


Figure 3-28 Scenario 3: Capture Digital I/Q Data Indefinitely Across VXI Bus

```
/*
Scenario 3:Capture an indefinite number of samples of digital
IQ data and output to the VXI bus without a trigger.

Notes:
1. The following files must exist in the same directory as the
   source code.
   "hpe650x.h"
   "commonex.h"
   "commonex.c"
   "visatype.h"
   "vpptype.h"

2. The file "hpe650x.lib" must be available during linking.
3. This program requires the installation of SRAM on Mezzanine 1
   of the IF Processor whose VXI address is stored in IFPvxiID[0].
   SRAM is also required on other mezzanines in the system
   if more than one IFP exists or if MEZZperIFP is greater than 1.

4. The user should read through the code modifying the
   following, as necessary:
   a. Constants that define the number of IF channels, IF
      processors and mezzanines
   b. Variables that define VXI addresses
   c. SYNC_AUTORANGE controls whether all IFP autoranging
      is synchronized.
   d. TunerExists should be set to VI_TRUE if a tuner exists
      otherwise is should be set to VI_FALSE.
   f. "fcenter" which defines the tuner's center frequency
   g. "ddcfreq" sets the frequency tuning of the DDCs relative
      to the IF.
   h. "ddcbw" sets the bandwidth of the DDCs. See the user
      documentation for a table of appropriate values.
   i. "AnalogFilter", "suspend", "format", "collect" and "output"
      can be set to predefined constants as described below.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5. The commands hpe650x_init(), hpe650x_initIFChannel() and hpe650x_setMonitoringMode() must be executed before other commands, such as hpe650x_setTunerFrequency().
6. Note: This example has been simplified to operate with no triggers. Please see the other scenarios for examples with triggers.

Disclaimer: This code is provided AS IS. It is a sample and unsupported.

```
*/
#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "Commonex.h"
/* Number of IF channels */
#define IFCHNLS      1
/* Number of IF processors installed */
#define SYSIFPS      1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs */
ViRsrc IFPvxiID[ SYSIFPS] = { "VXI0::43::INSTR"};
/* Extend for IFs.
      There are arrays for which each element contains a VXI address integer.
      The first dimension spans the system IFPs. The second dimension spans
      the IF channels for each IF processor. This example shows the VXI
      module addresses for one 3 GHz tuner. */
ViInt32 LO_log_addr[ SYSIFPS][ IFCHNLS] = { 141};
ViInt32 OneG_log_addr[ SYSIFPS][ IFCHNLS] = { 170};
ViInt32 ThreeG_log_addr[ SYSIFPS][ IFCHNLS] = { 40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_FALSE
/*      If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/*****************************************************************************
    Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1  0
#define MEZZ1 0
#define MEZZ2 1
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE      1
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING  0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY   3
#define DIGITALIQ          0
#define ADCDATA            1
#define VXIBUS             0
#define PORT               1
#define TRIGGER            1
#define FREERUN            0
#define SRAM               1
#define LINKPORT0
/*****************************************************************************/
/*
  The macro 'rcheck' saves the return status in the variable 'ret'.
  If the return status indicates an error, 'rcheck' will call 'error_exit'.
  The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.
*/
#define rcheck(A) ( ((result = A) == VI_SUCCESS) ?\
 (result) : (error_exit(sessionID[ifp],result,  __LINE__,  __FILE__)) )
int main()
{
      int                  i, ifp, mezz, chan;
      ViStatusresult;
      ViSessionsessionID[ SYSIFPS];
      ViInt32      correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
      ViInt32      sample = 0;
      ViInt32      length = 2000;
      ViInt32      totlength = 4000;
      ViInt16      IData[ 4000], QData[ 4000];
      ViReal64 finit = 20e6;                  /*          This number should be less than 1 GHz.*/
      ViReal64 fcenter = 2600e6;              /*  This number can be any valid frequency for the tuner.*/
      ViInt16      ddcfreq = 0;               /*     See user manual for "hpe650x_setDDCFrequency"*/
      ViInt16      ddcbw = 8;                 /*  *See user manual for "hpe650x_setDigitalIFBandwidth"*/
      ViInt32      AnalogFilter = IF700KHZ;   /* Other choices are IF30KHZ or IF8MHZ          */
      ViBoolean suspend = FREERUN;            /* Choices are FREERUN or TRIGGER               */
      ViBoolean format = DIGITALIQ;           /* Choices are DIGITALIQ or ADCDATA             */
      ViBoolean collect = SRAM;               /* Choices are SRAM or LINKPORT                 */
      ViBoolean output = VXIBUS;              /* Choices are VXIBUS or PORT                   */

      /* Configure the receiver state, initialize all IFP's and IF's */
      for( ifp = 0; ifp < SYSIFPS; ifp++)
      {
            rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
            for( chan = 0; chan < IFCHNLS; chan++)
                  /*    Initialize the IF channels including establishing communication to their addresses. The initial      frequency needs to be below 1 GHz.*/
                  rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],
OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan]));
      }

      /* Put all IFPs into monitoring mode */
      for( ifp = 0; ifp < SYSIFPS; ifp++)
            for( mezz = 0; mezz < MEZZperIFP; mezz++)
                  rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* Set up and tune the IFs */
for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( chan = 0; chan < IFCHNLS; chan++)
        {
                /*      Set the tuner frequency to the value of "fcenter"*/
                rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter ));
                /*      Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
                rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));
                /*      Activate autorange once after the initialization of each mezzanine.*/
                rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
        }
/* Set up and tune DDCs */
for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
                /* The following function can be used for each DDC installed */
                rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC1, RELATIVE, ddcfreq));
                rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC5, RELATIVE, ddcfreq));
                /*rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC2, RELATIVE, ddcfreq));
                rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC3, RELATIVE, ddcfreq));
                rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC4, RELATIVE, ddcfreq));*/
                /* only need to call this function once per mezzanine */
                rcheck( hpe650x_setDigitalIFBandwidth( sessionID[ifp], mezz, ddcbw));
        }
/* synchronize autoranging on all modules in system */
if( VI_TRUE == SYNC_AUTORANGE)
{
        /* sync autoranging on master channel */
        rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile,      &Gsystem));

        /* synch autoranging on the slave channels */
        if( IFCHNLS > 1)
                rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));

        for( ifp = 1; ifp < SYSIFPS; ifp++)
                for( chan = 0; chan < IFCHNLS; chan++)
                        rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile,          Gsystem));
}
/* Set up to do the actual data capture */
        for( ifp = 0; ifp < SYSIFPS; ifp++)
                for( mezz = 0; mezz < MEZZperIFP; mezz++)
                {
                        rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC1, VI_TRUE ));
                        rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC5, VI_TRUE ));
                        /*rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC2, VI_TRUE ));
                        rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC3, VI_TRUE ));
                        rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC4, VI_TRUE ));*/
                        mSleep( 50);        /* allow for DDC transients */
                        rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
                        rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
                        rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
                        rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
                        rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
                        rcheck( hpe650x_startCapture( sessionID[ifp], mezz));
                }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
mSleep(1000);/* let capture run for 1 second */
/* now stop capture */
for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_stopCapture( sessionID[ ifp], mezz ));

/*      The remaining code only accommodates one IFP and two Mezzanines.
It will gather measurements taken from two DDCs.
The data output will be interleaved:

I[ddc1.0] Q[ddc1.0] I[ddc2.0] Q[ddc2.0]
I[ddc1.1] Q[ddc1.1] I[ddc2.1] Q[ddc2.1]
                        ...
I[ddc1.n] Q[ddc1.n] I[ddc2.n] Q[ddc2.n]*/

/* Retrieve data from mezzanine 1 */
do
{
        /* Loop until the DSP finishes giving us the data across the bus.  Notice that we gather
        a quantity of samples defined by "totlength", which should be the total number of DDCs
        enabled for capturing times "length."In this example, we've enabled DDC1 and DDC5,
        each taking 2000 samples, making "totlength" equal to 4000.*/
        result = hpe650x_getCaptureDigitalIQData( sessionID[ 0], MEZZ1, IData, QData, &totlength);
} while ( DATANOTREADY == result);
/*  Output the captured data to the monitor or a file.*/
if( VI_SUCCESS == result)
{
        /*  This writes the I and Q data to the file results1.txt in the same directory as the executable file.  */
        FILE *stream;
        stream = fopen( "mez1ddc1.txt", "w" );
        for( i = 0; i < length; i++)
                fprintf( stream, "%5d%5d\n", IData[ i*2], QData[ i*2]);
        fclose( stream );

        stream = fopen( "mez1ddc5.txt", "w" );
        for( i = 0; i < length; i++)
                fprintf( stream, "%5d%5d\n", IData[ i*2+1], QData[ i*2+1]);
        fclose( stream );

        stream = fopen( "mez1all.txt", "w" );
        for( i = 0; i < totlength; i++)
                fprintf( stream, "%5d%5d\n", IData[ i], QData[ i]);
        fclose( stream );
        /*This is a print statement to display the data on the monitor.*/
        for( i = 0; i < totlength; i++)
                printf(" I[ %4d]  %5d   Q[ %4d]   %5d\n", i, IData[ i], i, QData[ i]);
}



/* Retrieve data from mezzanine 2 */
do
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
{
        /* Loop until the DSP finishes giving us the data across the bus.  Notice that we gather
        a quantity of samples defined by "totlength", which should be the total number of DDCs
        enabled for capturing times "length."In this example, we've enabled DDC1 and DDC5,
        each taking 2000 samples, making "totlength" equal to 4000.*/
        result = hpe650x_getCaptureDigitalIQData( sessionID[ 0], MEZZ2, IData, QData, &totlength);
} while ( DATANOTREADY == result);
/*  Output the captured data to the monitor or a file.*/
if( VI_SUCCESS == result)
{
        /*  This writes the I and Q data to the file results1.txt in the same directory as the executable file.  */
        FILE *stream;
        stream = fopen( "mez2ddc1.txt", "w" );
        for( i = 0; i < length; i++)
                fprintf( stream, "%5d%5d\n", IData[ i*2], QData[ i*2]);
        fclose( stream );

        stream = fopen( "mez2ddc5.txt", "w" );
        for( i = 0; i < length; i++)
                fprintf( stream, "%5d%5d\n", IData[ i*2+1], QData[ i*2+1]);
        fclose( stream );

        stream = fopen( "mez2all.txt", "w" );
        for( i = 0; i < totlength; i++)
                fprintf( stream, "%5d%5d\n", IData[ i], QData[ i]);
        fclose( stream );
        /*This is a print statement to display the data on the monitor.*/
        for( i    0; i < totlength; i++)
                printf(" I[ %4d] = %5d   Q[ %4d] = %5d\n", i, IData[ i], i, QData[ i]);
}
/* Close all IFPs*/
for( ifp = 0; ifp < SYSIFPS; ifp++)
        rcheck( hpe650x_close(sessionID[ ifp]));
return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 4: Capture Digital I/Q Data Indefinitely From VXI Bus Using a Trigger

Use the following programming example for capturing an indefinite number of digital I/Q data samples across the VXI bus using a trigger to begin the capture process.

Figure 3-29 shows the main process steps for this scenario.

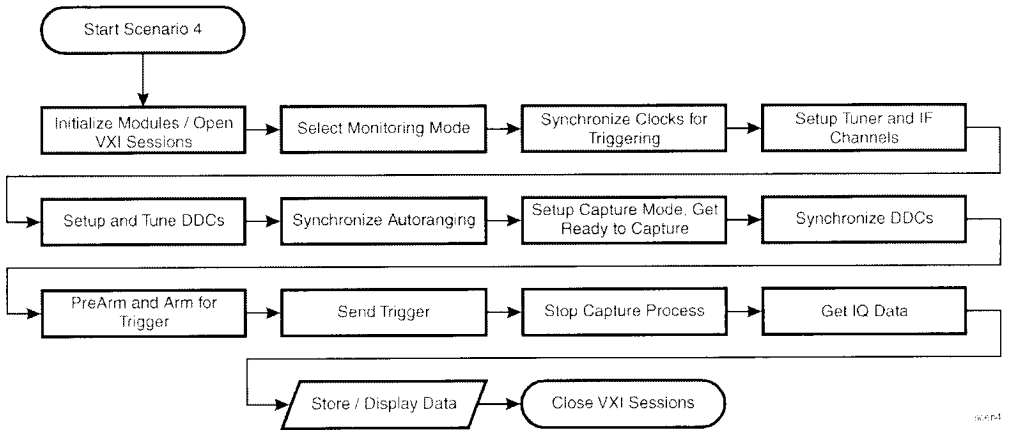


Figure 3-29 Scenario 4: Capture Digital I/Q Data Indefinitely From VXI Bus Using a Trigger

```
/*
Scenario 4:Capture an indefinite number of samples of digital
IQ data and output to the VXI bus with an external trigger.

Notes:
1. The following files must exist in the same directory as the
      source code.
      "hpe650x.h"
      "commonex.h"
      "commonex.c"
      "visatype.h"
      "vpptype.h"

2. The file "hpe650x.lib" must be available during linking.
3. This program requires the installation of SRAM on Mezzanine 1
      of the IF Processor whose VXI address is stored in IFPvxiID[0].
      SRAM is also required on other mezzanines in the system
      if more than one IFP exists or if MEZZperIFP is greater than 1.

4.    The user should read through the code modifying the
      following, as necessary:
      a.    Constants that define the number of IF channels, IF
            processors and mezzanines
      b.    Variables that define VXI addresses
      c.    SYNC_AUTORANGE controls whether all IFP autoranging
            is synchronized.
      d.    TunerExists should be set to VI_TRUE if a tuner exists
            otherwise is should be set to VI_FALSE.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

e. SYNC_CLOCKS is set to VI_TRUE when coherent measurements across multiple mezzanines and/or multiple IFPs is desired.

f. "fcenter" which defines the tuner's center frequency

g. "ddcfreq" sets the frequency tuning of the DDCs relative to the IF.

h. "ddcbw" sets the bandwidth of the DDCs. See the user documentation for a table of appropriate values.

i. "AnalogFilter", "suspend", "format", "collect" and "output" can be set to predefined constants as described below.

5. The commands hpe650x_init(), hpe650x_initIFChannel() and hpe650x_setMonitoringMode() must be executed before other commands, such as hpe650x_setTunerFrequency().

6. To synchronize DDCs across multiple mezzanines, the third parameter of hpe650x_armDDCsForSynchronization in "commonex.c" must be set to VI_TRUE. In addition, two trigger signals must be sent from a external source to mezzanine 1.

7. IMPORTANT: In this example, the two trigger signals MUST be sent when prompted, and BEFORE ANY SUBSEQUENT COMMANDS, otherwise the program will hang at the hpe650x_getCaptureDigitalIQData() command.

Disclaimer: This code is provided AS IS. It is a sample and unsupported.

```
*/
#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS     1
/* Number of IF processors installed */
#define SYSIFPS     1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs.
      This is an array for which each element contains a VXI address string.
      This example shows a system with only one IFP. */
ViRsrc IFPvxiID[SYSIFPS] = {"VXI0::43::INSTR"};
/* Extend for IFs.
      There are arrays for which each element contains a VXI address integer.
      The first dimension spans the system IFPs. The second dimension spans
      the IF channels for each IF processor. This example shows the VXI
      module addresses for one 3 GHz tuner. */
ViInt32 LO_log_addr[SYSIFPS][IFCHNLS]   {41};
ViInt32 OneG_log_addr[SYSIFPS][IFCHNLS] = {42};
ViInt32 ThreeG_log_addr[SYSIFPS][IFCHNLS] = {40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_FALSE
/*      If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* Use this to switch on synchronized IFP clocks (VI_TRUE).  This MUST be
       done if there is more than one mezzanine on an IFP, even if only one
       mezzanine is used in the measurement.  This is because of how the
       trigger signal is propagated throughout multiple mezzanines.*/
#define SYNC_CLOCKS VI_TRUE
/*****************************************************************************
     Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1   0
#define MEZZ1  0
#define MEZZ2  1
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE       1
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING   0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY    3
#define DIGITALIQ         0
#define ADCDATA           1
#define VXIBUS            0
#define PORT              1
#define TRIGGER           1
#define FREERUN           0
#define SRAM              1
#define LINKPORT0
/*****************************************************************************/
/*
  The macro 'rcheck' saves the return status in the variable 'ret'.
  If the return status indicates an error, 'rcheck' will call 'error_exit'.
  The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.
*/
#define rcheck(A) ( ((result = A) == VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp],result,   LINE   ,   FILE   )) )
int main()
{
       int                 i;
       int                 ifp, mezz, chan;
       ViStatusresult;
       ViSessionsessionID[ SYSIFPS];
       ViInt32       correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
       ViInt32       sample = 0;
       ViInt32       length = 2000;
       ViInt32       totlength = 4000;
       ViInt16       IData[ 4000], QData[ 4000];
       ViReal64 finit = 20e6;                    /*     This number should be less than 1 GHz.                  */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
ViReal64 fcenter = 2600e6;          /*  This number can be any valid frequency for the tuner.      */
ViInt16  ddcfreq = 0;               /*  See user manual for "hpe650x_setDDCFrequency"              */
ViInt16  ddcbw = 8;                 /*  See user manual for "hpe650x_setDigitalIFBandwidth"*/
ViInt32  AnalogFilter = IF700KHZ;   /*  Other choices are IF30KHZ or IF8MHZ                        */
ViBoolean suspend = TRIGGER;        /*  Choices are FREERUN or TRIGGER                             */
ViBoolean format = DIGITALIQ;       /*  Choices are DIGITALIQ or ADCDATA                           */
ViBoolean collect = SRAM;           /*  Choices are SRAM or LINKPORT                               */
ViBoolean output = VXIBUS;          /*  Choices are VXIBUS or PORT                                 */

/* Configure the receiver state, initialize all IFP's and IF's */
for( ifp = 0; ifp < SYSIFPS; ifp++)
{
    rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp])):
    for( chan = 0; chan < IFCHNLS; chan++)
        /*  Initialize the IF channels including establishing communication to their addresses. The initial  frequency needs to be below 1 GHz.*/
        rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan], OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan])):
}

/* Put all IFPs into monitoring mode */
for( ifp = 0; ifp < SYSIFPS; ifp++)
    for( mezz = 0; mezz < MEZZperIFP; mezz++)
        rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz)):

/*  Synchronize clocks between all mezzanines and IFPs.  The IFP at sessionID[ 0] is always the master in this  example.*/
if( VI_TRUE  SYNC_CLOCKS)
{
    /* Instructs the master IFP to send its clock out VXI.  Since the sample clock is also the DSP clock, this  action riskscausing the DSPs to hang.  Therefore
all the modulues are also reinitialized to reboot the  DSPs.*/
    rcheck( ConfigureMasterIFP( sessionID[ 0])):
    rcheck( hpe650x_init( IFPvxiID[ IF1], VI_TRUE, VI_TRUE, &sessionID[ IF1])):
    for( chan = 0; chan < IFCHNLS; chan++)
        rcheck( hpe650x_initIFChannel( sessionID[ IF1], chan, TunerExists, finit, LO_log_addr[ IF1][ chan], OneG_log_addr[ IF1][ chan],
ThreeG_log_addr[ IF1][ chan])):
    for( mezz = 0; mezz < MEZZperIFP; mezz++)
        rcheck( hpe650x_setMonitoringMode( sessionID[ IF1], mezz)):
    /* Now configure all slave IFPs.  The slaves are instructed to accept the master clock.  Finally, the slaves  must be reinitialized to boot the DSPs.*/
    for( ifp = 1; ifp < SYSIFPS; ifp++)
    {
        rcheck( ConfigureSlaveIFP( sessionID[ ifp])):
        rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp])):
        for( chan = 0; chan < IFCHNLS; chan++)
            rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan], OneG_log_addr[ ifp][ chan],
ThreeG_log_addr[ ifp][ chan])):
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz)):
    }
}
/* Set up and tune the IFs */
for( ifp = 0; ifp < SYSIFPS; ifp++)
    for( chan = 0; chan < IFCHNLS; chan++)
    {

    /*  Set the tuner frequency to the value of "fcenter"*/
        rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter )):
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
            /*      Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
            rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));
            /*      Activate autorange once after the initialization of each mezzanine.*/
            rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
        }
        /* Set up and tune DDCs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            /* The following function can be used for each DDC installed */
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC1, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC5, RELATIVE, ddcfreq));
        /*  rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC2, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC3, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC4, RELATIVE, ddcfreq));*/
            /* only need to call this function once per mezzanine */
            rcheck( hpe650x_setDigitalIFBandwidth( sessionID[ifp], mezz, ddcbw));
        }
    /* synchronize autoranging on all modules in system */
    if( VI_TRUE == SYNC_AUTORANGE)
    {
        /* sync autoranging on master channel */
        rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile,      &Gsystem));

        /* synch autoranging on the slave channels */
        if( IFCHNLS > 1)
            rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));

        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile,          Gsystem));
    }
/* Set up to do the actual data capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC1, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC5, VI_TRUE ));
        /*  rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC2, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC3, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC4, VI_TRUE ));*/
            mSleep( 50);        /* allow for DDC transients */
            rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
            rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
            rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
            rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
            rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
            rcheck( hpe650x_startCapture( sessionID[ifp], mezz ));
        }



    /*      Synchronize all DDCs*/
rcheck( SynchronizeDDCs( SYSIFPS, sessionID, MEZZperIFP));
    /*      Prearm -- required before arming.  This stops any current activity.*/
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_prearmDSPForDataCollection( sessionID[ ifp], mezz));
/* Arm the master DSP.  The master is defined as sessionID[ 0] in this example.*/
rcheck( hpe650x_armDSPForDataCollection( sessionID[ 0], MEZZ1, VI_FALSE));
/* Arm the slave DSPs */
if( MEZZperIFP > 1)
        rcheck( hpe650x_armDSPForDataCollection( sessionID[0], MEZZ2, VI_TRUE ));
if( SYSIFPS > 1)
        for( ifp = 1; ifp < SYSIFPS; ifp++)
                for( mezz = 0; mezz < MEZZperIFP; mezz++)
                        rcheck( hpe650x_armDSPForDataCollection( sessionID[ifp], mezz, VI_TRUE ));
/*      Prompt user for the trigger.*/
printf("Fire the trigger.  Press Enter to continue\n");
getchar();
/* sleep while data is captured */
mSleep( 1000);
/* now stop capture process */
for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
                result = hpe650x_stopCapture( sessionID[ ifp], mezz );
/* We've specified 0 samples above for indefinite capture.
Below, we pass &totlength into hpe650x_getCaptureDigitalIQData()
to tell the function the number of samples for which we've
allocated space.
The remaining code only accommodates one IFP and two Mezzanines.
It will gather measurements taken from two DDCs.
The data output will be interleaved:

I[ddc1,0] Q[ddc1,0] I[ddc2,0] Q[ddc2,0]
I[ddc1,1] Q[ddc1,1] I[ddc2,1] Q[ddc2,1]
                        ...
I[ddc1,n] Q[ddc1,n] I[ddc2,n] Q[ddc2,n]*/

/* Retrieve data from mezzanine 1 */
do
{
        /* Loop until the DSP finishes giving us the data across the bus.  Notice that we gather
        a quantity of samples defined by "totlength", which should be the total number of DDCs
        enabled for capturing times "length."In this example, we've enabled DDC1 and DDC5,
        each taking 2000 samples, making "totlength" equal to 4000.*/
        result = hpe650x_getCaptureDigitalIQData( sessionID[ 0], MEZZ1, IData, QData, &totlength);
} while ( DATANOTREADY == result);
/*  Output the captured data to the monitor or a file.*/
if( VI_SUCCESS == result)
{
        /*  This writes the I and Q data to the file results1.txt in the same directory as the executable file.  */
        FILE *stream;
        stream = fopen( "mez1ddc1.txt", "w" );
        for( i = 0; i < length; i++)
                fprintf( stream, "%5d%5d\n", IData[ i*2], QData[ i*2]);
        fclose( stream );

        stream = fopen( "mez1ddc5.txt", "w" );
        for( i = 0; i < length; i++)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
            fprintf( stream, "%5d%5d\n", IData[ i*2+1], QData[ i*2+1]);
        fclose( stream );

        stream = fopen( "mez1all.txt", "w" );
        for( i = 0; i < totlength; i++)
            fprintf( stream, "%5d%5d\n", IData[ i], QData[ i]);
        fclose( stream );
        /*This is a print statement to display the data on the monitor.*/
        for( i = 0; i < totlength; i++)
            printf(" I[ %4d] = %5d   Q[ %4d] = %5d\n", i, IData[ i], i, QData[ i]);
    }

    /* Retrieve data from mezzanine 2 */
    do
    {
        /* Loop until the DSP finishes giving us the data across the bus.  Notice that we gather
        a quantity of samples defined by "totlength", which should be the total number of DDCs
        enabled for capturing times "length."In this example, we've enabled DDC1 and DDC5,
        each taking 2000 samples, making "totlength" equal to 4000.*/
        result = hpe650x_getCaptureDigitalIQData( sessionID[ 0], MEZZ2, IData, QData, &totlength);
    } while ( DATANOTREADY == result);
    /*  Output the captured data to the monitor or a file.*/
    if( VI_SUCCESS == result)
    {
        /*  This writes the I and Q data to the file results1.txt in the same directory as the executable file.  */
        FILE *stream;
        stream = fopen( "mez2ddc1.txt", "w" );
        for( i = 0; i < length; i++)
            fprintf( stream, "%5d%5d\n", IData[ i*2], QData[ i*2]);
        fclose( stream );

        stream = fopen( "mez2ddc5.txt", "w" );
        for( i = 0; i < length; i++)
            fprintf( stream, "%5d%5d\n", IData[ i*2+1], QData[ i*2+1]);
        fclose( stream );

        stream = fopen( "mez2all.txt", "w" );
        for( i = 0; i < totlength; i++)
            fprintf( stream, "%5d%5d\n", IData[ i], QData[ i]);
        fclose( stream );
        /*This is a print statement to display the data on the monitor.*/
        for( i = 0; i < totlength; i++)
            printf(" I[ %4d] = %5d   Q[ %4d] = %5d\n", i, IData[ i], i, QData[ i]);
    }
    /* Close all IFPs*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        rcheck( hpe650x_close(sessionID[ ifp]));
    return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 5: Stream Digital I/Q Data Indefinitely to the Link Port Using a Trigger

Use the following programming example to stream an indefinite number of digital I/Q data samples to the link port using a trigger to begin the streaming process.

Figure 3-30 shows the main process steps for this scenario.

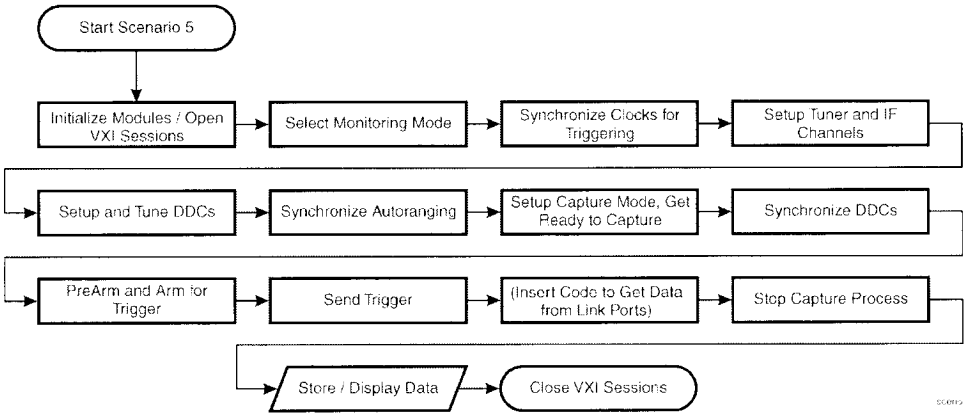


Figure 3-30 Scenario 5: Stream Digital I/Q Data Indefinitely to the Link Port Using a Trigger

```
/*
Scenario 5:Stream digital IQ data indefinitely to the link port
using a trigger.
Notes:
1. The following files must exist in the same directory as the
      source code.
      "hpe650x.h"
      "commonex.h"
      "commonex.c"
      "visatype.h"
      "vpptype.h"

2. The file "hpe650x.lib" must be available during linking.
3. This program requires the installation of SRAM on Mezzanine 1
      of the IF Processor whose VXI address is stored in IFPvxiID[0].
      SRAM is also required on other mezzanines in the system
      if more than one IFP exists or if MEZZperIFP is greater than 1.

4.    The user should read through the code modifying the
      following, as necessary:
      a.    Constants that define the number of IF channels, IF
            processors and mezzanines
      b.    Variables that define VXI addresses
      c.    SYNC_AUTORANGE controls whether all IFP autoranging
            is synchronized.
      d.    TunerExists should be set to VI_TRUE if a tuner exists
            otherwise is should be set to VI_FALSE.
      e.    SYNC_CLOCKS is set to VI_TRUE when coherent measurements
            across multiple mezzanines and/or multiple IFPs is desired.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

f. "fcenter" which defines the tuner's center frequency

g. "ddcfreq" sets the frequency tuning of the DDCs relative to the IF.

h. "ddcbw" sets the bandwidth of the DDCs. See the user documentation for a table of appropriate values.

i. "AnalogFilter", "suspend", "format", "collect" and "output" can be set to predefined constants as described below.

5. The commands hpe650x_init(), hpe650x_initIFChannel() and hpe650x_setMonitoringMode() must be executed before other commands, such as hpe650x_setTunerFrequency().
6. To synchronize DDCs across multiple mezzanines, the third parameter of hpe650x_armDDCsForSynchronization in "commonex.c" must be set to VI_TRUE. In addition, two trigger signals must be sent from a external source to mezzanine 1.
7. IMPORTANT: In this example, the two trigger signals MUST be sent when prompted, and BEFORE ANY SUBSEQUENT COMMANDS.

Disclaimer: This code is provided AS IS. It is a sample and unsupported.

```
*/
#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS      1
/* Number of IF processors installed */
#define SYSIFPS      1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs.
      This is an array for which each element contains a VXI address string.
      This example shows a system with only one IFP. */
ViRsrc  IFPvxiID[ SYSIFPS] = { "VXI0::43::INSTR"};
/* Extend for IFs.
      There are arrays for which each element contains a VXI address integer.
      The first dimension spans the system IFPs. The second dimension spans
      the IF channels for each IF processor. This example shows the VXI
      module addresses for one 3 GHz tuner. */
ViInt32 LO_log_addr[ SYSIFPS][ IFCHNLS] = { 41};
ViInt32 OneG_log_addr[ SYSIFPS][ IFCHNLS] = { 42};
ViInt32 ThreeG_log_addr[ SYSIFPS][ IFCHNLS] = { 40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_FALSE
/*    If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/* Use this to switch on synchronized IFP clocks (VI_TRUE). This MUST be
      done if there is more than one mezzanine on an IFP, even if only one
      mezzanine is used in the measurement. This is because the
      trigger signal is propagated throughout multiple mezzanines.*/
#define SYNC_CLOCKS VI_TRUE
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/*************************************************************************
     Don't change these, otherwise the code won't work anymore :-(
*************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1   0
#define MEZZ1  0
#define MEZZ2  1
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE       1
#define DATANOTREADY       -1
#define CAPTURENOTRUNNING   0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY    3
#define DIGITALIQ          0
#define ADCDATA            1
#define VXIBUS             0
#define PORT               1
#define TRIGGER            1
#define FREERUN            0
#define SRAM               1
#define LINKPORT0
/*************************************************************************/
/*
  The macro 'rcheck' saves the return status in the variable 'ret'.
  If the return status indicates an error, 'rcheck' will call 'error_exit'.
  The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.
*/
#define rcheck(A) ( ((result = A) == VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp].result,   LINE   ,   FILE   )) )
int main()
{
      int                  ifp, mezz, chan;
      ViStatusresult;
      ViSessionsessionID[ SYSIFPS];
      ViInt32      correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
      ViInt32      sample = 0;                  /*       Number of samples or 0 if taking indefinite length. */
      ViReal64 finit = 20e6;                    /*                  This number should be less than 1 GHz.*/
      ViReal64 fcenter = 2600e6;                /*    This number can be any valid frequency for the tuner.*/
      ViInt16      ddcfreq = 0;                 /*      See user manual for "hpe650x_setDDCFrequency"*/
      ViInt16      ddcbw = 8;                   /*   See user manual for "hpe650x_setDigitalIFBandwidth"*/
      ViInt32      AnalogFilter = IF700KHZ;     /* Other choices are IF30KHZ or IF8MHZ                  */
      ViBoolean suspend = TRIGGER;              /* Choices are FREERUN or TRIGGER                       */
      ViBoolean format = DIGITALIQ;             /* Choices are DIGITALIQ or ADCDATA                     */
      ViBoolean collect = LINKPORT;             /* Choices are SRAM or LINKPORT                         */
      ViBoolean output = PORT;                  /* Choices are VXIBUS or PORT                           */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* Configure the receiver state, initialize all IFP's and IF's */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
    {
        rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
        for( chan = 0; chan < IFCHNLS; chan++)
            /*  Initialize the IF channels including establishing communication to their addresses. The initial  frequency needs to be below 1 GHz.*/
            rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],  OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan]));
    }

    /* Put all IFPs into monitoring mode */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
    /*  Synchronize clocks between all mezzanines and IFPs.  The IFP at sessionID[ 0] is always the master in this  example.*/
    if( VI_TRUE == SYNC_CLOCKS)
    {
        /* Instructs the master IFP to send its clock out VXI.  Since the sample clock is also the DSP clock, this  action risks causing the DSPs to hang.  Therefore
all the modulues are also reinitialized to reboot the  DSPs.*/
        rcheck( ConfigureMasterIFP( sessionID[ 0]));
        rcheck( hpe650x_init( IFPvxiID[ IF1], VI_TRUE, VI_TRUE, &sessionID[ IF1]));
        for( chan = 0; chan < IFCHNLS; chan++)
            rcheck( hpe650x_initIFChannel( sessionID[ IF1], chan, TunerExists, finit, LO_log_addr[ IF1][ chan],  OneG_log_addr[ IF1][ chan],
ThreeG_log_addr[ IF1][ chan]));
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ IF1], mezz));
        /* Now configure all slave IFPs.  The slaves are instructed to accept the master clock.  Finally, the slaves  must be reinitialized to boot the DSPs.*/
        for( ifp = 1; ifp < SYSIFPS; ifp++)
        {
            rcheck( ConfigureSlaveIFP( sessionID[ ifp]));
            rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],  OneG_log_addr[ ifp][ chan],
ThreeG_log_addr[ ifp][ chan]));
            for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
        }
    }
    /* Set up and tune the IFs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( chan = 0; chan < IFCHNLS; chan++)
        {
            /*  Set the tuner frequency to the value of "fcenter"*/
            rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter ));
            /*  Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
            rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));
            /*  Activate autorange once after the initialization of each mezzanine.*/
            rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
        }


        /* Set up and tune DDCs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        {
            /* The following function can be used for each DDC installed */
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC1, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC5, RELATIVE, ddcfreq));
        /*  rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC2, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC3, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC4, RELATIVE, ddcfreq));*/
            /* only need to call this function once per mezzanine */
            rcheck( hpe650x_setDigitalIFBandwidth( sessionID[ifp], mezz, ddcbw));
        }
    /* synchronize autoranging on all modules in system */
    if( VI_TRUE == SYNC_AUTORANGE)
    {
        /* sync autoranging on master channel */
        rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile,  &Gsystem));
        /* synch autoranging on the slave channels */
        if( IFCHNLS > 1)
            rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile,  Gsystem));
    }
/* Set up to do the actual data capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC1, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC5, VI_TRUE ));
        /*  rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC2, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC3, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC4, VI_TRUE ));*/
            mSleep( 50);        /* allow for DDC transients */
            rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
            rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
            rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
            rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
            rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
            rcheck( hpe650x_startCapture( sessionID[ifp], mezz ));
        }
    /*  Synchronize all DDCs*/
rcheck( SynchronizeDDCs( SYSIFPS, sessionID, MEZZperIFP));
    /*  Prearm -- required before arming.  This stops any current activity.*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_prearmDSPForDataCollection( sessionID[ ifp], mezz));



    /* Arm the master DSP.  The master is defined as sessionID[ 0] in this example.*/
    rcheck( hpe650x_armDSPForDataCollection( sessionID[ 0], MEZZ1, VI_FALSE));

    /* Arm the slave DSPs */
    if( MEZZperIFP > 1)
        rcheck( hpe650x_armDSPForDataCollection( sessionID[0], MEZZ2, VI_TRUE ));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    if( SYSIFPS > 1)
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_armDSPForDataCollection( sessionID[ifp], mezz, VI_TRUE ));

    /*      Prompt user for the trigger.*/
    printf("Fire the trigger.  Press Enter to continue\n");
    getchar();
    /* sleep while data is captured */
    mSleep( 1000);
    /* now stop the capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_stopCapture( sessionID[ ifp], mezz ));
    /* Close all IFPs*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        rcheck( hpe650x_close(sessionID[ ifp]));
    return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 6: Stream Digital I/Q Data Indefinitely to the Link Port

Use the following programming example to stream an indefinite number of digital I/Q data samples to the link port without the use of a trigger to begin the streaming process.

Figure 3-31 shows the main process steps for this scenario.

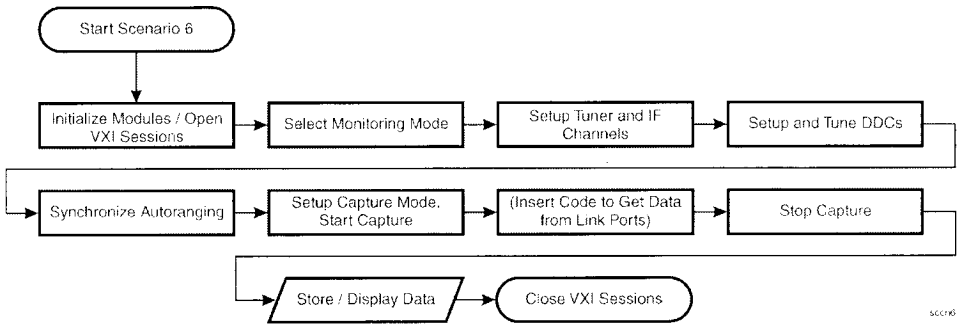


Figure 3-31 Scenario 6: Stream Digital I/Q Data Indefinitely to the Link Port

```
/*
Scenario 6:Stream digital IQ data indefinitely to the Link Port
without a trigger.
Notes:
1. The following files must exist in the same directory as the
      source code.
      "hpe650x.h"
      "commonex.h"
      "commonex.c"
      "visatype.h"
      "vpptype.h"
2. The file "hpe650x.lib" must be available during linking.
3. This program requires the installation of SRAM on Mezzanine 1
      of the IF Processor whose VXI address is stored in IFPvxiID[0].
      SRAM is also required on other mezzanines in the system
      if more than one IFP exists or if MEZZperIFP is greater than 1.
4.    The user should read through the code modifying the
      following, as necessary:
      a.    Constants that define the number of IF channels, IF
            processors and mezzanines
      b.    Variables that define VXI addresses
      c.    SYNC_AUTORANGE controls whether all IFP autoranging
            is synchronized.
      d.    TunerExists should be set to VI_TRUE if a tuner exists
            otherwise is should be set to VI_FALSE.
      f.    "fcenter" which defines the tuner's center frequency
      g.    "ddcfreq" sets the frequency tuning of the DDCs relative
            to the IF.
      h.    "ddcbw" sets the bandwidth of the DDCs.  See the user
            documentation for a table of appropriate values.
      i.    "AnalogFilter", "suspend", "format", "collect" and "output"
            can be set to predefined constants as described below.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
5.      The commands hpe650x_init(), hpe650x_initIFChannel() and
            hpe650x_setMonitoringMode() must be executed before other
            commands, such as hpe650x_setTunerFrequency().


        Disclaimer: This code is provided AS IS.  It is a sample and unsupported.
*/

#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS      1
/* Number of IF processors installed */
#define SYSIFPS      1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs.
        This is an array for which each element contains a VXI address string.
        This example shows a system with only one IFP. */
ViRsrc  IFPvxiID[ SYSIFPS] = { "VXI0::43::INSTR"};
/* Extend for IFs.
        There are arrays for which each element contains a VXI address integer.
        The first dimension spans the system IFPs.  The second dimension spans
        the IF channels for each IF processor.  This example shows the VXI
        module addresses for one 3 GHz tuner.  */
ViInt32 LO_log_addr[ SYSIFPS][ IFCHNLS] = { 41};
ViInt32 OneG_log_addr[ SYSIFPS][ IFCHNLS] = { 42};
ViInt32 ThreeG_log_addr[ SYSIFPS][ IFCHNLS] = { 40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_FALSE
/*      If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/*****************************************************************************
     Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1   0
#define MEZZ1  0
#define MEZZ2  1
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE      1
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING   0
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY    3
#define DIGITALIQ           0
#define ADCDATA             1
#define VXIBUS              0
#define PORT                1
#define TRIGGER             1
#define FREERUN             0
#define SRAM                1
#define LINKPORT0
/*****************************************************************************/
/*      The macro 'rcheck' saves the return status in the variable 'ret'.
        If the return status indicates an error, 'rcheck' will call 'error_exit'.
        The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.*/
#define rcheck(A) ( ((result = A) == VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp],result, __LINE__, __FILE__)) )
int main()
{
        int             ifp, mezz, chan;
        ViStatusresult;
        ViSessionsessionID[ SYSIFPS];
        ViInt32         correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
        ViInt32         sample = 0;                     /*      Number of samples or 0 if taking indefinite length. */
        ViReal64 finit = 20e6;                          /*              This number should be less than 1 GHz.*/
        ViReal64 fcenter = 2600e6;                      /*      This number can be any valid frequency for the tuner.*/
        ViInt16         ddcfreq = 0;                    /*      See user manual for "hpe650x_setDDCFrequency"*/
        ViInt16         ddcbw = 8;                      /*      See user manual for "hpe650x_setDigitalIFBandwidth"*/
        ViInt32         AnalogFilter = IF700KHZ;        /* Other choices are IF30KHZ or IF8MHZ                  */
        ViBoolean suspend = FREERUN;                    /* Choices are FREERUN or TRIGGER                       */
        ViBoolean format = DIGITALIQ;                   /* Choices are DIGITALIQ or ADCDATA                     */
        ViBoolean collect = LINKPORT;                   /* Choices are SRAM or LINKPORT                         */
        ViBoolean output = PORT;                        /* Choices are VXIBUS or PORT                           */
        /* Configure the receiver state, initialize all IFP's and IF's */
        for( ifp = 0; ifp < SYSIFPS; ifp++)
        {
                rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
                for( chan = 0; chan < IFCHNLS; chan++)
                        /*      Initialize the IF channels including establishing communication to their addresses. The initial         frequency needs to be below 1 GHz.*/
                        rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],     OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan]));
        }

        /* Put all IFPs into monitoring mode */
        for( ifp = 0; ifp < SYSIFPS; ifp++)
                for( mezz = 0; mezz < MEZZperIFP; mezz++)
                        rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
        /* Set up and tune the IFs */
        for( ifp = 0; ifp < SYSIFPS; ifp++)
                for( chan = 0; chan < IFCHNLS; chan++)
                {
                        /*      Set the tuner frequency to the value of "fcenter"*/
                        rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter ));
                        /*      Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
                        rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));

                /*      Activate autorange once after the initialization of each mezzanine.*/
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
            rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
        }

        /* Set up and tune DDCs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            /* The following function can be used for each DDC installed */
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC1, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC5, RELATIVE, ddcfreq));
        /*  rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC2, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC3, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC4, RELATIVE, ddcfreq));*/
            /* only need to call this function once per mezzanine */
            rcheck( hpe650x_setDigitalIFBandwidth( sessionID[ifp], mezz, ddcbw));
        }
    /* synchronize autoranging on all modules in system */
    if( VI_TRUE == SYNC_AUTORANGE)
    {
        /* sync autoranging on master channel */
        rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile,  &Gsystem));
        /* synch autoranging on the slave channels */
        if( IFCHNLS > 1)
            rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile,  Gsystem));
    }
/* Set up to do the actual data capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC1, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC5, VI_TRUE ));
        /*  rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC2, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC3, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC4, VI_TRUE ));*/
            mSleep( 50);        /* allow for DDC transients */
            rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
            rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
            rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
            rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
            rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
            rcheck( hpe650x_startCapture( sessionID[ifp], mezz ));
        }
    /*  Prompt the user to stop the data capture.  */
    printf("Data Streaming from Link Port.  Press Enter to stop capture\n");
    getchar();
    /* Stop the data streaming */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_stopCapture( sessionID[ ifp], mezz));

/* Close all IFPs*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        rcheck( hpe650x_close(sessionID[ ifp])):
    return VI_SUCCESS:
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 7: Stream N Samples of Digital I/Q Data to the Link Port

Use the following programming example to stream a specified number of digital I/Q data samples to the link port without the use of a trigger to begin the streaming process.

Figure 3-32 shows the main process steps for this scenario.

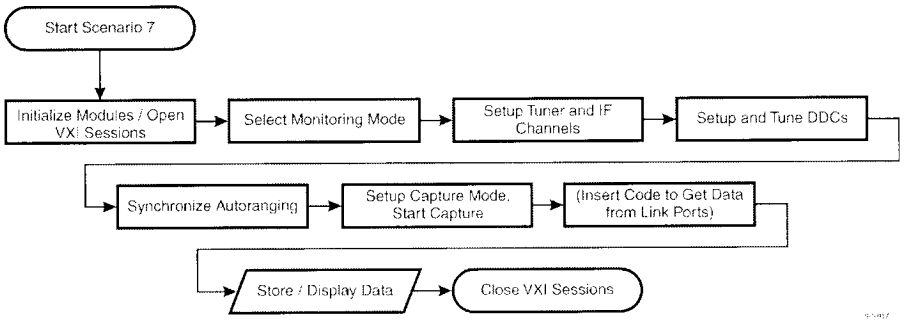


Figure 3-32 Scenario 7: Stream N Samples of Digital I/Q Data to the Link Port

```
/*
Scenario 7:Stream n-samples of digital IQ data to the Link Port
without a trigger.
Notes:
1. The following files must exist in the same directory as the
      source code.
      "hpe650x.h"
      "commonex.h"
      "commonex.c"
      "visatype.h"
      "vpptype.h"

2. The file "hpe650x.lib" must be available during linking.

3.    The user should read through the code modifying the
      following, as necessary:
      a.    Constants that define the number of IF channels, IF
            processors and mezzanines
      b.    Variables that define VXI addresses
      c.    SYNC_AUTORANGE controls whether all IFP autoranging
            is synchronized.
      d.    TunerExists should be set to VI_TRUE if a tuner exists
            otherwise is should be set to VI_FALSE.
      e.    "fcenter" which defines the tuner's center frequency
      f.    "ddcfreq" sets the frequency tuning of the DDCs relative
            to the IF.
      g.    "ddcbw" sets the bandwidth of the DDCs.  See the user
            documentation for a table of appropriate values.
      h.    "AnalogFilter", "suspend", "format", "collect" and "output"
            can be set to predefined constants as described below.

5.    The commands hpe650x_init(), hpe650x_initIFChannel() and
      hpe650x_setMonitoringMode() must be executed before other
      commands, such as hpe650x_setTunerFrequency().
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        Disclaimer: This code is provided AS IS.  It is a sample and unsupported.
        */

#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS      1
/* Number of IF processors installed */
#define SYSIFPS      1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs.
        This is an array for which each element contains a VXI address string.
        This example shows a system with only one IFP. */
ViRsrc  IFPvxiID[ SYSIFPS] = { "VXI0::43::INSTR"};
/* Extend for IFs.
        There are arrays for which each element contains a VXI address integer.
        The first dimension spans the system IFPs.  The second dimension spans
        the IF channels for each IF processor.  This example shows the VXI
        module addresses for one 3 GHz tuner.  */
ViInt32 LO_log_addr[ SYSIFPS][ IFCHNLS] = { 41};
ViInt32 OneG_log_addr[ SYSIFPS][ IFCHNLS]  = { 42};
ViInt32 ThreeG_log_addr[ SYSIFPS][ IFCHNLS] = { 40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_FALSE
/*      If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/*****************************************************************************
     Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1  0
#define MEZZ1 0
#define MEZZ2 1
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE      1
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING  0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY   3
#define DIGITALIQ         0
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#define ADCDATA          1
#define VXIBUS           0
#define PORT             1
#define TRIGGER          1
#define FREERUN          0
#define SRAM             1
#define LINKPORT0
/*************************************************************************/
/*      The macro 'rcheck' saves the return status in the variable 'ret'.
        If the return status indicates an error, 'rcheck' will call 'error_exit'.
        The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.*/
#define rcheck(A) ( ((result = A) == VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp],result,  __LINE__,__FILE__)) )
int main()
{
        int                     ifp, mezz, chan;
        ViStatusresult;
        ViSessionsessionID[ SYSIFPS];
        ViInt32         correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
        ViInt32         sample = 2000;                  /* Number of samples or 0 if taking indefinite length. */
        ViReal64 finit = 20e6;                          /*                      This number should be less than 1 GHz.*/
        ViReal64 fcenter = 2600e6;                      /*      This number can be any valid frequency for the tuner.*/
        ViInt16         ddcfreq = 0;                    /*          See user manual for "hpe650x_setDDCFrequency"*/
        ViInt16         ddcbw = 8;                      /*   See user manual for "hpe650x_setDigitalIFBandwidth"*/
        ViInt32         AnalogFilter = IF700KHZ;        /* Other choices are IF30KHZ or IF8MHZ                  */
        ViBoolean suspend = FREERUN;                    /* Choices are FREERUN or TRIGGER                       */
        ViBoolean format = DIGITALIQ;                   /* Choices are DIGITALIQ or ADCDATA                     */
        ViBoolean collect = LINKPORT;                   /* Choices are SRAM or LINKPORT                         */
        ViBoolean output = PORT;                        /* Choices are VXIBUS or PORT                           */
        /* Configure the receiver state, initialize all IFP's and IF's */
        for( ifp = 0; ifp < SYSIFPS; ifp++)
        {
                rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
                for( chan = 0; chan < IFCHNLS; chan++)
                        /*      Initialize the IF channels including establishing communication to their addresses. The initial         frequency needs to be below 1 GHz.*/
                        rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],      OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan]));
        }

        /* Put all IFPs into monitoring mode */
        for( ifp = 0; ifp < SYSIFPS; ifp++)
                for( mezz = 0; mezz < MEZZperIFP; mezz++)
                        rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
        /* Set up and tune the IFs */
        for( ifp = 0; ifp < SYSIFPS; ifp++)
                for( chan = 0; chan < IFCHNLS; chan++)
                {
                        /*      Set the tuner frequency to the value of "fcenter"*/
                        rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter ));
                        /*      Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
                        rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));
                        /*      Activate autorange once after the initialization of each mezzanine.*/
                        rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
                }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    /* Set up and tune DDCs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            /* The following function can be used for each DDC installed */
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC1, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC5, RELATIVE, ddcfreq));
        /*  rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC2, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC3, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC4, RELATIVE, ddcfreq));*/
            /* only need to call this function once per mezzanine */
            rcheck( hpe650x_setDigitalIFBandwidth( sessionID[ifp], mezz, ddcbw));
        }
    /* synchronize autoranging on all modules in system */
    if( VI_TRUE == SYNC_AUTORANGE)
    {
        /* sync autoranging on master channel */
        rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile,      &Gsystem));
        /* synch autoranging on the slave channels */
        if( IFCHNLS > 1)
            rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile,      Gsystem));
    }

/* Set up to do the actual data capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC1, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC5, VI_TRUE ));
        /*  rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC2, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC3, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC4, VI_TRUE ));*/
            mSleep( 50);        /* allow for DDC transients */
            rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
            rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
            rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
            rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
            rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
            rcheck( hpe650x_startCapture( sessionID[ifp], mezz ));
        }
    /*  Insert your code here for capturing data from the link ports.*/
    /*  Prompt user to continue*/
    printf("N samples has streamed from Link Port.  Press Enter to close session.\n");
    getchar();

    /* Close all IFPs*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        rcheck( hpe650x_close(sessionID[ ifp]));
    return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 8: Stream N Samples of Digital I/Q Data to the Link Port Using Multiple Triggers

Use the following programming example to stream a specified number of digital I/Q data samples to the link port using multiple triggers to begin the streaming process.

Figure 3-33 shows the main process steps for this scenario.

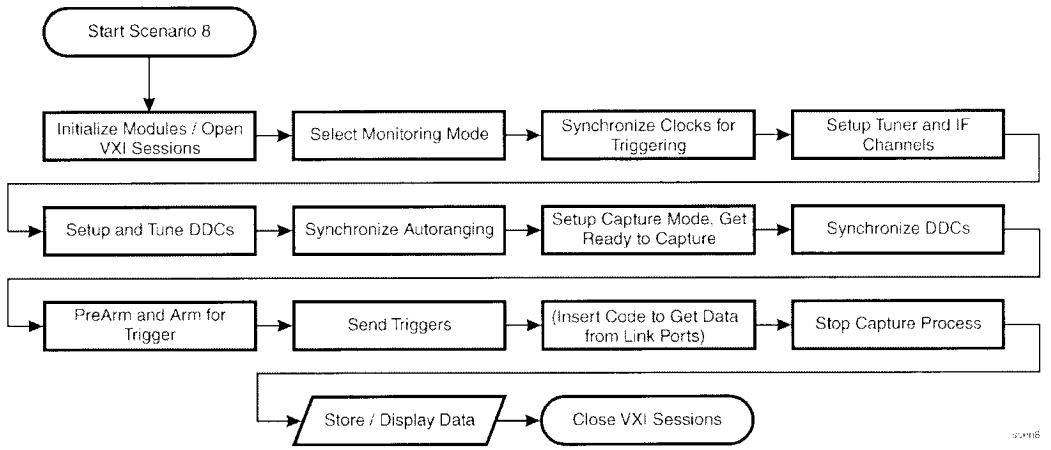


Figure 3-33 Scenario 8: Stream N Samples of Digital I/Q Data to the Link Port Using Multiple Triggers

```
/*  Scenario 8:Stream n-samples of Digital IQ data out the link port
    using multiple triggers.
    Notes:
    1.  The following files must exist in the same directory as the
        source code.
        "hpe650x.h"
        "commonex.h"
        "commonex.c"
        "visatype.h"
        "vpptype.h"
    2.  The file "hpe650x.lib" must be available during linking.

    3.  The user should read through the code modifying the
        following, as necessary:
        a.  Constants that define the number of IF channels, IF
            processors and mezzanines
        b.  Variables that define VXI addresses
        c.  SYNC_AUTORANGE controls whether all IFP autoranging
            is synchronized.
        d.  TunerExists should be set to VI_TRUE if a tuner exists
            otherwise is should be set to VI_FALSE.
        e.  SYNC_CLOCKS is set to VI_TRUE when coherent measurements
            across multiple mezzanines and/or multiple IFPs is desired.
        f.  "fcenter" which defines the tuner's center frequency
        g.  "ddcfreq" sets the frequency tuning of the DDCs relative
            to the IF.
        h.  "ddcbw" sets the bandwidth of the DDCs.  See the user
            documentation for a table of appropriate values.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

i. "AnalogFilter", "suspend", "format", "collect" and "output" can be set to predefined constants as described below.

4. The commands hpe650x_init(), hpe650x_initIFChannel() and hpe650x_setMonitoringMode() must be executed before other commands, such as hpe650x_setTunerFrequency().
5. To synchronize DDCs across multiple mezzanines, the third parameter of hpe650x_armDDCsForSynchronization in "commonex.c" must be set to VI_TRUE. In addition, two trigger signals must be sent from a external source to mezzanine 1.
6. IMPORTANT: In this example, the two trigger signals MUST be sent when prompted, and BEFORE ANY SUBSEQUENT COMMANDS. Two trigger signals must be sent for each trigger event.

Disclaimer: This code is provided AS IS. It is a sample and unsupported.

```
*/
#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS     1
/* Number of IF processors installed */
#define SYSIFPS     1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs.
      This is an array for which each element contains a VXI address string.
      This example shows a system with only one IFP. */
ViRsrc  IFPvxiID[ SYSIFPS] = { "VXI0::43::INSTR"};
/* Extend for IFs.
      There are arrays for which each element contains a VXI address integer.
      The first dimension spans the system IFPs.  The second dimension spans
      the IF channels for each IF processor.  This example shows the VXI
      module addresses for one 3 GHz tuner.  */
ViInt32 LO_log_addr[ SYSIFPS][ IFCHNLS] = { 41};
ViInt32 OneG_log_addr[ SYSIFPS][ IFCHNLS] = { 42};
ViInt32 ThreeG_log_addr[ SYSIFPS][ IFCHNLS] = { 40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_FALSE
/*      If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/* Use this to switch on synchronized IFP clocks (VI_TRUE).  This MUST be
      done if there is more than one mezzanine on an IFP, even if only one
      mezzanine is used in the measurement.  This is because of how the
      trigger signal is propagated throughout multiple mezzanines.*/
#define SYNC_CLOCKS VI_TRUE
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/*****************************************************************************
    Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1   0
#define MEZZ1  0
#define MEZZ2  1
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE      1
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING   0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY    3
#define DIGITALIQ         0
#define ADCDATA           1
#define VXIBUS            0
#define PORT              1
#define TRIGGER           1
#define FREERUN           0
#define SRAM              1
#define LINKPORT0
/*****************************************************************************/
/*
  The macro 'rcheck' saves the return status in the variable 'ret'.
  If the return status indicates an error, 'rcheck' will call 'error_exit'.
  The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.
*/
#define rcheck(A) ( ((result = A) != VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp],result,__LINE__, __FILE__)) )
int main()
{
      int                 ifp, mezz, chan;
      ViStatusresult;
      ViSessionsessionID[ SYSIFPS];
      ViInt32      correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
      ViInt32      sample = 2000;                /*      Number of samples or 0 if taking indefinite length. */
      ViInt32      numberoftriggers = 5;
      ViReal64 finit = 20e6;                     /*                  This number should be less than 1 GHz.*/
      ViReal64 fcenter = 2600e6;                 /*    This number can be any valid frequency for the tuner.*/
      ViInt16      ddcfreq = 0;                  /*      See user manual for "hpe650x_setDDCFrequency"*/
      ViInt16      ddcbw = 8;                    /*  See user manual for "hpe650x_setDigitalIFBandwidth"*/
      ViInt32      AnalogFilter = IF700KHZ;      /* Other choices are IF30KHZ or IF8MHZ                */
      ViBoolean suspend = TRIGGER;               /* Choices are FREERUN or TRIGGER                     */
      ViBoolean format = DIGITALIQ;              /* Choices are DIGITALIQ or ADCDATA                   */
      ViBoolean collect = LINKPORT;              /* Choices are SRAM or LINKPORT                       */
      ViBoolean output = PORT;                   /* Choices are VXIBUS or PORT                         */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* Configure the receiver state, initialize all IFP's and IF's */
for( ifp = 0; ifp < SYSIFPS; ifp++)
{
    rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp])):
    for( chan = 0; chan < IFCHNLS; chan++)
        /*  Initialize the IF channels including establishing communication to their addresses. The initial  frequency needs to be below 1 GHz.*/
        rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],  OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan])):
}


/* Put all IFPs into monitoring mode */
for( ifp = 0; ifp < SYSIFPS; ifp++)
    for( mezz = 0; mezz < MEZZperIFP; mezz++)
        rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz)):
/*  Synchronize clocks between all mezzanines and IFPs.  The IFP at sessionID[ 0] is always the master in this  example.*/
if( VI_TRUE == SYNC_CLOCKS)
{
    /* Instructs the master IFP to send its clock out VXI.  Since the sample clock is also the DSP clock, this  action risks causing the DSPs to hang.  Therefore
all the modulues are also reinitialized to reboot the  DSPs.*/
    rcheck( ConfigureMasterIFP( sessionID[ 0])):
    rcheck( hpe650x_init( IFPvxiID[ IF1], VI_TRUE, VI_TRUE, &sessionID[ IF1])):
    for( chan = 0; chan < IFCHNLS; chan++)
        rcheck( hpe650x_initIFChannel( sessionID[ IF1], chan, TunerExists, finit, LO_log_addr[ IF1][ chan],  OneG_log_addr[ IF1][ chan],
ThreeG_log_addr[ IF1][ chan])):
    for( mezz = 0; mezz < MEZZperIFP; mezz++)
        rcheck( hpe650x_setMonitoringMode( sessionID[ IF1], mezz)):
    /* Now configure all slave IFPs.  The slaves are instructed to accept the master clock.  Finally, the slaves  must be reinitialized to boot the DSPs.*/
    for( ifp = 1; ifp < SYSIFPS; ifp++)
    {
        rcheck( ConfigureSlaveIFP( sessionID[ ifp])):
        rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp])):
        for( chan = 0; chan < IFCHNLS; chan++)
            rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],  OneG_log_addr[ ifp][ chan],
ThreeG_log_addr[ ifp][ chan])):
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz)):
    }
}
/* Set up and tune the IFs */
for( ifp = 0; ifp < SYSIFPS; ifp++)
    for( chan = 0; chan < IFCHNLS; chan++)
    {
        /*  Set the tuner frequency to the value of "fcenter"*/
        rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter ));
        /*  Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
        rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));
        /*  Activate autorange once after the initialization of each mezzanine.*/
        rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
    }


/* Set up and tune DDCs */
for( ifp = 0; ifp < SYSIFPS; ifp++)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            /* The following function can be used for each DDC installed */
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC1, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC5, RELATIVE, ddcfreq));
        /*  rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC2, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC3, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC4, RELATIVE, ddcfreq));*/
            /* only need to call this function once per mezzanine */
            rcheck( hpe650x_setDigitalIFBandwidth( sessionID[ifp], mezz, ddcbw));
        }
    /* synchronize autoranging on all modules in system */
    if( VI_TRUE == SYNC_AUTORANGE)
    {
        /* sync autoranging on master channel */
        rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile,  &Gsystem));
        /* synch autoranging on the slave channels */
        if( IFCHNLS > 1)
            rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile,  Gsystem));
    }
/* Set up to do the actual data capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC1, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC5, VI_TRUE ));
        /*  rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC2, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC3, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC4, VI_TRUE ));*/
            mSleep( 50);        /* allow for DDC transients */
            rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
            rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
            rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
            rcheck( hpe650x_setMultipleTriggerAction( sessionID[ifp], mezz, VI_TRUE ));
            rcheck( hpe650x_setCaptureTrigger( sessionID[ifp], mezz, numberoftriggers ));
            rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
            rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
            rcheck( hpe650x_startCapture( sessionID[ifp], mezz ));
        }
    /*  Synchronize all DDCs*/
rcheck( SynchronizeDDCs( SYSIFPS, sessionID, MEZZperIFP));
    /*  Prearm -- required before arming.  This stops any current activity.*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_prearmDSPForDataCollection( sessionID[ ifp], mezz));



    /* Arm the master DSP.  The master is defined as sessionID[ 0] in this example.*/
    rcheck( hpe650x_armDSPForDataCollection( sessionID[ 0], MEZZ1, VI_FALSE));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    /* Arm the slave DSPs */
    if( MEZZperIFP > 1)
        rcheck( hpe650x_armDSPForDataCollection( sessionID[0], MEZZ2, VI_TRUE ));
    if( SYSIFPS > 1)
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_armDSPForDataCollection( sessionID[ifp], mezz, VI_TRUE ));

    /*      Prompt user for the trigger.*/
    printf("Fire the triggers.  Press Enter to continue\n");
    getchar();
    /* sleep while data is captured */
    mSleep( 1000);
    /* now stop the capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_stopCapture( sessionID[ ifp], mezz ));
    /* Close all IFPs*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        rcheck( hpe650x_close(sessionID[ ifp]));
    return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 9: Stream N Samples of I/Q Data to the Link Port Using a Trigger

Use the following programming example to stream a specified number of digital I/Q data samples to the link port using a trigger to begin the streaming process.

Figure 3-34 shows the main process steps for this scenario.

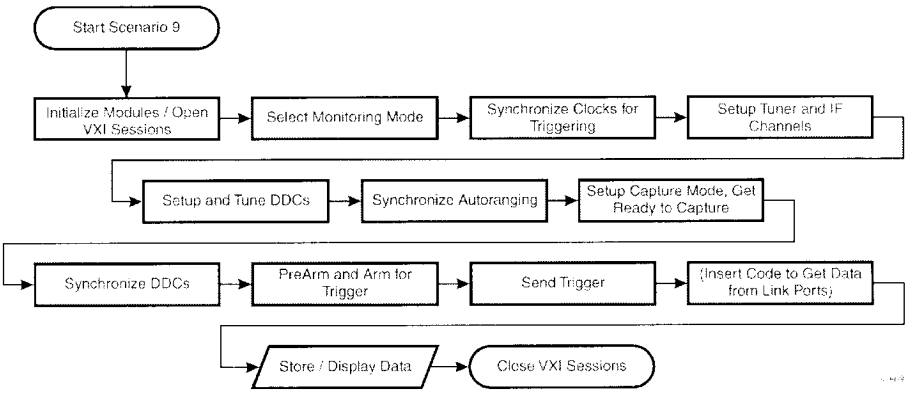


Figure 3-34 Scenario 9: Stream N Samples of I/Q Data to the Link Port Using a Trigger

```
/* Scenario 9:Stream n-samples of Digital IQ data out the link port
   using a trigger.
   Notes:

   1. The following files must exist in the same directory as the
      source code.
      "hpe650x.h"
      "commonex.h"
      "commonex.c"
      "visatype.h"
      "vpptype.h"
   2. The file "hpe650x.lib" must be available during linking.
   3. The user should read through the code modifying the
      following, as necessary:
      a. Constants that define the number of IF channels, IF
         processors and mezzanines
      b. Variables that define VXI addresses
      c. SYNC_AUTORANGE controls whether all IFP autoranging
         is synchronized.
      d. TunerExists should be set to VI_TRUE if a tuner exists
         otherwise is should be set to VI_FALSE.
      e. SYNC_CLOCKS is set to VI_TRUE when coherent measurements
         across multiple mezzanines and/or multiple IFPs is desired.
      f. "fcenter" which defines the tuner's center frequency
      g. "ddcfreq" sets the frequency tuning of the DDCs relative
         to the IF.
      h. "ddcbw" sets the bandwidth of the DDCs. See the user
         documentation for a table of appropriate values.
      i. "AnalogFilter", "suspend", "format", "collect" and "output"
         can be set to predefined constants as described below.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4. The commands hpe650x_init(), hpe650x_initIFChannel() and hpe650x_setMonitoringMode() must be executed before other commands, such as hpe650x_setTunerFrequency().
5. To synchronize DDCs across multiple mezzanines, the third parameter of hpe650x_armDDCsForSynchronization in "commonex.c" must be set to VI_TRUE. In addition, two trigger signals must be sent from a external source to mezzanine 1.
6. IMPORTANT: In this example, the two trigger signals MUST be sent when prompted, and BEFORE ANY SUBSEQUENT COMMANDS.

Disclaimer: This code is provided AS IS. It is a sample and unsupported.

```
*/
#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS      1
/* Number of IF processors installed */
#define SYSIFPS      1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs.
      This is an array for which each element contains a VXI address string.
      This example shows a system with only one IFP. */
ViRsrc IFPvxiID[SYSIFPS] = {"VXI0::43::INSTR"};
/* Extend for IFs.
      There are arrays for which each element contains a VXI address integer.
      The first dimension spans the system IFPs. The second dimension spans
      the IF channels for each IF processor. This example shows the VXI
      module addresses for one 3 GHz tuner. */
ViInt32 LO_log_addr[SYSIFPS][IFCHNLS]   {41};
ViInt32 OneG_log_addr[SYSIFPS][IFCHNLS] = {42};
ViInt32 ThreeG_log_addr[SYSIFPS][IFCHNLS] = {40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_FALSE
/*      If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/* Use this to switch on synchronized IFP clocks (VI_TRUE). This MUST be
      done if there is more than one mezzanine on an IFP, even if only one
      mezzanine is used in the measurement. This is because of how the
      trigger signal is propagated throughout multiple mezzanines.*/
#define SYNC_CLOCKS VI_TRUE
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/*****************************************************************************
    Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1   0
#define MEZZ1  0
#define MEZZ2  1
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE      1
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING   0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY    3
#define DIGITALIQ         0
#define ADCDATA           1
#define VXIBUS            0
#define PORT              1
#define TRIGGER           1
#define FREERUN           0
#define SRAM              1
#define LINKPORT0
/*****************************************************************************/
/*
  The macro 'rcheck' saves the return status in the variable 'ret'.
  If the return status indicates an error, 'rcheck' will call 'error_exit'.
  The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.
*/
#define rcheck(A) ( ((result = A) == VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp],result,  LINE ,  FILE )) )
int main()
{
      int                 ifp, mezz, chan;
      ViStatusresult;
      ViSessionsessionID[ SYSIFPS];
      ViInt32      correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
      ViInt32      sample = 2000;                /*      Number of samples or 0 if taking indefinite length. */
      ViReal64 finit = 20e6;                     /*                  This number should be less than 1 GHz.*/
      ViReal64 fcenter = 2600e6;                 /*    This number can be any valid frequency for the tuner.*/
      ViInt16      ddcfreq = 0;                  /*      See user manual for "hpe650x_setDDCFrequency"*/
      ViInt16      ddcbw = 8;                    /*   See user manual for "hpe650x_setDigitalIFBandwidth"*/
      ViInt32      AnalogFilter = IF700KHZ;      /* Other choices are IF30KHZ or IF8MHZ                  */
      ViBoolean suspend = TRIGGER;               /* Choices are FREERUN or TRIGGER                       */
      ViBoolean format = DIGITALIQ;              /* Choices are DIGITALIQ or ADCDATA                     */
      ViBoolean collect  LINKPORT;               /* Choices are SRAM or LINKPORT                         */
      ViBoolean output = PORT;                   /* Choices are VXIBUS or PORT                           */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* Configure the receiver state, initialize all IFP's and IF's */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
    {
        rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
        for( chan = 0; chan < IFCHNLS; chan++)
            /*  Initialize the IF channels including establishing communication to their addresses. The initial  frequency needs to be below 1 GHz.*/
            rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],  OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan]));
    }
        /* Put all IFPs into monitoring mode */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
    /*  Synchronize clocks between all mezzanines and IFPs.  The IFP at sessionID[ 0] is always the master in this  example.*/
    if( VI_TRUE == SYNC_CLOCKS)
    {
        /* Instructs the master IFP to send its clock out VXI.  Since the sample clock is also the DSP clock, this  action risks causing the DSPs to hang.  Therefore
all the modulues are also reinitialized to reboot the  DSPs.*/
        rcheck( ConfigureMasterIFP( sessionID[ 0]));
        rcheck( hpe650x_init( IFPvxiID[ IF1], VI_TRUE, VI_TRUE, &sessionID[ IF1]));
        for( chan = 0; chan < IFCHNLS; chan++)
            rcheck( hpe650x_initIFChannel( sessionID[ IF1], chan, TunerExists, finit, LO_log_addr[ IF1][ chan],  OneG_log_addr[ IF1][ chan],
ThreeG_log_addr[ IF1][ chan]));
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ IF1], mezz));
        /* Now configure all slave IFPs.  The slaves are instructed to accept the master clock.  Finally, the slaves  must be reinitialized to boot the DSPs.*/
        for( ifp = 1; ifp < SYSIFPS; ifp++)
        {
            rcheck( ConfigureSlaveIFP( sessionID[ ifp]));
            rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],  OneG_log_addr[ ifp][ chan],
ThreeG_log_addr[ ifp][ chan]));
            for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
        }
    }
    /* Set up and tune the IFs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( chan = 0; chan < IFCHNLS; chan++)
        {
            /*  Set the tuner frequency to the value of "fcenter"*/
            rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter ));
            /*  Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
            rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));
            /*  Activate autorange once after the initialization of each mezzanine.*/
            rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
        }

        /* Set up and tune DDCs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            /* The following function can be used for each DDC installed */
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC1, RELATIVE, ddcfreq));
            rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC5, RELATIVE, ddcfreq));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        /*      rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC2, RELATIVE, ddcfreq));
                rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC3, RELATIVE, ddcfreq));
                rcheck( hpe650x_setDDCFrequency( sessionID[ifp], mezz, DDC4, RELATIVE, ddcfreq));*/
                /* only need to call this function once per mezzanine */
                rcheck( hpe650x_setDigitalIFBandwidth( sessionID[ifp], mezz, ddcbw));
        }
    /* synchronize autoranging on all modules in system */
    if( VI_TRUE == SYNC_AUTORANGE)
    {
        /* sync autoranging on master channel */
        rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile,      &Gsystem));
        /* synch autoranging on the slave channels */
        if( IFCHNLS > 1)
            rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile,          Gsystem));
    }

/* Set up to do the actual data capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC1, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC5, VI_TRUE ));
        /*  rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC2, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC3, VI_TRUE ));
            rcheck( hpe650x_setCaptureDataDDCNum( sessionID[ifp], mezz, DDC4, VI_TRUE ));*/
            mSleep( 50);        /* allow for DDC transients */
            rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
            rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
            rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
            rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
            rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
            rcheck( hpe650x_startCapture( sessionID[ifp], mezz ));
        }
    /*      Synchronize all DDCs*/
rcheck( SynchronizeDDCs( SYSIFPS, sessionID, MEZZperIFP));
    /*      Prearm -- required before arming.  This stops any current activity.*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_prearmDSPForDataCollection( sessionID[ ifp], mezz));
    /* Arm the master DSP.  The master is defined as sessionID[ 0] in this example.*/
    rcheck( hpe650x_armDSPForDataCollection( sessionID[ 0], MEZZ1, VI_FALSE));
    /* Arm the slave DSPs */
    if( MEZZperIFP > 1)
        rcheck( hpe650x_armDSPForDataCollection( sessionID[0], MEZZ2, VI_TRUE ));
    if( SYSIFPS > 1)
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_armDSPForDataCollection( sessionID[ifp], mezz, VI_TRUE ));

    /*      Prompt user for the trigger.*/
    printf("Fire the trigger.  Press Enter to continue\n");
    getchar();
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    /* sleep while data is captured */
    mSleep( 1000);
    /* Close all IFPs*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        rcheck( hpe650x_close(sessionID[ ifp]));
    return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 10: Stream ADC Data Indefinitely to the Link Port

Use the following programming example to stream an indefinite number of ADC (full rate) data samples to the link port.

Figure 3-35 shows the main process steps for this scenario.

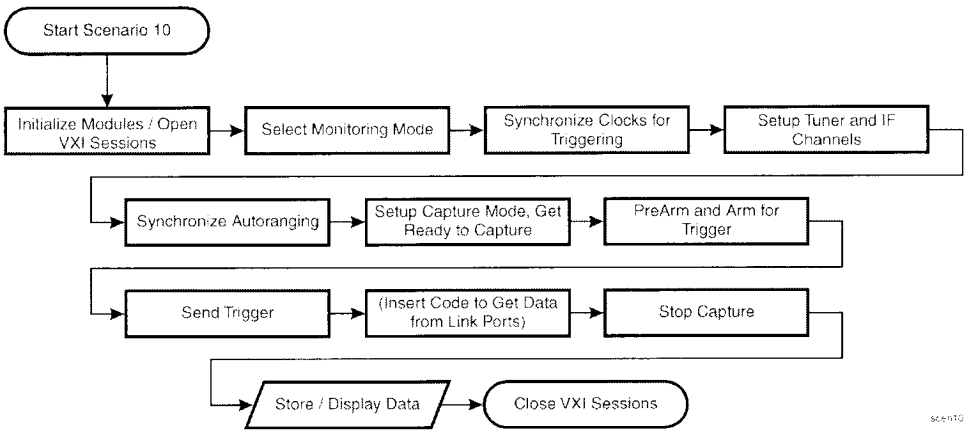


Figure 3-35 Scenario 10: Stream ADC Data Indefinitely to the Link Port

```
/*  Scenario 10:Stream ADC data indefinitely to the Link Port
        with a trigger.

        Notes:

        1.  The following files must exist in the same directory as the
                source code.
                "hpe650x.h"
                "commonex.h"
                "commonex.c"
                "visatype.h"
                "vpptype.h"

        2.  The file "hpe650x.lib" must be available during linking.

        3.      The user should read through the code modifying the
                following, as necessary:
                a.      Constants that define the number of IF channels, IF
                        processors and mezzanines
                b.      Variables that define VXI addresses
                c.      SYNC_AUTORANGE controls whether all IFP autoranging
                        is synchronized.
                d.      TunerExists should be set to VI_TRUE if a tuner exists
                        otherwise is should be set to VI_FALSE.
                e.      SYNC_CLOCKS is set to VI_TRUE when coherent measurements
                        across multiple mezzanines and/or multiple IFPs is desired.
                f.      "fcenter" which defines the tuner's center frequency
                g.      "AnalogFilter", "suspend", "format", "collect" and "output"
                        can be set to predefined constants as described below.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4. The commands hpe650x_init(), hpe650x_initIFChannel() and hpe650x_setMonitoringMode() must be executed before other commands, such as hpe650x_setTunerFrequency().

Disclaimer: This code is provided AS IS. It is a sample and unsupported.

```
*/
#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS      1
/* Number of IF processors installed */
#define SYSIFPS      1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs.
      This is an array for which each element contains a VXI address string.
      This example shows a system with only one IFP. */
ViRsrc  IFPvxiID[ SYSIFPS] = { "VXI0::43::INSTR"};
/* Extend for IFs.
      There are arrays for which each element contains a VXI address integer.
      The first dimension spans the system IFPs.  The second dimension spans
      the IF channels for each IF processor.  This example shows the VXI
      module addresses for one 3 GHz tuner.  */
ViInt32 LO_log_addr[ SYSIFPS][ IFCHNLS] = { 41};
ViInt32 OneG_log_addr[ SYSIFPS][ IFCHNLS]   { 42};
ViInt32 ThreeG_log_addr[ SYSIFPS][ IFCHNLS] = { 40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_FALSE
/*      If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/* Use this to switch on synchronized IFP clocks (VI_TRUE).  This MUST be
      done if there is more than one mezzanine on an IFP, even if only one
      mezzanine is used in the measurement.  This is because of how the
      trigger signal is propagated throughout multiple mezzanines.*/
#define SYNC_CLOCKS VI_TRUE
/*****************************************************************************
    Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1   0
#define MEZZ1  0
#define MEZZ2  1
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE      1
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING  0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY   3
#define DIGITALIQ         0
#define ADCDATA           1
#define VXIBUS            0
#define PORT              1
#define TRIGGER           1
#define FREERUN           0
#define SRAM              1
#define LINKPORT0
/*****************************************************************************/
/*
  The macro 'rcheck' saves the return status in the variable 'ret'.
  If the return status indicates an error, 'rcheck' will call 'error_exit'.
  The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.
*/
#define rcheck(A) ( ((result = A) == VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp],result, __LINE__, __FILE__)) )
int main()
{
      int                ifp, mezz, chan;
      ViStatusresult;
      ViSessionsessionID[ SYSIFPS];
      ViInt32      correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
      ViInt32      sample = 0;                  /*      Number of samples or 0 if taking indefinite length. */
      ViReal64 finit = 20e6;                    /*                  This number should be less than 1 GHz.*/
      ViReal64 fcenter = 2600e6;                /*    This number can be any valid frequency for the tuner.*/
      ViInt32      AnalogFilter = IF700KHZ;     /* Other choices are IF30KHZ or IF8MHZ                  */
      ViBoolean suspend = TRIGGER;              /* Choices are FREERUN or TRIGGER                       */
      ViBoolean format = ADCDATA;               /* Choices are DIGITALIQ or ADCDATA                     */
      ViBoolean collect = LINKPORT;             /* Choices are SRAM or LINKPORT                         */
      ViBoolean output = PORT;                  /* Choices are VXIBUS or PORT                           */
      /* Configure the receiver state, initialize all IFP's and IF's */
      for( ifp = 0; ifp < SYSIFPS; ifp++)
      {
            rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
            for( chan = 0; chan < IFCHNLS; chan++)
                  /*      Initialize the IF channels including establishing communication to their addresses. The initial      frequency needs to be below 1 GHz.*/
                  rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],      OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan]));
      }
```

```
/* Put all IFPs into monitoring mode */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
    /*  Synchronize clocks between all mezzanines and IFPs.  The IFP at sessionID[ 0] is always the master in this example.*/
    if( VI_TRUE == SYNC_CLOCKS)
    {
        /* Instructs the master IFP to send its clock out VXI.  Since the sample clock is also the DSP clock, this action riskscausing the DSPs to hang.  Therefore all the modulues are also reinitialized to reboot the DSPs.*/
        rcheck( ConfigureMasterIFP( sessionID[ 0]));
        rcheck( hpe650x_init( IFPvxiID[ IF1], VI_TRUE, VI_TRUE, &sessionID[ IF1]));
        for( chan = 0; chan < IFCHNLS; chan++)
            rcheck( hpe650x_initIFChannel( sessionID[ IF1], chan, TunerExists, finit, LO_log_addr[ IF1][ chan], OneG_log_addr[ IF1][ chan], ThreeG_log_addr[ IF1][ chan]));
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ IF1], mezz));
        /* Now configure all slave IFPs.  The slaves are instructed to accept the master clock.  Finally, the slaves must be reinitialized to boot the DSPs.*/
        for( ifp = 1; ifp < SYSIFPS; ifp++)
        {
            rcheck( ConfigureSlaveIFP( sessionID[ ifp]));
            rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan], OneG_log_addr[ ifp][ chan], ThreeG_log_addr[ ifp][ chan]));
            for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
        }
    }
    /* Set up and tune the IFs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( chan = 0; chan < IFCHNLS; chan++)
        {
            /*  Set the tuner frequency to the value of "fcenter"*/
            rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter ));
            /*  Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
            rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));
            /*  Activate autorange once after the initialization of each mezzanine.*/
            rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
        }
    /* synchronize autoranging on all modules in system */
    if( VI_TRUE == SYNC_AUTORANGE)
    {
        /* sync autoranging on master channel */
        rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile, &Gsystem));
        /* synch autoranging on the slave channels */
        if( IFCHNLS > 1)
            rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));
    }

/* Set up to do the actual data capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
            rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
            rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
            rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
            rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
            rcheck( hpe650x_startCapture( sessionID[ifp], mezz ));
        }
    /*      Prearm -- required before arming.  This stops any current activity.*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_prearmDSPForDataCollection( sessionID[ ifp], mezz));

    /* Arm the master DSP.  The master is defined as sessionID[ 0] in this example.*/
    rcheck( hpe650x_armDSPForDataCollection( sessionID[ 0], MEZZ1, VI_FALSE));

    /* Arm the slave DSPs */
    if( MEZZperIFP > 1)
        rcheck( hpe650x_armDSPForDataCollection( sessionID[0], MEZZ2, VI_TRUE ));
    if( SYSIFPS > 1)
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_armDSPForDataCollection( sessionID[ifp], mezz, VI_TRUE ));

    /*      Prompt user for the trigger.*/
    printf("Fire the trigger.  Press Enter to continue\n");
    getchar();
    /* sleep while data is captured */
    mSleep( 1000);
    /* now stop the capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_stopCapture( sessionID[ ifp], mezz ));
    /* Close all IFPs*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        rcheck( hpe650x_close(sessionID[ ifp]));
    return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 11: Stream N Samples of ADC Data to the Link Port

Use the following programming example to stream a specified number of ADC (full rate) data samples to the link port without the use of a trigger to begin the streaming process.

Figure 3-36 shows the main process steps for this scenario.

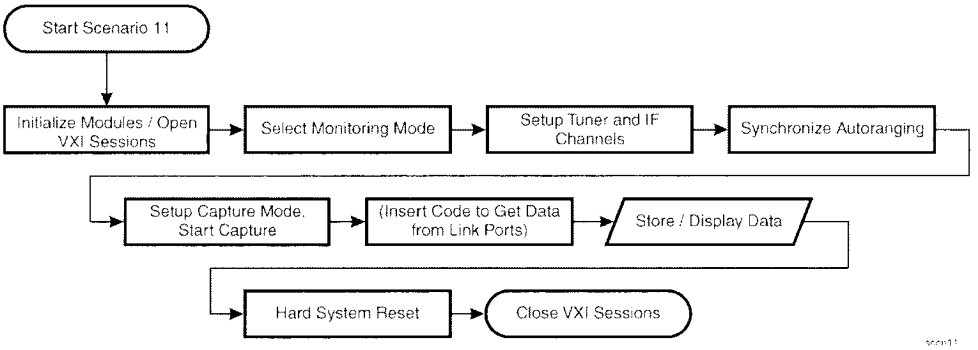


Figure 3-36 Scenario 11: Stream N Samples of ADC Data to the Link Port

```
/*  Scenario 11: Stream n-samples of ADC data to the Link Port
    with no trigger.
    Notes:
    1.  The following files must exist in the same directory as the
        source code.
        "hpe650x.h"
        "commonex.h"
        "commonex.c"
        "visatype.h"
        "vpptype.h"

    2.  The file "hpe650x.lib" must be available during linking.

    3.  The user should read through the code modifying the
        following, as necessary:
        a.  Constants that define the number of IF channels, IF
            processors and mezzanines
        b.  Variables that define VXI addresses
        c.  SYNC_AUTORANGE controls whether all IFP autoranging
            is synchronized.
        d.  TunerExists should be set to VI_TRUE if a tuner exists
            otherwise is should be set to VI_FALSE.
        e.  "fcenter" which defines the tuner's center frequency
        f.  "AnalogFilter", "suspend", "format", "collect" and "output"
            can be set to predefined constants as described below.

    4.  The commands hpe650x_init(), hpe650x_initIFChannel() and
        hpe650x_setMonitoringMode() must be executed before other
        commands, such as hpe650x_setTunerFrequency().


    Disclaimer: This code is provided AS IS.  It is a sample and unsupported.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        */

#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS      1
/* Number of IF processors installed */
#define SYSIFPS      1
/* Number of mezzanines per IFP */
#define MEZZperIFP    2
/* Extend for IFPs.
        This is an array for which each element contains a VXI address string.
        This example shows a system with only one IFP. */
ViRsrc  IFPvxiID[ SYSIFPS] = { "VXI0::43::INSTR"};
/* Extend for IFs.
        There are arrays for which each element contains a VXI address integer.
        The first dimension spans the system IFPs.  The second dimension spans
        the IF channels for each IF processor.  This example shows the VXI
        module addresses for one 3 GHz tuner.  */
ViInt32 LO_log_addr[ SYSIFPS][ IFCHNLS] = { 41};
ViInt32 OneG_log_addr[ SYSIFPS][ IFCHNLS] = { 42};
ViInt32 ThreeG_log_addr[ SYSIFPS][ IFCHNLS] = { 40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_FALSE
/*      If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/*****************************************************************************
      Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1    0
#define IF2    1
#define IFP1    0
#define MEZZ1  0
#define MEZZ2  1
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE       1
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING   0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY    3
#define DIGITALIQ          0
#define ADCDATA            1
#define VXIBUS             0
#define PORT               1
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#define TRIGGER          1
#define FREERUN          0
#define SRAM             1
#define LINKPORT0
/*****************************************************************************/
/*
  The macro 'rcheck' saves the return status in the variable 'ret'.
  If the return status indicates an error, 'rcheck' will call 'error_exit'.
  The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.
*/
#define rcheck(A) ( ((result = A) == VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp],result,__LINE__, __FILE__)) )
int main()
{
        int             ifp, mezz, chan;
        ViStatusresult;
        ViSessionsessionID[ SYSIFPS];
        ViInt32     correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
        ViInt32     sample = 2000;              /*      Number of samples or 0 if taking indefinite length. */
        ViReal64 finit = 20e6;                  /*              This number should be less than 1 GHz.*/
        ViReal64 fcenter = 2600e6;              /*      This number can be any valid frequency for the tuner.*/
        ViInt32     AnalogFilter = IF700KHZ;    /* Other choices are IF30KHZ or IF8MHZ          */
        ViBoolean suspend = FREERUN;            /* Choices are FREERUN or TRIGGER               */
        ViBoolean format = ADCDATA;             /* Choices are DIGITALIQ or ADCDATA             */
        ViBoolean collect = LINKPORT;           /* Choices are SRAM or LINKPORT                 */
        ViBoolean output = PORT;                /* Choices are VXIBUS or PORT                   */
        /* Configure the receiver state, initialize all IFP's and IF's */
        for( ifp = 0; ifp < SYSIFPS; ifp++)
        {
                rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
                for( chan = 0; chan < IFCHNLS; chan++)
                        /*      Initialize the IF channels including establishing communication to their addresses. The initial         frequency needs to be below 1 GHz.*/
                        rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],     OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan]));
        }



        /* Put all IFPs into monitoring mode */
        for( ifp = 0; ifp < SYSIFPS; ifp++)
                for( mezz = 0; mezz < MEZZperIFP; mezz++)
                        rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
        /* Set up and tune the IFs */
        for( ifp = 0; ifp < SYSIFPS; ifp++)
                for( chan = 0; chan < IFCHNLS; chan++)
                {
                        /*      Set the tuner frequency to the value of "fcenter"*/
                        rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter ));
                        /*      Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
                        rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));
                        /*      Activate autorange once after the initialization of each mezzanine.*/
                        rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
                }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    /* synchronize autoranging on all modules in system */
    if( VI_TRUE == SYNC_AUTORANGE)
    {
        /* sync autoranging on master channel */
        rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile,
                &Gsystem));
        /* synch autoranging on the slave channels */
        if( IFCHNLS > 1)
            rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile,
                        Gsystem));
    }
/* Set up to do the actual data capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
            rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
            rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
            rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
            rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
            rcheck( hpe650x_startCapture( sessionID[ifp], mezz ));
        }
    /*      Prompt user to continue*/
    printf("N Samples streamed from Link Port.  Press Enter to continue.\n");
    getchar();
    /*      Reset IFPs*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        hpe650x_hardSystemReset( sessionID[ifp]);
    /* Close all IFPs*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        rcheck( hpe650x_close(sessionID[ ifp]));

    return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 12: Stream N Samples of ADC Data to the Link Port Using a Single Trigger

Use the following programming example to stream a specified number of ADC (full rate) data samples to the link port using a single trigger to begin the streaming process.

Figure 3-37 shows the main process steps for this scenario.

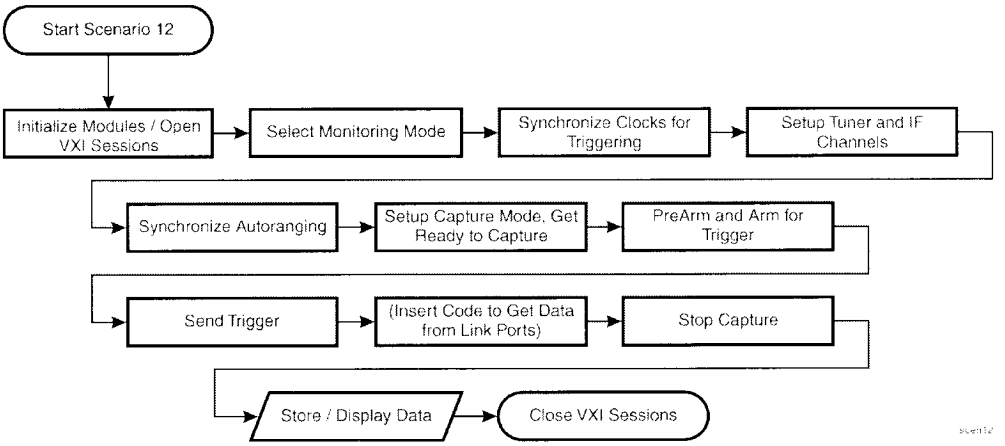


Figure 3-37 Scenario 12: Stream N Samples of ADC Data to the Link Port Using a Single Trigger

```
/* Scenario 12:  Stream n-samples of ADC data to the link port
      using a single trigger.

      Notes:

      1.  The following files must exist in the same directory as the
            source code.
            "hpe650x.h"
            "commonex.h"
            "commonex.c"
            "visatype.h"
            "vpptype.h"

      2.  The file "hpe650x.lib" must be available during linking.
      3.    The user should read through the code modifying the
            following, as necessary:
            a.    Constants that define the number of IF channels, IF
                  processors and mezzanines
            b.    Variables that define VXI addresses
            c.    SYNC_AUTORANGE controls whether all IFP autoranging
                  is synchronized.
            d.    TunerExists should be set to VI_TRUE if a tuner exists
                  otherwise is should be set to VI_FALSE.
            e.    SYNC_CLOCKS is set to VI_TRUE when coherent measurements
                  across multiple mezzanines and/or multiple IFPs is desired.
            f.    "fcenter" which defines the tuner's center frequency
            g.    "AnalogFilter", "suspend", "format", "collect" and "output"
                  can be set to predefined constants as described below.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4. The commands hpe650x_init(), hpe650x_initIFChannel() and hpe650x_setMonitoringMode() must be executed before other commands, such as hpe650x_setTunerFrequency().

Disclaimer: This code is provided AS IS. It is a sample and unsupported.

*/

```
#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS      1
/* Number of IF processors installed */
#define SYSIFPS      1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs.
      This is an array for which each element contains a VXI address string.
      This example shows a system with only one IFP. */
ViRsrc IFPvxiID[ SYSIFPS] = { "VXI0::43::INSTR"};
/* Extend for IFs.
      There are arrays for which each element contains a VXI address integer.
      The first dimension spans the system IFPs. The second dimension spans
      the IF channels for each IF processor. This example shows the VXI
      module addresses for one 3 GHz tuner. */
ViInt32 LO_log_addr[ SYSIFPS][ IFCHNLS] = { 41};
ViInt32 OneG_log_addr[ SYSIFPS][ IFCHNLS] = { 42};
ViInt32 ThreeG_log_addr[ SYSIFPS][ IFCHNLS] = { 40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_FALSE
/*    If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/* Use this to switch on synchronized IFP clocks (VI_TRUE). This MUST be
      done if there is more than one mezzanine on an IFP, even if only one
      mezzanine is used in the measurement. This is because of how the
      trigger signal is propagated throughout multiple mezzanines.*/
#define SYNC_CLOCKS VI_TRUE
/*****************************************************************************
    Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1   0
#define MEZZ1  0
#define MEZZ2  1
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE      1
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING  0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY   3
#define DIGITALIQ         0
#define ADCDATA           1
#define VXIBUS            0
#define PORT              1
#define TRIGGER           1
#define FREERUN           0
#define SRAM              1
#define LINKPORT0
/*************************************************************************/
/*
  The macro 'rcheck' saves the return status in the variable 'ret'.
  If the return status indicates an error, 'rcheck' will call 'error_exit'.
  The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.
*/
#define rcheck(A) ( ((result = A) != VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp],result,__LINE__,__FILE__)) )
int main()
{
      int                ifp, mezz, chan;
      ViStatusresult;
      ViSessionsessionID[ SYSIFPS];
      ViInt32      correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
      ViInt32      sample = 2000;                  /*      Number of samples or 0 if taking indefinite length. */
      ViReal64 finit = 20e6;                       /*            This number should be less than 1 GHz.*/
      ViReal64 fcenter = 2600e6;                   /*     This number can be any valid frequency for the tuner.*/

      ViInt32      AnalogFilter = IF700KHZ;        /* Other choices are IF30KHZ or IF8MHZ                */
      ViBoolean suspend = TRIGGER;                 /* Choices are FREERUN or TRIGGER                     */
      ViBoolean format = ADCDATA;                  /* Choices are DIGITALIQ or ADCDATA                   */
      ViBoolean collect = LINKPORT;                /* Choices are SRAM or LINKPORT                       */
      ViBoolean output = PORT;                     /* Choices are VXIBUS or PORT                         */

      /* Configure the receiver state, initialize all IFP's and IF's */
      for( ifp = 0; ifp < SYSIFPS; ifp++)
      {
            rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
            for( chan = 0; chan < IFCHNLS; chan++)
                  /*      Initialize the IF channels including establishing communication to their addresses. The initial      frequency needs to be below 1 GHz.*/
                  rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],      OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan]));
      }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* Put all IFPs into monitoring mode */
for( ifp = 0; ifp < SYSIFPS; ifp++)
    for( mezz = 0; mezz < MEZZperIFP; mezz++)
        rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
/*  Synchronize clocks between all mezzanines and IFPs.  The IFP at sessionID[ 0] is always the master in this  example.*/
if( VI_TRUE == SYNC_CLOCKS)
{
    /* Instructs the master IFP to send its clock out VXI.  Since the sample clock is also the DSP clock, this  action risks causing the DSPs to hang.  Therefore all the modulues are also reinitialized to reboot the  DSPs.*/
    rcheck( ConfigureMasterIFP( sessionID[ 0])):
    rcheck( hpe650x_ init( IFPvxiID[ IF1], VI_TRUE, VI_TRUE, &sessionID[ IF1])):
    for( chan = 0; chan < IFCHNLS; chan++)
        rcheck( hpe650x_ initIFChannel( sessionID[ IF1], chan, TunerExists, finit, LO_log_addr[ IF1][ chan],  OneG_log_addr[ IF1][ chan], ThreeG_log_addr[ IF1][ chan])):
    for( mezz = 0; mezz < MEZZperIFP; mezz++)
        rcheck( hpe650x_setMonitoringMode( sessionID[ IF1], mezz)):
    /* Now configure all slave IFPs.  The slaves are instructed to accept the master clock.  Finally, the slaves  must be reinitialized to boot the DSPs.*/
    for( ifp = 1; ifp < SYSIFPS; ifp++)
    {
        rcheck( ConfigureSlaveIFP( sessionID[ ifp])):
        rcheck( hpe650x_ init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp])):
        for( chan = 0; chan < IFCHNLS; chan++)
            rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],  OneG_log_addr[ ifp][ chan], ThreeG_log_addr[ ifp][ chan])):
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_ setMonitoringMode( sessionID[ ifp], mezz)):
    }
}
/* Set up and tune the IFs */
for( ifp = 0; ifp < SYSIFPS; ifp++)
    for( chan = 0; chan < IFCHNLS; chan++)
    {
        /*  Set the tuner frequency to the value of "fcenter"*/
        rcheck( hpe650x_ setTunerFrequency( sessionID[ ifp], chan, fcenter )):
        /*  Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
        rcheck( hpe650x_ setAnalogFilter( sessionID[ifp], chan, AnalogFilter)):
        /*  Activate autorange once after the initialization of each mezzanine.*/
        rcheck( hpe650x_ activateAutoranging( sessionID[ifp], chan)):
    }
/* synchronize autoranging on all modules in system */
if( VI_TRUE == SYNC_AUTORANGE)
{
    /* sync autoranging on master channel */
    rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile,  &Gsystem));
    /* synch autoranging on the slave channels */
    if( IFCHNLS > 1)
        rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));
    for( ifp = 1; ifp < SYSIFPS; ifp++)
        for( chan = 0; chan < IFCHNLS; chan++)
            rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile,  Gsystem));
}

/* Set up to do the actual data capture */
for( ifp = 0; ifp < SYSIFPS; ifp++)
    for( mezz = 0; mezz < MEZZperIFP; mezz++)
    {
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
            rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
            rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
            rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
            rcheck( hpe650x_setMultipleTriggerAction( sessionID[ifp], mezz, VI_FALSE ));
            rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
            rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
            rcheck( hpe650x_startCapture( sessionID[ifp], mezz ));
        }
/*      Prearm -- required before arming.  This stops any current activity.*/
for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_prearmDSPForDataCollection( sessionID[ ifp], mezz));



/* Arm the master DSP.  The master is defined as sessionID[ 0] in this example.*/
rcheck( hpe650x_armDSPForDataCollection( sessionID[ 0], MEZZ1, VI_FALSE));

/* Arm the slave DSPs */
if( MEZZperIFP > 1)
        rcheck( hpe650x_armDSPForDataCollection( sessionID[0], MEZZ2, VI_TRUE ));
if( SYSIFPS > 1)
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_armDSPForDataCollection( sessionID[ifp], mezz, VI_TRUE ));

/*      Prompt user for the trigger.*/
printf("Fire the trigger.  Press Enter to continue\n");
getchar();
/* sleep while data is captured */
mSleep( 1000);
/* now stop the capture */
for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_stopCapture( sessionID[ ifp], mezz ));
/* Close all IFPs*/
for( ifp = 0; ifp < SYSIFPS; ifp++)
        rcheck( hpe650x_close(sessionID[ ifp]));
return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 13: Stream N Samples of ADC Data to the Link Port Using Multiple Triggers

Use the following programming example to stream a specified number of ADC (full rate) data samples to the link port using multiple triggers to begin the streaming process.

Figure 3-38 shows the main process steps for this scenario.

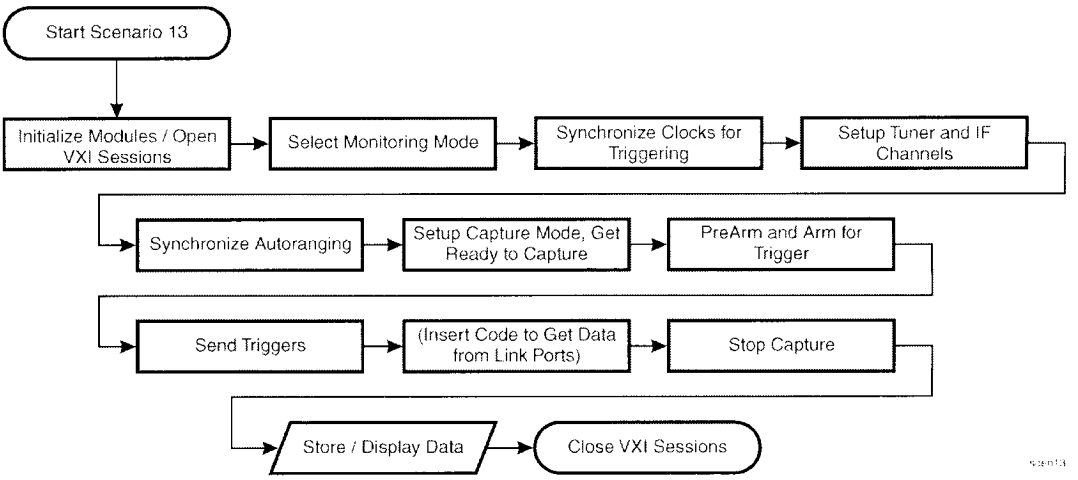


Figure 3-38 Scenario 13: Stream N Samples of ADC Data to the Link Port Using Multiple Triggers

```
/* Scenario 13: Stream n-samples of ADC data to the link port
      using multiple triggers.

      Notes:

      1. The following files must exist in the same directory as the
            source code.
            "hpe650x.h"
            "commonex.h"
            "commonex.c"
            "visatype.h"
            "vpptype.h"

      2. The file "hpe650x.lib" must be available during linking.
      3. This program requires the installation of SRAM on Mezzanine 1
            of the IF Processor whose VXI address is stored in IFPvxiID[0].
            SRAM is also required on other mezzanines in the system
            if more than one IFP exists or if MEZZperIFP is greater than 1.

      4.    The user should read through the code modifying the
            following, as necessary:
            a.    Constants that define the number of IF channels, IF
                  processors and mezzanines
            b.    Variables that define VXI addresses
            c.    SYNC_AUTORANGE controls whether all IFP autoranging
                  is synchronized.
            d.    TunerExists should be set to VI_TRUE if a tuner exists
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

otherwise is should be set to VI_FALSE.

e. SYNC_CLOCKS is set to VI_TRUE when coherent measurements across multiple mezzanines and/or multiple IFPs is desired.

f. "fcenter" which defines the tuner's center frequency

i. "AnalogFilter", "suspend", "format", "collect" and "output" can be set to predefined constants as described below.

5. The commands hpe650x_init(), hpe650x_initIFChannel() and hpe650x_setMonitoringMode() must be executed before other commands, such as hpe650x_setTunerFrequency().

Disclaimer: This code is provided AS IS. It is a sample and unsupported.

```
*/
#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS      1
/* Number of IF processors installed */
#define SYSIFPS      1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs.
      This is an array for which each element contains a VXI address string.
      This example shows a system with only one IFP. */
ViRsrc IFPvxiID[ SYSIFPS] = { "VXI0::43::INSTR"};
/* Extend for IFs.
      There are arrays for which each element contains a VXI address integer.
      The first dimension spans the system IFPs. The second dimension spans
      the IF channels for each IF processor. This example shows the VXI
      module addresses for one 3 GHz tuner. */
ViInt32 LO_log_addr[ SYSIFPS][ IFCHNLS] = { 41};
ViInt32 OneG_log_addr[ SYSIFPS][ IFCHNLS] = { 42};
ViInt32 ThreeG_log_addr[ SYSIFPS][ IFCHNLS] = { 40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_FALSE
/*    If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/* Use this to switch on synchronized IFP clocks (VI_TRUE). This MUST be
      done if there is more than one mezzanine on an IFP, even if only one
      mezzanine is used in the measurement. This is because of how the
      trigger signal is propagated throughout multiple mezzanines.*/
#define SYNC_CLOCKS VI_TRUE
/*****************************************************************************
    Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1  0
#define MEZZ1 0
#define MEZZ2 1
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE      1
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING   0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY    3
#define DIGITALIQ         0
#define ADCDATA           1
#define VXIBUS            0
#define PORT              1
#define TRIGGER           1
#define FREERUN           0
#define SRAM              1
#define LINKPORT0
/*****************************************************************************/
/*
  The macro 'rcheck' saves the return status in the variable 'ret'.
  If the return status indicates an error, 'rcheck' will call 'error_exit'.
  The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.
*/
#define rcheck(A) ( ((result = A) >= VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp],result,  LINE  ,  FILE  )) )
int main()
{
      int                ifp, mezz, chan;
      ViStatusresult;
      ViSessionsessionID[ SYSIFPS];
      ViInt32      correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
      ViInt32      sample = 2000;                /*      Number of samples or 0 if taking indefinite length. */
      ViInt32      numberoftriggers = 5;
      ViReal64 finit = 20e6;                     /*                  This number should be less than 1 GHz.*/
      ViReal64 fcenter   2600e6;                 /*    This number can be any valid frequency for the tuner.*/
      ViInt32      AnalogFilter = IF700KHZ;      /* Other choices are IF30KHZ or IF8MHZ                    */
      ViBoolean suspend = TRIGGER;               /* Choices are FREERUN or TRIGGER                         */
      ViBoolean format = ADCDATA;                /* Choices are DIGITALIQ or ADCDATA                       */
      ViBoolean collect = LINKPORT;              /* Choices are SRAM or LINKPORT                           */
      ViBoolean output = PORT;                   /* Choices are VXIBUS or PORT                             */
/* Configure the receiver state, initialize all IFP's and IF's */
      for( ifp = 0; ifp < SYSIFPS; ifp++)
      {
            rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
            for( chan = 0; chan < IFCHNLS; chan++)
                  /*      Initialize the IF channels including establishing communication to their addresses. The initial      frequency needs to be below 1 GHz.*/
                  rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],      OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan]));
      }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    /* Put all IFPs into monitoring mode */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
    /*  Synchronize clocks between all mezzanines and IFPs.  The IFP at sessionID[ 0] is always the master in this  example.*/
    if( VI_TRUE == SYNC_CLOCKS)
    {
        /* Instructs the master IFP to send its clock out VXI.  Since the sample clock is also the DSP clock, this  action risks causing the DSPs to hang.  Therefore all the modulues are also reinitialized to reboot the  DSPs.*/
        rcheck( ConfigureMasterIFP( sessionID[ 0]));
        rcheck( hpe650x_init( IFPvxiID[ IF1], VI_TRUE, VI_TRUE, &sessionID[ IF1]));
        for( chan = 0; chan < IFCHNLS; chan++)
            rcheck( hpe650x_initIFChannel( sessionID[ IF1], chan, TunerExists, finit, LO_log_addr[ IF1][ chan],  OneG_log_addr[ IF1][ chan], ThreeG_log_addr[ IF1][ chan]));
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ IF1], mezz));
        /* Now configure all slave IFPs.  The slaves are instructed to accept the master clock.  Finally, the slaves  must be reinitialized to boot the DSPs.*/
        for( ifp = 1; ifp < SYSIFPS; ifp++)
        {
            rcheck( ConfigureSlaveIFP( sessionID[ ifp]));
            rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],  OneG_log_addr[ ifp][ chan], ThreeG_log_addr[ ifp][ chan]));
            for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
        }
    }
    /* Set up and tune the IFs */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( chan = 0; chan < IFCHNLS; chan++)
        {
            /*  Set the tuner frequency to the value of "fcenter"*/
            rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter ));
            /*  Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
            rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));
            /*  Activate autorange once after the initialization of each mezzanine.*/
            rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
        }
    /* synchronize autoranging on all modules in system */
    if( VI_TRUE == SYNC_AUTORANGE)
    {
        /* sync autoranging on master channel */
        rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile,  &Gsystem));
        /* synch autoranging on the slave channels */
        if( IFCHNLS > 1)
            rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile,  Gsystem));
    }


/* Set up to do the actual data capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        {
            rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
            rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
            rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
            rcheck( hpe650x_setMultipleTriggerAction( sessionID[ifp], mezz, VI_TRUE ));
            rcheck( hpe650x_setCaptureTrigger( sessionID[ifp], mezz, numberoftriggers ));
            rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
            rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
            rcheck( hpe650x_startCapture( sessionID[ifp], mezz ));
        }
    /*      Prearm -- required before arming.  This stops any current activity.*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_prearmDSPForDataCollection( sessionID[ ifp], mezz));



    /* Arm the master DSP.  The master is defined as sessionID[ 0] in this example.*/
    rcheck( hpe650x_armDSPForDataCollection( sessionID[ 0], MEZZ1, VI_FALSE));

    /* Arm the slave DSPs */
    if( MEZZperIFP > 1)
        rcheck( hpe650x_armDSPForDataCollection( sessionID[0], MEZZ2, VI_TRUE ));
    if( SYSIFPS > 1)
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_armDSPForDataCollection( sessionID[ifp], mezz, VI_TRUE ));

    /*      Prompt user for the trigger.*/
    printf("Fire the triggers.  Press Enter to continue\n");
    getchar();
    /* sleep while data is captured */
    mSleep( 1000);
    /* now stop the capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_stopCapture( sessionID[ ifp], mezz ));
    /* Close all IFPs*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        rcheck( hpe650x_close(sessionID[ ifp]));
    return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 14: Capture N Samples of Full Rate ADC Data Across the VXI Bus

Use the following programming example to capture a specified number of ADC (full rate) data samples across the VXI bus without the use of a trigger to begin the capture process.

Figure 3-39 shows the main process steps for this scenario.

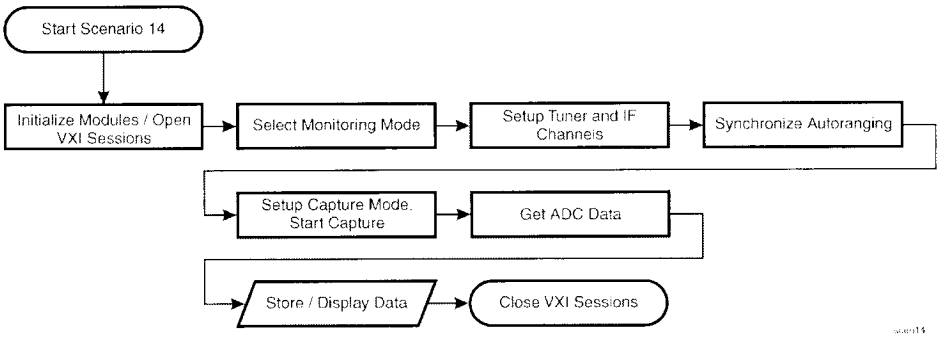


Figure 3-39 Scenario 14: Capture N Samples of Full Rate ADC Data Across the VXI Bus

```
/*
Scenario 14:Capture n-samples of ADC data and output to
the VXI bus without an external trigger.
Notes:
1. The following files must exist in the same directory as the
      source code.
      "hpe650x.h"
      "commonex.h"
      "commonex.c"
      "visatype.h"
      "vpptype.h"
2. The file "hpe650x.lib" must be available during linking.
3. This program requires the installation of SRAM on Mezzanine 1
      of the IF Processor whose VXI address is stored in IFPvxiID[0].
      SRAM is also required on other mezzanines in the system
      if more than one IFP exists or if MEZZperIFP is greater than 1.

4.    The user should read through the code modifying the
      following, as necessary:
      a.    Constants that define the number of IF channels, IF
            processors and mezzanines
      b.    Variables that define VXI addresses
      c.    SYNC_AUTORANGE controls whether all IFP autoranging
            is synchronized.
      d.    TunerExists should be set to VI_TRUE if a tuner exists
            otherwise is should be set to VI_FALSE.
      e.    SYNC_CLOCKS is set to VI_TRUE when coherent measurements
            across multiple mezzanines and/or multiple IFPs is desired.
      f.    "fcenter" which defines the tuner's center frequency
      g.    "AnalogFilter", "suspend", "format", "collect" and "output"
            can be set to predefined constants as described below.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5. The commands hpe650x_init(), hpe650x_initIFChannel() and hpe650x_setMonitoringMode() must be executed before other commands, such as hpe650x_setTunerFrequency().

Disclaimer: This code is provided AS IS. It is a sample and unsupported.

```
*/
#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS     1
/* Number of IF processors installed */
#define SYSIFPS     1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs.
     This is an array for which each element contains a VXI address string.
     This example shows a system with only one IFP. */
ViRsrc IFPvxiID[ SYSIFPS] = { "VXI0::43::INSTR"};
/* Extend for IFs.
     There are arrays for which each element contains a VXI address integer.
     The first dimension spans the system IFPs. The second dimension spans
     the IF channels for each IF processor. This example shows the VXI
     module addresses for one 3 GHz tuner. */
ViInt32 LO_log_addr[ SYSIFPS][ IFCHNLS] = { 141};
ViInt32 OneG_log_addr[ SYSIFPS][ IFCHNLS] = { 170};
ViInt32 ThreeG_log_addr[ SYSIFPS][ IFCHNLS] = { 40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_TRUE
/*     If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/*****************************************************************************
    Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1   0
#define MEZZ1  0
#define MEZZ2  1
#define DDC1 0
#define DDC2 1
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE     1
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING  0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY   3
#define DIGITALIQ          0
#define ADCDATA            1
#define VXIBUS             0
#define PORT               1
#define TRIGGER            1
#define FREERUN            0
#define SRAM               1
#define LINKPORT0
/*****************************************************************************/
/*
  The macro 'rcheck' saves the return status in the variable 'ret'.
  If the return status indicates an error, 'rcheck' will call 'error_exit'.
  The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.
*/
#define rcheck(A) ( ((result = A) == VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp],result,__LINE__,__FILE__)) )
int main()
{
        int                     i;
        int                     ifp, mezz, chan;
        ViStatusresult;
        ViSessionsessionID[ SYSIFPS];
        ViInt32         correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
        ViInt32         sample = 2000;
        ViInt32         length = 2000;
        ViInt16         ADCData[ 2000];
        ViReal64 finit = 20e6;                  /*              This number should be less than 1 GHz.*/
        ViReal64 fcenter   2600e6;              /*      This number can be any valid frequency for the tuner.*/
        ViInt32         AnalogFilter = IF8MHZ;  /* Other choices are IF30KHZ or IF8MHZ          */
        ViBoolean suspend = FREERUN;            /* Choices are FREERUN or TRIGGER               */
        ViBoolean format = ADCDATA;             /* Choices are DIGITALIQ or ADCDATA             */
        ViBoolean collect = SRAM;               /* Choices are SRAM or LINKPORT                 */
        ViBoolean output = VXIBUS;              /* Choices are VXIBUS or PORT                   */
        /* Configure the receiver state, initialize all IFP's and IF's */
        for( ifp = 0; ifp < SYSIFPS; ifp++)
        {
                rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
                for( chan = 0; chan < IFCHNLS; chan++)
                        /*      Initialize the IF channels including establishing communication to their addresses. The initial         frequency needs to be below 1 GHz.*/
                        rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],      OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan]));
        }


        /* Put all IFPs into monitoring mode */
        for( ifp = 0; ifp < SYSIFPS; ifp++)
                for( mezz = 0; mezz < MEZZperIFP; mezz++)
                        rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));



        /* Set up and tune the IFs */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( chan   0; chan < IFCHNLS; chan++)
        {
            /*      Set the tuner frequency to the value of "fcenter"*/
            rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter ));
            /*      Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
            rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));
            /*      Activate autorange once after the initialization of each mezzanine.*/
            rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
        }
    /* synchronize autoranging on all modules in system */
    if( VI_TRUE == SYNC_AUTORANGE)
    {
        /* sync autoranging on master channel */
        rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile,        &Gsystem));

        /* synch autoranging on the slave channels */
        if( IFCHNLS > 1)
            rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));

        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( chan = 0; chan < IFCHNLS; chan++)
                rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile,              Gsystem));
    }
/* Set up to do the actual data capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
            rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
            rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
            rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
            rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
            rcheck( hpe650x_startCapture( sessionID[ifp], mezz ));
        }
    do
    {
        /* Loop until the DSP finishes giving us the code across the bus */
        result = hpe650x_getCaptureFullRateADCData( sessionID[ 0], MEZZ1, ADCData, &length );
    } while ( DATANOTREADY == result);
    if( VI_SUCCESS == result)
    {
        FILE *stream;
        stream = fopen( "mez1adc.txt", "w" );
        for( i = 0; i < length; i++)
            fprintf( stream, "%5d\n", ADCData[ i]);
        fclose( stream );

        for( i = 0; i < length; i++)
            printf(" ADC[ %4d] = %5d \n", i, ADCData[ i]);
    }

do
    {
        /* Loop until the DSP finishes giving us the code across the bus */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        result = hpe650x_getCaptureFullRateADCData( sessionID[ 0], MEZZ2, ADCData, &length );
    } while ( DATANOTREADY == result);
    if( VI_SUCCESS == result)
    {
        FILE *stream;
        stream = fopen( "mez2adc.txt", "w" );
        for( i = 0; i < length; i++)
            fprintf( stream, "%5d\n", ADCData[ i]);
        fclose( stream );

        for( i = 0; i < length; i++)
            printf(" ADC[ %4d] = %5d \n", i, ADCData[ i]);
    }
    /* Close all IFPs*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        rcheck( hpe650x_close(sessionID[ ifp]));
    return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Scenario 15: Capture N Samples of ADC Data From VXI Bus Using a Trigger

Use the following programming example to capture a specified number of ADC (full rate) data samples across the VXI bus using a trigger to begin the capture process.

Figure 3-40 shows the main process steps for this scenario.

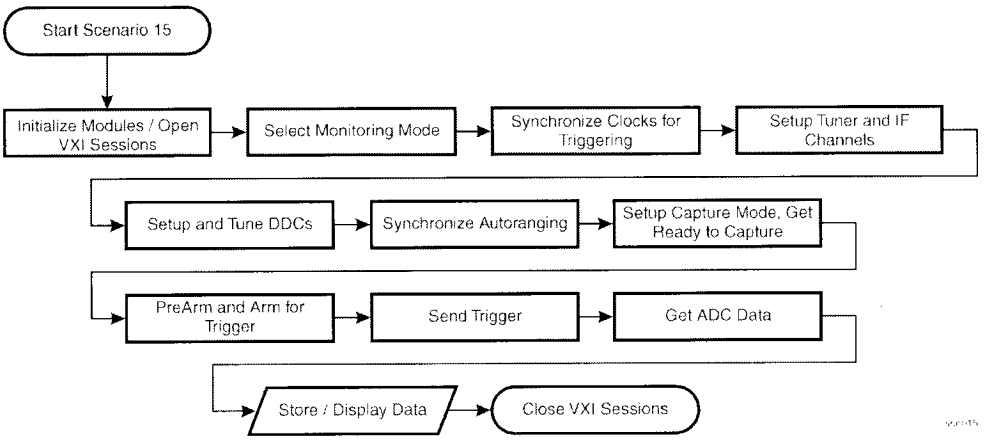


Figure 3-40 Scenario 15: Capture N Samples of ADC Data From VXI Bus Using a Trigger

```
/*  Scenario 15:Capture n-samples of ADC data and output to
    the VXI bus with an external trigger.

    Notes:

    1. The following files must exist in the same directory as the
       source code.
       "hpe650x.h"
       "commonex.h"
       "commonex.c"
       "visatype.h"
       "vpptype.h"

    2. The file "hpe650x.lib" must be available during linking.
    3. This program requires the installation of SRAM on Mezzanine 1
       of the IF Processor whose VXI address is stored in IFPvxiID[0].
       SRAM is also required on other mezzanines in the system
       if more than one IFP exists or if MEZZperIFP is greater than 1.

    4. The user should read through the code modifying the
       following, as necessary:
       a. Constants that define the number of IF channels, IF
          processors and mezzanines
       b. Variables that define VXI addresses
       c. SYNC_AUTORANGE controls whether all IFP autoranging
          is synchronized.
       d. TunerExists should be set to VI_TRUE if a tuner exists
          otherwise is should be set to VI_FALSE.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

e. SYNC_CLOCKS is set to VI_TRUE when coherent measurements across multiple mezzanines and/or multiple IFPs is desired.

f. "fcenter" which defines the tuner's center frequency

g. "AnalogFilter", "suspend", "format", "collect" and "output" can be set to predefined constants as described below.

5. The commands hpe650x_init(), hpe650x_initIFChannel() and hpe650x_setMonitoringMode() must be executed before other commands, such as hpe650x_setTunerFrequency().

```
    Disclaimer: This code is provided AS IS.  It is a sample and unsupported.
*/
#include <stdlib.h>
#include <stdio.h>
#define VISA
#include "hpe650x.h"
#include "commonex.h"
/* Number of IF channels */
#define IFCHNLS     1
/* Number of IF processors installed */
#define SYSIFPS     1
/* Number of mezzanines per IFP */
#define MEZZperIFP   2
/* Extend for IFPs.
    This is an array for which each element contains a VXI address string.
    This example shows a system with only one IFP. */
ViRsrc IFPvxiID[ SYSIFPS] = { "VXI0::43::INSTR"};
/* Extend for IFs.
    There are arrays for which each element contains a VXI address integer.
    The first dimension spans the system IFPs.  The second dimension spans
    the IF channels for each IF processor.  This example shows the VXI
    module addresses for one 3 GHz tuner.  */
ViInt32 LO_log_addr[ SYSIFPS][ IFCHNLS] = { 41};
ViInt32 OneG_log_addr[ SYSIFPS][ IFCHNLS] = { 42};
ViInt32 ThreeG_log_addr[ SYSIFPS][ IFCHNLS] = { 40};
/* Use this to switch on sychronized autorange (Change "VI_FALSE" to "VI_TRUE") */
#define SYNC_AUTORANGE VI_TRUE
/*    If the tuner exists, this constant should be set to VI_TRUE*/
#define TunerExists VI_TRUE
/* Use this to switch on synchronized IFP clocks (VI_TRUE).  This MUST be
    done if there is more than one mezzanine on an IFP, even if only one
    mezzanine is used in the measurement.  This is because of how the
    trigger signal is propagated throughout multiple mezzanines.*/
#define SYNC_CLOCKS VI_TRUE
/*****************************************************************************
    Don't change these, otherwise the code won't work anymore :-(
*****************************************************************************/
#define IF1   0
#define IF2   1
#define IFP1   0
#define MEZZ1  0
#define MEZZ2  1
#define DDC1 0
#define DDC2 1
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#define DDC3 2
#define DDC4 3
#define DDC5 4
#define IF30KHZ 0
#define IF700KHZ 1
#define IF8MHZ 2
#define RELATIVE0
#defineABSOLUTE      1
#define DATANOTREADY      -1
#define CAPTURENOTRUNNING   0
#define CAPTUREDATANOTREADY 2
#define CAPTUREDATAREADY    3
#define DIGITALIQ         0
#define ADCDATA           1
#define VXIBUS            0
#define PORT              1
#define TRIGGER           1
#define FREERUN           0
#define SRAM              1
#define LINKPORT0
/*************************************************************************/
/*
  The macro 'rcheck' saves the return status in the variable 'ret'.
  If the return status indicates an error, 'rcheck' will call 'error_exit'.
  The 'rcheck' macro requires variables 'ret' and 'instrumentID' be defined.
*/
#define rcheck(A) ( ((result = A) == VI_SUCCESS) ? \
 (result) : (error_exit(sessionID[ifp],result,  __LINE__,  __FILE__)) )
int main()
{
      int                 i;
      int                 ifp, mezz, chan;
      ViStatusresult;
      ViSessionsessionID[ SYSIFPS];
      ViInt32      correction_RAMpg, Gprofile, Gsubprofile, Gsystem;
      ViInt32      sample = 2000;
      ViInt32      length = 2000;
      ViInt16      ADCData[ 2000];
      ViReal64 finit = 20e6;                  /*          This number should be less than 1 GHz.*/
      ViReal64 fcenter = 2600e6;              /*    This number can be any valid frequency for the tuner.*/
      ViInt32      AnalogFilter = IF8MHZ;     /* Other choices are IF30KHZ or IF8MHZ          */
      ViBoolean suspend = TRIGGER;            /* Choices are FREERUN or TRIGGER               */
      ViBoolean format = ADCDATA;             /* Choices are DIGITALIQ or ADCDATA             */
      ViBoolean collect = SRAM;               /* Choices are SRAM or LINKPORT                 */
      ViBoolean output = VXIBUS;              /* Choices are VXIBUS or PORT                   */
      /* Configure the receiver state, initialize all IFP's and IF's */
      for( ifp = 0; ifp < SYSIFPS; ifp++)
      {
            rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp])):
            for( chan = 0; chan < IFCHNLS; chan++)
                  /*    Initialize the IF channels including establishing communication to their addresses. The initial      frequency needs to be below 1 GHz.*/
                  rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],    OneG_log_addr[ ifp][ chan], ThreeG_log_addr[
ifp][ chan])):
      }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* Put all IFPs into monitoring mode */
for( ifp = 0; ifp < SYSIFPS; ifp++)
    for( mezz = 0; mezz < MEZZperIFP; mezz++)
        rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
/*  Synchronize clocks between all mezzanines and IFPs.  The IFP at sessionID[ 0] is always the master in this  example.*/
if( VI_TRUE == SYNC_CLOCKS)
{
    /* Instructs the master IFP to send its clock out VXI.  Since the sample clock is also the DSP clock, this  action risks causing the DSPs to hang.  Therefore all the modulues are also reinitialized to reboot the  DSPs.*/
    rcheck( ConfigureMasterIFP( sessionID[ 0]));
    rcheck( hpe650x_init( IFPvxiID[ IF1], VI_TRUE, VI_TRUE, &sessionID[ IF1]));
    for( chan = 0; chan < IFCHNLS; chan++)
        rcheck( hpe650x_initIFChannel( sessionID[ IF1], chan, TunerExists, finit, LO_log_addr[ IF1][ chan],  OneG_log_addr[ IF1][ chan], ThreeG_log_addr[ IF1][ chan]));
    for( mezz = 0; mezz < MEZZperIFP; mezz++)
        rcheck( hpe650x_setMonitoringMode( sessionID[ IF1], mezz));
    /* Now configure all slave IFPs.  The slaves are instructed to accept the master clock.  Finally, the slaves  must be reinitialized to boot the DSPs.*/
    for( ifp = 1; ifp < SYSIFPS; ifp++)
    {
        rcheck( ConfigureSlaveIFP( sessionID[ ifp]));
        rcheck( hpe650x_init( IFPvxiID[ ifp], VI_TRUE, VI_TRUE, &sessionID[ ifp]));
        for( chan = 0; chan < IFCHNLS; chan++)
            rcheck( hpe650x_initIFChannel( sessionID[ ifp], chan, TunerExists, finit, LO_log_addr[ ifp][ chan],  OneG_log_addr[ ifp][ chan], ThreeG_log_addr[ ifp][ chan]));
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_setMonitoringMode( sessionID[ ifp], mezz));
    }
}
/* Set up and tune the IFs */
for( ifp = 0; ifp < SYSIFPS; ifp++)
    for( chan = 0; chan < IFCHNLS; chan++)
    {
        /*  Set the tuner frequency to the value of "fcenter"*/
        rcheck( hpe650x_setTunerFrequency( sessionID[ ifp], chan, fcenter ));
        /*  Set the analog filter to either 30 kHz, 700 kHz or 8 MHz using "AnalogFilter"*/
        rcheck( hpe650x_setAnalogFilter( sessionID[ifp], chan, AnalogFilter));
        /*  Activate autorange once after the initialization of each mezzanine.*/
        rcheck( hpe650x_activateAutoranging( sessionID[ifp], chan));
    }
/* synchronize autoranging on all modules in system */
if( VI_TRUE == SYNC_AUTORANGE)
{
    /* sync autoranging on master channel */
    rcheck( ConfigureMasterChannel( sessionID[ IFP1], IF1, &correction_RAMpg, &Gprofile, &Gsubprofile,  &Gsystem));
```

```
/* synch autoranging on the slave channels */
    if( IFCHNLS > 1)
        rcheck( ConfigureSlaveChannel( sessionID[ IFP1], IF2, correction_RAMpg, Gprofile, Gsubprofile, Gsystem));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
                    for( ifp = 1; ifp < SYSIFPS; ifp++)
                for( chan = 0; chan < IFCHNLS; chan++)
                    rcheck( ConfigureSlaveChannel( sessionID[ ifp], chan, correction_RAMpg, Gprofile, Gsubprofile,          Gsystem));
    }
/* Set up to do the actual data capture */
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
        {
            rcheck( hpe650x_setCaptureDataFormat( sessionID[ifp], mezz, format ));
            rcheck( hpe650x_setCaptureCollectInSRAM( sessionID[ifp], mezz, collect ));
            rcheck( hpe650x_setCaptureDataOutput( sessionID[ifp], mezz, output ));
            rcheck( hpe650x_setSuspendedCaptureTask( sessionID[ifp], mezz, suspend ));
            rcheck( hpe650x_setNumberOfSamplesToCapture( sessionID[ifp], mezz, sample ));
            rcheck( hpe650x_startCapture( sessionID[ifp], mezz ));
        }
    /*      Prearm -- required before arming.  This stops any current activity.*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        for( mezz = 0; mezz < MEZZperIFP; mezz++)
            rcheck( hpe650x_prearmDSPForDataCollection( sessionID[ ifp], mezz));

    /* Arm the master DSP.  The master is defined as sessionID[ 0] in this example.*/
    rcheck( hpe650x_armDSPForDataCollection( sessionID[ 0], MEZZ1, VI_FALSE));

    /* Arm the slave DSPs */
    if( MEZZperIFP > 1)
        rcheck( hpe650x_armDSPForDataCollection( sessionID[0], MEZZ2, VI_TRUE ));
    if( SYSIFPS > 1)
        for( ifp = 1; ifp < SYSIFPS; ifp++)
            for( mezz = 0; mezz < MEZZperIFP; mezz++)
                rcheck( hpe650x_armDSPForDataCollection( sessionID[ifp], mezz, VI_TRUE ));

    /*      Prompt user for the trigger.*/
    printf("Fire the trigger.  Press Enter to continue\n");
    getchar();
    /* sleep while data is captured */
    mSleep( 1000);
    do
    {
        /* Loop until the DSP finishes giving us the code across the bus */
        result = hpe650x_getCaptureFullRateADCData( sessionID[ 0], MEZZ1, ADCData, &length );
    } while ( DATANOTREADY == result);
    if( VI_SUCCESS == result)
    {
        FILE *stream;
        stream = fopen( "mezIadc.txt", "w" );
        for( i = 0; i < length; i++)
            fprintf( stream, "%5d\n", ADCData[ i]);
        fclose( stream );
        for( i = 0; i < length; i++)
            printf(" ADC[ %4d] = %5d \n", i, ADCData[ i]);
    }
    do
    {
        /* Loop until the DSP finishes giving us the code across the bus */
        result = hpe650x_getCaptureFullRateADCData( sessionID[ 0], MEZZ2, ADCData, &length );
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    } while ( DATANOTREADY == result);
    if( VI_SUCCESS == result)
    {
        FILE *stream;
        stream = fopen( "mez2adc.txt", "w" );
        for( i = 0; i < length; i++)
            fprintf( stream, "%5d\n", ADCData[ i]);
        fclose( stream );

        for( i = 0; i < length; i++)
            printf(" ADC[ %4d] = %5d \n", i, ADCData[ i]);
    }
    /* Close all IFPs*/
    for( ifp = 0; ifp < SYSIFPS; ifp++)
        rcheck( hpe650x_close(sessionID[ ifp]));
    return VI_SUCCESS;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4 Theory of Operation

In This Chapter

- E6501A, E6502A, E6503A VXI Receiver Description
- E6401A 20 to 1000 MHz Downconverter Operation
- E6402A Local Oscillator Operation
- E6403A 1000 to 3000 MHz Block Downconverter Operation
- E6404A IF Processor Operation

Note

The software driver supports tuning the receiver down to 2 MHz. However, specifications, typicals, and characteristics do not apply below 20 MHz.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6501A/E6502A/E6503A VXI Receiver Description

Refer to the end of this chapter for simplified block diagrams of the IF processor, LO, and downconverters.

The standard E6501A VXI receiver consists of an E6401A 20–1000 MHz downconverter module, an E6402A local oscillator module, and an E6404A IF processor module. This configuration covers an input frequency range from 20 to 1000 MHz. The E6401A translates this input signal to a 21.4 MHz IF output signal. The IF output signal has approximately 16 MHz of bandwidth. The E6402A module provides the 1st and 2nd LO signals for the E6401A module.

The E6403A option 003 downconverter extends the input frequency range to provide continuous frequency coverage from 20 to 3000 MHz. The E6403A module translates the 1000 to 3000 MHz band into the frequency range of the E6401A 20–1000 MHz downconverter module. The E6402A module provides the block downconverter LO signal to the E6403A module.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Preselector Bands

There are ten preselector bands defined for the E6401A downconverter module and an additional four bands defined for the E6403A block downconverter module.

Table 4-1 Preselector Bands

Module	Preselector Band	Characteristic Filter Frequency Range and Band Switching Points (MHz)
E6401A Downconverter	1	(40 LPF)
	2	40 to 60
	3	60 to 84
	4	84 to 118
	5	118 to 170
	6	170 to 230
	7	230 to 350
	8	350 to 450
	9	450 to 750
	10	750 to 1000
E6403A Block Downconverter	11	1000 to 1250
	12	1250 to 1800
	13	1800 to 2400
	14	2400 to 3000

Note: All preselector bands except 11 and 12 are frequency inverted at the tuner output. The receiver performs this inversion automatically.

Each preselector band (except band 1) is suboctave, ranging in frequency by less than two-to-one from highest to lowest frequency. Using suboctave preselection decreases the chance of the second harmonic of a strong signal (at half the desired frequency) from appearing as the desired signal. Suboctave preselection increases the second-order spurious-free dynamic range of the receiver.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6401A 20 to 1000 MHz Downconverter Operation

Functions

- Attenuation
- Preselection
- Preamplification
- Upconversion
- Image filtering
- Downconversion
- Gain control

Description

The E6401A 20–1000 MHz downconverter module converts input signals in the frequency range of 20 to 1000 MHz to a 21.4 MHz IF output frequency.

The 20–1000 MHz input signal path splits into two main paths, each having a solid-state 0 to 30 dB input attenuator (adjustable in 10 dB steps) to allow increased signal handling capability. Signals below 450 MHz are switched to the path that splits into eight preselector band paths (preselector bands 1 through 8). Signals above 450 MHz are switched to the path that splits into two preselector band paths (preselector bands 9 and 10). Nine of the ten preselector paths each have a filter to allow for suboctave preselection. After preselector filtering, the signal passes through a 550 MHz low-pass filter in preselector bands 1 through 8 path or a 1 GHz low-pass filter in the preselector bands 9 and 10 path. These low-pass filters offer added rejection of the 1st LO and IF signals. The two signal paths are switched into one common signal path.

Note

For the unused input path (*low path* or *high path*), the input attenuator is set to 30 dB of attenuation and the preselector switches are set for optimum isolation.

The common signal path passes through a preamplifier which compensates for losses in the preselector switches and filters. An additional 1 GHz low-pass filter follows for more LO and IF rejection. The signal is then upconverted in the first mixer to a 1st IF frequency of 1221.4 MHz. Bandpass filters provide image rejection at the 1st IF frequency. Two amplifiers in the 1st IF path compensate for loss in these filters. The 1st IF signal is downconverted to the 2nd IF frequency of 21.4 MHz in the 2nd mixer. The signal passes through a 30 MHz low-pass filter and a final amplifier stage. Before the 21.4 MHz IF output, a solid-state 0 to 15 dB variable output attenuator, adjustable in 1 dB steps, sets the approximate gain of the E6401A module to +10 dB in the receiver system.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Output attenuator values for several frequencies in each preselector band are stored by the factory in each module's EEPROM. These values are read once and stored as a table in the IF processor.

The block downconverter input path accepts the output signal from the optional E6403A 1000–3000 MHz block downconverter module. This signal path has two switched filters. These filters prevent undesired mixing products from mixing with the input signal and giving the appearance of degraded IF rejection. After passing through one of the two filters, the signal is switched into the path used by preselector bands 9 and 10 just before the 1 GHz low-pass filter.

The system bandwidth is set to approximately 16 MHz by the 1st IF bandpass filters centered at 1221.4 MHz. However, other filters in the signal path can also affect the overall bandwidth. In the lower preselector bands, if the tuned frequency is near the edge of a preselector band, the stopband skirts of the preselector filter will contribute to a narrowing of the overall bandwidth.

The LO signals for the first and second mixers come from the E6402A. Each of the two LO signals goes through automatic level control loops to control the power at the LO port of each mixer.

Inputs and Outputs

E6401A

- 20–1000 MHz input
- Block downconverter input (Block downconv Input), 250 to 900 MHz
- 1st LO input, approximately 1223.4 to 2221.4 MHz
- 2nd LO input, approximately 1200 MHz
- 21.4 MHz IF output with +10 dB of gain

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6402A Local Oscillator Operation

Functions

- Synthesize 1st LO and 2nd LO
- Filtering
- Ovenized reference
- LO distribution

Description

The E6402A local oscillator module provides the LO signals needed by the E6401A 20–1000 MHz downconverter and the E6403A 1000–3000 MHz block downconverter. The LOs are phase-locked to the 10 MHz reference. The E6402A module also provides reference distribution circuitry and a built-in oven-controlled crystal oscillator (OCXO).

2nd Local Oscillator

Synthesis starts with a narrowband voltage-controlled oscillator (VCO) tuned over a frequency range of 1200 MHz +/-5 MHz. The VCO signal is buffered and sent to one of two output signal paths. One path goes to the 2nd LO Output port: its signal drives the E6401A. The other path goes to the BD LO Output port: its signal drives the E6403A. The only difference between the 2nd LO and BD LO output signals is the power level.

10 MHz Reference

An internal 10 MHz OCXO is available as the frequency reference for the E6402A. This oscillator is *only* powered up when it is providing the 10 MHz reference (internal reference mode) for the LO module. Selection of the internal versus external reference is achieved using the hpe650x_**selectTuner10MHzReference** command. When the internal oscillator is active, the external reference path is disabled and all the reference signals of the module will come from the internal OCXO. The oscillator is guaranteed to meet specified frequency accuracy only after it has been selected and allowed at least thirty minutes to stabilize. The output frequency of the OCXO is controlled by a digital-to-analog converter (DAC). The value for this DAC is generated at the factory and stored in the module's EEPROM. This EEPROM is read by the E6404A IF processor and automatically sent to the LO DAC to set the OCXO frequency.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The external reference input goes through a limiter and is buffered. Then the reference, either external or internal, is split into paths leading to the:

- Ref TTL Out port used to lock the VXI backplane 10 MHz signal to the system reference
- Ref Out port with an attenuated version of the 10 MHz reference at approximately 0 dBm

The 3rd LO output is created by tripler circuitry, which takes the 10 MHz reference signal through a series of amplifiers and bandpass filters, and yields a 30 MHz fundamental signal with very low subharmonics.

If either of the LOs is in an unlocked state, an indicator light on the E6402A front panel will illuminate.

First Local Oscillator

The 1st LO provides a synthesized leveled signal from 1223.4 to 2221.4 MHz (settable to 1 Hz), nonvolatile storage (EEPROM) of calibration data, and a 10 MHz output at 0 dBm.

The VCO, labeled VCO1, produces a signal in the range of 1223.4/2 to 1110.7 MHz. Bias of the VCO, controlled by a DAC, is aligned at the factory. Its value is stored in the EEPROM. This value is read from the EEPROM by the E6404A IF processor and set automatically. The VCO is temperature compensated.

The VCO signal is buffered and then switched to one of three bands. Each band has an amplifier, frequency doubler, bandpass filters, and three-to-one band switches. The first band ranges from approximately 1200 to 1350 MHz. The bandpass filters and amplifier in this band are used to filter out the fundamental VCO signal and submultiples of the desired signals. The amplifier helps maintain a low noise floor. The 2nd band ranges from approximately 1350 to 1675 MHz; it also needs filtering of the fundamental and submultiples of the desired signal. In addition, the bandpass filter must attenuate the noise at 1221.4 MHz (the 1st IF frequency of the E6401A 20 to 1000 MHz downconverter). The third band ranges from approximately 1675 to 2300 MHz; its filtering characteristics are similar to those of the second band.

After the three-to-one bandswitch, the signal is amplified and passed through an automatic leveling control (ALC) modulator. The signal then passes through a directional coupler and a splitter. This output leveling circuit compensates for changes in the output power versus frequency and temperature.

E6402A Option 002 Module

The E6402A Option 002 offers dual LO output signals for two 20–1000 MHz downconverter modules and two 1000–3000 MHz block downconverter modules.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Inputs and Outputs

E6402A

- External reference input (Ext Ref In), 10 MHz
- Reference TTL output (Ref TTL Out)
- Reference output (Ref Out), 10 MHz
- Block downconverter LO output (BD LO Output), approximately 1200 MHz
- 1st LO output, approximately 1223.4 to 2221.4 MHz
- 2nd LO output, approximately 1200 MHz
- 3rd LO output, 30 MHz

E6402A Option 002

- External reference input (Ext Ref In), 10 MHz
- Reference TTL output (Ref TTL Out)
- Reference output (Ref Out), 10 MHz
- Block downconverter LO output (BD LO Output), approximately 1200 MHz (dual outputs)
- 1st LO output, approximately 1223.4 to 2221.4 MHz (dual outputs)
- 2nd LO output, approximately 1200 MHz (dual outputs)
- 3rd LO output, 30 MHz

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6403A 1000 to 3000 MHz Block Downconverter Operation

Functions

- Attenuation
- Preselection
- Preamplification
- Downconversion
- Image filtering
- Gain control

Description

The E6403A 1000–3000 MHz block downconverter module is a frequency extension module for the receiver. It converts input signals in the range of 1000 to 3000 MHz down to the range of 250 to 900 MHz. This puts the signals within the tuning range of the E6401A downconverter module.

The 20–3000 MHz input path goes through a solid-state input switch. When the receiver is tuned to a signal below 1000 MHz, the switch routes the input signal to the 20–1000 MHz Input port of the E6401A downconverter. When the receiver is tuned to an input signal above 1000 MHz, the switch routes the signal to the E6403A block downconverter (BD) path.

After the input switch in the block downconverter path, there is a programmable solid-state attenuator. The attenuator can be set to 0, 10, 20, or 30 dB of attenuation. The attenuator improves the dynamic range of the receiver when large signals are present at the input. Without attenuation, large signals can overload the receiver and cause spurious responses.

After the attenuator, the signal is routed through a bank of four preselector filters (preselector bands 11 through 14), a preamplifier, and a second, identical bank of preselector filters. These filters not only provide preselection, but image rejection for the mixer, and they prevent leakage of the LO signal from the mixer out the RF input connector. Two banks of filters are used to provide the isolation necessary to achieve good image rejection and LO emissions. The preamplifier compensates for loss in the preselectors, switches, and mixer, achieving good sensitivity.

The 2nd LO signal, provided by the E6402A module, is routed to the E6403A module's BD LO input port. The E6403A manipulates the 2nd LO input signal of approximately 1200 MHz, generating an LO frequency of 1.25 times (LO_{low}) or 1.75 times (LO_{high}) the 2nd LO. The value of the LO frequency depends on the frequency of the input signal as shown in Table 4-2.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-2 Frequency Translations

Tuned Frequency (MHz)	E6403A Generated LO Frequency (Approximate) (MHz)	Block Downconverter Output Frequency (MHz)	E6401A Block Downconv Input Filter
1000–1250	1500 (LO_{low})	500–250	BP
1250–1500	2100 (LO_{high})	850–600	HP
1500–**1800**		600–300	BP
1800–2100	1500 (LO_{low})	300–600 [1]	BP
2100–**2400**		600–900 [1]	HP
2400–2700	2100 (LO_{high})	300–600 [1]	BP
2700–**3000**		600–900 [1]	HP

Boldface numbers indicate preselector range

BP = bandpass, HP =high-pass

1. A reversed frequency spectrum exists at the block downconverter output and the 21.4 MHz output for these ranges.

After the second bank of filters, the input signal is downconverted in the mixer, using the E6403A generated LO frequency, to produce the IF output.

The IF output from the mixer is amplified and filtered to remove the image frequencies and the LO signal from the output of the mixer. The IF output goes through a programmable solid-state attenuator which corrects for the frequency response of the block downconverter. The IF output is then referred to as the block downconverter output. The attenuator settings are calibrated at the factory and stored in the module's EEPROM.

Inputs and Outputs

- Block downconverter LO input (BD LO Input), approximately 1200 MHz
- 20–3000 MHz input
- Block downconverter output (Block Downconv Output), 250 to 900 MHz
- 20–1000 MHz output

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E6404A IF Processor Operation

Functions

- IF analog filtering
- Automatic gain control
- Analog-to-digital conversion
- Digital downconversion and filtering
- Digital signal processing
- VXI interfacing

Description

The E6404A IF processor (IFP) module performs the final intermediate frequency (IF) signal processing on the downconverted RF signal from the tuner. Two separate channels allow independent processing of up to two IF signals. For each channel, the analog 21.4 MHz IF signal is first bandwidth limited with selectable 8 MHz, 700 kHz, or 30 kHz roofing filters. It is then amplified and sent to a 28.53 MSamples/sec analog to digital converter (ADC) and digitized. An automatic gain control (AGC) system optimizes the dynamic range of the ADC by maximizing the analog input signal to the converter. In manual gain mode, the gain can be set in 2 dB steps from –12 dBm to 48 dBm. Signals within the digitized bandwidth (set by the selected roofing filter) can then be converted to baseband by means of hardware digital downconverters (DDC) which perform complex mixing and digital filtering. Depending on the module option, multiple signals can be simultaneously downconverted, one for each DDC installed. The I and Q data streams from the downconverters are then sent to programmable digital signal processing (DSP) chips (up to two, depending on the option), where the final signal processing is performed (demodulation, FFTs, etc.). In the case of demodulation, data from the digital signal processor (DSP) is converted to analog and sent to the front panel audio connector. Other DSP data, as well as commands, are sent to or from a host computer over the VXI bus or can be captured from the front-panel link port connectors.

Mezzanine Board Description

The mezzanine board contains hardware for digital signal processing of the input IF signal. The following functions are performed on the mezzanine board: general digital signal processing, digital filtering, digital downconversion, digital demodulation, and FFT processing. The devices that perform these functions are the DSP and the DDC.

The IF signal is converted to digital form before it reaches the mezzanine board. The digital samples are sent to the DDCs and to two FIFO devices simultaneously. This allows the DSP to switch between the ADC output sample data and the DDC output sample data. The FIFOs act as synchronous buffers between the sample data coming from the ADCs and the DSP. By

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

processing ADC samples directly, the DSP can perform batch mode processing on signals with up to 8 MHz of bandwidth. The actual bandwidth will be determined by the IF analog input filter bandwidth. The possible bandwidths are 8 MHz, 700 kHz, and 30 kHz. The alternative to processing ADC samples directly is to process the sample data in the DDCs.

The DDC devices perform digital (filtering) downconversion and decimation of the ADC sample data. Thus, the wide band input data is converted to narrow band data (centered at DC). The first step of this process is to mix the samples with a signal from a numerically controlled oscillator. The DDC clock is twice the sample rate (57.066 MHz). The mixer output is then low-pass filtered. The low-pass filter is in fact a quadrature filter so the filter has in-phase and quadrature (I/Q) outputs. The output bandwidth determines the decimation rate and is programmable. The DDC output data is sent through a formatter and then to a serial port where it is sent to the DSP.

The DSP runs at 28.5333 MHz (ADC sample rate). For this reason, no processing can be done on "full span" data (data coming from the FIFO devices) in real-time. This data is processed in batch mode using SRAM buffers internal to the DSP. Alternatively, the DSP can process DDC output data in either real time or batch mode as the situation requires. This is because the DDC has decimated (reduced the sample rate) the input data. This decimation reduces the data stream thus providing the DSP with time to process data rather than having to continuously retrieve data from the data source. Within the DSP the data is processed in a variety of ways. The data is filtered, decimated, demodulated, and many other generic DSP algorithms or tasks are performed.

Since real-time demodulation is available, an output path for low bandwidth signals has been provided in the form of audio frequency output DACs. Ten DACs are available, though a single mezzanine can only have five DDCs. Therefore, if the receiver has two mezzanines, up to ten DDCs can be installed so that the demodulated data from the two mezzanines can be output on a single cable. In addition to providing analog outputs, full rate digital data and digital I/Q data may be retrieved from a mezzanine board.

Digital data, in batch form, is available from a mezzanine via the VXI backplane. Low bandwidth real-time data may also be available on the VXI backplane by restricting backplane traffic to a minimum. High bandwidth data may be retrieved via the front-panel link ports. Two DSP link ports are available and their combined bandwidth is sufficient to deliver "full span" raw ADC data in real time (8 MHz bandwidth) to an external DSP or other link port compatible device.

The final function of the mezzanine board is to be an embedded controller. The E6404A IF processor is digitally controlled. Input bandwidth, input gain, and gain correction are a few of the IF circuits that are digitally controlled. For this reason, the DSP firmware incorporates embedded control software as well as DSP software. All DSP firmware is stored in on-board flash memory.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Overview of Automatic Gain Control (AGC)

The receiver does not use the traditional AGC that analog receivers employ. Instead, autoranging and dynamic range optimization are used in the IF processor to maintain the optimum level to the ADC and to the DDCs, thereby optimizing dynamic range.

Gain is used to raise the signal to the detector's measurement range. Without AGC, the detector may be overloaded, or sensitivity may be insufficient to measure low-level signals. The E650XA employs autoranging, which functions similar to AGC in an analog receiver, but it does not require as much gain, since a wide dynamic range ADC is utilized rather than an analog detector.

Autoranging Operation

Autoranging uses the analog gain in the IF processor to maintain an optimum signal level at the input to the analog-to-digital converter (ADC) in the main signal path. The autoranging routine uses a 6 bit ADC (separate from the 12 bit ADC in the main signal path) and a log amplifier to measure the voltage envelope of the signal within the analog bandwidth of the IF processor. Custom logic then uses the code from the 6 bit ADC to automatically set step attenuators in the main signal path, resulting in an optimum signal level at the 12 bit ADC input. The relative gain range is –12 dB to +48 dB, or 60 dB overall.

The gain switches rapidly (<1 μs) and is adjustable in 2 dB increments; consequently, every time the gain changes, a step change in signal amplitude is seen at the output of the 12 bit ADC. Since for real-time operation it is undesirable to pass these abrupt amplitude changes through the narrow bandwidth digital filters in the DDCs, they are effectively removed by scaling the data stream using a RAM look-up table (called the correction RAM) located between the 12 bit ADC and the DDCs. This results in a 16 bit word that is passed to the DDCs. Changes in correction RAM scale factors are precisely timed with the analog gain changes such that the net gain change is transparent to the DDCs. In addition to removing the gain changes, the correction RAM also corrects inaccuracies in the analog gains using calibration data measured at the factory and stored in the IF processor. This technique makes it possible to perform calibrated measurements concurrently with real-time processing, such as measuring RSSI during demodulation.

It should be noted that the largest signal within the bandwidth of the IF analog filter (30 kHz, 700 kHz, or 8 MHz) will set the gain for the bandwidth. For example, in search mode where the 8 MHz filter is used and 8 MHz FFTs are performed, one high-level signal in that 8 MHz will cause the gain for the entire 8 MHz to be reduced, thereby exhibiting a higher receiver noise floor than an adjacent 8 MHz band that only has low-level signals. The adjacent band with low-level signals will have an AGC gain

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

automatically set to 48 dB, or some other high value, thus reducing the noise floor for that 8 MHz band.

Note

There is a 0 dB to 30 dB RF attenuator in the E6401A downconverter module to help set the level of signals in the receiver. This RF attenuator is *NOT* part of the autoranging algorithm. This attenuator can be programmed using the driver software, but it is not automatically adjusted.

Autoranging Benefits

First, autoranging optimizes the level to the 12 bit ADC to minimize the possibility of overload. Second, it effectively extends the dynamic range of the ADC. Third, its benefits can be seen when doing signal searches. Autoranging occurs for each 8 MHz spectrum. So, if there is a very large signal in one 8 MHz spectrum, the autoranging will decrease the gain to ensure the ADC is not overloaded, but the result is that the noise floor increases (because there is less gain). Also, the other bands are not affected by the low gain level set by the band having the very large signal. In other words, if the other bands have low level signals, the gain can be set high by the autoranging routine and consequently the noise level is lowered to allow these signals to be seen.

Other digitizers have one gain setting that is locked down for *all* bands, not just the one with the high level signal. The benefit of the autoranging is that it only decreases the gain in bands with high level signals, thereby leaving the other bands with high gains (and more importantly low noise floors).

Autoranging Routine Attack and Decay Time

The autoranging can respond to signal level increases in about 1 µs. Thus, the attack time can be considered to be about 1 µs. The hold time is 2.3 ms and the decay time will vary from approximately 9 µs to a maximum of 574 µs, depending on the size of the signal amplitude drop. An approximation would be about 10 µs for each dB of amplitude drop.

30 kHz and 700 kHz Analog Filters

When the receiver is tuned to a signal and the signal is being received through a DDC centered at 21.4 MHz, the excellent rejection of the digital filter in the DDC will suppress any adjacent channels present in the analog passband of the IF processor. However, since the autoranging will set the gain based on the sum of all signals present, a large signal (or several small ones) inside the analog passband will cause the gain to be set lower than the optimum for the signal of interest. By selecting the narrowest analog filter which will accommodate the DDC bandwidth, the additional signals can be suppressed, allowing the gain to be increased, thus improving sensitivity. Note that it is still possible to use multiple DDCs with these narrower analog filters, as long as their bandwidths fit inside the analog passband.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Processing Gain

Recall that the sampling rate is 28.533 MSamples/sec in the IF processor. This centers the 21.4 MHz IF signal in the second Nyquist band between 14.267 MHz and 28.533 MHz (IF = 21.4 =¾ x 28.533). The Nyquist bandwidth, then, is 1/2 of the sample rate, or around 14 MHz. The characteristic signal-to-noise ratio (SNR) of the ADC used in the IF processor sampled at 28.533 MS/Sec is 62 dB. If digital filtering and decimation are performed using the DDC, the resulting bandwidth reduction improves the SNR. This increase in SNR is called processing gain. For example, a 15 kHz DDC bandwidth provides 10 log (14 MHz/ 15 kHz) processing gain, or around 30 dB. Thus, the SNR in a 15 kHz bandwidth would be 62 dB + 30 dB = 92 dB. This is analogous to decreasing the resolution bandwidth on a spectrum analyzer to get a lower noise floor.

Dynamic Range Optimization

Dynamic range optimization (DRO) is a feature in the IF processor that works in conjunction with the autoranging gain to optimize the receiver's dynamic range. Autoranging maintains the optimum analog signal level at the ADC, while DRO maintains the optimum digital signal level at the DDC input.

As previously mentioned, in order to avoid gain switching transients from being sent through the digital filters in the DDCs, they are removed by normalizing the gain in the correction RAM. As the analog gain increases, the correction RAM effectively divides the 12 bit output from the ADC by the gain. The output of the correction RAM (and the input to the DDCs) is 16 bits wide. So, as the ADC output is divided by larger and larger numbers, the 12 ADC bits migrate further to the right in the 16 bit field. As the analog gain continues to increase, eventually the low order bits begin to be truncated as they are shifted beyond the 16 bit field, resulting in increased quantization spurs. To avoid this situation, the DRO continuously monitors the composite signal level in the analog passband by reading the 6 bit autorange ADC and, when conditions allow, adds back some of the gain that was removed by the correction RAM. The result is that the 12 bits from the ADC are kept as far to the left in the 16 bit field as the peak composite signal will allow, thus maximizing the input to the DDC.

Because rewriting the correction RAM momentarily disrupts the data flow, it is done as infrequently as possible. A window comparator prevents the DRO process from responding to small envelope variations and the DRO attack and decay response times are adjustable by the user over a 500 µs to 1 s range. DRO can also be disabled completely while allowing autoranging to continue to run.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Dynamic Range Optimization Attack and Decay Time

The attack and decay response times (Ta, Td) are programmable by the user between 500 µs and 1 s. The DRO uses a window comparator to monitor the peak composite signal in the analog passband. When the peak signal level increases above the upper threshold of the window and remains there for a time equal to Ta, the DRO will immediately re-optimize the correction RAM. Once re-optimized, the window is moved up so that the new signal level is roughly centered in the window. As long as the signal stays inside the new window, no other changes will occur. To avoid responding to momentary bursts in signal amplitude, Ta can be increased; however, it is desirable to keep Ta short, since it is possible to overdrive the output of the correction RAM during that time. When the peak signal level decreases below the lower threshold of the window and remains there for a time equal to Td, the DRO does not necessarily re-optimize immediately. The time it takes depends on Td and the signal level change in dB. If P is the signal level change in dB (P>=1), then the approximate time before re-optimization is

$$Td * \left[\sum_{k=0}^{\text{ceil }(P/2)-1} 1/2^{K} \right]$$

where ceil is the ceiling function, which equals the next highest integer value of its argument. For example, suppose the power change is 7 dB, then ceil(P/2) – 1 = ceil(7/2) – 1 = 4 – 1 = 3, and the time before re-optimization would be Td* (1+1/2 +1/4 +1/8) = 1.875*Td.

Interrelationship Between Autoranging, DRO, Correction RAM, and DSP

The overall gain in the IF processor can be divided into three components: analog gain (autoranging), correction RAM gain (also part of autoranging), and DSP gain (part of dynamic range optimization). Refer to these components as Ga, Gc, and Gd, respectively. The equation that relates these terms at any point in time is Ga+Gc+Gd=0. The values of Ga, Gc, and Gd are continuously adjusted, depending on signal level and processing mode (real-time versus batch mode), to obtain the best possible performance from the hardware.

In general, autoranging deals with Ga and Gc, while dynamic range optimization (DRO) deals with Gc and Gd. Ga is controlled by autorange hardware and is always automatically adjusted to optimize the input signal to the main ADC. It has a range of -12 dB to +48 dB. During real-time processing (for example, demodulation), Gc is adjusted synchronously with, and exactly opposite to, Ga, such that Ga+Gc equals a constant at any instant of time. This removes unwanted transients from the signal before being sent to the digital filters in the DDCs.

DRO, which has about 53 dB of range, is used to optimize the input level to the DDCs when signal conditions allow it. DRO accomplishes this by effectively restoring some of the gain that was removed in the correction

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RAM by scaling and rewriting the correction RAM contents. To illustrate how DRO works in conjunction with autoranging, assume an input signal level whose peak amplitude never causes the analog gain to be less than 24 dB. In this situation, $24 \le Ga \le 48$, $-48 \le Gc \le -24$, and it is assumed that Gd is zero. Also, assume that the maximum allowable receiver input signal level (receiver full scale, F Srx) is defined such that the analog gain at that level is zero (Ga=0). Thus, in this example, when the analog gain is at its minimum of 24 dB, the input signal would be -24 dBFSrx. In this situation, it is obvious that 24 dB of the available DDC dynamic range is wasted, since the signal never exceeds -24 dB relative to the DDC's full scale input. In equation form, the signal at the DDC input is: -24 dBFSrx receiver input +24 dB analog gain -24 dB correction RAM = -24 dBFSddc DDC input, where dBFSrx and dBFSddc are dB relative to the receiver and DDC full scale inputs respectively. To improve this, the input to the DDC is increased by shifting 24 dB of attenuation from the correction RAM to the DSP (Ga+Gc+Gd = 24-24+0 ==> 24+0-24). Notice that Ga+Gc+Gd = 0 is still satisfied, but the equation for the DDC input now becomes -24 dBFSrx + 24 dB analog gain + 0 dB correction RAM = 0 dBFSddc, and the dynamic range of the DDC is fully utilized. Notice also that the autoranging is still free to change the analog and correction RAM gains, Ga and Gc, rapidly in response to varying signal levels, as long as they stay within the bounds stated above. If the signal level increases above -24 dBFSrx, however, the DDC input will clip unless attenuation is shifted back into the correction RAM from the DSP.

In summary, the DRO maximizes the DDC input by continually monitoring the signal envelope and shifting gain back and forth between the correction RAM and the DSP in response to amplitude changes. It responds rapidly to increasing signals to avoid clipping and responds slowly to decreasing signals to avoid following fast envelope changes. Unlike fast autoranging which can change gains without generating signal transients through the DDC, the DRO does generate transients, but they are usually only perceptible if it changes gains too often. Its slow response to decreasing signals and the use of a window comparator to avoid reacting to small envelope fluctuations prevent it from reacting unnecessarily.

Note that fast autoranging is fully functional during this process and will respond to fast envelope changes transparently.

Dynamic Range Optimization in Search Mode

Autoranging and DRO behave somewhat differently when non-realtime (batch mode) processing is used. Search mode is an example of batch mode processing, since each time the tuner moves to a different frequency, a data record is collected for the FFT and the data stream is then ignored between records. While the data stream is being ignored, there is ample time to change gains. Therefore, there is no need to use the correction RAM gain to remove the transients. So, in search mode, the correction RAM gain still

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

corrects for imperfections in the analog gain step sizes using the factory calibration data, but does not remove the gain change by attenuating the signal as it would do during autoranging (that is, Gc is always nominally 0 dB). Each time the frequency is changed, the fast autoranging is allowed to set the analog gain and then the gain is locked down prior to the data collection. The analog gain setting is then read by the DSP to determine the scale factor it will apply to the data to compensate for the analog gain. Even though there are differences in the way gain control operates during batch mode processing, it is still referred to as autoranging, since the effect is the same.

FFT-Based Measurements

A thorough understanding of FFTs is necessary before making FFT-based measurements with the E650XA. In addition to providing basic information about FFTs, this section provides information specific to the E650XA such as FFT resolution bandwidths and improved sensitivity using FFTs.

FFT Background

The Fourier transform integral converts data from the time domain into the frequency domain. However, this integral assumes the possibility of deriving a mathematical description of the waveform to be transformed. But, real-world signals are complex and defy description by a simple equation. The Fast Fourier Transform (FFT) algorithm operates on sampled data, and provides time-to-frequency domain transformations without the need to derive the waveform equation.

The FFT is an implementation of the Discrete Time Fourier Transform, the algorithm used for transforming data from the time domain to the frequency domain. Before a receiver uses the FFT algorithm, it samples the input signal with an analog-to-digital converter (the Nyquist sampling theorem states that if samples are taken of twice the bandwidth, the signal can be reconstructed exactly). This transforms the continuous (analog) signal into a discrete (digital) signal.

Because the input signal is sampled, an exact representation of this signal is not available in either the time domain or the frequency domain. However, by spacing the samples closely, the receiver provides an excellent approximation of the input signal.

FFT Properties

As with the swept-tuned receiver, the input to the E650XA is a continuous analog voltage. Whatever the source of the input signal, the FFT algorithm requires digital data. Therefore, the receiver must convert the analog voltage into a digital representation. The first steps in building an FFT receiver are to build a sampler and an analog-to-digital converter (ADC) in order to create the digitized stream of samples that feed the FFT processor.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The FFT algorithm works on sampled data in a special way. Rather than acting on each data sample as it is converted by the ADC, the FFT waits until a number of samples (N) have been taken and transforms the complete batch of data. The sampled data representing the time-domain waveform is typically called a time record of size-N samples.

But the FFT receiver cannot compute a valid frequency-domain result until at least one time record is acquired. This is analogous to the initial settling time in an analog receiver. After the initial time record is filled, the FFT receiver is able to determine very rapid changes in the frequency domain. A typical size for N might be 1024 samples in one time record.

FFT Process

Figure 4-1 shows a summary of how the E650XA implements the FFT in its time and frequency transformation process.

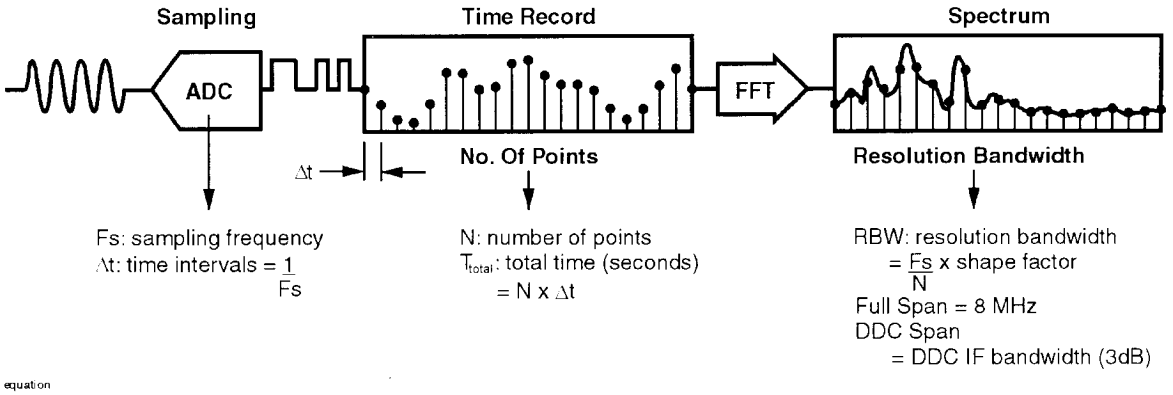


Figure 4-1 Summary of the Basic FFT Process

The input signal is sampled at the frequency of Fs, which produces a time data stream with Δt interval between samples. The time domain data batch accumulates N samples before sending it to the DSP where an FFT will be performed, and the length of the time record T equals N times Δt. After the time data is transformed into a spectrum, the resolution bandwidth is determined by the sampling frequency Fs, divided by the number of points N, times the window shape factor (1.5 for Hanning window). DSPs are optimized for performing FFTs very effectively.

Stepped FFT Measurements

The E650XA receivers use FFT technology to provide enhanced wideband measurements. Technological advances in designing ADCs and digital signal processors (DSPs) have been combined with FFT technology to provide the same results as a swept-tuned receiver but with additional capability and faster speed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The E650XA receivers are capable of making FFT measurements with excellent resolution at higher frequency ranges. This has been accomplished by translating the highest frequencies to a lower band, then performing FFTs on separate segments of the spectrum. These segments are displayed contiguously (in search mode) so that the result appears as it would with a swept-tuned receiver. Also, these receivers are capable of selecting spans and resolution bandwidths that were previously unavailable in swept-tuned receivers.

For narrow resolution bandwidths, the stepped technology in the E650XA receivers is much faster than swept-tuned receivers. In a swept-tuned receiver, time is required for the IF filters to settle on each input signal. With stepped technology, time is still required for settling, but fewer steps are required.

Windowing

This section describes why windowing is needed in the FFT process, describes window types and characteristics, and describes the relation to resolution bandwidth filters.

Windowing is a time domain function that is applied to the time data batch before it is transformed into a spectrum. Windows can be compared to the impulse response of a resolution bandwidth filter. In the frequency domain, the window determines the resolution bandwidth shape factor.

Purpose for Windowing

The FFT algorithm assumes that the signal is periodic in the time domain. If a sample can be taken of a periodic signal, such as a sine wave, with an integer number of cycles in the time domain data batch, then the assumption can be met. However, it is impossible to sample a complete cycle of a signal without knowing its frequency. In many cases, the signal is not periodic in nature. Therefore, if the signal is sampled with some truncations, errors will occur. These errors are called leakage. The purpose of windowing is to minimize these errors and correct the spectrum.

Although windowing correction does not produce a perfect spectrum, it is close enough to allow for accurate measurements to be performed.

Window Implementation

Windowing is implemented in the time domain. The time domain data batch, before it is transformed into a spectrum, is modified by the selected windowing function. The type of window must be selected by the user. Selecting an incorrect window type will result in errors. Users must have an understanding of the different window types and their characteristics.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Window Characteristics

There are three types of windows implemented in the E650XA receivers as shown in Table 4-3.

Table 4-3 Window Types and Characteristics

Parameter	Uniform	Hanning	Flat Top
window bandwidth factor	1.0	1.5	3.8
shape factor	716:1	9.1:1	2.45:1
amplitude accuracy	poor	good	best

The window types are selected using the hpe650x_**setFFTWindowType** command. Typically, the Uniform window is applied to "self-windowed" signals, such as burst and transient signals. The Hanning window has good frequency resolution but average amplitude accuracy. The Hanning window is applied to random types of signals and is the default used in the E650XA receivers. The Flat Top window has very good amplitude and average frequency resolution, making it good for measuring spurs or periodic signals.

Window as Resolution Bandwidth Filter

The E650XA receivers do not sweep resolution bandwidth filters across the frequency range. Instead, they use the DSP algorithm to generate a bank of parallel filters to compute spectrum components at the same time. The shape of the filter is determined by what window type is used. Therefore, the frequency line shape of the window is the resolution bandwidth filter shape.

FFT Resolution Bandwidth Range (Search Mode)

FFT resolution bandwidths refer to the "resolution" bandwidths of the spectral displays. For example, with the 8 MHz wide FFT, the minimum resolution bandwidth is 5 kHz, because the receiver utilizes the 8 MHz antialias filter. The resolution bandwidth can be less than 5 kHz with DDC filtering. An FFT is performed on the entire 8 MHz wide spectrum (converted to a spectral display in frequency domain) by the DSP. The 8 MHz analog antialias filter is used, but no DDC filtering is performed in search mode. The resolution bandwidth is determined by the sample frequency (28.533 MHz) divided by the FFT length. Also, the window shape factor is needed to account for the Hanning windowing. For example, 28.533 MHz divided by 8192 points then multiplied by the Hanning window shape factor of 1.5 is approximately 5 kHz. This is the minimum FFT resolution bandwidth, since it uses the maximum number of FFT points. Reducing the FFT length increases the resolution bandwidth.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

FFT Resolution Bandwidths <5 kHz (Search Mode)

FFT resolution bandwidths less than 5 kHz can be selected from the entry window on the search display of the virtual front panel. For FFT resolution bandwidths less than 5 kHz, the DDC is stepped across the 8 MHz spectrum to provide the resolution required. This process is transparent to the user.

Note that selecting FFT resolution bandwidths less than 5 kHz will affect the overall speed of the operation. As in any receiver, the lower the resolution bandwidth, the longer the sweep time.

FFT Resolution Bandwidths for DDC IF Pan Windows

The "span" of these windows is set by the DDC IF bandwidth (247 Hz to 462 kHz). The FFT resolution bandwidth of these displays is set by the FFT length.

The RBW = (sample frequency divided by FFT length) x 1.5, where 1.5 is the window resolution bandwidth factor; see Table 4-3.

Improved Sensitivity Using FFTs

Improved sensitivity using an FFT refers to increasing the processing gain. If the 8 MHz search mode or 8 MHz stare mode is used, the DDC IF bandwidths are bypassed. Therefore, the sensitivity is determined by the FFT resolution bandwidth. If the FFT resolution bandwidth is 100 kHz, the noise is determined by kTB in a 100 kHz bandwidth (plus the noise figure of the system).

In the case of the IF pan mode, assume that the DDC IF bandwidth is set to 10 kHz (which sets the span of the IF pan window), and an FFT resolution bandwidth of 1 kHz is selected. Since 1 kHz is the effective resolution bandwidth, it offers 10 log BW, or 10 dB lower noise floor than the 10 kHz DDC IF bandwidth.

Search mode provides a low noise floor by using the FFT resolution bandwidth and thus provides increased processing gain. Processing gain is further improved when the FFT resolution bandwidth and the DDC IF bandwidth ratio are considered (10 log the ratio of the two). Note that the FFT processing gain only applies to the signal display and not the signal demodulation process. The receiver sensitivity specification provides the performance of how well the receiver can demodulate small signals in the presence of noise.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Inputs and Outputs

- Ch 1 IF In
- Channel 1 Link Ports
- Channel 1 Audio/Trigger
- Ch 2 IF In
- Channel 2 Link Ports
- Channel 2 Audio/Trigger
- Ref In
- Ref Out

For a complete list of available options, refer to Table 1-1.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

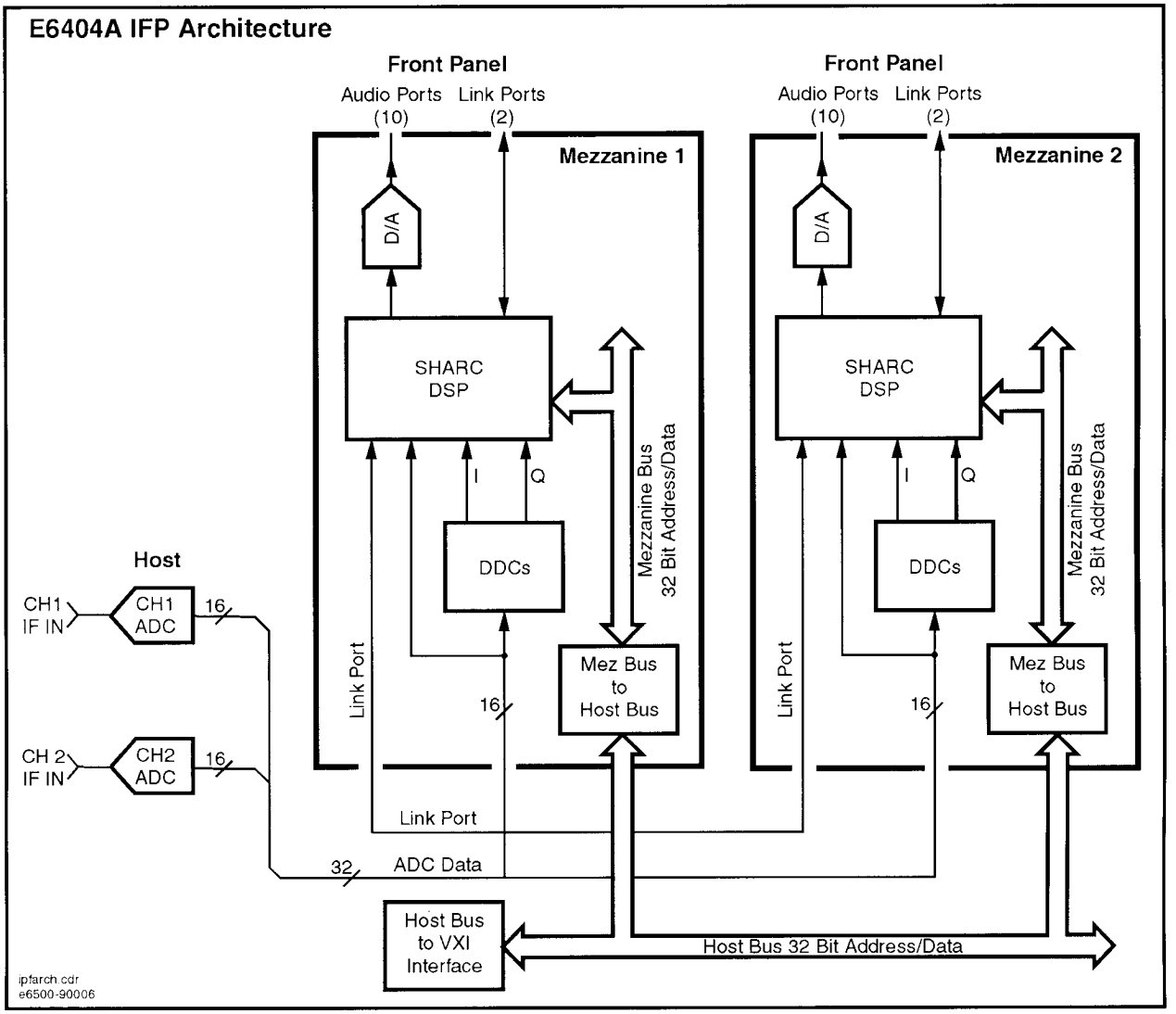


Figure 4-2 IF Processor Architecture Diagram

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

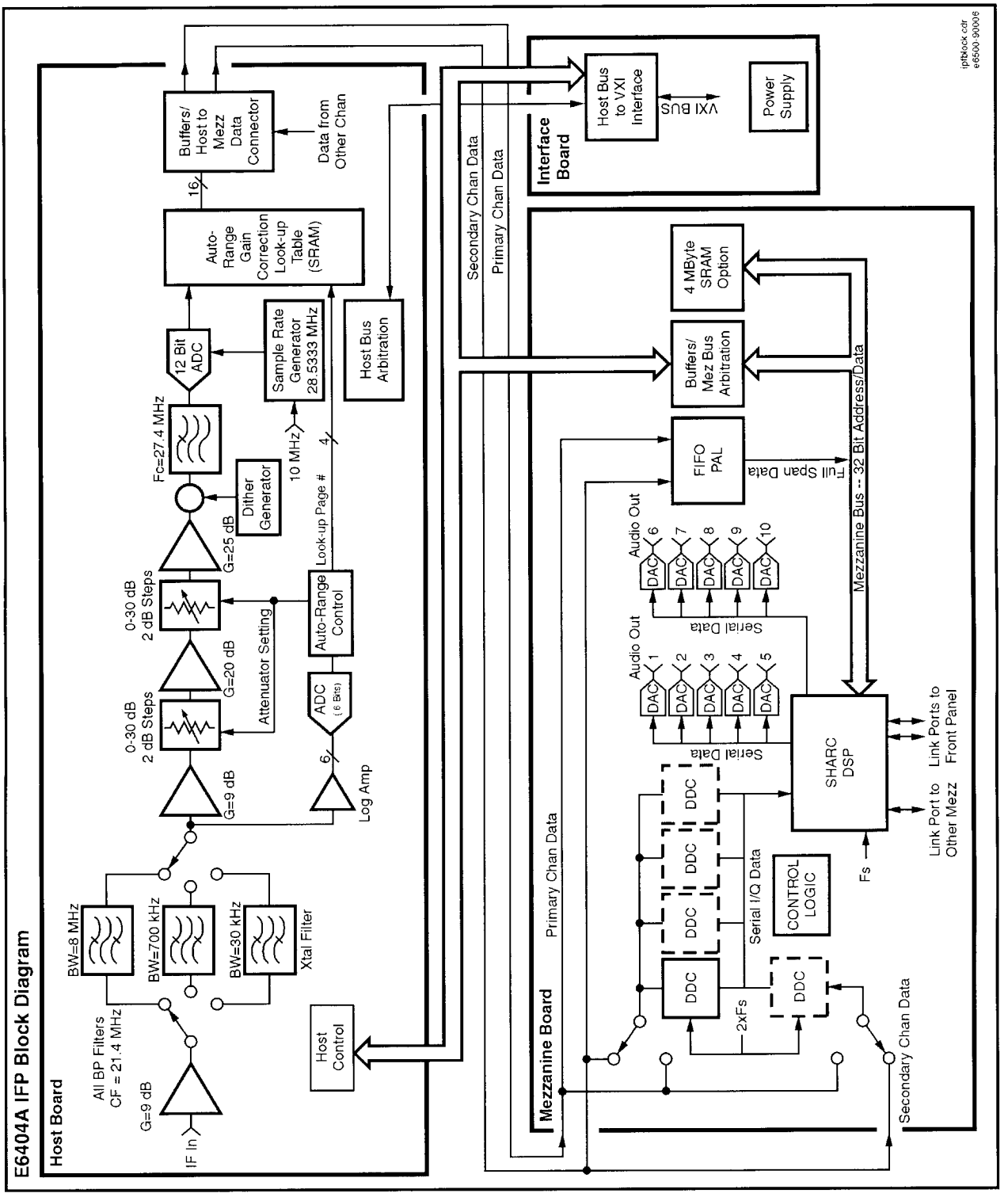


Figure 4-3 IF Processor Block Diagram (single channel, 1 mezzanine option)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

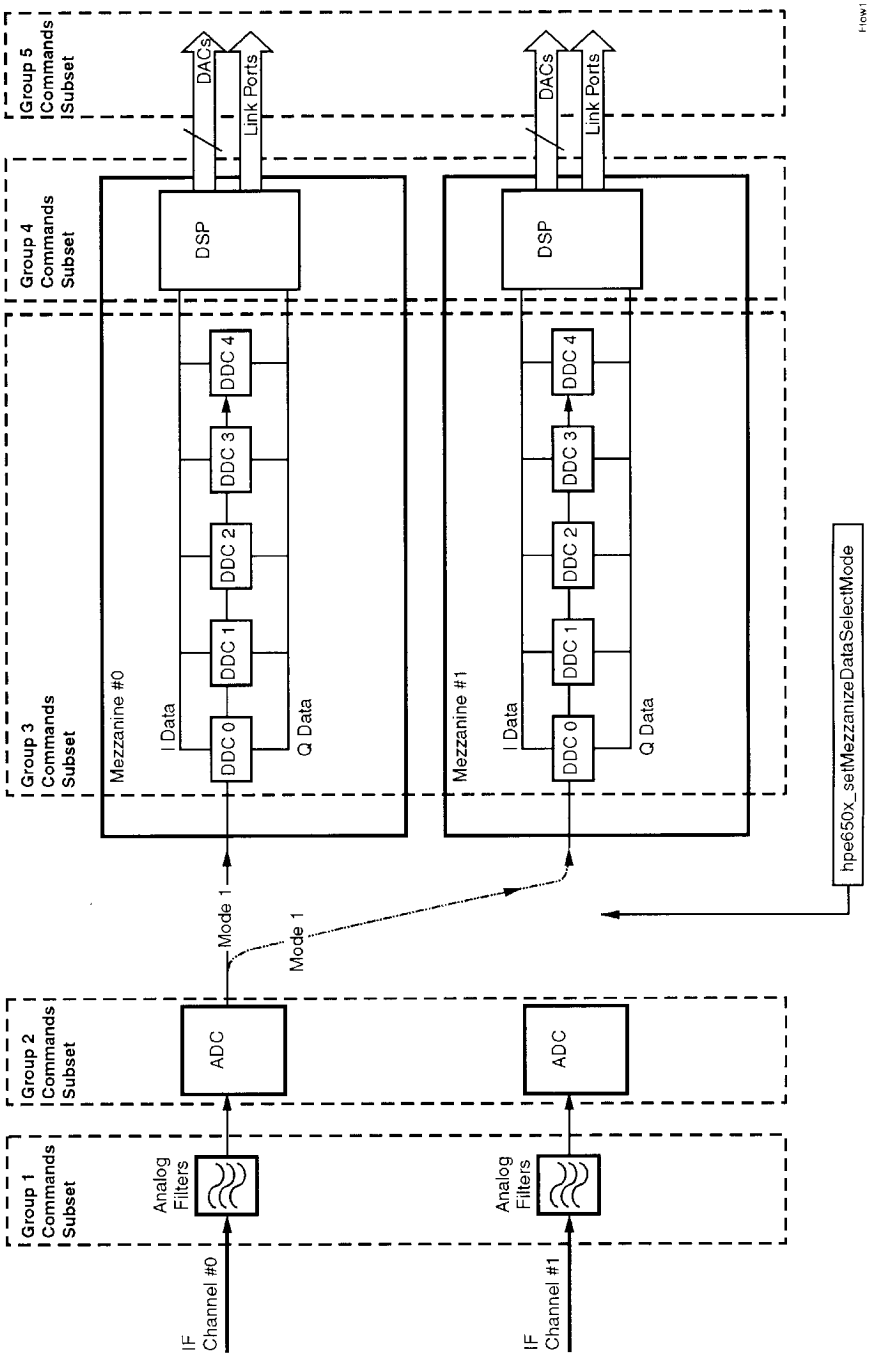


Figure 4-4 Mezzanine Data Select Mode 1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

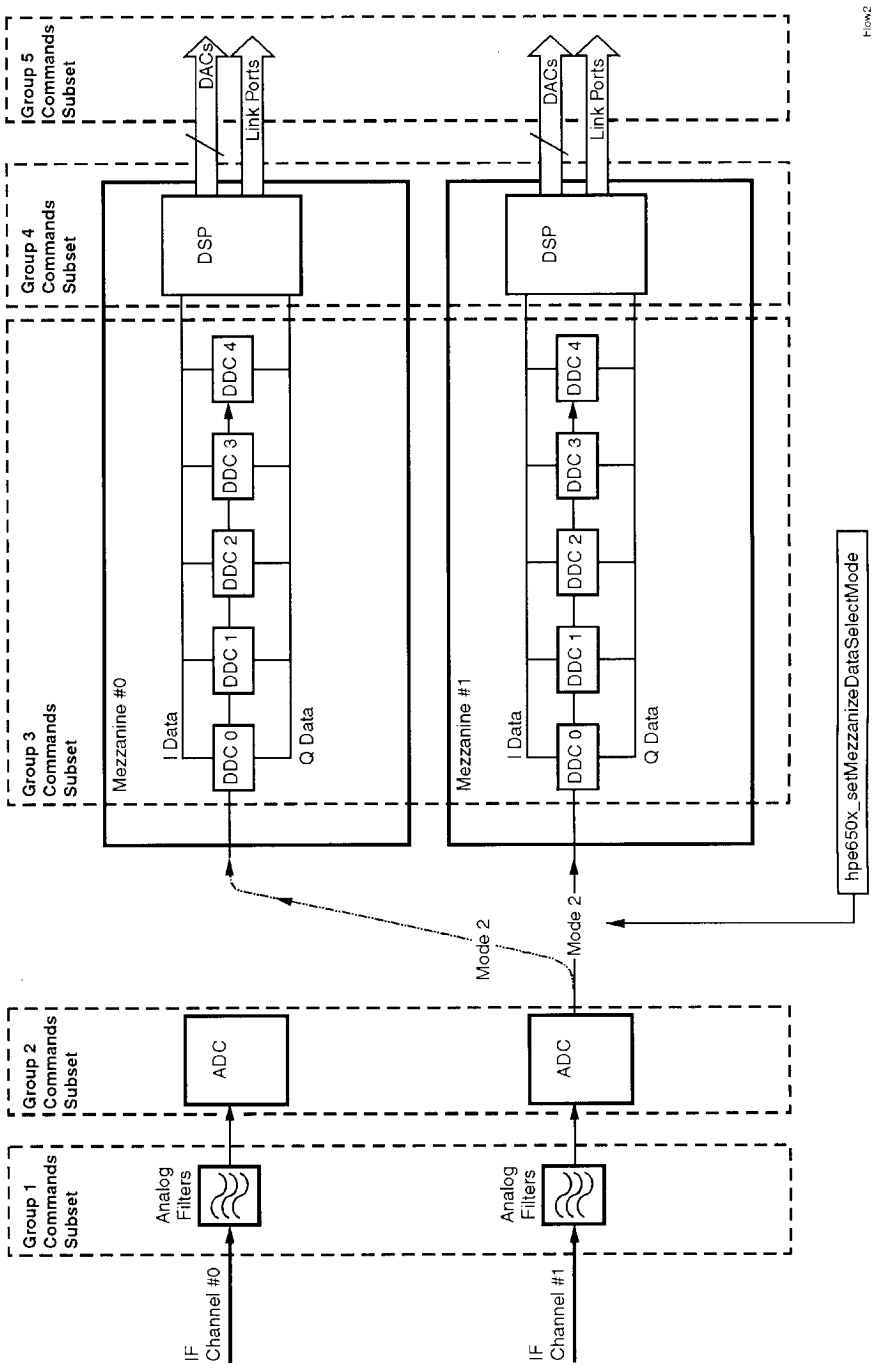


Figure 4-5 Mezzanine Data Select Mode 2

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

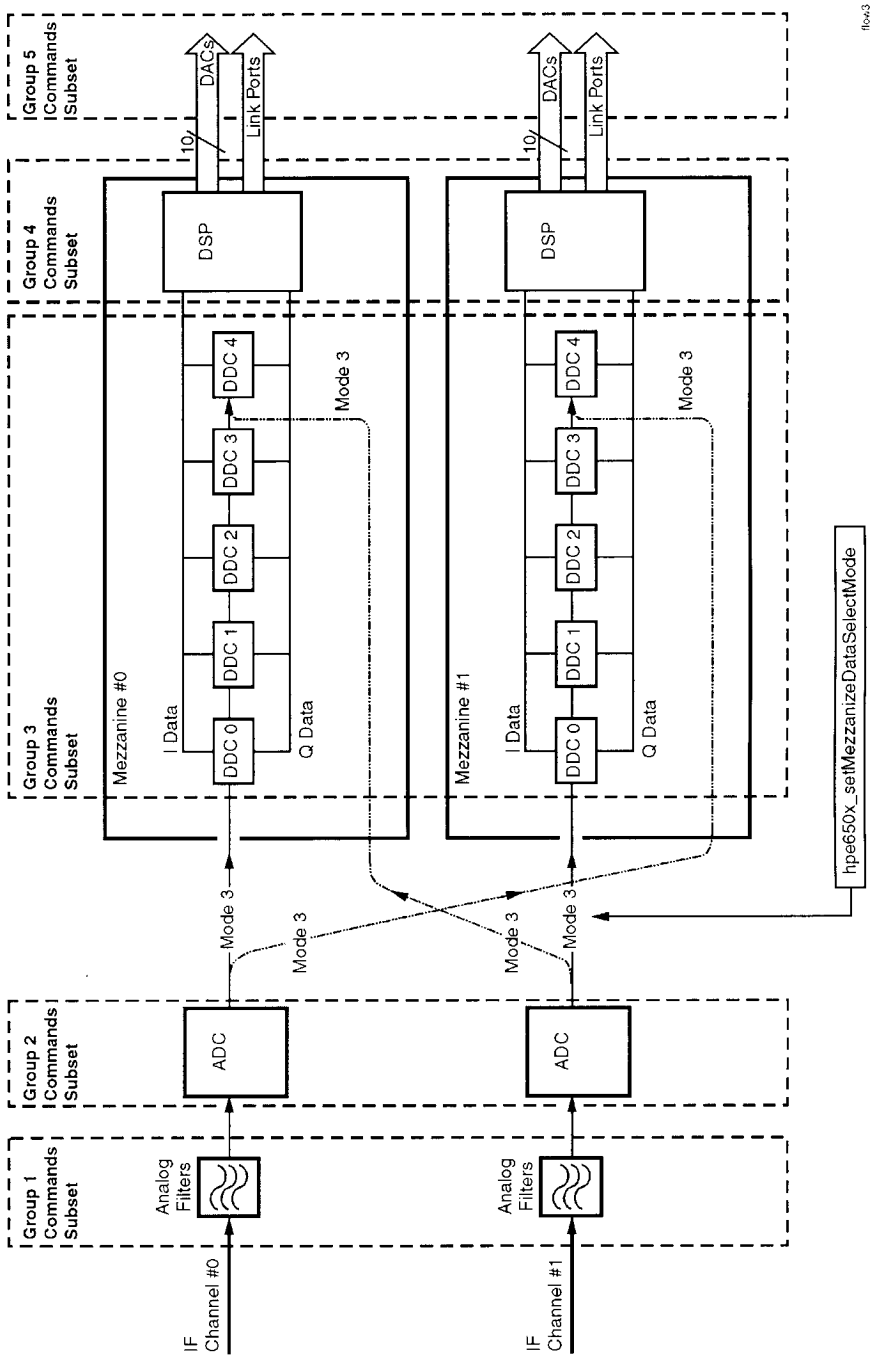


Figure 4-6 Mezzanine Data Select Mode 3

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

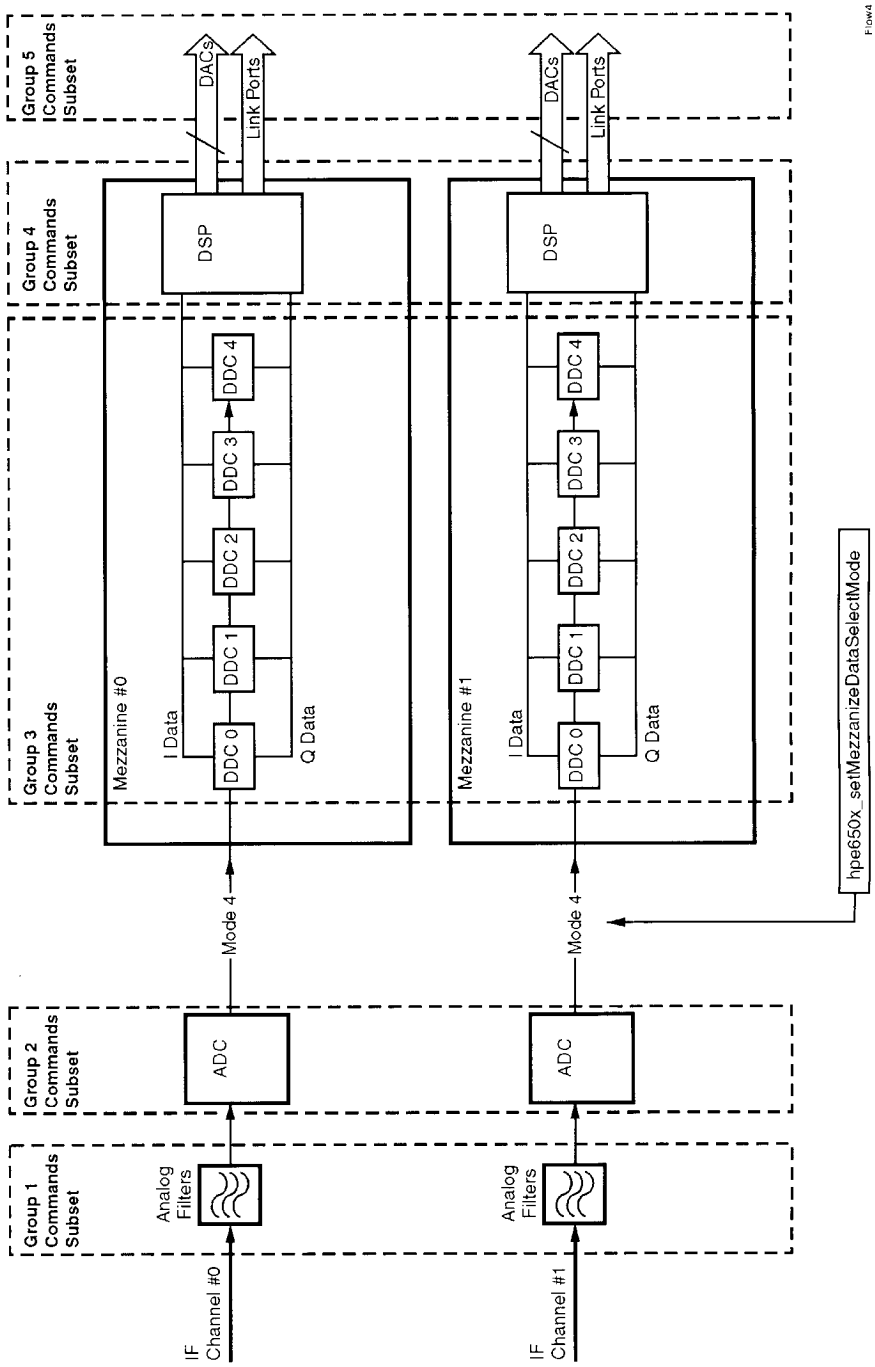


Figure 4-7 Mezzanine Data Select Mode 4

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5 Specifications

This chapter contains specifications, characteristics, and typical performance parameters for the E6501A, E6502A, and E6503A VXI receivers.

Values given are specifications unless labeled as characteristic or typical.

Specifications are not available at the individual E6400-Series VXI *module* level.

Definition of Terms

Specifications describe warranted performance over the temperature range of 0 °C to 55 °C after a 30-minute warmup from ambient conditions.

Characteristics describe product performance for parameters that are not subject to variation, not measurable, verifiable through functional pass or fail tests, or are not routinely measured. Characteristic performance parameters are non-warranted.

Typical refers to test data at the 50th percentile (averaged over the frequency range) and 25 °C. Typical indicates non-warranted performance parameters.

Note

Specifications are valid *only* if the modules are housed in an Agilent Technologies VXI mainframe. (E1421B and E1401B mainframes are recommended.) Performance is not specified below 20 MHz.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Frequency-Related Specifications

Frequency Range	20 MHz[1] to 1000 MHz 20 MHz[1] to 3000 MHz (Option 003)
Tuning Resolution	1 Hz
Synthesizer Tuning Speed (Characteristic)	1 ms (10 kHz settling) 2 ms (1 kHz settling) 4 ms (100 Hz settling) [Data derived from register-based programming]
Overall Receiver Sweep Speed (Characteristic)	2.8 GHz/s in 10.4 kHz resolution bandwidth (maximum speed using 4k FFTs of successive 8 MHz spans over the 20 MHz to 1000 MHz frequency range; does not account for computer overhead time)
Tuning Accuracy **(center frequency × reference accuracy)**	
Internal OCXO Reference Accuracy	1×10^{-6}/yr [1]
External Reference Input	Requires 10 MHz reference signal with level 0 dBm ±3 dB

1. The software driver supports tuning the receiver down to 2 MHz. However, specifications, typicals, and characteristics do not apply below 20 MHz.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RF Preselection	
Preselector Band	**Filter Frequency Range** (Characteristic)
1	(40 MHz LPF)
2	40 to 60 MHz
3	60 to 84 MHz
4	84 to 118 MHz
5	118 to 170 MHz
6	170 to 230 MHz
7	230 to 350 MHz
8	350 to 450 MHz
9	450 to 750 MHz
10	750 to 1000 MHz
11	1000 to 1250 MHz
12	1250 to 1800 MHz
13	1800 to 2400 MHz
14	2400 to 3000 MHz

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Amplitude-Related Specifications

Input Parameters	
RF Input Impedance (Characteristic)	50 ohms
RF Input Connector	SMA
Input VSWR (Characteristic)	2:1
Maximum Input without Damage (Characteristic)	
Average Continuous RF Power	+20 dBm
DC Voltage	20 volts
Maximum recommended operating level at RF input:	–20 dBm with 0 dB RF attenuation (+10 dBm with 30 dB attenuation)
RF Input Attenuation (Characteristic)	0 to 30 dB in 10 dB steps

Detection Modes	
DSP-based Demodulation	AM, LSB, USB, ISB, FM, CW, PM
Simultaneous Demods	up to 10 (optional configuration; see Table 5-3 for bandwidth requirements)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Dynamic Range Parameters	
Noise Figure (Typical)	10 dB (20 MHz to 1 GHz)[1] 14 dB (1 GHz to 3 GHz)
Sensitivity: (12 dB SINAD; 1 Vp-p audio output; modulation=1 kHz; analog filter = 30 kHz BW, FM de-emphasis on) (Typical)	See Table 5-1
Intermodulation: Second Order (Typical) SOI (Referenced to the RF input) (with 0 dB input attenuation) (0 dB RF attenuation)	 +67 dBm
Intermodulation: Third Order (Typical) TOI Narrowband Intermodulation Distortion for 2 signals at -20 dBm and 125 kHz spacing using the 8 MHz analog filter (with 0 dB input attenuation)	 +15 dBm (20 MHz spacing) (referenced to the RF input) 0 dB RF attenuation –63 dBc (-60 dBc for Option 001) (referenced to the RF input) 0 dB RF attenuation
Image Rejection	95 dB
IF Rejection	85 dB
Phase Noise @ 20 kHz Offset (Characteristic)	–100 dBc/Hz
Internally Generated Spurious (Typical)	< –100 dBm, equivalent input
LO Emissions	–110 dBm (in 20 to 1000 MHz RF band) –100 dBm (in 1000 to 3000 MHz RF band)
Blocking (Characteristic)	< 2dB[2]
Reciprocal Mixing (Characteristic)	< 3 dB[3]

1. Add 1 dB when using E6403A 20 to 3000 MHz input.
2. Attenuation of –100 dBm desired signal by unwanted –5 dBm signal at 250 kHz offset in 25 kHz DDC BW and 30 kHz analog BW.
3. S+N/N degradation of desired signal by unwanted signal 350 kHz offset and 70 dB higher in 20 kHz DDC BW and 30 kHz analog BW at RF-tuned frequency of 85 MHz.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 5-1 Typical Values

Frequency	AM Sensitivity (DDC BW = 5 kHz; 50% AM)	FM Sensitivity (DDC BW = 25 kHz; FM dev = 5 kHz)
500 MHz	–107 dBm	–112 dBm
1125 MHz	–105 dBm	–110 dBm
1500 MHz	–105 dBm	–110 dBm
2000 MHz	–105 dBm	–110 dBm
2700 MHz	–103 dBm	–108 dBm

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IF (Intermediate Frequency) Processing

IF Section: Analog Signal Conditioning and ADC	
Analog IF Input Filter Bandwidths (minimum) (Characteristic)	8 MHz, 700 kHz, and 30 kHz, (Filters centered at 21.4 MHz. The 8 MHz filter is an anti-alias filter. The 700 kHz and 30 kHz filters are adjacent channel rejection filters.)
Analog Gain Control	manual and autoranging
Input Range Settings (Characteristic)	–48 dBm to 0 dBm in 2 dB steps (overrange from +2 dBm to +12 dBm also provided)
Autoranging Gain Response Time (Characteristic)	< 1 µs attack: < 1 µs decay: < 2.9 ms
Overall Gain Control Range (Characteristic)	150 dB (includes RF attenuation, ADC dynamic range, and autoranging)[1]
Dual Input IF Channel Isolation (Characteristic)	≤–120 dBm. (For a signal at full scale at the ADC on channel 1, channel 2 at maximum gain will have an equivalent input signal level ≤–120 dBm, and vice versa. Applies to E6502A and E6503A dual channel receivers only.)
ADC Sampling Rate (Characteristic)	28.5333 MSa/s [2]
ADC Output Data Rates	Full rate ADC at 28.533 MSa/s using 2 link ports (16 bits wide; 57.0666 MBytes/sec data rate using 2 link ports. See IF Section for link port information. Spectral information is inverted for bands 11 and 12 for full rate ADC output.)

1. Actual adjustable range using the software driver is 90 dB (30 dB RF attenuation and 60 dB analog IF gain). Remaining 60 dB range is accomplished with the ADC dynamic range (S/N of > 62 dB at Nyquist sampling rate of 28.533 MSa/s).
2. Sample rate is calculated by multiplying 21.4 MHz by 4/3.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IF Section: IF Processor Dynamic Range Parameters	
Harmonic Distortion[1,2,3] (Characteristic)	–75 dBc or <–140 dBm[4]
Spurious Responses[1,2,5] (Characteristic)	–110 dBc or <–140 dBm[4]
Signal-to-Noise Ratio[6,7] (Characteristic)	62 dB
Internally Generated Spurious Responses[1,8,9] (Characteristic)	<–140 dBm[4]

1. 8 MHz bandwidth (undecimated ADC data).
2. Input signal equal to input range setting (see "Input Range Settings" on the previous page).
3. Includes aliased distortion components.
4. Referred to E6404A input.
5. Includes non-harmonically-related spurious, clock spurious, sidebands, etc.
6. 14.266 MHz Nyquist bandwidth.
7. For input range setting of –10 dBm and signal at ADC clipping level.
8. E6404A input terminated in 50Ω.
9. With input range setting of –48 dBm.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IF Section: Digital (General)	
Number of Digital Downconverters (DDC)	Standard E6501A receiver: 1 Standard E6502A receiver: 2 (1 per mezzanine) Standard E6503A receiver: 2 (on mezzanine #1) Optionally: 5 per mezzanine (10 total per IF Processor module with two mezzanines; when DSP on mezzanine #1 used for search, only the DDCs on mezzanine #2 are usable for digital I/Q output or demodulation)
DDC Tuning Range (Characteristic)	8 MHz, max (limited to analog IF input filter) centered at 21.4 MHz
DDC Resolution (Characteristic)	16 bits DDC full scale automatically adjusted over 54 dB range using dynamic range optimization.
Dynamic Range Optimization Response Time (Characteristic)	Variable, adjustable in 500 µs steps attack: 500 µs to 1 s decay: 500 µs to 1 s
Digital IF Bandwidths (Characteristics)	36 different filters with following 3 dB BWs: 247 Hz, 493 Hz, 740 Hz, 1 kHz, 2.4 kHz, 3.3 kHz, 5 kHz, 6.3 kHz, 10 kHz, 12 kHz, 15 kHz, 20 kHz, 25 kHz, 30 kHz, 34 kHz, 44 kHz, 54 kHz, 62 kHz, 74 kHz, 83 kHz, 93 kHz, 109 kHz, 119 kHz, 138 kHz, 154 kHz, 167 kHz, 187 kHz, 201 kHz, 218 kHz, 238 kHz, 262 kHz, 291 kHz, 327 kHz, 374 kHz, 436 kHz, 462 kHz. (all 5 DDCs per mezzanine are set to the same BW)
Digital Bandwidth Shape Factor (Characteristic)	< 1.5: 1 (102 dB to 3 dB BW ratio)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Digital Output Interface	Analog devices 2106X DSP (Sharc) link ports (2 link ports per mezzanine)
Link Port Connector Type	AMP 1-104074-0 connector on E6404A front panel (mating connector: AMP 487550-5 housing and contacts)
Number of Digital I/Q Outputs	1 per DDC (up to 10 DDCs available optionally) (The number of DDCs depends on model and option configuration.)

Bandwidth of Digital I/Q Outputs	From 247 Hz to 462 kHz (The BW depends on the number of DDCs simultaneously accessed. Refer to Table 5-2)
DDC Decimated Sample Rate (samples/second) (Characteristic)	1.8181 x DDC BW
4 MByte Data RAM	Optional (4 MBytes per mezzanine; 2 mezzanines total); used for data buffering or delay memory applications (not available for programming)

IF Section: Digital I/Q Outputs	
Number of Simultaneous Digital I/Q Outputs per Mezzanine (Characteristic)	Up to 5 (See Table 5-2)[1] (up to 10 using optional second mezzanine)
Output Bandwidth	Set by DDC
Output Interface	Sharc link ports
Link Port Output Data Rate (bytes/second) (Characteristic)	1.8181 x 4 x DDC BW I/Q complex data is 16 bits wide (16 bits for I and 16 bits for Q; 4 bytes per sample) (see "Digital IF Bandwidths" for DDC BW range)

1. The standard receiver includes one or two DDCs depending on the specific model number. Options are required for two or five DDCs per mezzanine. The E6501A and 6503A require an option for a second mezzanine.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 5-2 Number of Simultaneous I/Q Output

DDC Bandwidth	.247 to 34 kHz	44 to 54 kHz	62 to 83 kHz	93 to 187 kHz	201 to 462 kHz
Number of Simultaneous I/Q Outputs per Mezzanine	5	4	3	2	1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IF Section: Demodulated Analog Audio Outputs	
DSP-based Detection Modes (demodulation)	AM, LSB, USB, ISB,[1] FM, CW, PM
Number of Simultaneous Demodulated Signals per Mezzanine (Characteristic)	Up to 5 signals[2] (See Table 5-3) (up to a total of 10 using optional second mezzanine)
Maximum Realtime Demodulated Bandwidth[3] (single channel per mezzanine)	AM: 374 kHz FM: 462 kHz LSB/USB: 167 kHz ISB: 138 kHz CW: 462 kHz PM: 138 kHz
Analog Audio Output Bandwidth (Characteristic)	15 kHz, maximum
COR (carrier operated relay)	Use TTL trigger output signal (no traditional COR dry contacts)
Squelch Range (Characteristic)	–125 dBm to -20 dBm
ALC (automatic level control) Range (Characteristic)	Adjustable; > 100 dB (volume control; maintains audio level to within 25% full scale; used for USB, LSB, and ISB)
ALC Response Time (Characteristic)	Adjustable in 1 µs steps; range: 1 µs to 10 s
AFC (automatic frequency control) Tracking Range (Characteristic)	±1/2 DDC BW
FM De-emphasis (Characteristic)	75 µs

1. Independent sideband (ISB) is supported. However, ISB signal channel requires two DDC channels.
2. The standard receiver includes one or two DDCs depending on the specific model number. Options are required for two or five DDCs per mezzanine. The E6501A and E6503A require an option for a second mezzanine.
3. Maximum modulation frequency is dependent on modulation format. For example, the maximum modulation frequency for AM is half the bandwidth, so the maximum modulation frequency is 187 kHz.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 5-3 Number of Simultaneous Channels

AM **DDC Bandwidth (kHz)**	5 .247 - 30	4 34 - 54	3 62 - 83	2 93 - 167	1 187 - 374
FM **DDC Bandwidth (kHz)**	5 .247 - 30	4 34 - 54	3 62 - 83	2 93 - 187	1 201 - 462
USB/LSB[1] **DDC Bandwidth (kHz)**	5 .247 - 25	4 30 - 34	3 44	2 54 - 74	1 83 - 167
CW **DDC Bandwidth (kHz)**	5 .247 - 30	4 34 - 54	3 62 - 83	2 93 - 187	1 201 - 462
PM **DDC Bandwidth (kHz)**	5 .247 - 25	4 30	3 34	2 44 - 62	1 74 - 138

1. Independent sideband (ISB) is supported. However, each signal channel requires two DDC channels.

Maximum Audio Output (Characteristic)	1 volt RMS into 600 ohms
Audio Output Connector Type	AMP 750823-1 on front panel of E6404A IF Processor module for connection to separately ordered cable and audio breakout box (or order AMP 750833-1 cable connector and 750850-3 backshell kit to configure your own cable)
External Cable and Audio Breakout Box	Ordered separately as E3245A (includes 10 mini-phone plugs for connection to headphones or amplified speakers)
Trigger Input	TTL level (uses 2 pins on AMP 750823-1 audio connector); used to synchronize data. Refer to Table 3-3.
Trigger Output	TTL level (uses 2 pins); controllable from software driver. Refer to Table 3-3.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Physical Characteristics

Front Panel Connectors	
E6401A VXI Module (20 to 1000 MHz downconverter)	20 - 1000 MHz Input, SMA block downconverter input, SMA 1st LO input, SMC 2nd LO input, SMC 21.4 MHz IF output, SMB
E6402A VXI Module (local oscillator)	1st LO output, SMC 2nd LO output, SMC block downconverter LO output, SMC 3rd LO output, SMB reference output, SMB external reference input, SMB reference TTL output, SMB (The E6503A dual channel receiver includes the E6402A Option 002 LO module which adds a second set of the following outputs: 1st LO output, 2nd LO output, and block downconverter LO output.)
E6402A Option 002 Module (dual LO output)	Ext Reference in, SMB Reference TTL Out, SMB Reference out, SMB BD LO output, SMC (2) 1st LO output, SMC (2) 2nd LO output, SMC (2) 3rd LO output, SMB
E6403A VXI Module (1000 - 3000 MHz block downconverter)	20 to 3000 MHz input, SMA 20 to 1000 MHz output, SMA block downconverter output, SMA block downconverter LO input, SMC
E6404A VXI Module (IF processor)	Ch1 IF input, SMB Ch2 IF input, SMB (option required) reference input, SMB reference output, SMB audio output/trigger input (on mezzanine 1) audio output/trigger input (on optional mezzanine 2) link port 1/2 output (on mezzanine 1) link port 1/2 output (on optional mezzanine 2)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Weight (Characteristic)[1]	
E6501A (20 - 1000 MHz) Receiver	8.6 kg (18 lbs, 11 oz)
E6501A Option 003 (20 - 3000 MHz) Receiver	11.3 kg (24 lbs, 9 oz)
E6502A (20 - 1000 MHz) Receiver	14.4 kg (31 lbs, 5 oz)
E6502A Option 003 (20 - 3000 MHz) Receiver	19.8 kg (43 lbs, 1 oz)
E6503A (20 - 1000 MHz) Receiver	11.3 kg (24 lbs, 9 oz)
E6503A Option 003 (20 - 3000 MHz) Receiver	16.7 kg (36 lbs, 5 oz)
E6401A module (20 - 1000 MHz downconverter)	2.7 kg (5 lbs, 14 oz)
E6402A module (local oscillator)	3.1 kg (6 lbs, 12 oz)
E6403A module (1000 - 3000 MHz block downconverter)	2.7 kg (5 lbs, 14 oz)
E6404A module (IF processor)	2.8 kg (6 lbs, 1 oz)

1. E6404A IF processor weight includes complete set of options.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

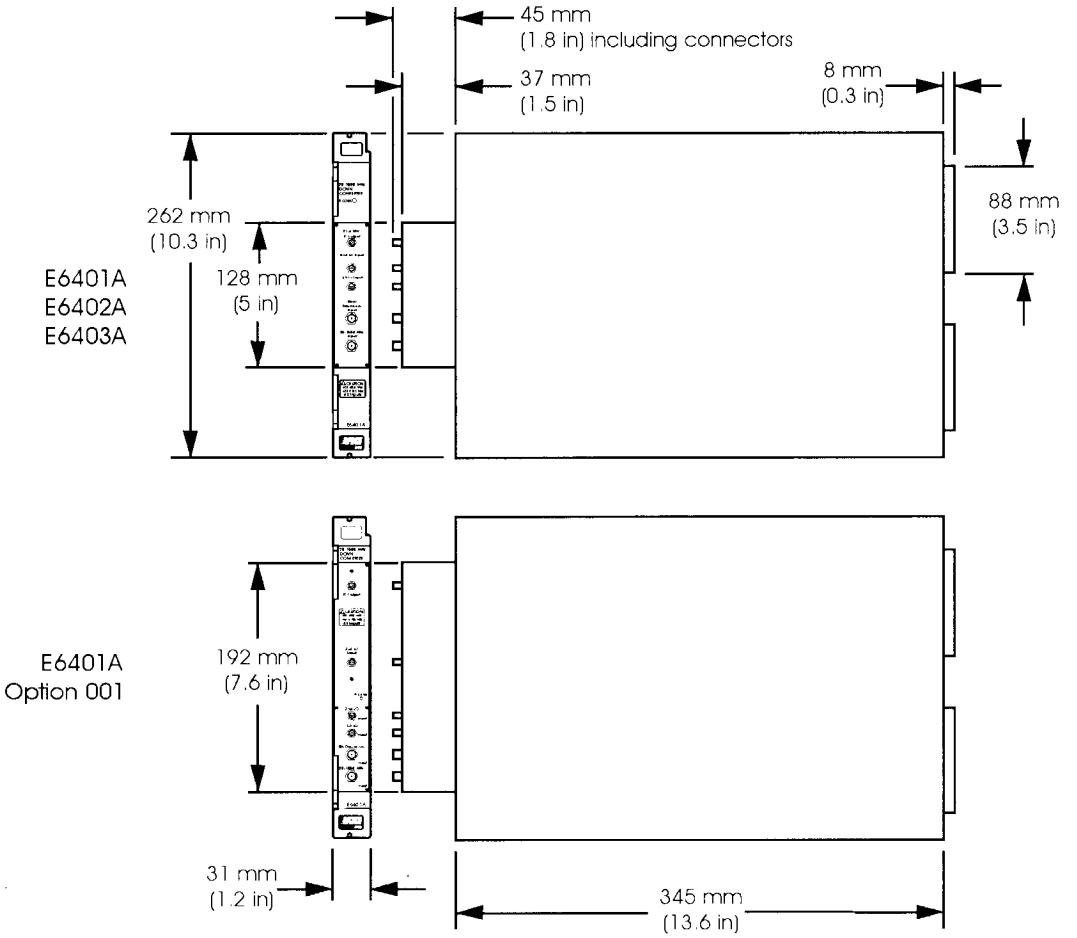


Figure 5-1 Dimensions of the E640XA Modules Comprising the E650XA VXI Receivers, (Front-panel connectors vary with model and option.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

General Information

Power Requirements[1] (Characteristic)	
E6501A (20 – 1000 MHz)	90 watts
E6501A Option 003 (20 – 3000 MHz)	104 watts
E6502A (20 – 1000 MHz)	126 watts
E6502A Option 003 (20 – 3000 MHz)	154 watts
E6503A (20 – 1000 MHz)	104 watts
E6503A Option 003 (20 – 3000 MHz)	132 watts

1. Power requirements based on the E6404A IF processor with two IF channels, two mezzanines, ten DDCs, two DSPs, and two 4 Mbyte RAMs.

Power Requirements
(characteristic)

Module	+5 Vdc	–5.2 Vdc	+24 Vdc	–24 Vdc	–2 Vdc	–12 Vdc
E6401A						
DC Current	0.48 A	0.34 A	0.58 A	0.022 A	NA	NA
Dynamic Current	0.10 A	0.07 A	0.10 A	0.00 A	NA	NA
E6401A Option 001						
DC Current	0.74 A	0.39 A	0.58 A	0.021 A	NA	NA
Dynamic Current	0.16 A	0.09 A	0.10 A	0.00 A	NA	NA

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Power Requirements
(characteristic)

Module	+5 Vdc	–5.2 Vdc	+24 Vdc	–24 Vdc	–2 Vdc	–12 Vdc
E6402A						
DC Current	0.75 A	0.012 A	1.035 A	0.124 A	NA	NA
Dynamic Current	0.17 A	0.00 A	0.19 A	0.02 A	NA	NA
E6403A						
DC Current	0.450 A	0.341 A	0.361 A	0.161 A	NA	NA
Dynamic Current	0.21 A	0.08 A	0.06 A	0.03 A	NA	NA
E6404A						
DC Current	8 A	3 A	0.3 A	NA	0.1 A	0.2 A
Dynamic Current	1 A	0.3 A	0.03 A	NA	0.01 A	0.02 A

Calibration and Adjustment	
Calibration Interval	2 years
Internal Timebase Adjustment Interval	1 year

Warranty
3 years

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Environmental Information

Operating Temperature	0 °C to 55 °C
Storage Temperature	-20 °C to +70 °C
EMC	CISPR 11 Class A; MIL-STD-461C RE02 Parts 5 and 7; IEC 801-3; HP ETM 765 RS2; IEC 1000-4-6; VXI-EDG B.8.6.3 and B.8.6.4; IEC 801-2 and HP ETM 765.
Humidity	HP ETM 758 Class A2, B1, B2 (40 °C, 95% RH.)
Shock	MIL-T-28800E; HP ETM 760
Vibration	MIL-T-28800E Class 3; HP ETM 759 Class B2; HP ETM 762

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

VXI Information

VXI Control	VXI plug-and-play driver for Windows® NT O/S; National Instruments MXI-2 interface required for NT O/S
Module Size	VXI C-size modules
Slots Used	3 slots (E6501A 1 GHz Rx) 4 slots (E6501A Option 003 3 GHz Rx) 5 slots (E6502A 1 GHz Rx) 7 slots (E6502A Option 003 3 GHz Rx) 4 slots (E6503A 1 GHz Rx) 6 slots (E6503A Option 003 3 GHz Rx)
VXI Interface	Requires MXI interface (not included)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Regulatory Information

The information on the following page applies to the E6501A, E6502A, and E6503A VXI receivers, and all options of these products.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DECLARATION OF CONFORMITY

according to ISO/IEC Guide 22 and EN 45014

Manufacturer's Name:	Hewlett-Packard Co.
Manufacturer's Address:	Santa Rosa Systems Division 1400 Fountaingrove Parkway Santa Rosa, CA 95403-1799 USA

declares that the products

Product Names:	VXI Receivers and IF Processor Module
Model Numbers:	HP E6501A, HP E6502A, HP E6503A, HP E6404A
Product Options:	This declaration covers all options of the above products.

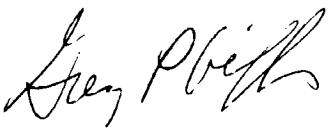
conform to the following Product specifications:

Safety: IEC 1010-1:1990+A1 / EN 61010-1:1993
CAN/CSA-C22.2 No. 1010.1-92

EMC: CISPR 11:1990/EN 55011:1991 Group 1, Class A
IEC 801-2:1984/EN 50082-1:1992 4 kV CD, 8 kV AD
IEC 801-3:1984/EN 50082-1:1992 3 V/m, 27-500 MHz
IEC 801-4:1988/EN 50082-1:1992 0.5 kV Sig. Lines, 1 kV Power Lines

Supplementary Information:
The products herewith comply with the requirements of the Low Voltage Directive 73/23/EEC and the EMC Directive 89/336/EEC and carry the CE-marking accordingly.

The HP E6501A, HP E6502A and HP E6503A VXI Receivers consist of various combinations of HP E6401A, HP E6402A, HP E6403A and HP E6404A modules. The systems were tested in HP E1401B mainframes.



Santa Rosa, CA, USA 1 May 1998

Greg Pfeiffer/Quality Engineering Manager

European Contact: Your local Hewlett-Packard Sales and Service Office or Hewlett-Packard GmbH, Department HQ-TRE, Herrenberger Strasse 130, D-71034 Böblingen, Germany (FAX +49-7031-14-3143)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6 Programming Command Reference

In This Chapter

- Overview
- Return Values
- Command Lists

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Overview

The driver software provided with the E650XA receiver allows the user to control the E650XA hardware from their custom software. This driver software is designed to meet the VXI plug & play standards set forth by the VXI Systems Alliance.

The driver software performs the following functions:

- Maps hardware to software.
- Passes data to and from the user's program.
- Passes task requests to the DSP software.
- Establishes tuner control mechanism.
- Establishes hardware signal paths.

Figure 6-1 shows a model of the software/hardware (hierarchy) structure.

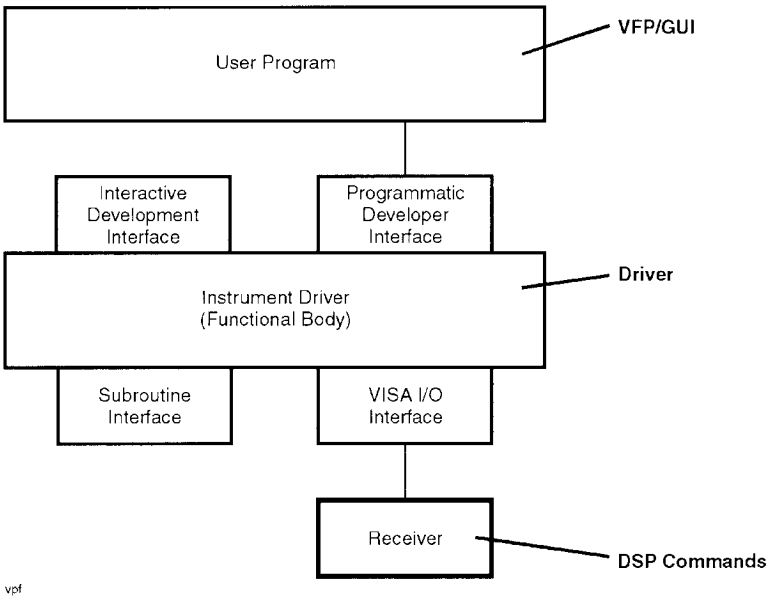


Figure 6-1 Model Applied to the E650X Receiver

Note that the VISA software library must be used with the E650X receiver regardless of the platform. There are two advantages with using the VISA library: single I/O library to communicate with most instrument interfaces, and it is a standard library that all plug & play drivers must use. The library is installed by default when the MXI controller card is installed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Driver Architecture

Figure 6-2 shows the architecture of the driver software.

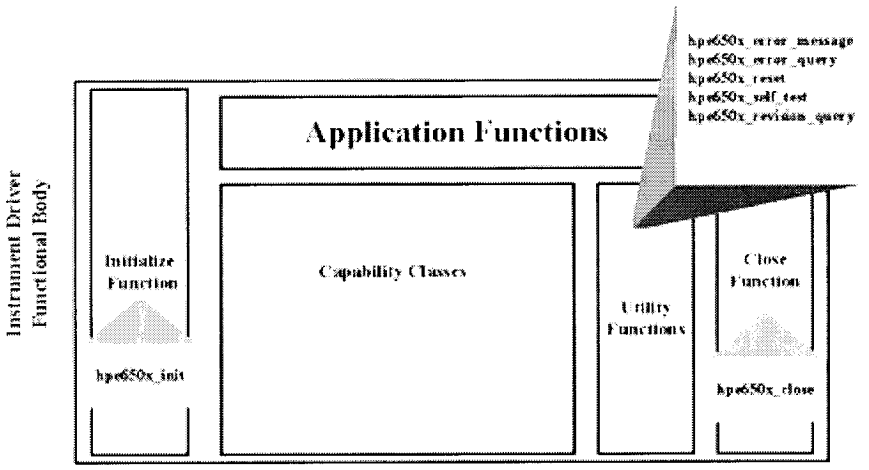


Figure 6-2 Driver Architecture

Capability Classes

The capability class is a collection of intermediate-level functions that allow developers and integrators greater access to instrument capabilities.

The capability classes are divided into the following files:

- Common.c
- Monitoring.c
- Search.c
- FFT.c

Capability class functions group the instrument driver functions according to the instrument's capabilities such as *measure*, *source*, *route*, etc. In the case of the E650XA receivers, only the measure and route class functions are applicable. At the highest level within a capability class, each function executes a complete action. No prior instrument state is assumed. This allows these functions to be order independent. Execution of a capability class function produces a finished result such as a returned value of a measurement or a connected route by a switch.

For example, the measure class is divided into *configure* and *read* function classes. Configure functions configure the device for a particular measurement but do not execute the measurement. Read functions initiate the measurement and read the result from the instrument.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Measure Capability Class Functions

Functions in the measure class configure a device for a particular measurement, initiate that measurement, and read the result in a single operation. Typically, these functions include multiple parameters and require no interaction with other instrument driver operations. The measure class functions consist of the following structure:

- Configure
- Read Function
  - Initiate
  - Fetch

The measure class functions are further divided into real-time demodulation or stepped-tuner mode. Driver software commands that control DDCs, capture data, or control FFT measurements are examples of measure capability class functions and can also be referred to as functions that control the mezzanine.

Configure and read functions of the measure class provide a highly abstract control interface to the capabilities of a measurement device. These functions correspond to the measure functions except they separate the functionality of configuration from the functionality of reading measured values from the instrument.

Configure functions configure an instrument for a particular measurement but do not initiate that measurement. No finished result is provided. Read functions complete the measurement function action by initiating the measurement and providing the result. Read functions have a specific sequence dependency with their complimentary configure function.

The read functions depend on the instrument state produced by the configure functions. However, because the configure and read functions are separated, there is an opportunity to send additional low level commands which modify the instrument state prior to calling the read function. This allows the default action of a measurement function to be modified, providing more control over the test execution. For example, the application program may choose to defer measurement execution until all of the instrument modules have been configured.

Read functions may be further divided into initiate and fetch functions. These functions separate the initiation of a measurement from the data retrieval. This gives the user finer control of the measurement operation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Route Class Functions

The route class functions consist of the following type of functions:

- Configure
- Initiate

The route class functions set signal path conditions or control signal routing through the receiver. Driver software commands having a parameter to control IF channel routing (IFchan), hpe650x_**setAnalogFilter** and hpe650x_**setIFGain**, are examples of route class functions and can also be referred to as receiver functions.

Application Functions

The application functions are a collection of high-level functions that perform complete tests and measurement operations.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Return Values

The E650X receiver functions return a defined value depending on the outcome of the process. For example, if the function was successful, 0 is returned. If the function failed, an error in the range of 0xFEC0 0000 to 0xFEC0 FFFF is returned. In other words, a zero corresponds to the successful execution of the function, while any negative value corresponds to an error condition.

Warnings have values greater than zero. A warning indicates a non-fatal problem (function succeeded) during the execution of the function process.

*Table 6-1 **Return Values***

Defined Symbol (defined in the hpe650x.h file)	Hexadecimal Value
VI_SUCCESS	0x0
VI_WARN_NSUP_ID_QUERY	0x3FFC 0101
VI_WARN_NSUP_RESET	0x3FFC 0102
VI_WARN_NSUP_SELF_TEST	0x3FFC 0103
VI_WARN_NSUP_ERROR_QUERY	0x3FFC 0104
VI_WARN_NSUP_REV_QUERY	0x3FFC 0105
VI_ERROR_PARAMETER1	(refer to the vpptype.h file)
VI_ERROR_PARAMETER2	(refer to the vpptype.h file)
VI_ERROR_PARAMETER3	(refer to the vpptype.h file)
VI_ERROR_PARAMETER4	(refer to the vpptype.h file)
VI_ERROR_PARAMETER5	(refer to the vpptype.h file)
VI_ERROR_PARAMETER6	(refer to the vpptype.h file)
VI_ERROR_PARAMETER7	(refer to the vpptype.h file)
VI_ERROR_PARAMETER8	(refer to the vpptype.h file)
VI_ERROR_FAIL_ID_QUERY	(refer to the vpptype.h file)
VI_ERROR_INV_RESPONSE	(refer to the vpptype.h file)
DEFAULT_IF_CONFIG_USED	0x0000 0001
MORE_DATA_REMAINING	0x0000 0002
SEARCH_ALREADY_STARTED	0x0000 0003
SEARCH_ALREADY_STOPPED	0x0000 0004
FREQUENCY_INDEX_OUTSIDE_TRACE	0x0000 0005

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Defined Symbol (defined in the hpe650x.h file)	Hexadecimal Value
WARN_NO_NEW_DATA	0x0000 0006
MAX_NUM_OF_IF_MODULES_REACHED	0xFEC0 0001
CANNOT_OPEN_DEFAULT_SESSION	0xFEC0 0002
CANNOT_OPEN_VI_SESSION	0xFEC0 0003
INVALID_INSTRUMENT_ID	0xFEC0 0004
INVALID_OPTION_STRING	0xFEC0 0005
INVALID_INDEX_NUMBER	0xFEC0 0006
TUNER_NOT_ATTACHED	0xFEC0 0007
INVALID_IF_CHANNEL	0xFEC0 0008
INVALID_MEZZANINE_NUMBER	0xFEC0 000A
INVALID_DAC_FILTER_NUMBER	0xFEC0 000B
INVALID_MODE_NUMBER	0xFEC0 000C
INVALID_RF_FREQUENCY	0xFEC0 000D
INVALID_LOGICAL_ADDRESS	0xFEC0 000E
TUNER_NOT_CONTROLLED	0xFEC0 000F
IMPROPER_MEZZANINE_MODE	0xFEC0 0010
INVALID_BANDWIDTH_VALUE	0xFEC0 0011
NOT_VALID_IN_MONITOR_MODE	0xFEC0 0012
INVALID_DDC_NUMBER	0xFEC0 0013
INVALID_GAIN_VALUE	0xFEC0 0014
INVALID_ATTENUATION_SETTING	0xFEC0 0015
INVALID_SOURCE_VALUE	0xFEC0 0016
INVALID_DIGITAL_BANDWIDTH	0xFEC0 0017
INVALID_DDC_FREQUENCY	0xFEC0 0018
INVALID_DAC_NUMBER	0xFEC0 0019
INVALID_ALC_SELECT_VALUE	0xFEC0 001A
INVALID_PROCESSS_NUM	0xFEC0 001B
CANNOT_ADJUST_IN_TIME_MODE	0xFEC0 001C
INVALID_FFT_LENGTH	0xFEC0 001D
MODULE_IS_NOT_HEWLETT_PACKARD	0xFEC0 001E
CANNOT_OPEN_LOADER_FILE	0xFEC0 001F
MODULE_IS_NOT_E6401	0xFEC0 0021
MODULE_IS_NOT_E6402	0xFEC0 0022

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Defined Symbol (defined in the hpe650x.h file)	Hexadecimal Value
MODULE_IS_NOT_E6403	0xFEC0 0023
MODULE_IS_NOT_E6404	0xFEC0 0024
UNABLE_TO_WRITE_TO_DSP	0xFEC0 0026
IF_MODULE_FAILED_TO_RESPOND	0xFEC0 0027
CANNOT_WRITE_TO_IF_MODULE	0xFEC0 0028
INVALID_AUDIO_VALUE	0xFEC0 0029
INVALID_DEMOD_VALUE	0xFEC0 002A
RSSI_DATA_NOT_YET_AVAILABLE	0xFEC0 002B
INALID_PROCESS_NUMBER	0xFEC0 002C
INVALID_SCALE_TYPE	0xFEC0 002D
FAILED_RAM_TEST	0xFEC0 002E
RAM_TEST_TIMED_OUT	0xFEC0 002F
INVALID_WINDOW_TYPE	0xFEC0 0030
FFT_NOT_A_POWER_OF_TWO	0xFEC0 0031
CANNOT_BEGIN_FFT_PROCESS	0xFEC0 0032
INALID_SETUP_PROCEDURE	0xFEC0 0033
SANITY_CHECK_FAILED	0xFEC0 0034
UNABLE_TO_READ_ATTENUATOR	0xFEC0 0035
AUTORANGE_NOT_ACTIVE	0xFEC0 0036
INVALID_SETTLING_TIME	0xFEC0 0037
INVALID_GAIN_SETTING	0xFEC0 0038
INVALID_CAPTURE_FORMAT	0xFEC0 0039
INVALID_CAPTURE_DESTINATION	0xFEC0 003A
FAILED_LINK_PORT_TEST	0xFEC0 003B
CAPTURE_PROCESS_RUNNING	0xFEC0 003C
INVALID_OUTPUT_DEVICE	0xFEC0 003D
AUDIO_CHANNEL_INACTIVE	0xFEC0 003E
ERROR_IN_CAPTURE_CONFIGURATION	0xFEC0 003F
FAILED_TO_ARM_DDC	0xFEC0 0041
NO_DDC_SPECIFIED_TO_CAPTURE	0xFEC0 0042
NO_CAPTURE_PROCESS_RUNNING	0xFEC0 0043
INVALID_TRACE_LENGTH	0xFEC0 0044
INVALID_SEARCH_TYPE	0xFEC0 0045

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Defined Symbol (defined in the hpe650x.h file)	Hexadecimal Value
INVALID_ALC_TIME_PARAMETER	0xFEC0 0046
INVALID_RSSI_TIME_PARAMETER	0xFEC0 0047
CAPTURE_PROCESS_FAILED	0xFEC0 0048
NO_CAPTURE_DATA_FOR_DDC	0xFEC0 004A
NO_ADC_DATA_CAPTURED	0xFEC0 004B
NO_ABSOLUTE_AMPLITUDE_OPTION	0xFEC0 004C
RSSI_NOT_ACTIVE	0xFEC0 004D
FFT_PROCESS_NOT_ACTIVE	0xFEC0 004E
INVALID_AVERAGE_VALUE	0xFEC0 004F
INVALID_TRIGGER_ACTION	0xFEC0 0050
FAILED_TO_ARM_DDCs	0xFEC0 0051
FAILED_TO_ARM_DSP	0xFEC0 0052
SRAM_NOT_INSTALLED	0xFEC0 0053
INVALID_NUMBER_OF_SAMPLES	0xFEC0 0054
INVALID_SCALE_SOURCE	0xFEC0 0055
CANNOT_GET_COR_VALUE	0xFEC0 0056
SEARCH_NOT_ACTIVE	0xFEC0 0057
INVALID_GAIN_NUMBER	0xFEC0 0058

Special Values

FFT and search functions that return trace data will return a value of -1 when the DSP does not have data ready for the user's software.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command Lists

Commands in this chapter are grouped by file source (capability class):

- VXI Plug and Play (from the file *common.c*)
- From the file *common.c*
- From the file *search.c*
- From the file *monitor.c*
- From the file *FFT.c*

The following commands must be called in every session and must be in the following sequence:

ViStatus_VI_FUNC hpe650x_init();
ViStatus_VI_FUNC hpe650x_initIFChannel();
ViStatus_VI_FUNC hpe650x_setMonitoringMode();

or

ViStatus_VI_FUNC hpe650x_setSearchMode();

.
.
.

ViStatus_VI_FUNC hpe650x_close();

Pointers to Memory Addresses

The success, error, and warning values are the only values returned. Retrieving actual data, such as IF attenuator setting, tuner frequency, digital IF bandwidth, etc., is accomplished by using pointers. A pointer is expected to be the address of allocated memory where the data is to be stored. These parameters have the letter "P" somewhere in their data type name. For example, ViPSession, ViPInt32, and ViPReal64 indicate that their corresponding parameters are pointers to the location where actual data is stored. Most commands starting with "get" have at least one pointer in the function's parameter list.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
VXI Plug and Play (from common.c)			
hpe650x_**autoConfigure**			
Returns all modules configured in the receiver.	ViUInt32	module_data[]	The array address passed by the user into which the module data will be passed. This array contains the following data: • Module ID number 270 - 1 GHz downconverter 271 - local oscillator 272 - 3 GHz block downconverter 273 - IF processor • Slot number • Logical address
Syntax:	ViStatus_VI_FUNC hpe650x_autoConfigure(module_data[]);		
hpe650x_**close**			
Terminates the session.	ViSession	instrumentID	Index to the array maintained by the driver.
Note:	• This function is required by the Plug and Play committee. • This command releases the IFP and its resources.		
Syntax:	ViStatus_VI_FUNC hpe650x_close(*instrumentID*);/*VISAIO.c*/		
Example:	Ret = hpe650x_close(instrumentID);		
hpe650x_**error_message**			
• Returns text error message. • VISA PnP standard command will return a text representation of errors in warning codes known to the hpe650x sub-system.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	error_code	Status code returned by the driver.
	ViString	error_message	Verbal translation of the numeric code.
Note:	• Will return VI_SUCCESS if code found. • Will return VI_WARN_UNKNOWN_STATUS if status code not known.		
Syntax:	ViStatus_VI_FUNC hpe650x_error_message(*instrumentID, error_code, error_message*);		
hpe650x_**error_query**			
• Returns current error state of hardware. • Not Supported. • VISA PnP standard command.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViPInt32	error_code	Not supported.
	ViString	error_message	Not supported.
Note:	Will return VI_WARN_NSUP_ERROR_QUERY		
Syntax:	ViStatus_VI_FUNC hpe650x_error_query(*instrumentID, error_code, error_message*);/*common.c*/		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**getIFAttenuator**			
Returns current IF attenuator setting.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication. 0 = IF channel 1 1 = IF channel 2
	ViPInt32	attenuator	Pointer to the address of the allocated memory for the attenuator setting.
	ViPInt32	rampage	Pointer to the address of allocated memory for rampage setting.

Note: This command and the hpe650x_setAutorangeLock command are linked. If you call the hpe650x_setAutorangeLock command with the use_external_gain parameter set to TRUE, then you must call the hpe650x_getIFAttenuator command first on the master IF processor.

Syntax: ViStatus_VI_FUNC hpe650x_getIFAttenuator(instrumentID, IFchan, attenuator, &rampage);

Example:
```
ViStatus Ret;
ViInt32 IFChannel=0;
ViInt32 att, rampage;
Ret = hpe650x_getIFAttenuator(instrumentID, IFChannel, att, &rampage);
```

Command/Action	Data Type	Parameters	
hpe650x_**getTunerTemperature**			
Returns internal temperature of tuner unit in receiver.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication. 0 = IF channel 1 1 = IF channel 2
	ViPReal64	temperature	Pointer to the address of allocated memory for temperature in Celsius.

Syntax: ViStatus_VI_FUNC hpe650x_getTunerTemperature(instrumentID, IFchan, temperature);

Example:
```
ViReal64 temp;
ViInt32 IF=0;
Ret = hpe650x_getTunerTemperature (instrumentID, IF, &temp);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**init**			
• Initializes communications to the IF processor. • Confirms that the logical address passed is that of an Agilent Technologies IF processor (to prevent initializing the incorrect module). • The only supported use is idQuery=true and resetInstr=true. • VISA PnP standard command.	ViRsrc	rsrcName	String built by the user interface and passed to the VISAIO library. Refer to the VISA documentation for more information.
	ViBoolean	idQuery	Determines if the option string will be read from the hardware by the driver. A value of true will force the driver to read the option string and enable error checking based on the results.
	ViBoolean	resetInstr	Determines if the DSP will be reset after the code is loaded. This value usually will be true, and MUST be true if you are booting from Flash ROM.
	ViPSession	pInstrumentID	Pointer to the address of the memory you have allocated for the instrument ID.

Note:
- This must be the first command called before any other command.
- This command must be called only once per IF module
- This function is required by the Plug and Play committee.
- The COMMON_STATE internal data structure contains the actual_instrument_id. The value passed to the functions is really an index to the array of structures containing information for the unit.

All subsequent commands to the E650X should be referenced by this InstrumentID number.

Syntax: ViStatus_VI_FUNC hpe650x_init(*rsrcName, idQuery, resetInstr, pInstrumentID*);

Example:
- ViStatus Ret;

ViSession InstrumentID;
Ret = hpe650x_init ("VXIO::43::INSTR", VI_TRUE, VI_TRUE, *&InstrumentID*);

hpe650x_**readOptionString**

Returns a list of options installed in the E6404A.

Data Type	Parameters	
ViSession	instrumentID	Index to the array maintained by the driver.
ViUInt16	option_buff[]	The array address passed by the user into which the option string will be passed.

Syntax: ViStatus_VI_FUNC hpe650x_readOptionString(*instrumentID, option_buff[]*);/*common.c*/

Example:
ViStatus Ret;
ViUInt16 option_buff]256];
Ret = hpe650x_readOptionString(instrumentID, option_buff);

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**reset**			
• Resets the DSP. • VISA PnP standard command.	ViSession	instrumentID	Index to the array maintained by the driver.
Syntax:	ViStatus_VI_FUNC hpe650x_reset(*instrumentID*);/*common.c*/		
Example:	ViStatus Ret; Ret = hpe650x_reset(instrumentID);		
hpe650x_**revision_query**			
• Returns current version of firmware and driver software. • VISA PnP standard command.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViChar_VI_FAR	driver_rev[]	The array address passed by the user that points to a memory block which the driver revision will be written to.
	ViChar_VI_FAR	instr_rev[]	The array address passed by the user that points to a memory block which the instrument revision will be written to.
Syntax:	ViStatus_VI_FUNC hpe650x_revision_query(*instrumentID, driver_rev[], instr_rev[]*);/*common.c*/		
Example:	Char driver[256].instrument[256]; Ret = hpe650x_revision_query(instrumentID, driver, instrument);		
hpe650x_**sanityCheck**			
Reads and writes to the DSP to determine if the processor is still able to both read and write commands.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine containing DSP to be checked. 0 = mezzanine 1 1 = mezzanine 2
Note:	This command is useful during development to ensure the processor is still operating. This is especially true when data is expected to be received but is not.		
Syntax:	ViStatus_VI_FUNC hpe650x_sanityCheck(*instrumentID, mezz_num*);		
hpe650x_**self_test**			
Not supported.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViPInt16	test_result	Not supported.
	ViChar _VI_FAR	test_message[]	Not supported.
Note:	Will return VI_WARN_NSUP_SELF_ TEST		
Syntax:	ViStatus_VI_FUNC hpe650x_self_test(*instrumentID, test_result,test_message[]*):/*common.c*/		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setAbsAmplitudeCalSignalCorr**			
Improves amplitude accuracy in certain extreme temperature conditions.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication. 0 = IF channel 1 1 = IF channel 2
	ViBoolean	enabled	VI_TRUE enables and VI_FALSE disables this function.
	ViReal64	calSignalLevel	Measured RSSI value of the front-panel 3rd LO Out from the E6402A LO module in dBm.
Note:	This is an option-dependent command. The absolute amplitude option must be installed or an error will be returned.		
Syntax:	ViStatus_VI_FUNC hpe650x_setAbsAmplitudeCalSignalCorr(instrumentID, IFchan, enabled, calSignalLevel);		
hpe650x_**setAbsAmplitudeTempComp**			
This is an option-dependent command. The absolute amplitude option must be installed or an error will be returned.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViBoolean	enabled	VI_TRUE enables and VI_FALSE disables built-in temperature table.
Syntax:	ViStatus_VI_FUNC hpe650x_setAbsAmplitudeTempComp(instrumentID, enabled);		
Example:	ViBoolean useTempComp=VI_TRUE; ViSession InstrumentID; ViStatus ret; ret=hpe650x_setAbsAmplitudeTempComp (instrumentID, useTempComp);		
hpe650x_**setAbsoluteAmplitude**			
The serial number for the tuner for each IF channel is stored in the IF processor.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication. 0 = IF channel 1 1 = IF channel 2
	ViBoolean	enabled	VI_TRUE enables and VI_FALSE disables optional absolute amplitude values.
Note:	This is an option-dependent command. The absolute amplitude option must be installed or an error will be returned.		
Syntax:	ViStatus_VI_FUNC hpe650x_setAbsoluteAmplitude(instrumentID, IFchan, enabled);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setAnalogFilter**			
Sets the analog filter in the IF processor. The 8 MHz bandwidth setting is the default.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication. 0 = IF channel 1 1 = IF channel 2
	ViInt32	bandwidth	0 = 30 kHz 1 = 700 kHz 2 = 8 MHz
Syntax:	ViStatus_VI_FUNC hpe650x_setAnalogFilter(instrumentID, IFchan, bandwidth);		
Example:	ViInt32 IF=0, filter=0; Ret = hpe650x_setAnalogFilter(instrumentID, IF, filter); See also "To set up an FFT measurement" in Chapter 3.		
hpe650x_**setAutorangeLock**			
This command either locks the autorange to a specified gain setting, or locks the autorange with a value from another IF channel.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication. 0 = IF channel 1 1 = IF channel 2
	ViBoolean	lock	Whether the autorange is locked or not, a value of VI_TRUE locks the range value, while VI_FALSE allows autorange to continue.
	ViInt32	value	Specified gain setting index (0 to 15).
	ViBoolean	use_external_gain	A value from another IF channel. Typically, this is used in a master-slave configuration.
Note:	• This command and the hpe650x_getIFAttenuator command are linked. If you call this command with the use_external_gain parameter set to TRUE, then you must call the hpe650x_getIFAttenuator command first on the master IF processor. • Lock the autorange control when performing any process where you do not want the gain changing values; for example, when capturing full span data.		
Syntax:	ViStatus_VI_FUNC hpe650x_setAutorangeLock(instrumentID, IFchan, lock, value, use_external_gain);		
Example:	See "To turn autoranging off" in Chapter 3.		
hpe650x_**setDitherState**			
Controls ADC dither operation.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel number: 0 = IF channel 1. 1 = IF channel 2.
	ViInt32	newstate	1 = Dither on, 0 = Dither off.
Syntax:	ViStatus_VI_FUNC hpe650x_setDitherState(*instrumentID, IFchan, newstate);*		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
From common			
hpe650x_**getIF3dBBandwidth**			
Computes the 3 dB points for the antialiasing filter being used.	ViInt32	index	Valid DDC index to the DDC_DecimationRate array.
	ViPReal64	bandwidth_hz	Address of the variable into which the bandwidth value will be placed.

Note:
- There is an array of pre-computed supported decimation rates, which may be altered (as in custom systems) for different sample rates. Error-checking is based on a constant in the *hpe650x.h* file.
- The DDC bandwidth setting is computed as:

$$DDC\ Bandwidth\ (Hz) = \frac{(DDC\ 3dB\ Factor) \cdot 2 \cdot (ADC\ Sampling\ Rate)}{Decimation\ Rate}$$

Syntax: ViStatus_VI_FUNC hpe650x_getIF3dBBandwidth(*index, bandwidth_hz*);

Example:
```
ViReal64 BW;
ViInt32 DDC=2;
Ret = hpe650x_getIF3dBBandwidth(DDC, &BW);
```

hpe650x_**initIFChannel**			
Initializes the variables for the IF channel, establishes communication to the logical addresses of the tuner cards, and establishes a connection between the mezzanine and the IF channel.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication. 0 = IF channel 1 1 = IF channel 2
	ViBoolean	tuner_attached	Confirms that the specified IF channel is controlling a tuner.
	ViReal64	InitialRFFrequency	Sets the state variable for the frequency.
	ViInt32	LO_LogicalAddr	Sets the logical address for the LO.
	ViInt32	OneGHz_LogicalAddr	Sets the logical address for the 1 GHz downconverter.
	ViInt32	ThreeGHz_LogicalAddr	Sets the logical address for the 3 GHz downconverter. If this module is not installed, you must pass a 0.

Note:
- If the E650x will control a VXI-based tuner, you must use this command to specify the logical addresses. Note that it is not possible to have two IF channels controlling a single tuner.
- A value of 0 passed to the tuner_attached parameter indicates that the tuner will be controlled by an external controller. A value of 1 passed to the tuner_attached parameter indicates that the tuner will be controlled by the internal IF processor (normal mode).
- In a shared LO configuration, the 3 GHz downconverter module is initialized for one channel but not for both.

Syntax: ViStatus_VI_FUNC hpe650x_initIFChannel(*instrumentID, IFchan, tuner_attached, InitialRFFrequency, LO_LogicalAddr, OneGHz_LogicalAddr, ThreeGHz_LogicalAddr*);/*These are hardware switch settings*/

Example:
```
ViSession InstrumentID
ViInt32 IF=0, LO=41, OneGHz=42, ThreeGHz=40;
ViBoolean tuner=1;
ViStatus ret
ret = hpe650x_InitIFChannel(instrumentID, IF, tuner, 20E6, LO, OneGHz, ThreeGHz)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**selectTuner10MHzReference**			
Selects either the internal 10 MHz reference or an external source. The internal source is switched on by default.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	tuner_num	Number of the tuner. 0 = tuner 1 1 = tuner 2
	ViInt32	source	1 = internal source 0 = external source
Syntax:	ViStatus_VI_FUNC hpe650x_selectTuner10MHzReference(*instrumentID, tuner_num, source*); enum source {_External=0,_Internal}; ViInt32 Tuner=0; ret= hpe650x_selectTuner10MHzReference(instrumentID, Tuner, _External);		
Example:			
hpe650x_**setDefaultIFConfig**			
Indicates whether one or two mezzanines are installed.	ViBoolean	mezzanine_one_exists	VI_TRUE indicates mezzanine one is installed; VI_FALSE if not.
	ViBoolean	mezzanine_two_exists	VI_TRUE indicates mezzanine two is installed; VI_FALSE if not.
Syntax:	ViStatus_VI_FUNC hpe650x_setDefaultIFConfig (*mezzanine_one_exists, mezzanine_two_exists*);		
hpe650x_**setIF10MHzReferenceOut**			
Turns on the 10 MHz reference output. (Off is the default.)	ViSession	instrumentID	Index to the array maintained by the driver.
	ViBoolean	enable	VI_TRUE enables and VI_FALSE disables this function.
Syntax:	ViStatus_VI_FUNC hpe650x_setIF10MHzReferenceOut (*instrumentID, enable*); See "To turn the IF processor 10 MHz reference on" in Chapter 3.		
Example:			
hpe650x_**setMezzanineDataSelectMode**			
• Establishes the IF signal routing to the mezzanines. Refer to Figures 3-42, 3-43, 3-44, and 3-45. • Mode 4 is the default.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mode	Valid modes and data routings are: 1 IF channel #1 data routed to all DDCs 2 IF channel #2 data routed to all DDCs 3 IF channel #1 data routed to 4 DDCs on mezzanine #1; data routed to 5th DDC on mezzanine #2. IF channel #2 data routed to 4 DDCs on mezzanine #2; data routed to 5th DDC on mezzanine #1 4 IF channel #1 data routed to all DDCs on mezzanine #1. IF channel #2 data routed to all DDCs on mezzanine #2
Note:	The mezzanine configuration determines which modes the driver will allow. For example, if mezzanine #2 does not exist, the driver will allow only mode 1 and mode 4. This prevents getting unexpected values when addressing the last DDC on a mezzanine.		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
Syntax:	ViStatus_VI_FUNC hpe650x_setMezzanineDataSelectMode (*instrumentID, mode*);		
	ViSession instrumentID;		
Example:	ViInt32 routingmode=4;		
	Ret= hpe650x_setMezzanineDataSelectMode(instrumentID, routingmode);		

hpe650x_**setMonitoringMode**

Sets the DSP to operate in monitoring mode.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine housing the Sharc chip to be booted. 0 = mezzanine 1 1 = mezzanine 2

Note: You can switch between search and monitoring modes with the same session (instrumentID).

Syntax: ViStatus_VI_FUNC hpe650x_setMonitoringMode(*instrumentID, mezz_num*);

Example: See "To set up an FFT measurement" in Chapter 3.

hpe650x_**setSearchMode**

Sets the DSP to operate in search mode.	ViSession ViInt32	instrumentID	Index to the array maintained by the driver.
		mezz_num	Mezzanine housing the Sharc chip to be booted. 0 = mezzanine 1 1 = mezzanine 2

Syntax: ViStatus_VI_FUNC hpe650x_setSearchMode(*instrumentID, mezz_num*);

Example: See "To set up a search process" in Chapter 3.

hpe650x_**setTunerAttenuation**

Adjusts the value of the tuner input attenuator (AT-1 or AT-2), depending on frequency.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	tuner_num	Number of the tuner for which the attenuator must be changed.
	ViInt32	attenuation	Amount of attenuation in dB from 0 to 30 in 10 dB steps.

Syntax: ViStatus_VI_FUNC hpe650x_setTunerAttenuation(*instrumentID, tuner_num, attenuation*);

Example:
```
ViInt32 IF=0, att=10;
ViStatus ret
ret = hpe650x_TunerAttenuation(instrumentID, IF, att);
```
See also "To set tuner attenuation" in Chapter 3.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
From search			
hpe650x_**getActualSearchStopFreq**			
Retrieves actual stop frequency.	ViSessions	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search 0 = mezzanine 1 1 = mezzanine 2
	ViPReal64	stop_freq	Actual stop frequency.
Syntax:	ViStatus_VI_FUNC hpe650x_getActualSearchStopFreq(instrumentID, mezz_num, stop_freq);		
hpe650x_**getFFTBinWidth**			
Shows how much bandwidth is contained in an FFT bin.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search. 0 = mezzanine 1. 1 = mezzanine 2.
	ViPReal64	fft_bin_width	Pointer to the address of memory allocated for the frequency value per bin.
Syntax:	ViStatus_VI_FUNC hpe650x_getFFTBinWidth(instrumentID, mezz_num, &fft_bin_width);		
hpe650x_**getNumberOfActiveModules**			
Returns the number of active modules in the driver.	ViPInt32	numOfActiveModules	Pointer to address of memory allocated for the number of active modules.
Note:	Nine IF modules may be controlled by a single controller.		
Syntax:	ViStatus_VI_FUNC hpe650x_getNumberOfActive Modules(*numOfActiveModules*); ViInt32 modules;		
Example:	ret= hpe650x_getNumberOfActiveModules(&modules); printf("There are %d IF modules active in", modules);		
hpe650x_**getSearchDecimationFactor**			
Decimates data down to fit within trace length.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search. 0 = mezzanine 1 1 = mezzanine 2
	ViPReal64	dec_factor	Pointer to the address of memory allocated for the decimation factor value.
Syntax:	ViStatus_VI_FUNC hpe650x_getSearchDecimationFactor (instrumentID, mezz_num, &dec_factor);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**getSearchFFTLength**			
Retrieves actual search FFT length.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search. 0 = mezzanine 1 1 = mezzanine 2
	ViPInt32	fft_length	Pointer to the address of memory allocated for the actual FFT length.
Syntax:	ViStatus_VI_FUNC hpe650x_getSearchFFTLength(instrumentID, mezz_num, fft_length);		
hpe650x_**getSearchIndexFrequency**			
This command returns the center frequency for the bin pointed to by the index parameter.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search. 0 = mezzanine 1 1 = mezzanine 2
	ViBoolean	decimated_indexes	Indicates whether the index has been decimated.
	ViInt32	index	Index into the search span corresponding to the index parameter.
	ViPReal64	frequency	Pointer to the address of the memory allocated for frequency.
Syntax:	ViStatus_VI_FUNC hpe650x_getSearchIndexFrequency(instrumentID, mezz_num, decimated_indexes, index, &frequency);		
hpe650x_**getSearchResolutionBW**			
Retrieves actual search resolution bandwidth.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search. 0 = mezzanine 1 1 = mezzanine 2
	ViPReal64	res_bw	Pointer to the address of memory allocated for the actual bandwidth value in Hz.
Syntax:	ViStatus_VI_FUNC hpe650x_getSearchResolutionBW(instrumentID, mezz_num, res_bw);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**getSearchTrace**			
This command returns the maximum and minimum power values (in dBm) for each decimated point.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search. 0 = mezzanine 1 1 = mezzanine 2
	ViReal64	maxtrace[]	The array address passed by the user into which the maximum trace value will be passed.
	ViReal64	mintrace[]	The array address passed by the user into which the minimum trace value will be passed.
Syntax:	ViStatus_VI_FUNC hpe650x_getSearchTrace(instrumentID, mezz_num, maxtrace[], mintrace[]);		
Example:	See "To set up a search process" in Chapter 3.		
hpe650x_**getSearchTraceBlock**			
Returns trace data in small blocks, rather than one long trace.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search. 0 = mezzanine 1 1 = mezzanine 2
	ViReal64	trace_block[]	The array address passed by the user into which the trace block will be passed.
	ViPInt32	index	The pointer to the sequence index of the block.
	ViPUInt32	length	Pointer to the address of the memory allocated for the length of the trace block vector.
	ViPBoolean	last_one	The pointer to the last block indicator.
Syntax:	ViStatus_VI_FUNC hpe650x_getSearchTraceBlock(instrumentID, mezz_num, trace_block[], &index, &length, &last_one);		
hpe650x_**getSearchTraceLength**			
Returns one amount of amplitude points than will be returned by the call to hpe650x_get SearchTrace( ).	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search. 0= mezzanine 1 1 = mezzanine 2
	ViPInt32	trace_length	Pointer to the address of memory allocated for the actual FFT length.
Syntax:	ViStatus_VI_FUNC hpe650x_getSearchTraceLength(instrumentID, mezz_num, trace_length);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setSearchOutputTraceLength**			
Decimates trace data automatically.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	target_trace_length	Specified trace length. The minimum is 10, and the maximum is the actual length of the FFT.
Syntax:	ViStatus_VI_FUNC hpe650x_setSearchOutputTraceLength (instrumentID, mezz_num, target_trace_length);		
Example:	See "To set up a search process" in Chapter 3.		
hpe650x_**setSearchResBWParameters**			
Use this command if you want bandwidths <5 kHz in the search process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	use_ddc	DDC number on mezzanine (DDC 0 only)
	ViInt32	res_bw_index	DDC bandwidth. See the setDigitalIFBandwidth command for bandwidth index numbers.
	ViInt32	fft_length	Requested length of the FFT. The range is 64 to 8,192 points.
Syntax:	ViStatus_VI_FUNC hpe650x_setSearchResBWParameters (instrumentID, mezz_num, use_ddc, res_bw_index, fft_length);		
Example:	See "To set up a search process" in Chapter 3.		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setSearchResolutionBW**			
Sets the resolution bandwidth for the search.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	res_bw_index	Index to BW table value must be 0 through 43 (44 possible values as shown below).

Supported Search Resolution Bandwidths:

Index Number	Bandwidth Setting	Index Number	Bandwidth Setting
0	668.75 kHz	22	62.198 Hz
1	334.38 kHz	23	55.581 Hz
2	167.19 kHz	24	51.223 Hz
3	83.594 kHz	25	45.83 Hz
4	41.797 kHz	26	39.58 Hz
5	20.898 kHz	27	36.282 Hz
6	10.45 kHz	28	31.099 Hz
7	5.225 kHz	29	27.8 Hz
8	4.9173 kHz	30	24.64 Hz
9	2.459 kHz	31	20.73 Hz
10	1.229 kHz	32	18.14 Hz
11	614.66 Hz	33	14.76 Hz
12	307.33 Hz	34	11.46 Hz
13	153.665 Hz	35	9.86 Hz
14	145.128 Hz	36	8.215 Hz
15	124.395 Hz	37	6.698 Hz
16	108.846 Hz	38	4.92 Hz
17	96.752 Hz	39	4.107 Hz
18	87.077 Hz	40	3.286 Hz
19	79.161 Hz	41	2.093 Hz
20	72.564 Hz	42	1.643 Hz
21	66.982 Hz	43	1.0884 Hz

Syntax: ViStatus_VI_FUNC hpe650x_setSearchResolutionBW(instrumentID, mezz_num, res_bw_index);

Example: See "To set up a search process" in Chapter 3.

hpe650x_**setSearchSpan**			
Specifies the start and stop frequencies to be searched.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search. 0 = mezzanine 1 1 = mezzanine 2
	ViReal64	start_freq	Start frequency (2 MHz).
	ViReal64	stop_freq	Stop frequency (1 GHz or 3 GHz depending on installed option).

Syntax: ViStatus_VI_FUNC hpe650x_setSearchSpan(instrumentID, mezz_num, start_freq, stop_freq);

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setSearchType**			
Determines if internal DSP will perform data decimation, or user's application will perform data decimation.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	search_type	0 = decimated video data 1 = undecimated video data 2 = no data/not supported
Syntax:	ViStatus_VI_FUNC hpe650x_setSearchType(instrumentID, mezz_num, search_type);		
hpe650x_**startSearch**			
Starts the search process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine to be used for search. 0 = mezzanine 1 1 = mezzanine 2
Note:	Mezzanine must be in search mode.		
Syntax:	ViStatus_VI_FUNC hpe650x_startSearch(instrumentID, mezz_num);		
Example:	See "To set up a search process" in Chapter 3.		
hpe650x_**stopSearch**			
Stops the search process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for search. 0 = mezzanine 1 1 = mezzanine 2
Syntax:	ViStatus_VI_FUNC hpe650x_stopSearch(instrumentID, mezz_num);		
Example:	See "To set up a search process" in Chapter 3.		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
From monitor			
hpe650x_**abortDataCollection**			
Aborts the multiple IF processor data collection process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine to be controlled. 0 = mezzanine 1 1 = mezzanine 2
Syntax:	ViStatus_VI_FUNC hpe650x_abortDataCollection(instrumentID, mezz_num, hardware_reset);		
hpe650x_**activateAFC**			
Activates automatic frequency control for the specified mezzanine and DDC.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine in which AFC is to activated. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DDC_num	DDC in which AFC is to be activated (0 through 4).
Syntax:	ViStatus_VI_FUNC hpe650x_activateAFC(instrumentID, mezz_num, DDC_num); See "To activate automatic frequency control" in Chapter 3.		
Example:			
hpe650x_**activateAutoranging**			
Activates the autoranging function for the specified IF channel. On is the default setting.	ViSession	instrumentID	Index to the array maintained by the driver
	ViInt32	IFchan	IF channel for communication 0 = IF channel 1 1 = IF channel 2
Note:	This command is called only once per setup. For example, if autoranging is activated when the mezzanine is initialized, you do not need to call the function again when the tuner is retuned.		
Syntax:	ViStatus_VI_FUNC hpe650x_activateAutoranging(*instrumentID, IFchan*); See "To set up an FFT measurement" in Chapter 3.		
Example:			
hpe650x_**armDDCsForSynchronization**			
Prepares DDCs for multiple IF processor synchronization process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine to be controlled 0 = mezzanine 1 1 = mezzanine 2
	ViBoolean	hardware_reset	VI_TRUE indicates an external device will provide trigger signal.
Syntax:	ViStatus_VI_FUNC hpe650x_armDDCsForSynchronization(instrumentID, mezz_num, hardware_reset);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**armDSPForDataCollection**			
Prepares DSP for multiple IF processor synchronization process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine to be controlled 0 = mezzanine 1 1 = mezzanine 2
	ViBoolean	servant	VI_TRUE indicates that the specified mezzanine is a servant.
Syntax:	ViStatus_VI_FUNC hpe650x_armDSPForDataCollection(instrumentID, mezz_num, servant);		
hpe650x_**checkDDCsAvailable**			
Returns a vector of the currently available DDCs and the maximum bandwidth that can be used.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine that has the DDCs of interest. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	chan_array[ ]	The array address corresponding to the available DDCs.
	ViPInt32	max_bw_index	Pointer to the address of the allocated memory for the maximum bandwidth setting.
Note:	If the frequency is beyond the tuner's range, the command returns an error (some considerations are made when tuning around a preselector band).		
Syntax:	ViStatus_VI_FUNC hpe650x_checkDDCsAvailable(*instrumentID, mezz_num, chan_array[ ], &max_bw_index);*		
hpe650x_**clearCaptureDDCNum**			
Clears the specified DDC in the I/Q data capture process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DDC_num	DDC number to clear (0 through 4).
Note:	• This command will be set for each DDC. • At least one DDC must be assigned for an I/Q capture process		
Syntax:	ViStatus_VI_FUNC hpe650x_clearCaptureDDCNum(instrumentID, mezz_num, DDC_num);		
hpe650x_**deactivateAFC**			
Deactivates automatic frequency control for the specified mezzanine and DDC.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine in which AFC is to deactivated. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DDC_num	DDC in which AFC is to deactivated (0 through 4).
Syntax:	ViStatus_VI_FUNC hpe650x_deactivateAFC(instrumentID, mezz_num, DDC_num);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**disableAFC**			
Disables automatic frequency control for the specified mezzanine and DDC.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine for which AFC is to be disabled. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DDC_num	DDC on mezzanine (0 through 4).
Syntax:	ViStatus_VI_FUNC hpe650x_disableAFC(*instrumentID, mezz_num, DDC_num*);		
hpe650x_**disableVCO**			
Disables clock VCO on slave IF processor.	ViSession	instrumentID	Index to the array maintained by the driver.
Syntax:	ViStatus_VI_FUNC hpe650x_disableVCO(*instrumentID*);		
hpe650x_**enableAFC**			
Enables automatic frequency control for the specified mezzanine and DDC.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine for which AFC is to be enabled. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DDC_num	DDC on mezzanine (0 through 4).
Syntax:	ViStatus_VI_FUNC hpe650x_enableAFC(*instrumentID, mezz_num, DDC_num*);		
hpe650x_**gangTuneDDC**			
Tunes all DDCs to the specified frequency simultaneously.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine containing DDCs to be tuned. 0 = mezzanine 1 1 = mezzanine 2
	ViBoolean	absolute	VI_TRUE will tune DDCs to absolute frequency; VI_FALSE will tune DDCs relative to antialiasing filter.
	ViReal64	DDC0Frequency	Frequency setting for DDC 0 (±1/2 analog filter bandwidth).
	ViReal64	DDC1Frequency	Frequency setting for DDC 1 (±1/2 analog filter bandwidth).
	ViReal64	DDC2Frequency	Frequency setting for DDC 2 (±1/2 analog filter bandwidth).
	ViReal64	DDC3Frequency	Frequency setting for DDC 3 (±1/2 analog filter bandwidth).
	ViReal64	DDC4Frequency	Frequency setting for DDC 4 (±1/2 analog filter bandwidth).
Note:	This command minimizes command overhead by tuning all DDCs simultaneously.		
Syntax:	ViStatus_VI_FUNC hpe650x_gangTuneDDC(instrumentID, mezz_num, absolute, DDC0Frequency, DDC1Frequency, DDC2Frequency, DDC3Frequency, DDC4Frequency);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**getADCClippingIndicator**			
Returns clipping state of an ADC.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication 0 = IF channel 1 1 = IF channel 2
	ViPBoolean	overload	VI_TRUE = ADC is clipping VI_FALSE = ADC is not clipping
Syntax:	ViStatus_VI_FUNC hpe650x_getADCClippingIndicator(instrumentID, IFchan, overload);		
hpe650x_**getAutorangeState**			
Reads complete autorange state from an IF channel.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication 0 = IF channel 1 1 = IF channel 2
	ViPInt32	gain_profile	Pointer to receiver gain profile
	ViPInt32	gain_subprofile	Pointer to receiver gain sub-profile
	ViPInt32	sys_gain	Pointer to receiver system gain
Syntax:	ViStatus_VI_FUNC hpe650x_getAutorangeState(instrumentID, IFchan, &gain_profile, &gain_subprofile, &sys_gain);		
hpe650x_**getCaptureCollectInSRAM**			
Retrieves the capture state corresponding to SRAM or Linkport data capture.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViPBoolean	collect_in_SRAM	A pointer to a value indicating VI_TRUE indicates capture data will be stored in SRAM, otherwise it will stream to the link port.
Note:	SRAM is a product option and must be installed for capture data to be stored in SRAM.		
Syntax:	ViStatus_VI_FUNC hpe650x_getCaptureCollectInSRAM(instrumentID, mezz_num, &collect_in_SRAM);		
hpe650x_**getCaptureDataDDCNum**			
Retrieves capture data from DDC.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DDC_num	Number of the DDC (0 through 4).
	ViPBoolean	active	Pointer indicating whether the specified DDC is active during capture.
Syntax:	ViStatus_VI_FUNC hpe650x_getCaptureDataDDCNum(instrumentID, mezz_num, DDC_num, &active);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**getCaptureDataFormat**			
Retrieves whether I/Q or full rate data has been set.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViPInt32	format	Pointer to the address in memory allocated for format state, either digital I/Q or full rate ADC. 0 = Digital I/Q 1=Full rate ADC
Syntax:	ViStatus_VI_FUNC hpe650x_getCaptureDataFormat(instrumentID, mezz_num, &format);		
hpe650x_**getCaptureDataOutput**			
Retrieves the output to VXI bus or link ports.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViPInt32	output	Pointer to the address in memory allocated for the output state, either VXI, SRAM, or linkport. 0 = VXI 1 = SRAM 2 = linkport
Syntax:	ViStatus_VI_FUNC hpe650x_getCaptureDataOutput(instrumentID, mezz_num, &output);		
hpe650x_**getCaptureDigitalIQData**			
Returns I/Q data.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViInt16	IBuff[]	The array address passed by the user into which the "I" data will be passed.
	ViInt16	QBuff[]	The array address passed by the user into which the "Q" data will be passed.
	ViPInt32	length	Pointer to the length of the I:Q vectors (arrays). Their lengths are equal.

Note:

- Returned data is interlaced. The IBuff[] array contains the following:

  $[I_1DDC_1, I_1DDC_2, \ldots I_1DDC_m, \ldots I_2DDC_1, I_2DDC_2, \ldots I_2DDC_m, \ldots I_nDDC_1, I_nDDC_2, \ldots I_nDDC_m,]$.

  See Table 3-14 and Scenario 1 sample code.
- This command returns data which is scaled to the output of the ADC. In order to scale the number to that of the input voltage, the results must be multiplied by the result of hpe650x_getDataCorrectionValue.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**getCaptureDataFormat**			
Retrieves whether I/Q or full rate data has been set.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViPInt32	format	Pointer to the address in memory allocated for format state, either digital I/Q or full rate ADC. 0 = Digital I/Q 1=Full rate ADC
Syntax:	ViStatus_VI_FUNC hpe650x_getCaptureDataFormat(instrumentID, mezz_num, &format);		
hpe650x_**getCaptureDataOutput**			
Retrieves the output to VXI bus or link ports.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViPInt32	output	Pointer to the address in memory allocated for the output state, either VXI, SRAM, or linkport. 0 = VXI 1 = SRAM 2 = linkport
Syntax:	ViStatus_VI_FUNC hpe650x_getCaptureDataOutput(instrumentID, mezz_num, &output);		
hpe650x_**getCaptureDigitalIQData**			
Returns I/Q data.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViInt16	IBuff[]	The array address passed by the user into which the "I" data will be passed.
	ViInt16	QBuff[]	The array address passed by the user into which the "Q" data will be passed.
	ViPInt32	length	Pointer to the length of the I:Q vectors (arrays). Their lengths are equal.

Note:

- Returned data is interlaced. The IBuff[] array contains the following:

  $[I_1DDC_1, I_1DDC_2, \ldots I_1DDC_m, \ldots I_2DDC_1, I_2DDC_2, \ldots I_2DDC_m, \ldots I_nDDC_1, I_nDDC_2, \ldots I_nDDC_m,]$.

  See Table 3-14 and Scenario 1 sample code.
- This command returns data which is scaled to the output of the ADC. In order to scale the number to that of the input voltage, the results must be multiplied by the result of hpe650x_getDataCorrectionValue.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
Syntax:	ViStatus_VI_FUNC hpe650x_getCaptureDigitalIQData(instrumentID, mezz_num, IBuff[], QBuff[], &length);		
hpe650x_**getCaptureFullRateADCData**			
Returns full rate data.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViInt16	ADCBuffer[ ]	ADC data return array (unpacked).
	ViPInt32	length	Pointer to memory address allocated for vector length.
Note:	• See Table 3-16 and Scenario 14 sample code. • This command returns data which is scaled to the output of the ADC. In order to scale the number to that of the input voltage, the results must be multiplied by the result of hpe650x_getDataCorrectionValue.		
Syntax:	ViStatus_VI_FUNC hpe650x_getCaptureFullRateADCData(instrumentID, mezz_num, ADCBuffer[ ], &length);		
hpe650x_**getCaptureTrigger**			
Gets the number of trigger cycles expected when streaming n-samples of ADC data to the link port using multiple triggers (scenario 13), or n-samples of I/Q to Linkport (scenario 8).	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture 0 = mezzanine 1 1 = mezzanine 2
	ViPBoolean	hardware_trigger	Pointer to the address of allocated memory for the trigger status.
	ViPInt32	trigger_cycles	Pointer to the number of triggers expected.
Syntax:	ViStatus_VI_FUNC hpe650x_getCaptureTrigger(instrumentID, mezz_num, &trigger, &trigger_cycles);		
hpe650x_**getDataCorrectionValue**			
Reads data correction value (linear) that normalizes receiver gain.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine containing correction value. 0= mezzanine 1 1 = mezzanine 2
	ViInt32	source	0 – 4, DDC 1 – 5 5, ADC
	ViPReal32	cor_value	Pointer to the address in memory allocated for the correction value.
Syntax:	ViStatus_VI_FUNC hpe650x_getDataCorrectionValue(*instrumentID, mezz_num, source, &cor_value*);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**getCurrentDemodType**			
Returns the current demodulation mode for the specified DDC and mezzanine board. FM = 0 AM = 1 CW = 2 USB = 3 LSB = 4 PM = 5	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine which contains the AFC. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DDC_num	DDC index
	ViPInt32	DemodType	Defines specified frequency as absolute (true frequency) or relative (relative to the given anti-aliasing filter bandwidth).
			Pointer to the address of allocated memory for the returned frequency of the DDC.

Note: DDCs are tuned based on tuner frequency.

Syntax: ViStatus_VI_FUNC hpe650x_getCurrentDemodType(*instrumentID, mezz_num, DDC_num, &DemodType*); ViInt32 demodmode;

Example: ret= hpe650x_getCurrentDemodType(instumentID, mezz, &demodmode);

hpe650x_**getDDCFrequency**

Returns the frequency value of the specified DDC.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine which contains the AFC. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DDC_num	DDC index
	ViBoolean	absolute	Defines specified frequency as absolute (true frequency) or relative (relative to the given anti-aliasing filter bandwidth).
	ViPreal64	freq	Pointer to the address of allocated memory for the returned frequency of the DDC.

Note: DDCs are tuned based on tuner frequency.

Syntax: ViStatus_VI_FUNC hpe650x_getDDCFrequency(*instrumentID, mezz_num, DDC_num, absolute, &freq*);

Example:
```
ViStatus ret;
ViInt32 mezz=0, DDC=0;
ViBoolean absolute=1;
ViSession instrumentID;
ret= hpe650x_getDDCFrequency(instrumentID, mezz, DDC, absolute, &fHz;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**getDigitalIFBandwidth**			
Retrieves the bandwidth setting for the specified mezzanine board.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine from which the bandwidth is to be retrieved. 0 = mezzanine 1 1 = mezzanine 2
	ViPInt32	bandwidth_index	Pointer to the address of allocated memory for the value to which the mezzanine is set (returned).
	ViPReal64	bandwidth_hz	Pointer to the address of allocated memory for the physical bandwidth setting (in Hz) to which the mezzanine is set.
Syntax:	ViStatus_VI_FUNC hpe650x_getDigitalIFBandwidth(*instrumentID, mezz_num, bandwidth_index, &bandwidth_hz*);		
hpe650x_**getIFChannelForDDC**			
Returns the IF channel associated with a particular DDC.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine that has the DDCs of interest. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DDC_num	DDC number on mezzanine (0 through 4).
	ViPInt32	IFchan	Pointer to the address of allocated memory for the IF channel routed to the specified DDC or mezzanine.
Syntax:	ViStatus_VI_FUNC hpe650x_getIFChannelForDDC(*instrumentID, mezz_num, DDC_num, &IFchan*);		
hpe650x_**getMultipleTriggerAction**			
Gets the capture mode so that multiple triggers are expected.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine to be triggered. 0 = mezzanine 1 1 = mezzanine 2
	ViPBoolean	multiple	Pointer to indicate that multiple triggers will occur.
Syntax:	ViStatus_VI_FUNC hpe650x_getMultipleTriggerAction(instrumentID, mezz_num, &multiple);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**getNumberOfSamplesToCapture**			
Retrieves the number of samples that were captured.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0= mezzanine 1 1 = mezzanine 2
	ViPInt32	num_samples	Pointer to the address in memory allocated to the number of I and Q samples that were captured.

Syntax: ViStatus_VI_FUNC hpe650x_getNumberOfSamplesToCapture(instrumentID, mezz_num, &num_samples);

hpe650x_**getRSSIvalue**

Retrieves the corresponding RSSI measurement from the specified mezzanine and DDC.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine containing the DDC. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	chan	DDC channel (0 through 4).
	ViPReal64	rssi_val	Pointer to the address of allocated memory for the returned value.

Note:
- DDC channel is mode-specific. By default, the mezzanine data select mode is #4.
- There is a group delay of approximately 10 msec. Therefore, an RSSI reading cannot be taken within the first 10 msec.

Syntax: ViStatus_VI_FUNC hpe650x_getRSSIvalue(*instrumentID, mezz_num, chan, &rssi_val*);

Example: See "To set up a channelized power measurement" in Chapter 3.

hpe650x_**getSuspendedCaptureTask**

Gets the suspended mode indication. See the hpe650x_set SuspendedCapture Task command.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViPBoolean	suspended	Pointer to a value of VI_TRUE indicating capture will be started in suspended mode, and started by a trigger.

Syntax: ViStatus_VI_FUNC hpe650x_getSuspendedCaptureTask(instrumentID, mezz_num, &suspended);

hpe650x_**getTunerFrequency**

Reports the value of the RF frequency for a given IF channel.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication 0 = IF channel 1 1 = IF channel 2
	ViPReal64	frequency	Pointer to the address of the allocated memory for the frequency value.

Syntax: ViStatus_VI_FUNC hpe650x_getTunerFrequency(*instrumentID, IFchan, &frequency*);

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**hardSystemReset**			
• Performs reset on IF processor hardware. • Group deemphasis command.	ViSession	instrumentID	Index to the array maintained by the driver.
Syntax: ViStatus_VI_FUNC hpe650x_hardSystemReset(*instrumentID*);			
hpe650x_**prearmDDCsForSynchronization**			
• Performs prearm functions prior to data collection.	ViSession	instrumentID	Index to the array maintained by the driver.
• Called before arming the DDCs and DSP for capture. Sets DSP state. See capture examples in Chapter 3.	ViInt32	mezz_num	Mezzanine used for synchronization. 0 = mezzanine 1 1 = mezzanine 2
Syntax: ViStatus_VI_FUNC hpe650x_prearmDDCsForSynchronization(instrumentID, mezz_num);			
hpe650x_**prearmDSPForDataCollection**			
• Preforms prearm functions prior to data collection	ViSession	instrumentID	Index to the array maintained by the driver.
• Called before arming the DSP for capture. Sets DSP state. See capture examples in Chapter 3.	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
Syntax: ViStatus_VI_FUNC hpe650x_prearmDSPForDataCollection(instrumentID, mezz_num);			
hpe650x_**selectBackplaneFs**			
Sets clock source to the VXI back-plane.	ViSession	instrumentID	Index to the array maintained by the driver.
Syntax: ViStatus_VI_FUNC hpe650x_selectBackplaneFs(*instrumentID*);			
hpe650x_**setALCRate**			
Sets the automatic level control (ALC) for the audio output. Controls both attack time and decay time of the ALC.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine for which ALC will be changed. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	select	0 = change decay time 1 = adjust attack time
	ViInt32	m_sec	Value to adjust ALC rate in milliseconds (0.001 to 10,000).
Note: This command applies only to SSB, ISB, USB, and LSB.			
Syntax: ViStatus_VI_FUNC hpe650x_setALCRate(*instrumentID, mezz_num, select, m_sec*);			

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setAutorangeState**			
Sets complete autorange state on an IF channel.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication 0 = IF channel 1 1 = IF channel 2
	ViInt32	gain_profile	Receiver gain profile.
	ViInt32	gain_subprofile	Receiver gain sub-profile.
	ViInt32	sys_gain	Receiver system gain.
Note:	If you want to set the autorange state, you must first use the hpe650x_getAutorangeState command to get the autorange state from a master IF.		
Syntax:	ViStatus_VI_FUNC hpe650x_setAutorangeState(instrumentID, IFchan, gain_profile, gain_subprofile, sys_gain);		
hpe650x_**setCaptureCollectInSRAM**			
Sets the captured I/Q data to either store in SRAM or stream to the link port.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0= mezzanine 1 1 = mezzanine 2
	ViBoolean	collect_in_SRAM	Indicates whether data samples will be stored in SRAM. A value of VI_TRUE indicates capture data will be stored in SRAM. Otherwise, capture data will be streamed to the link port.
Note:	• This must be enabled for captured I and Q data to be output to VXI bus. • SRAM is a product option and must be installed for capture data to be stored in SRAM.		
Syntax:	ViStatus_VI_FUNC hpe650x_setCaptureCollectInSRAM(instrumentID, mezz_num, collect_in_SRAM);		
Example:	See "To set up digital I/Q data output" in Chapter 3.		
hpe650x_**setCaptureDataDDCNum**			
Sets DDC for capture process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DDC_num	Number of the DDC (0 through 4).
	ViBoolean	active	VI_TRUE indicates that the specified DDC will be used for capture.
Syntax:	ViStatus_VI_FUNC hpe650x_setCaptureDataDDCNum(instrumentID, mezz_num, DDC_num, active);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setCaptureDataFormat**			
Sets either I/Q or full rate data.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	format	Digital I/Q = 0 Full rate ADC data = 1
Syntax:	ViStatus_VI_FUNC hpe650x_setCaptureDataFormat(instrumentID, mezz_num, format);		
Example:	See "To set up digital I/Q data output" in Chapter 3.		
hpe650x_**setCaptureDataOutput**			
Sets the output to VXI bus or link ports.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	output	Indicates whether output should go to the VXI bus (0) or linkport (1).
Note:	SRAM option must be installed and data collected in SRAM for output to VXI bus to be valid.		
Syntax:	ViStatus_VI_FUNC hpe650x_setCaptureDataOutput(instrumentID, mezz_num, output);		
Example:	See "To set up digital I/Q data output" in Chapter 3.		
hpe650x_**setCaptureTrigger**			
Sets the number of trigger cycles expected when streaming n-samples of ADC data to the link port using multiple triggers (scenario 13), or n-samples of I/Q to Linkport (scenario 8).	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViBoolean	hardware_trigger	VI_TRUE indicates an external trigger; VI_FALSE indicates an internal trigger.
	ViInt32	trigger_cycles	Number of triggers expected.
Syntax:	ViStatus_VI_FUNC hpe650x_setCaptureTrigger(instrumentID, mezz_num, trigger, trigger_cycles);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setDDCFrequency**			
Sets the DDC frequency by absolute (the actual frequency of interest) or to relative (the frequency relative to the IF bandwidth, which is centered at 21.4 MHz).	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine selected 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DDC_num	DDC number on mezzanine (0 through 4).
	ViBoolean	absolute	If VI_TRUE, defines specified frequency as absolute (true frequency) or, if VI_FALSE, relative to the tuning frequency (the tuner frequency is subtracted from the "frequency" parameter).
	ViReal64	frequency	Frequency setting (cannot exceed ±1/2 analog filter bandwidth).

Note: The DDC frequency may be set relative to the IF channel tuned frequency or set to absolute frequency.

Syntax: ViStatus_VI_FUNC hpe650x_setDDCFrequency(*instrumentID, mezz_num, DDC_num, absolute, frequency*);

Example: See "To set up an FFT measurement" in Chapter 3.

hpe650x_**setDeemphasis**			
Enables 75 μs FM de-emphasis for audio.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for deemphasis. 0 = mezzanine 1 1 = mezzanine 2
	ViBoolean	enabled	VI_TRUE enables and VI_FALSE disables this function.

Syntax: ViStatus_VI_FUNC hpe650x_setDeemphasis(*instrumentID, mezz_num, enabled*);

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setDemodType**			
Sets the demodulation mode for the specified DDC and mezzanine board.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for demodulation. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DDC_num	DDC on mezzanine (0 through 4).
	ViInt32	demod	Demodulation type. FM = 0 AM = 1 CW = 2 USB = 3 LSB = 4 PM = 5

Syntax: ViStatus_VI_FUNC hpe650x_setDemodType(*instrumentID, mezz_num, DDC_num, demod*);
enum demodtypes {FM=0,AM,CW,USB,LSB,PM};

Example:
```
demodtypes mode=AM;
ViInt32 DDC=0, mezz=0;
ret= hpe650x_setDemodType(instrumentID, mezz, DDC, mode);
/*set DDC 0 on mezzanine 0 to AM*/
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setDigitalIFBandwidth**			
Sets the digital IF bandwidth to a value specified by the passed index for the specified mezzanine.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine to be controlled. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	bandwidth_index	Index corresponding to the supported DDC bandwidth table.

Note: All DDCs assume the same digital IF bandwidths per mezzanine except in Mode 3.

Syntax: ViStatus_VI_FUNC hpe650x_setDigitalIFBandwidth(*instrumentID, mezz_num, bandwidth_index*);

Example: See "To set span in monitor mode" in Chapter 3.

Supported Digital IF Bandwidths:

Index Number	Bandwidth Setting
0	247 Hz
1	493 Hz
2	740 Hz
3	1 kHz
4	2 kHz
5	3 kHz
6	5 kHz
7	6 kHz
8	10 kHz
9	12 kHz
10	15 kHz
11	20 kHz
12	25 kHz
13	29 kHz
14	34 kHz
15	44 kHz
16	54 kHz
17	64 kHz
18	74 kHz
19	83 kHz
20	93 kHz
21	109 kHz
22	123 kHz
23	138 kHz
24	154 kHz
25	171 kHz
26	187 kHz
27	201 kHz
28	218 kHz
29	238 kHz
30	262 kHz
31	291 kHz
32	327 kHz
33	374 kHz
34	436 kHz
35	462 kHz

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setDROAttackTime**			
Sets the dynamic range optimization (DRO) attack time.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication. 0 = IF channel 1 1 = IF channel 2
	ViInt32	delay_units	2 to 2000 delay units.
Note:	• If a peak signal level is above an upper threshold for a time equal to the attack time setting, then the correction RAM is re-optimized. • Avoids responding to every fluctuation in signal amplitudes.		
Syntax:	ViStatus_VI_FUNC hpe650x_setDROAttackTime(*instrumentID, IFchan,* delay_units);		
Example:	See "To set up dynamic range optimization" in Chapter 3.		
hpe650x_**setDRODecayTime**			
Sets the dynamic range optimization (DRO) decay time.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication. 0 = IF channel 1. 1 = IF channel 2.
	ViInt32	delay_units	2 to 2000 delay units. $Td * \left[\sum_{k=0}^{ceil\,(P/2)-1} 1/2^{K} \right]$
Note:	• If a peak signal level is below a lower threshold for a time equal to the decay time setting, then the correction RAM is re-optimized. • Avoids responding to every fluctuation in signal amplitudes.		
Syntax:	ViStatus_VI_FUNC hpe650x_setDRODecayTime(*instrumentID, IFchan,* delay_units);		
Example:	See "To set up dynamic range optimization" in Chapter 3.		
hpe650x_**setIFGain**			
Sets the overall gain of the IF module; determines the correct settings for the attenuators, amplifiers, etc.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication 0 = IF channel 1 1 = IF channel 2
	ViInt32	gain	Gain setting for the IF channel; -48 dBm to 0 dBm in 2 dB steps. Overrange from +2 dBm to +12 dBm also available. Refer to Table 3-12 for index numbers used for setting gain.
Note:	If autoranging is on for this channel, then this gain setting is disabled.		
Syntax:	ViStatus_VI_FUNC hpe650x_setIFGain(*instrumentID, IFchan, gain*);		
Example:	See "To set the IF gain" in Chapter 3.		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setInterMezzanineAudio**			
Allows the routing of audio from one mezzanine to the other. Using the audio breakout box, up to 10 audio outputs are possible with two mezzanines installed.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine to be controlled. 0 = mezzanine 1 1 = mezzanine 2
	ViBoolean	enable	Indicates whether to enable (VI_TRUE) or disable (VI_FALSE) intermezzanine audio.
Syntax:	ViStatus_VI_FUNC hpe650x_setInterMezzanineAudio(instrumentID, mezz_num, enable);		
hpe650x_**setMasterIFClock**			
Sets the selected module as the master IF processor and will distribute its clock to the back-plane.	ViSession	instrumentID	Index to the array maintained by the driver.
Syntax:	ViStatus_VI_FUNC hpe650x_setMasterIFClock(*instrumentID*);		
hpe650x_**setMultipleTriggerAction**			
Sets the capture mode so that multiple triggers are expected.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine to be triggered. 0= mezzanine 1 1 = mezzanine 2
	ViBoolean	multiple	Indicates that multiple triggers will occur.
Syntax:	ViStatus_VI_FUNC hpe650x_setMultipleTriggerAction(instrumentID, mezz_num, multiple);		
hpe650x_**setNumberOfSamplesToCapture**			
Sets the number of samples to capture or sets the capture length to indefinite.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0= mezzanine 1 1 = mezzanine 2
	ViInt32	num_samples	Specifies the number of I and Q samples that should be captured (1 to 1,048,576 for one DDC), or 0 for indefinite.
Note:	• If data is being captured to SRAM, amount of data captured is limited to available memory. • If data is being streamed out of the link ports, then up to 2^{31} - 1 samples are possible.		
Syntax:	ViStatus_VI_FUNC hpe650x_setNumberOfSamplesToCapture(instrumentID, mezz_num, num_samples); See "To set up digital I/Q data output" in Chapter 3.		
Example:			

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setRSSIMeasTime**			
Sets the dwell time for all DDCs when making received signal strength indication (RSSI).	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine for which RSSI measurement time is to be set. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	m_sec	Dwell time in milliseconds (0.001 to 10,000).
Note:	Influences measured results tremendously.		
Syntax:	ViStatus_VI_FUNC hpe650x_setRSSIMeasTime(*instrumentID, mezz_num, m_sec*); See "To set up a channelized power measurement" in Chapter 3.		
Example:			
hpe650x_**setSlaveIFClock**			
Sets the selected module as the servant IF processor.	ViSession	instrumentID	Index to the array maintained by the driver.
Syntax:	ViStatus_VI_FUNC hpe650x_setSlaveIFClock(*instrumentID*);		
hpe650x_**setSquelchLevel**			
Establishes the squelch level for the audio output of the specified mezzanine.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine for which squelch is to be adjusted. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	level	Set squelch level in dBm (20 dBm to 120 dBm).
Note:	Squelch level is established per mezzanine value.		
Syntax:	ViStatus_VI_FUNC hpe650x_setSquelchLevel(*instrumentID, mezz_num, level*); See "To set up demodulation, turn on an audio channel, and set squelch" in Chapter 3.		
Example:			
hpe650x_**setSquelchState**			
Permits the user to enable or disable the squelch for the specified mezzanine.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine to be controlled. 0 = mezzanine 1 1= mezzanine 2
	ViBoolean	activate	VI_TRUE activates and VI_FALSE deactivates squelch.
Syntax:	ViStatus_VI_FUNC hpe650x_setSquelchState(*instrumentID, mezz_num, activate*); /*turn squelch on for mezzanine 0*/		
Example:	ret= hpe650x_setSquelchState(instrumentID, mezz, VI_TRUE);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setSuspendedCaptureTask**			
Used to set whether capture will be started in suspended mode and started by a trigger.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
	ViBoolean	suspended	A value of VI_TRUE indicates that a trigger will start capture.
Syntax:	ViStatus_VI_FUNC hpe650x_setSuspendedCaptureTask(instrumentID, mezz_num, suspended);		
hpe650x_**setTunerFrequency**			
Sets the value of the RF frequency for a given IF channel.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel for communication 0 = IF channel 1 1 = IF channel 2
	ViReal16	frequency	Sets the frequency value (2 MHz to 1 GHz, or 2 MHz to 3 GHz depending on the installed option).
Syntax:	ViStatus_VI_FUNC hpe650x_setTunerFrequency(*instrumentID, IFchan, frequency*);		
hpe650x_**setVolumeLevel**			
Sets the digital gain of the audio process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine where the audio level will be set. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DACnum	DAC for which the gain will be adjusted (0 through 9).
	ViReal64	volume_level	Value for volume level setting in dB: 0 = maximum volume level -50 = mute
Syntax:	ViStatus_VI_FUNC hpe650x_setVolumeLevel(*instrumentID, mezz_num, DACnum, volume_level*);		
Example:	See "To set up demodulation, turn on an audio channel, and set squelch" in Chapter 3.		
hpe650x_**startCapture**			
Starts capture process based on previously defined parameters such as length and DDC number, etc.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
Syntax:	ViStatus_VI_FUNC hpe650x_startCapture(instrumentID, mezz_num,);		
Example:	See "To set up digital I/Q data output" in Chapter 3.		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**startDynamicRangeOptimization**			
Enables dynamic range optimization on an IF channel.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	IFchan	IF channel selection for optimization 0 = IF channel 1 1 = IF channel 2
Syntax:	ViStatus_VI_FUNC hpe650x_startDynamicRangeOptimization(instrumentID, IFchan);		
hpe650x_**startRSSI**			
Start measuring signal strength data on the specified mezzanine and DDC.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine to be controlled. 0 = mezzanine 1 1= mezzanine 2
	ViInt32	DDC_num	DDC used to measure signal strength (0 through 4).
Syntax:	ViStatus_VI_FUNC hpe650x_startRSSI(*instrumentID, mezz_num, DDC_num)*; ret= hpe650x_startRSSI(instrumentID, mezz, DDC);		
Example:			
hpe650x_**stopCapture**			
Stops capture process. Must be run to end capture process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used for capture. 0 = mezzanine 1 1 = mezzanine 2
Syntax:	ViStatus_VI_FUNC hpe650x_stopCapture(instrumentID, mezz_num);		
hpe650x_**stopDynamicRangeOptimization**			
Disables dynamic range optimization on an IF channel.	ViSession	instrumentID	Index to the array maintained by the driver
	ViInt32	IFchan	IF channel selection for optimization 0 = IF channel 1 1 = IF channel 2
Syntax:	ViStatus_VI_FUNC hpe650x_stopDynamicRangeOptimization(instrumentID,IFchan);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**stopRSSI**			
Stops measuring signal strength data on the specified mezzanine and DDC.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine to be controlled. 0 = mezzanine 1 1= mezzanine 2
	ViInt32	DDC_num	DDC used to measure signal strength (0 through 4).
Syntax:	ViStatus_VI_FUNC hpe650x_stopRSSI(*instrumentID, mezz_num, DDC_num)*; See "To set up a channelized power measurement" in Chapter 3.		
Example:			
hpe650x_**turnOffAudioChannel**			
Deactivates the specified audio channel (DAC) on the mezzanine.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine that has the specified DDC. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DAC_num	DAC channel number (0 through 9).
Syntax:	ViStatus_VI_FUNC hpe650x_turnOffAudioChannel(*instrumentID, mezz_num, DAC_num*); ret= hpe650x_turnOffAudioChannel(instrumentID, mezz, DAC);		
Example:			
hpe650x_**turnOnAudioChannel**			
Activates the specified audio channel (DAC) on the mezzanine.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine that has the specified DDC. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	DDC_num	DDC on mezzanine (0 through 4) to use for demodulation.
	ViInt32	DAC_num	DAC channel number (0 through 9).
Syntax:	ViStatus_VI_FUNC hpe650x_turnOnAudioChannel(*instrumentID, mezz_num, DAC_num, DDC_num);*		
Example:	ViInt32 mezz=0, DDC=0, DAC=0; ret= hpe650x_turnOnAudioChannel(instrumentID, mezz, DAC, DDC);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
From FFT			
hpe650x_**getFFTFreqScale**			
Returns the FFT frequency scale as a vector.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine from which to return the FFT frequency scale. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	process_num	Process number. Maximum of four FFT processes can be run (0 through 3).
	ViInt32	scale_type	0 = return absolute frequencies 1 = return purely offset frequencies 2 = return offset from IF frequencies
	ViReal64	freq_scale	Returns vector with all frequency points.
Syntax:	ViStatus_VI_FUNC hpe650x_getFFTFreqScale(*instrumentID, mezz_num, process_num, scale_type, freq_scale*);		
hpe650x_**getFFTLength**			
Returns the length of the process' FFT length.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine containing the DDC. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	process_num	Process number. Maximum of four FFT processes can be run (0 through 3).
	ViPInt32	length	Pointer to the address of memory allocated for FFT length.
Syntax:	ViStatus_VI_FUNC hpe650x_getFFTLength(*instrumentID, mezz_num, process_num, &length*);		
hpe650x_**getFFTResolutionBW**			
Returns the resolution bandwidth used by the FFT.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine from which resolution bandwidth is returned. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	process_num	Process number. Maximum of four FFT processes can be run (0 through 3).
	ViPReal64	res_bw	Pointer to the address of memory allocated for the resolution bandwidth.
Syntax:	ViStatus_VI_FUNC hpe650x_getFFTResolutionBW(*instrumentID, mezz_num, process_num, &res_bw*);		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**getFFTTrace**			
Returns the amplitude trace data as a vector.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine from which amplitude will be returned. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	process_num	Process number. Maximum of four FFT processes can be run (0 through 3).
	ViReal64	buff[]	The array address passed by the user into which the amplitude trace data will be passed.

Note: Return code of -1 indicates FFT data is not ready yet. DSP is still processing.

Syntax: ViStatus_VI_FUNC hpe650x_getFFTTrace(*instrumentID, mezz_num, process_num, buff[]*);

Example:
```
ViSession instrumentID;
ViReal64 amp_data[4096];
ViInt32 mezz=0, pid=0, DDC=0, length=4096;
ViStatus ret;
.
.
.
ret= hpe650x_setFFTDDCNumber(instrumentID, mezz, pid, DDC);
ret= hpe650x_setFFTLength(instrumentID, mezz, pid, length);
ret= hpe650x_setReturnAllFFTData(instrumentID, mezz, pid, VI_FALSE);
ret= hpe650x_startFFT(instrumentID, mezz, pid);
while {ret= hpe650x_getFFTTrace(instrumentID, mezz, pid, amp_data);}
(ret <0);
```

hpe650x_**getFFTTraceLength**			
Returns the length of the trace data vector.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine from which trace data will be returned. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	process_num	Process number. Maximum of four FFT processes can be run (0 through 3).
	ViPInt32	tracelength	Pointer to trace length

Syntax: ViStatus_VI_FUNC hpe650x_getFFTTraceLength(*instrumentID, mezz_num, process_num, &tracelength*);

Example: See "To set up an FFT measurement" in Chapter 3.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setFFTAverages**			
Sets the number of trace averages performed by the DSP code.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine on which to set number of averages. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	process_num	Process number. Maximum of four FFT processes can be run (0 through 3).
	ViInt32	numOfAverages	Specified number of averages (1 to 200).
Syntax:	ViStatus_VI_FUNC hpe650x_setFFTAverages(*instrumentID, mezz_num, process_num, numOfAverages*);		
Example:	See "To set up an FFT measurement" in Chapter 3.		
hpe650x_**setFFTDDCNumber**			
Assigns a specified mezzanine and DDC to an FFT process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine containing the DDC. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	process_num	Process number. Maximum of four FFT processes can be run (0 through 3).
	ViInt32	DDC_num	DDC on mezzanine (0 through 4).
Note:	There is a special case of using DDC #5 which indicates that the FFT will return full span data.		
Syntax:	ViStatus_VI_FUNC hpe650x_setFFTDDCNumber(*instrumentID, mezz_num, process_num, DDC_num*);		
Example:	See "To set up an FFT measurement" in Chapter 3.		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setFFTLength**			
Establishes the FFT length for the specified process number.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine containing the DSP that maintains this process number. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	process_num	Process number. Maximum of four FFT processes can be run (0 through 3).
	ViInt32	length	*Requested* length of the FFT: minimum = 64 maximum = 4096
Note:	Because the actual FFT length the driver passes to the DSP for processing can differ from the selected value, you should also call hpe650x_getFFTLength to determine the actual FFT length passed to the DSP.		
Syntax:	ViStatus_VI_FUNC hpe650x_setFFTLength(*instrumentID, mezz_num, process_num, length*);		
	See "To set up an FFT measurement" in Chapter 3.		
Example:			
hpe650x_**setFFTWindowType**			
Sets the type of filter window used by the FFT process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine on which to set number of averages. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	process_num	Process number. Maximum of four FFT processes can be run (0 through 3).
	ViInt32	window_type	Specified filter window. 0 = Hanning 1 = Flat Top 2 = Rectangular
Syntax:	ViStatus_VI_FUNC hpe650x_setFFTWindowType(*instrumentID, mezz_num, process_num, window_type*);		

Example:	enum wintypes {Hanning=0, Flat, Rectangular};
	ViInt32 mezz=0, pid=0,
	ViSession InstrumentID;
	wintypes usewin= Hanning;
	ret= hpe650x_setFFTWindowType(instrumentID, mezz, pid, usewin);
	See also "To set up an FFT measurement" in Chapter 3.		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command/Action	Data Type	Parameters	
hpe650x_**setReturnAllFFTData**			
Indicates whether all data about the FFT should be returned in the trace, or just frequency amplitude data.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine from which trace data will be returned. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	process_num	Process number. Maximum of four FFT processes can be run (0 through 3).
Note:	All data including the dither signal is returned.		
Syntax:	ViStatus_VI_FUNC hpe650x_setReturnAllFFTData(*instrumentID, mezz_num, process_num,returnAllData*);		
Example:	See "To set up an FFT measurement" in Chapter 3.		
hpe650x_**startFFT**			
Starts the FFT for the specified process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used to perform FFT. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	process_num	Process number. Maximum of four FFT processes can be run (0 through 3).
Syntax:	ViStatus_VI_FUNC hpe650x_startFFT(*instrumentID, mezz_num, process_num*);		
Example:	See "To set up an FFT measurement" in Chapter 3.		
hpe650x_**stopFFT**			
Stops the FFT for the specified process.	ViSession	instrumentID	Index to the array maintained by the driver.
	ViInt32	mezz_num	Mezzanine used to perform FFT. 0 = mezzanine 1 1 = mezzanine 2
	ViInt32	process_num	Process number. Maximum of four FFT processes can be run (0 through 3).
Syntax:	ViStatus_VI_FUNC hpe650x_stopFFT(*instrumentID, mezz_num, process_num*);		
Example:	See "To set up an FFT measurement" in Chapter 3.		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Index

Numerics

0 samples, capturing, 3-56
10 MHz reference, 4-6
10 MHz reference, turning off, 3-37
16 MHz Span, 3-21
16 MHz Stare, 3-18
1st LO, 4-7
2nd LO, 4-6
4 MByte Data RAM, 5-10

A

accessories
  E6401A, 1-29
  E6401A option 001, 1-29
  E6402A, 1-29
  E6402A option 002, 1-29
  E6403A, 1-29
  E6404A, 1-30
  E6404A options 031, 040, 1-30
  static-safe, 2-2
ADC Output Data Rates, 5-7
ADC Sampling Rate, 5-7
adress switches, setting, 2-5
AFC Active, 3-19
AFC Tracking Range, 5-12
AFC, activating, 3-38
AGC, theory of, 4-13
ALC Attack Rate, 3-13
ALC Decay Rate, 3-13
ALC Range, 5-12
ALC Response Time, 5-12
AM Sensitivity, 5-6
Analog Filter, 3-22
analog filters, theory of, 4-14
Analog Gain Control, 5-7
Analog IF Input Filter Bandwidths, 5-7
analog outputs, 3-7
antialiasing filters, 3-3
application functions, 6-5
arming DSP, 3-49
attack/decay, autoranging, 4-14
Attenuation, 3-23
audio breakout box, 3-7
audio channel, turning on, 3-41
audio connector, 3-8
Audio Controls (Mezz 1), 3-20
Audio Output Connector Type, 5-13
audio trigger, 3-8
Autorange, 3-23
autoranging attack/decay, 4-14
Autoranging Gain Response Time, 5-7autoranging, benefits of, 4-14
autoranging, locking, 3-38
autoranging, theory of, 4-13

B

Bandwidth of Digital IQ Outputs, 5-10
BFO Control, 3-24
Blocking, 5-5
breakout box, 3-7
Bufferred Sync Rx, 2-9
bus timeout, 2-14

C

cable, MXI controller, 2-10
Calibration Interval, 5-18
capabilities, receiver, 3-2
capability classes, 6-3
capture data, 3-57
Chan, 3-17
channelization, 3-5
channelized power, example of, 3-39
characteristics, definition of, 5-1
characteristics, windowing, 4-21
check operation, 2-19
clocks, distributing, 3-48
close instrument session, 3-28
commands
  , 6-16, 6-18
  hpe650x_abortDataCollection, 6-27
  hpe650x_activateAFC, 6-27
  hpe650x_activateAutoranging, 6-27
  hpe650x_armDDCsForSynchronization, 6-27
  hpe650x_armDSPForDataCollection, 6-28
  hpe650x_autoConfigure, 6-11
  hpe650x_checkDDCsAvailable, 6-28
  hpe650x_clearCaptureDDCNum, 6-28
  hpe650x_close, 6-11
  hpe650x_deactivateAFC, 6-29
  hpe650x_disableAFC, 6-29
  hpe650x_disableVCO, 6-29
  hpe650x_enableAFC, 6-29
  hpe650x_error_query, 6-11
  hpe650x_gangTuneDDC, 6-30
  hpe650x_getActualScarchStopFreq, 6-21
  hpe650x_getADCClippingIndicator, 6-30
  hpe650x_getAutorangeState, 6-30
  hpe650x_getCaptureCollectInSRAM, 6-31
  hpe650x_getCaptureDataDDCNum, 6-31
  hpe650x_getCaptureDataFormat, 6-31

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

hpe650x_getCaptureDataOutput, 6-32
hpe650x_getCaptureDigitalIQData, 6-32
hpe650x_getCaptureFullRateADCData, 6-33
hpe650x_getCaptureTrigger, 6-33, 6-37
hpe650x_getCurrentDemodType, 6-34
hpe650x_getDataCorrectionValue, 6-33
hpe650x_getDDCFrequency, 6-34
hpe650x_getDigitalIFBandwidth, 6-35
hpe650x_getFFTBinWidth, 6-21
hpe650x_getFFTFreqScale, 6-49
hpe650x_getFFTLength, 6-49
hpe650x_getFFTResolutionBW, 6-49
hpe650x_getFFTTrace, 6-50
hpe650x_getFFTTraceLength, 6-50
hpe650x_getIF3dBBandwidth, 6-17
hpe650x_getIFAttenuator, 6-12
hpe650x_getIFChannelForDDC, 6-35
hpe650x_getMultipleTriggerAction, 6-35
hpe650x_getNumberOfActiveModules, 6-21
hpe650x_getNumberOfSamplesToCapture, 6-36
hpe650x_getRSSIvalue, 6-36
hpe650x_getSearchDecimationFactor, 6-21
hpe650x_getSearchFFTLength, 6-22
hpe650x_getSearchIndexFrequency, 6-22
hpe650x_getSearchResolutionBW, 6-22
hpe650x_getSearchTrace, 6-23
hpe650x_getSearchTraceBlock, 6-23
hpe650x_getSearchTraceLength, 6-23
hpe650x_getSuspendedCaptureTask, 6-36
hpe650x_getTunerFrequency, 6-37
hpe650x_getTunerTemperature, 6-12
hpe650x_hardSystemReset, 6-37
hpe650x_init, 6-13
hpe650x_initIFChannel, 6-17
hpe650x_prearmDSPForDataCollection, 6-37
hpe650x_readOptionString, 6-13
hpe650x_reset, 6-14
hpe650x_revision_query, 6-11, 6-14
hpe650x_sanityCheck, 6-14
hpe650x_selectBackplaneFs, 6-37
hpe650x_selectTuner10MHzReference, 6-18
hpe650x_self_test, 6-14
hpe650x_setAbsAmplitudeCalSignalCorr, 6-15
hpe650x_setAbsoluteAmplitude, 6-15
hpe650x_setALCRate, 6-38
hpe650x_setAnalogFilter, 6-16
hpe650x_setAutorangeLock, 6-16
hpe650x_setAutorangeState, 6-38
hpe650x_setCaptureCollectInSRAM, 6-38
hpe650x_setCaptureDataDDCNum, 6-39
hpe650x_setCaptureDataFormat, 6-39
hpe650x_setCaptureDataOutput, 6-39
hpe650x_setCaptureTrigger, 6-40
hpe650x_setDDCFrequency, 6-40
hpe650x_setDeemphasis, 6-40
hpe650x_setDefaultIFConfig, 6-18
hpe650x_setDemodType, 6-41
hpe650x_setDigitalIFBandwidth, 6-42
hpe650x_setDitherState, 6-16
hpe650x_setDROAttackTime, 6-43
hpe650x_setDRODecayTime, 6-43
hpe650x_setFFTAverages, 6-51
hpe650x_setFFTDDCNumber, 6-51
hpe650x_setFFTLength, 6-52
hpe650x_setFFTWindowType, 6-52
hpe650x_setIFGain, 6-43
hpe650x_setInterMezzanineAudio, 6-44
hpe650x_setMasterIF, 6-44
hpe650x_setMezzanineDataSelectMode, 6-19
hpe650x_setMonitoringMode, 6-19
hpe650x_setMultipleTriggerAction, 6-44
hpe650x_setNumberOfSamplesToCapture, 6-44
hpe650x_setReturnAllFFTData, 6-53
hpe650x_setRSSIMeasTime, 6-45
hpe650x_setSearchMode, 6-19
hpe650x_setSearchOutputTraceLength, 6-24
hpe650x_setSearchResBWParameters, 6-24
hpe650x_setSearchResolutionBW, 6-25
hpe650x_setSearchSpan, 6-25
hpe650x_setSearchType, 6-26
hpe650x_setServantIF, 6-45
hpe650x_setSquelchLevel, 6-45
hpe650x_setSquelchState, 6-45
hpe650x_setSuspendedCaptureTask, 6-46
hpe650x_setTunerAttenuation, 6-20
hpe650x_setTunerFrequency, 6-46
hpe650x_setVolumeLevel, 6-46
hpe650x_startCapture, 6-46, 6-47
hpe650x_startDynamicRangeOptimization, 6-47
hpe650x_startFFT, 6-53
hpe650x_startRSSI, 6-47
hpe650x_startSearch, 6-26
hpe650x_stopDynamicRangeOptimization, 6-47
hpe650x_stopFFT, 6-53
hpe650x_stopRSSI, 6-48
hpe650x_stopSearch, 6-26
hpe650x_turnOffAudioChannel, 6-48
hpe650x_turnOnAudioChannel, 6-48
common commands, 6-10
configuration
E6501A, 1-23
E6501A option 003, 1-24
E6502A, 1-25
E6502A option 003, 1-26
E6503A, 1-27
E6503A option 003, 1-28
configure function, 6-4
COR, 5-12

D

data collection, 3-57
common functions, 3-60
scenario 1, 3-63
scenario 10, 3-115
scenario 11, 3-120
scenario 12, 3-124
scenario 13, 3-129
scenario 14, 3-134

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

scenario 15, 3-139
scenario 2, 3-68
scenario 3, 3-75
scenario 4, 3-81
scenario 5, 3-88
scenario 6, 3-94
scenario 7, 3-99
scenario 8, 3-103
scenario 9, 3-109
data format, captured, 3-51
data select mode, 3-3
dB/div, 3-22
DDC, 4-25
DDC Decimated Sample Rate, 5-10
DDC Freq (MHz), 3-19
DDC Resolution, 5-9
DDC synchronization, 3-47
DDC Tuning Range, 5-9
DDCs, synchronizing, 3-48
declaration of conformity, 5-22
De-emphasis, 3-19
default settings, 3-28
Demo.exe, 2-14
Demod Type, 3-19
demodulation
capabilities, 3-5
dependencies, 3-5
example of, 3-41
number of simultaneous, 3-5
description
E6501A, 1-2
E6502A, 1-2
E6503A, 1-3
diagrams, programmers block, 3-25
Digital Bandwidth Shape Factor, 5-9
digital drop receiver, 3-3
digital IF bandwidth, 3-6
Digital IF Bandwidths, 5-9
digital IF bandwidths, 3-3
Digital Output Interface, 5-10
digital outputs, 3-9
Dimensions, 5-16
distributing clocks, 3-48
driver revision, 3-28
DRO attack/decay, 4-16
DRO, RAM, DSP, 4-16
DRO, search mode, 4-17
DSP, 3-4
DSP loading, 3-27
DSP, arming, 3-49
DSP-based Demodulation, 5-4
DSP-based Detection Modes, 5-12
Dual Input IF Channel Isolation, 5-7
dynamic range optimization, 3-7
Dynamic Range Optimization Response Time, 5-9
dynamic range optimization, setting, 3-38
dynamic range optimization, theory of, 4-15

E

E6401A, detailed description, 4-4
E6402A option 002, 4-7
E6402A, detailed description, 4-6
E6403A, detailed description, 4-9
E6404A, detailed description, 4-11
E6501A configuration, 1-23
E6501A option 003 configuration, 1-24
E6502A configuration, 1-25
E6502A option 003 configuration, 1-26
E6503A configuration, 1-27
E6503A option 003 configuration, 1-28
ECL, 2-8
EMC, 5-19
Enable DAC, 3-19
Enable IF Reference Out, 3-14
equipment, test, 2-19
ESD (electrostatic discharge), 2-2
examples
10 MHz reference, turning off, 3-37
AFC, activating, 3-38
audio channel, turning on, 3-41
channelized power, 3-39
demodulation, 3-41
dynamic range optimization, setting up, 3-38
FFT measurement, 3-33
IF bandpass filter, setting, 3-34
IF gain, setting, 3-35
lock autoranging, 3-38
mezzanine data select mode, setting, 3-37
monitor process, 3-40
multi-threading, 3-42
search mode RBW, setting, 3-36
search process, 3-31
span, setting in monitor mode, 3-36
squelch, setting, 3-41
tuner attenuation, setting, 3-36
tuner frequency, 3-34
Exit, 3-12
External, 3-13
External Cable and Audio Breakout Box, 5-13

F

FFT commands, 6-10
FFT measurements
algorithm, 4-19
background, 4-18
example of, 3-33
IF pan windows, RBW, 4-22
maximum FFT length, 3-26
maximum FFT processes, 3-26
monitor mode, 3-3
process, 4-19
properties, 4-18
resolution bandwidth range, 4-21
resolution bandwidths, 4-22
stepped, 4-19

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

windowing, 4-20
File menu, 3-12
files, list of installation, 2-14
Filter Frequency Range, 5-3
filters, analog, 4-14
flat top, 4-21
FM De-emphasis, 5-12
FM Sensitivity, 5-6
format, captured data, 3-51
frequency range, 5-2
frequency translations, 4-10
Front Panel Connectors, 5-14
front panel features
E6401A, 1-7
E6402 A option 002, 1-11
E6402A, 1-9
E6403A, 1-13
E6404A, 1-15
E6404A options 022, 040, 1-20
E6404A options 031, 040, 1-17
full rate digitized data, 3-9
Full Span, 3-22
full span spectral display, 3-3
Full Span Start Freq, 3-21
Full Span Stop Freq, 3-21

G

Gain, 3-23
gain control, 3-7
gain settings, IF, 3-35
gain, processing, 4-15
group 0 commands, 3-25
group 1 commands, 3-25
group 2 commands, 3-25
group 3 commands, 3-25
group 4 commands, 3-25
group 5 commands, 3-25
group numbers, 3-25

H

Hanning, 4-21
hardware configuration, synchronization, 3-46
hardware installation, 2-9
hardware trigger, 3-47
Harmonic Distortion, 5-8
hpe650x.dll, 2-14
hpe650x.exp, 2-14
hpe650x.h, 2-14
hpe650x.hlp, 2-14
hpe650x.lib, 2-14
hpe650x_abortDataCollection, 6-27
hpe650x_activateAFC, 6-27
hpe650x_activateAutoranging, 6-27
hpe650x_armDDCsForSynchronization, 6-27
hpe650x_armDSPForDataCollection, 6-28
hpe650x_autoConfigure, 6-11
hpe650x_checkDDCsAvailable, 6-28
hpe650x_clearCaptureDDCNum, 6-28
hpe650x_close, 6-11
hpe650x_deactivateAFC, 6-29
hpe650x_disableAFC, 6-29
hpe650x_disableVCO, 6-29
hpe650x_enableAFC, 6-29
hpe650x_error_query, 6-11
hpe650x_gangTuneDDC, 6-30
hpe650x_getActualSearchStopFreq, 6-21
hpe650x_getADCClippingIndicator, 6-30
hpe650x_getAutorangeState, 6-30
hpe650x_getCaptureCollectInSRAM, 6-31
hpe650x_getCaptureDataDDCNum, 6-31
hpe650x_getCaptureDataFormat, 6-31
hpe650x_getCaptureDataOutput, 6-32
hpe650x_getCaptureDigitalIQData, 6-32
hpe650x_getCaptureFullRateADCData, 6-33
hpe650x_getCaptureTrigger, 6-33, 6-37
hpe650x_getCurrentDemodType, 6-34
hpe650x_getDataCorrectionValue, 6-33
hpe650x_getDDCFrequency, 6-34
hpe650x_getDigitalIFBandwidth, 6-35
hpe650x_getFFTBinWidth, 6-21
hpe650x_getFFTFreqScale, 6-49
hpe650x_getFFTLength, 6-49
hpe650x_getFFTResolutionBW, 6-49
hpe650x_getFFTTrace, 6-50
hpe650x_getFFTTraceLength, 6-50
hpe650x_getIF3dBBandwidth, 6-17
hpe650x_getIFAttenuator, 6-12
hpe650x_getIFChannelForDDC, 6-35
hpe650x_getMultipleTriggerAction, 6-35
hpe650x_getNumberOfActiveModules, 6-21
hpe650x_getNumberOfSamplesToCapture, 6-36
hpe650x_getRSSIvalue, 6-36
hpe650x_getSearchDecimationFactor, 6-21
hpe650x_getSearchFFTLength, 6-22
hpe650x_getSearchIndexFrequency, 6-22
hpe650x_getSearchResolutionBW, 6-22
hpe650x_getSearchTrace, 6-23
hpe650x_getSearchTraceBlock, 6-23
hpe650x_getSearchTraceLength, 6-23
hpe650x_getSuspendedCaptureTask, 6-36
hpe650x_getTunerFrequency, 6-37
hpe650x_getTunerTemperature, 6-12
hpe650x_hardSystemReset, 6-37
hpe650x_init, 6-13
hpe650x_initIFChannel, 6-17
hpe650x_prearmDSPForDataCollection, 6-37
hpe650x_readOptionString, 6-13
hpe650x_reset, 6-14
hpe650x_revision_query, 6-11, 6-14
hpe650x_sanityCheck, 6-14
hpe650x_selectBackplaneFs, 6-37
hpe650x_selectTuner10MHzReference, 6-18
hpe650x_self_test, 6-14
hpe650x_setAbsAmplitudeCalSignalCorr, 6-15
hpe650x_setAbsAmplitudeTempComp, 6-15
hpe650x_setAbsoluteAmplitude, 6-15

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

hpe650x_setALCRate, 6-38
hpe650x_setAnalogFilter, 6-16
hpe650x_setAutorangeLock, 6-16
hpe650x_setAutorangeState, 6-38
hpe650x_setCaptureCollectInSRAM, 6-38
hpe650x_setCaptureDataDDCNum, 6-39
hpe650x_setCaptureDataFormat, 6-39
hpe650x_setCaptureDataOutput, 6-39
hpe650x_setCaptureTrigger, 6-40
hpe650x_setDDCFrequency, 6-40
hpe650x_setDeemphasis, 6-40
hpe650x_setDefaultIFConfig, 6-18
hpe650x_setDemodType, 6-41
hpe650x_setDigitalIFBandwidth, 6-42
hpe650x_setDitherState, 6-16
hpe650x_setDROAttackTime, 6-43
hpe650x_setDRODecayTime, 6-43
hpe650x_setFFTAverages, 6-51
hpe650x_setFFTDDCNumber, 6-51
hpe650x_setFFTLength, 6-52
hpe650x_setFFTWindowType, 6-52
hpe650x_setIF10MHzReferenceOut, 6-18
hpe650x_setIFGain, 6-43
hpe650x_setInterMezzanineAudio, 6-44
hpe650x_setMasterIF, 6-44
hpe650x_setMezzanineDataSelectMode, 6-19
hpe650x_setMonitoringMode, 6-19
hpe650x_setMultipleTriggerAction, 6-44
hpe650x_setNumberOfSamplesToCapture, 6-44
hpe650x_setReturnAllFFTData, 6-53
hpe650x_setRSSIMeasTime, 6-45
hpe650x_setSearchMode, 6-19
hpe650x_setSearchOutputTraceLength, 6-24
hpe650x_setSearchResBWParameters, 6-24
hpe650x_setSearchResolutionBW, 6-25
hpe650x_setSearchSpan, 6-25
hpe650x_setSearchType, 6-26
hpe650x_setServantIF, 6-45
hpe650x_setSquelchLevel, 6-45
hpe650x_setSquelchState, 6-45
hpe650x_setSuspendedCaptureTask, 6-46
hpe650x_setTunerAttenuation, 6-20
hpe650x_setTunerFrequency, 6-46
hpe650x_setVolumeLevel, 6-46
hpe650x_startCapture, 6-46, 6-47
hpe650x_startDynamicRangeOptimization, 6-47
hpe650x_startFFT, 6-53
hpe650x_startRSSI, 6-47
hpe650x_startSearch, 6-26
hpe650x_stopDynamicRangeOptimization, 6-47
hpe650x_stopFFT, 6-53
hpe650x_stopRSSI, 6-48
hpe650x_stopSearch, 6-26
hpe650x_turnOffAudioChannel, 6-48
hpe650x_turnOnAudioChannel, 6-48
Humidity, 5-19

I

I/Q data, 3-9
IF bandpass filter, setting, 3-34
IF bandwidth, 3-19
IF Channel Controls, 3-22
IF gain settings, 3-35
IF Rejection, 5-5
Image Rejection, 5-5
Input Range Settings, 5-7
Input VSWR, 5-4
installation
  hardware, 2-9
  MXI cable, 2-10
  software, 2-14
installation files, 2-14
Instrument Preset, 3-12
Inter mezzanine audio, 3-19
Intermodulation, 5-5
Internal, 3-13
Internal Timebase Adjustment Interval, 5-18
Internally Generated Spurious, 5-5
Internally Generated Spurious Responses, 5-8

L

Layout menu, 3-14
length, maximum FFT, 3-26
Link Port Connector Type, 5-10
Link Port Output Data Rate, 5-10
link port pin orientation, 3-9
LO Emissions, 5-5
loading weight, 3-28
local bus, 2-8
local bus switch, settings for synchronization, 3-47
lockout key, 2-8
logical address switches, setting, 2-5

M

mainframes, VXI, 1-22
Marker, 3-18
master IFP, 3-47
Maximum Audio Output, 5-13
Maximum Input without Damage, 5-4
Maximum Realtime Demodulated Bandwidth, 5-12
measure capability class, 6-4
memory pointers, 3-31
Mezz. 1 Controls, 3-18
Mezzanine, 4-25
Mezzanine 1, 2, 3-18
mezzanine assembly, theory of, 4-11
mezzanine data select mode
  general description, 3-3
  setting, 3-37
Mezzanine menu, 3-14
mode 1, 3-25
mode 1, 2, 3, 4, 3-13

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

mode 2, 3-25
mode 3, 3-25
mode 4, 3-25
Module Size, 5-20
monitor commands, 6-10
monitor mode, 3-2
monitor process, setting up and starting, 3-40
multiple IFPs, synchronization, 3-46
multi-threading, 3-42
MXI cable, installing, 2-10

N

New Spectral Display, 3-16
Noise Figure, 5-5
Number of Digital Downconverters, 5-9
Number of Digital IQ Outputs, 5-10
Number of Simultaneous Channels, 5-13
Number of Simultaneous Demodulated Signals per Mezzanine, 5-12
Number of Simultaneous Digital I/Q Outputs, 5-10
Number of simultaneous I/Q outputs, 5-11
Nyquist, 4-18

O

OCXO, 4-6
Off/On, 3-17, 3-22
open instrument session, 3-28
Open menu, 3-16
Operating Temperature, 5-19
operation, checking, 2-19
options, adding functionality, 1-4
Output Bandwidth, 5-10, 5-12
Output Interface, 5-10
Overall Gain Control Range, 5-7
Overall Receiver Sweep Speed, 5-2

P

Phase Noise, 5-5
pin numbers, link port, 3-9
plug and play commands, 6-10
pointers, 6-10
pointers, memory, 3-31
Power Requirements, 5-17
Preselector Band, 5-3
preselector bands, 4-3
processes, maximum FFT, 3-26
processing gain, 4-15
programmer block diagram, 3-25

R

read function, 6-4
Reciprocal Mixing, 5-5
requirements, system, 2-13
Res. BW, 3-21
Res. BW Averages dB/div, 3-17
Reset DDCs, 3-19
resource manager, running, 2-15
return values, 3-31, 6-6
return values, special cases, 6-9
revision, driver, 3-28
RF Input Attenuation, 5-4
RF Input Connector, 5-4
RF Input Impedance, 5-4
route class, 6-5
RSSI for Mezzanine 1, 3-20
RSSI Meas. Time, 3-13

S

samples, capturing 0, 3-56
Save Current Layout, 3-12
search commands, 6-10
Search Controls for Mezzanine 1, 3-20
Search Display for Mezzanine 1, 3-22
search mode, 3-4
search mode RBW, setting, 3-36
search process, example of, 3-31
search rate, 3-4
Sensitivity, 5-5
sensitivity, using FFTs, 4-22
serial loading, 3-27
servant module, 3-47
session, opening/closing, 3-28
setting address switches, 2-5
Settings menu, 3-12
settings, default, 3-28
shape factor, 4-21
Shock, 5-19
shortcut menu, 3-17
Signal-to-Noise Ratio, 5-8
Simultaneous Demods, 5-4
Slots Used, 5-20
software configuration, sychronization, 3-47
software trigger, 3-47
software, installing, 2-14
sole module, DDC synchronization, 3-47
Span, 3-21
span, setting in monitor mode, 3-36
specifications
  4 MByte Data RAM, 5-10
  ADC Output Data Rates, 5-7
  ADC Sampling Rate, 5-7
  AFC Tracking Range, 5-12
  ALC Range, 5-12
  ALC Response Time, 5-12
  AM Sensitivity, 5-6
  Analog Gain Control, 5-7
  Analog IF Input Filter Bandwidths, 5-7
  Audio Output Connector Type, 5-13
  Autoranging Gain Response Time, 5-7
  Bandwidth of Digital IQ Outputs, 5-10

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Blocking, 5-5
Calibration Interval, 5-18
COR, 5-12
DDC Decimated Sample Rate, 5-10
DDC Resolution, 5-9
DDC Tuning Range, 5-9
Declaration of Conformity, 5-22
definition of, 5-1
Digital Bandwidth Shape Factor, 5-9
Digital IF Bandwidths, 5-9
Digital Output Interface, 5-10
Dimensions, 5-16
DSP-based Demodulation, 5-4
DSP-based Detection Modes, 5-12
Dual Input IF Channel Isolation, 5-7
Dynamic Range Optimization Response Time, 5-9
EMC, 5-19
External Cable and Audio Breakout Box, 5-13
Filter Frequency Range, 5-3
FM De-emphasis, 5-12
FM Sensitivity, 5-6
frequency range, 5-2
Front Panel Connectors, 5-14
Harmonic Distortion, 5-8
Humidity, 5-19
IF Rejection, 5-5
Image Rejection, 5-5
Input Range Settings, 5-7
Input VSWR, 5-4
Intermodulation, 5-5
Internal Timebase Adjustment Interval, 5-18
Internally Generated Spurious, 5-5
Internally Generated Spurious Responses, 5-8
Link Port Connector Type, 5-10
Link Port Output Data Rate, 5-10
LO Emissions, 5-5
Maximum Audio Output, 5-13
Maximum Input without Damage, 5-4
Maximum Realtime Demodulated Bandwidth, 5-12
Module Size, 5-20
Noise Figure, 5-5
Number of Digital Downconverters, 5-9
Number of Digital IQ Outputs, 5-10
Number of Simultaneous Channels, 5-13
Number of Simultaneous Demodulated Signals per Mezzanine, 5-12
Number of Simultaneous Digital I/Q Outputs, 5-10
Operating Temperature, 5-19
Output Bandwidth, 5-10, 5-12
Output Interface, 5-10
Overall Gain Control Range, 5-7
Phase Noise, 5-5
Power Requirements, 5-17
Preselector Band, 5-3
Reciprocal Mixing, 5-5
RF Input Attenuation, 5-4
RF Input Connector, 5-4
RF Input Impedance, 5-4
Sensitivity, 5-5
Shock, 5-19
Signal-to-Noise Ratio, 5-8
Simultaneous Demods, 5-4
Slots Used, 5-20
Spurious Responses, 5-8
Squelch Range, 5-12
Storage Temperature, 5-19
Synthesizer Tuning Speed, 5-2
Trigger Input, 5-13
Trigger Output, 5-13
tuning resolution, 5-2
Vibration, 5-19
VXI Control, 5-20
VXI Interface, 5-20
Warranty, 5-18
Weight, 5-15
Spurious Responses, 5-8
Squelch Range, 5-12
squelch, setting, 3-41
Start Freq, 3-21
static-safe accessories, 2-2
Step Size, 3-23
Stop Freq, 3-21
Storage Temperature, 5-19
switch settings, local bus, 2-9
Sync Tx, 2-9
synchronization, multiple IFPs, 3-46
Synthesizer Tuning Speed, 5-2
system requirements, 2-13

T

Terminal Sync Rx, 2-9
test equipment, 2-19
test operation, 2-19
timeout, configuring VXI bus, 2-15
top logo base, 2-8
Trigger Input, 5-13
Trigger Output, 5-13
trigger, audio, 3-8
trigger, software/hardware, 3-47
TTL, 2-8
Tune IF Chan 1, 3-21
Tune Select, 3-19
tuner attenuation, setting, 3-36
Tuner Controls, 3-23
Tuner Freq, 3-23
tuner frequency, example of, 3-34
Tuning Accuracy, 5-2
tuning resolution, 5-2
Turbo Speed, 3-18
typical, definition of, 5-1

U

UnBufferred Sync Rx, 2-9
uniform, 4-21
UNIX operation, 2-13

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

V

VCO, 4-6
Vibration, 5-19
View, 3-21
virtual front panel
  16 MHz Span, 3-21
  16 MHz Stare, 3-18
  AFC Active, 3-19
  ALC Attack Rate, 3-13
  ALC Decay Rate, 3-13
  Analog Filter, 3-22
  Attenuation, 3-23
  Audio Controls (Mezz 1), 3-20
  Autorange, 3-23
  BFO Control, 3-24
  Chan, 3-17
  dB/div, 3-22
  DDC Freq (MHz), 3-19
  De-emphasis, 3-19
  Demod Type, 3-19
  Enable DAC, 3-19
  Enable IF Reference Out, 3-14
  Exit, 3-12
  External, 3-13
  File menu, 3-12
  Full Span, 3-22
  Full Span Start Freq, 3-21
  Full Span Stop Freq, 3-21
  Gain, 3-23
  IF bandwidth, 3-19
  IF Channel Controls, 3-22
  Instrument Preset, 3-12
  Inter mezzanine audio, 3-19
  Internal, 3-13
  Layout menu, 3-14
  Marker, 3-18
  Mezz. 1 Controls, 3-18
  Mezzanine 1, 2, 3-18
  Mezzanine menu, 3-14
  mode 1, 2, 3, 4, 3-13
  New Spectral Display, 3-16
  Off/On, 3-17, 3-22
  Open menu, 3-16
  Res. BW, 3-21
  Res. BW Averages dB/div, 3-17
  Reset DDCs, 3-19
  RSSI for Mezzanine 1, 3-20
  RSSI Meas. Time, 3-13
  Save Current Layout, 3-12
  Search Controls for Mezzanine 1, 3-20
  Search Display for Mezzanine 1, 3-22
  search window, 3-4
  Settings menu, 3-12
  shortcut menu, 3-17
  Span, 3-21
  Start Freq, 3-21
  starting, 2-15
  Step Size, 3-23
  Stop Freq, 3-21
  Tune IF Chan 1, 3-21
  Tune Select, 3-19
  Tuner Controls, 3-23
  Tuner Freq, 3-23
  Turbo Speed, 3-18
  View, 3-21
  Window menu, 3-24
  Window Type, 3-18
VISA library, 6-2
VXI bus timeout, 2-14
VXI Control, 5-20
VXI Interface, 5-20
VXI mainframes
  option 006, 1-22
  option 013, 1-22

W

Warranty, 5-18
window characteristics, 4-21
Window menu, 3-24
Window Type, 3-18
windowing, 4-20

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP