

**Operators Manual
For**

Pulse Instruments

**PI-3105 Multi-Channel
Data Acquisition System
and
PI-Controller Software**

November 2015

Copyright © Pulse Instruments 2003-2015
1234 Francisco Street
Torrance, California 90502
310-515-5330 (voice), 310-515-0068 (fax)
<mailto:support@pulseinstruments.com>
<http://www.pulseinstruments.com/>

CONTENTS

Contents.....	i
List of Figures	vii
List of Tables.....	ix
1. Introduction	1
1.1. How to Read This Manual	2
1.2. Known Issues, PI-DATS 2.708	3
1.2.1. PI-Controller and dacq.dll (common issues)	3
1.2.2. Hardware.....	3
2. Hardware	5
2.1. CPU and Chassis	5
2.2. Front Panel Connections	5
2.2.1 CLK Connectors	5
2.2.2 Control Cable	6
2.2.3 Timing Cables	7
2.2.4 PI-3100 Acquisition Interface Module, Front Panel.....	7
2.2.5 PI-3100 Acquisition Interface Module, Rear Panel	8
2.2.6 PI-41000 Digital Acquisition Card	11
2.2.6.1 Data/Clock Connectors	12
2.2.6.2 Acquisition Arm Connector	12
3. Software.....	15
3.1. Channel Assignments.....	15
3.2. PI-Controller	15
3.3. Hardware Configuration.....	15
3.3.1. Simulated Components.....	16
3.3.2. Connecting Components.....	17
3.3.2.1. Timing Channels	17
3.3.2.2. Data Channels	18
3.3.2.3. Master/Slave Remote Arm.....	19
3.3.2.4. Confirming Simulated Components/Hardware Device Not Responding.....	20
3.4. Setting Up Test Plans.....	22
3.5. TIMING mnemonic	22
3.5.1. Overview	22
3.5.2. Convert and CDS strobe timing	22
3.5.3. Frame Sync and Line Sync	23
3.6. AIM mnemonic.....	24
3.6.1. Overview	24
3.6.2. Gain.....	25
3.6.3. Offset.....	25
3.6.4. CDS Mode.....	26
3.6.5. Filter	26
3.6.6. Monitor	26
3.7. DEFINE mnemonic.....	27
3.7.1. Overview	27
3.7.1.1. Multiple Configurations	28
3.7.1.2. Multiple Acquisition Cards.....	28
3.7.2. Master/Slave setup	28
3.7.3. FPA size and DACQ card selection	30
3.7.4. FPA Orientation.....	31
3.7.5. Maximum FPA Size.....	31
3.7.6. Channel Parameters	31
3.7.7. FPA Definition Constraints	32
3.8. Channel Definition Examples	32

3.8.1. Contiguous FPA Definitions	33
3.8.1.1. Columnar Device, Vertically Clocked from Lower Left Corner.....	33
3.8.1.2. Columnar Device, Horizontally Clocked from Lower Left Corner.....	34
3.8.2. Interleaved FPA Definitions.....	35
3.8.2.1. Alternating Lines	35
3.8.2.2. Interleaved Grid	37
3.8.3. Stitching Across Multiple Data Acquisition Cards	38
3.8.4. Acquiring Across Multiple Masters	40
3.9. DACQ mnemonic.....	44
3.9.1. General Controls	45
3.9.1.1. Start Acq/Stop Acq button.....	45
3.9.1.2. PI-Plot button	46
3.9.1.3. Save File button.....	47
3.9.2. FPA Property Page	49
3.9.2.1. AOI Start and AOI Size	49
3.9.2.2. Frames.....	50
3.9.2.3. Stitch	50
3.9.3. Data Property Page.....	51
3.9.3.1. Data Type	51
3.9.3.2. Bits	52
3.9.3.3. VMin.....	52
3.9.3.4. VMax.....	52
3.9.3.5. Setup Mode.....	52
3.9.3.6. Average Frames	53
3.9.4. Post Property Page	54
3.9.5. File Property Page	55
3.9.5.1. Format.....	55
3.9.5.2. Auto.....	55
3.9.5.3. Over	55
3.9.5.4. FileName.....	56
3.9.5.5. Format.....	56
3.9.6. Hardware Property Page.....	57
3.9.6.1. Clock Source Selection.....	57
3.9.7. Timing Requirements	58
3.9.7.1. Data Valid Window.....	58
3.9.7.2. Frame Sync, Line Sync, and First Valid Pixel.....	59
3.9.7.3. Multiplexed Data	59
3.10. In-line Post-processing.....	60
3.10.1. Implementation.....	60
3.10.2. INI File.....	62
3.10.3. Configuration Utility	63
3.10.4. Examples	65
3.10.5. Support.....	65
4. PI-Plot.....	67
4.1. Introduction.....	67
4.2. Data Sources	68
4.3. Data Ranges.....	69
4.4. Histograms and Cursors	69
4.5. Managing Views	71
4.6. Common Controls.....	72
4.6.1. Multiple Frames.....	72
4.6.2. Zooming	73
4.7. Acquisition Controls.....	73
4.7.1. Start.....	73
4.7.2. Continuous Acquisition.....	73
4.8. Scope View.....	74

4.9. Histogram View	75
4.9.1. Histogram Cursors	76
4.10. Grayscale View.....	77
4.10.1. Grayscale Luminosity.....	77
4.11. False Color View	80
4.11.1. False Color Table.....	80
5. Sample Files and Tutorial	85
5.1. Quick Start—Acquisition Timing Setup.....	85
5.2. Quick Start—Multiple Channel Acquisition	88
6. Troubleshooting	107
6.1. Common Problems.....	107
7. PI-3105 Programmer's Reference	110
7.1. Introduction.....	110
7.2. Local Operation	111
7.2.1. Configuring the GPIB Interface	111
7.2.2. Linking to dacq.dll	111
7.3. Remote IEEE-488 Interface Setup	113
7.4. Input/Output Protocol.....	114
7.4.1. Command Strings and File Exchange.....	114
7.4.1.1. Input to the PI-3105	114
7.4.1.2. Output From the PI-3105	114
7.4.1.3. Checking Status of the PI-3105	115
7.5. Sample Application/Testing Utility (GPIBCom).....	115
7.5.1. Local Setup	115
7.5.2. Remote Setup	116
7.5.3. General Operation.....	117
8. PI-3105 Command Reference	119
8.1. Chart of Commands	119
8.1.1. System Commands	119
8.1.2. Card-Specific Commands (Including port-specific commands)	120
8.1.3. Channel-Specific Commands.....	121
8.2. Command Sequencing	122
8.3. Command Reference Format.....	123
; (REM).....	123
@.....	123
ACQSTATUS	124
ACQTIME	124
ALLCARD	125
ALLCHAN	125
AOI	125
ATTACH DAQ	126
ATTACH TIMING	127
AVERAGE	128
BIASCLKMSGs.....	128
BITSPERPIX	129
BLOCKING.....	129
CDS.....	130
CDS STROBE	130
CHAN	131
CLOCK SRC	131
CONVERT STROBE	131
DACQMSGs	132
DAQ CARD	133
FILTERS.....	133
FLOAT.....	133
FRAME COUNT	134

FRAME SYNC/LINE SYNC.....	134
GAIN.....	135
GET ACQTIME.....	135
GET ACTIVE CHAN.....	135
GET ADC IDS.....	135
GET AOI.....	136
GET ATTACHMENTS.....	136
GET AVERAGE.....	136
GET BITSPERPIX.....	137
GET BLOCKING.....	137
GET CDS.....	137
GET CDS STROBE.....	137
GET CHAN.....	138
GET CLOCK SRC.....	138
GET CONVERT STROBE.....	138
GET DAQ CARDS.....	138
GET FILTERS.....	139
GET FLOAT.....	139
GET FPA.....	139
GET FRAMECOUNT.....	140
GET GAIN.....	140
GET INVERT.....	140
GET MONITOR.....	140
GET MUX.....	141
GET MUX CARDS.....	141
GET NAMES.....	141
GET OFFSET.....	142
GET OUTPUT.....	142
GET PORT.....	142
GET PIXEL PERIOD.....	142
GET POSITION.....	143
GET PREAMP ID.....	143
GET SAVETOFILE.....	143
GET SETUP MODE.....	144
GET SLAVE.....	144
GET STITCHING.....	144
GET SYNC.....	144
GET TIMING CARDS.....	145
GET TIMING MAXIMA.....	145
GET TIMING STEPS.....	145
GET VERTICAL.....	146
ID.....	146
INVERT/NINVERT.....	146
MUX MULTIPLEX MUX SWITCH.....	147
NAME AD.....	148
OFFSET.....	148
OUTPUT.....	148
PATMSGs.....	149
PAUSE.....	150
PIXEL PERIOD.....	150
PORT.....	150
POSITION.....	151
RESET.....	151
SAVEFILE.....	152
SAVETOFILE.....	152
SET ACTIVE CHAN.....	153

SETUP MODE.....	154
SLAVE.....	154
STARTACQ.....	154
STATUS.....	155
STITCHING.....	155
STOPACQ.....	156
VERTICAL.....	156
VIDEO/CONVERT/CDS.....	156
VMAX.....	157
VMIN.....	157
9. Index.....	159

LIST OF FIGURES

Figure 1: PI-3105 Multi-Channel Data Acquisition System	1
Figure 2: Micro D connectors, front view of card	5
Figure 3: PI-3100 Acquisition Interface Module, USB In and Address DIP Switch.....	7
Figure 4: PI-3100 Acquisition Interface Module, Front.....	7
Figure 5: AIM Front Panel Connections.....	8
Figure 6: PI-3100 Acquisition Interface Module, Rear	8
Figure 7: J101/J201 Power Connector Pinout and Wire Coloring.....	9
Figure 8: Rear Panel Connections, PI-3100-USB, AIM with Mux	10
Figure 9: Rear Panel Connections, PI-3100-USB, AIM without Mux	11
Figure 10: PI-41000 Digital Acquisition Card	11
Figure 11: DATA connector, front view	12
Figure 12: ACQ ARM connector, front view	12
Figure 14: Advanced Hardware Configuration.....	16
Figure 15: Add Hardware Component (Timing & Control).....	17
Figure 16: Connections Property Page (Timing Assignment).....	17
Figure 17: Connections Property Page (Timing Assignment Connected)	18
Figure 18: Connections Property Page (un-connectable components)	18
Figure 19: Connections Property Page (AIM to Mux connection).....	19
Figure 20: Connections Property Page (MUX to DACQ connection)	19
Figure 21: Connections Property Page (DACQ Arm Out to DACQ Arm In).....	20
Figure 22: Hardware Not Responding (Timing & Control Card).....	21
Figure 23: Simulating Hardware Components	21
Figure 24: TIMING mnemonic	22
Figure 25: AIM mnemonic	24
Figure 26: DEFINE mnemonic.....	27
Figure 27: Master/Slave Tree with Master Group.....	29
Figure 28: Master with Slave.....	29
Figure 29: Master and Independent	30
Figure 30: DACQ Slot Selection	30
Figure 31: Contiguous FPA Definition, Vertically Clocked	33
Figure 32: Contiguous FPA Definition, Horizontally Clocked	34
Figure 33: Interleaved FPA Definition, Alternating Lines.....	36
Figure 34: Interleaved FPA Definition, Interleaved Grids.....	37
Figure 35: 8-Channel Device, Contiguous Regions.....	38
Figure 36: Multiple Device Setup, Master/Slave Configuration	41
Figure 37: Multiple Device Setup, FPA 1 definition	42
Figure 38: Multiple Device Setup, FPA 2 Definition.....	42
Figure 39: DACQ mnemonic	44
Figure 40: Data Acquisition mnemonic, FPA Property Page.....	49
Figure 41: Data Acquisition mnemonic, Data Property Page	51
Figure 42: Data Acquisition mnemonic, Post Property Page	54
Figure 43: Data Acquisition mnemonic, File Property Page.....	55
Figure 44: Data Acquisition mnemonic, Hardware Property Page.....	57
Figure 45: PI-41000 Timing Relationship.....	59
Figure 46: Configuration Utility, Function Selection	63
Figure 47: Configuration Utility, Arguments List.....	64
Figure 48: Configuration Utility, Argument Entry	64
Figure 49: DACQ Mnemonic	67
Figure 50: PI-Plot Main Window	68
Figure 51: Data Information Dialog Box	69
Figure 52: Data Range	69
Figure 53: Cursor Pairs and Histogram Bins	70

Figure 54: Views and Data Sources	71
Figure 55: PI-Plot (Grayscale View)	72
Figure 56: Playback Buttons (First, Prev, Play, Next, Last)	72
Figure 57: Zoom Buttons (In, Out, 1:1, Fit)	73
Figure 58: Scope View	74
Figure 59: Scope View settings	75
Figure 60: Histogram View	75
Figure 61: Histogram Settings, Histogram Property Page	76
Figure 62: Histogram Settings, Axis Property Page	76
Figure 63: Grayscale View	77
Figure 64: Grayscale View, scaled to cursors	78
Figure 65: Grayscale View, Inverted	79
Figure 66: False Color View	80
Figure 67: False Color View, Properties, Color Table	81
Figure 68: False Color View, scaled to cursors	82
Figure 69: False Color View, Inverted	83
Figure 70: Sample PI-PAT file for acquisition timing	85
Figure 71: DEFINE setup for 16 x 1 frame	86
Figure 72: DACQ setup for 1 frame	86
Figure 73: DEFINE setup for 16 x 16 frame	87
Figure 74: DEFINE setup for 256 x 256 frame	87
Figure 75: WelcomeToPulse4.pip, Test Plan File	88
Figure 76: WelcomeToPulse4.w25, PI-PAT file with “walking” image	89
Figure 77: WelcomeToPulse4.w25, PI-PAT file, zoomed view	90
Figure 78: WelcomeToPulse4, AIM mnemonic	91
Figure 79: WelcomeToPulse4, Timing mnemonic	91
Figure 80: WelcomeToPulse4, Oscilloscope Capture	92
Figure 81: WelcomeToPulse4, Define Mnemonic	93
Figure 82: WelcomeToPulse4, False-Color Plot	94
Figure 83: WelcomeToPulse4, Oscilloscope Capture, Low Amplitude	95
Figure 84: WelcomeToPulse4, False Color Plot, Low Amplitude	96
Figure 85: WelcomeToPulse4, ‘Scope Plot, Low Amplitude	97
Figure 86: WelcomeToPulse4, Oscilloscope Capture, Medium Amplitude	98
Figure 87: WelcomeToPulse4, False Color Plot, Medium Amplitude	99
Figure 88: WelcomeToPulse4, ‘Scope Plot, Medium Amplitude	100
Figure 89: WelcomeToPulse4, Oscilloscope Capture, Full Amplitude	101
Figure 90: WelcomeToPulse4, False Color Plot, Full Amplitude	102
Figure 91: WelcomeToPulse4, ‘Scope Plot, Full Amplitude	103
Figure 92: Define Mnemonic, Rotated 90 Degrees	104
Figure 93: Image Rotated 90 Degrees	104
Figure 94: Define Mnemonic, Flipped Horizontally	105
Figure 95: Image Flipped Horizontally	105
Figure 96: Define Mnemonic, Flipped Vertically	106
Figure 97: Image Flipped Vertically	106
Figure 98: GPIBCom	116
Figure 99: GPIBCom communicating with dacq.dll	117

LIST OF TABLES

Table 1: Available Gain Values	25
Table 2: Filter Positions	26
Table 3: FPA Definition Parameters	31
Table 4: Parameters for Contiguous FPA Definition, Vertically Clocked	33
Table 5: Parameters for Contiguous FPA Definition, Vertically Clocked	35
Table 6: Parameters for Interleaved FPA Definition, Alternating Lines	36
Table 7: Parameters for Interleaved FPA Definition, Interleaved Grid	37
Table 8: Parameters for 8-Channel FPA Definition	39
Table 9: Variable Types for Post-Processing	63

1. INTRODUCTION

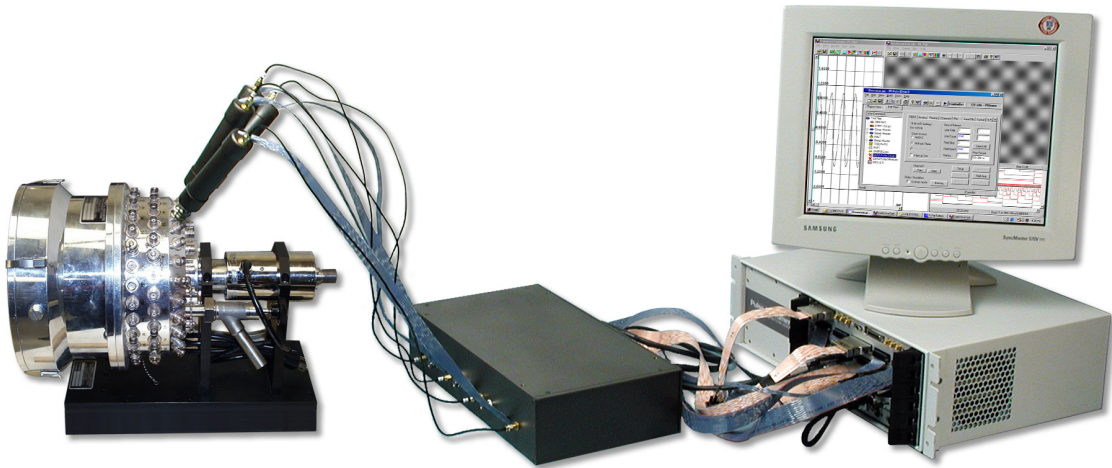


Figure 1: PI-3105 Multi-Channel Data Acquisition System

The PI-3105 is an integrated ADC and data acquisition system designed for acquiring high-resolution, low-noise image data from visible and infra-red sensors (CCD, CMOS, IR FPA and related technologies). There are several components to the PI-3105:

- One or more PI-3100 Acquisition Interface Modules (AIMs), containing an optional 4:1 digital multiplexer module and 1-4 each of:
 - ADC modules (PI-41040, PI-41060, PI-41070, PI-41080)
 - Gain/Filter modules (PI-41010, or integrated with the ADC module)
- Preamplifier Module(s) (PI-3150, PI-3170, or PI-3180)
- PI-3103D Low-Noise Linear DC Supply Mainframe(s)
- PI-41000 Digital Acquisition Card(s) (CompactPCI)
- CompactPCI Bus Interface Cards (GPIB, SCSI, PI-Bus, etc., Optional)
- CompactPCI CPU card, hard drive, optical drive (optional), and software
- PI-11000 CompactPCI mainframe(s)

The CompactPCI cards reside in a rack-mountable instrument mainframe. The AIM is a separate enclosure that can be placed near your device-under-test (DUT), such as on your optical bench or prober. The Gain/Filter and ADC modules are housed inside the AIM and are not accessible to the user. The Preamplifiers should be connected to your Dewar or test fixture, either directly via BNC connectors on a Dewar or test fixture, or via a short length of coaxial cable.

The PI-3105 Data Acquisition System must be used with a functioning imager that is already properly biased, clocked, and programmed for readout. These “stimulus” signals may be provided by other Pulse Instruments components, by 3rd-party instrumentation, or by the device itself in the case of “system on a chip” (SOC) devices.,

The components of the PI-3105 can be configured as a stand-alone data acquisition system or integrated with additional Pulse Instruments components to form a complete imaging test station. Pulse Instruments stimulus components may include:

- Pattern generators
- Programmable low-noise clock drivers
- Programmable low-noise DC bias supplies

The PI-3105 is controlled via PI-Controller software, included with your instrument and installed on the CompactPCI CPU running Windows 7 from the internal hard disk drive. The PI-3105 may also be controlled by optional PI-DATS automated test software, or by 3rd-party applications running under Windows 7. PI-Controller, PI-DATS, or your 3rd-party software may also run on an external Windows PC equipped with a GPIB interface card. The CPU card has standard PC-style peripheral ports for a user-supplied monitor, keyboard, and mouse. It also includes multiple Gigabit-Ethernet interfaces for connection to your network and an optional DVD-R drive for data storage, software installation, and system restoration.

1.1. How to Read This Manual

This manual specifically addresses operation of the PI-3105 Data Acquisition System with PI-Controller software. It assumes basic familiarity with PI-PAT software and with general operation of PI-Controller software. Subjects such as card installation and test plan generation are covered in a separate manual for the PI-11000 Instrument Mainframe and PI-Controller, and are not repeated here.

In general, users should read the PI-2005/PI-PAT manual first, followed by the PI-11000/PI-Controller manual. Once users are familiar with pattern generation and test plan generation, they should then proceed with this manual.

In a system without ADC conversion, only the PI-41000 cards will be present. Sections of this manual pertaining to the PI-3100, PI-3103D, ADC and preamplifier hardware may be skipped, but please read through all the software sections to ensure your familiarity with the channel assignment and data acquisition setup concepts. In particular, the Timing and Control/AIM components must be simulated in PI-Controller in order to provide the software with a complete data path schema. If your hardware and system software was installed at the factory, then all simulated components will already be configured, but you should still read the relevant sections of the manual in case reconfiguration is necessary (e.g. if hardware components are added or re-cabled).

Operators must also be familiar with correct usage of their focalplane device. In addition to knowing the required input timing patterns, clock driver voltages, and DC bias voltages for their device, users must also know:

- 1) Pixel count (horizontal and vertical, including any dark/reference pixels) for each video tap (output)
- 2) Pixel orientation and location for each video tap (output)
- 3) Timing relationship between clock inputs and valid video output
- 4) Output voltage swing and DC offset at the desired integration times and/or gain settings

For customer support, please contact support@pulseinstruments.com.

1.2. Known Issues, PI-DATS 2.708

1.2.1. PI-Controller and dacq.dll (common issues)

- **Save File** as ASCII is slow at this time, and saves floating point data only in 4.4 format.
- Setup mode control is not currently implemented, and is currently “On” at all times. When data are acquired as Floating Point (via the **Float** checkbox), the data reflect the conditioned signal at the input of the ADC circuit, and does not include correction for programmed gains and offsets.
- Data files currently use fixed header format that is not extensible and that currently omits several features. A future release will use an extensible, tags-based header that will be more flexible, but that may break applications that rely on the present format. Third-party applications written to read Pulse Instruments acquisition data files should optimally have an abstraction layer to separate the file parsing from the functions that use the data.
- Currently, **FILTERS** and **GAIN** are global settings, instead of settable for each Master Group. This may be updated in a future release. To ensure forward compatibility with in custom applications, you should use a **DAQ CARD** argument before a **FILTERS** or **GAIN** command to ensure that the proper channels are programmed when the DLL is updated.

1.2.2. Hardware

- In systems using both the A and B ports of DACQ cards, when switching from using the A port only to using both the A and B ports simultaneously, the first acquired frame will be invalid. Collect one “dummy” frame and discard it to work around this problem.
- Any of these actions can cause PI-41000 Data Acquisition cards to stop acquiring. Card(s) will arm, but the acquisition will not complete:
 - Data/timing cables are plugged/unplugged while clocks are running.
 - PI-3103D is power-cycled while clocks are running.Any of these occurrences may require the system to be rebooted in order to continue acquiring data.

2. HARDWARE

2.1. CPU and Chassis

Please read the accompanying PI-11000 Operators Manual for general instructions about the CompactPCI mainframe and about inserting and removing CompactPCI cards.

2.2. Front Panel Connections

There are three types of connections between the ADC hardware and the CompactPCI mainframes:

- Control signals
- Timing signals
- Data

If you have purchased a fully-integrated system from Pulse Instruments, consult the system block diagram and the wire list to see how your system is connected.

Timing signals are sent from the Pulse Instruments Pattern Generator (or from the FPA/ROIC) to the PI-3100 AIM(s), and from the AIMs to the ADCs. Control signals are sent from the Pulse Instruments computer over USB to the PI-3100 AIM(s) and then to the preamplifier modules. Digitized data are sent back from the AIM(s) to the PI-41000 Digital Acquisition Card(s).

In most Pulse Instruments systems the Timing Signals will originate from the Pulse Instruments pattern generator, typically from connector J1 on the first installed (top) pattern card. In systems with multiple AIMs, subsequent AIMs may be timed either from J2-J4 of the pattern card or from a PRL-4523 Fanout Buffer System.

2.2.1 CLK Connectors

The pattern generator J1-J4 output connectors and the AIM CLK IN connectors are a 9 pin MOLEX 1.27mm (.050") Pitch Ulti-Mate Commercial Micro D. The MOLEX description and part number are: "Board Mount, 9 Pin Right Angle Plug, with Pin Contacts, 83611-9006." The same connector is used on the PI-2100x Pattern Cards.

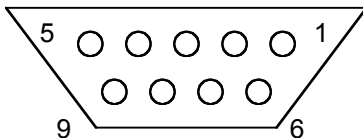


Figure 2: Micro D connectors, front view of card

The pin assignments for the J1 connectors are as follows:

Board: Pin:	PI-21x0y Pattern Card Output	PI-3100 CLK IN
Pin 1	Ch. 3 out	Frame Sync In
Pin 2	GND	GND
Pin 3	N/C	N/C
Pin 4	Ch. 4 out	CDS Clock In
Pin 5	GND	GND
Pin 6	GND	GND
Pin 7	Ch. 1 out	Line Sync In
Pin 8	GND	GND
Pin 9	Ch. 2 out	Pixel Clock In

When used with a Pulse Instruments cable 88001030-x or 88001035-x, the numbered coaxial outputs will correspond to channels 1-4 as indicated for the Pattern Card. When used with Pulse Instruments output cable 88001120-x, the cable is the same on both ends (e.g. wired straight through), and pin 1 is wired to pin 1.

The CLK Inputs to the PI-41000 are not terminated. If 50 Ohm termination is desired, there are jumpers available on the board adjacent to the input connector to provide 50 Ohm terminations. Please be sure that your timing source can drive a 50 Ohm load before installing the termination jumpers. If your timing source cannot drive a 50 Ohm load, then it may be necessary to use a 50 Ohm line driver module such as a Pulse Research Lab PRL-444.

2.2.2 Control Cable

Connect a standard USB A/B cable (P/N 88006001-x) from a USB connector on the CPU card or rear I/O card the **USB In** connector on the AIM. If you are using more than one AIM in your system, the AIMS can be connected via a USB hub. AIMS are logically numbered in software according to DIP switches to the left of the USB input on the AIM, and data channels also derive their numbering from this position:

- Bits 1-2: Virtual Slot Number
 - 00 = Slot 9
 - 01 = Slot 10
 - 10 = Slot 11
 - 11 = Slot 12
- Bits 3-4: AIM Number
 - 00 = AIM 1, Channels 1-4
 - 01 = AIM 2, Channels 5-8
 - 10 = AIM 3, Channels 9-12
 - 11 = AIM 4, Channels 13-16

When the switches are in the Down position (as shown, below), the bit value is 0, e.g. the AIM below is at Virtual Slot 9, AIM #1:

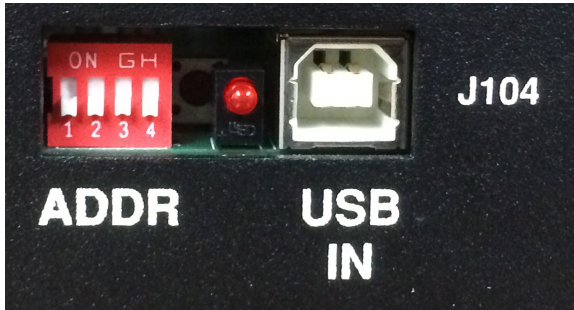


Figure 3: PI-3100 Acquisition Interface Module, USB In and Address DIP Switch

2.2.3 Timing Cables

Using the long Micro-D to Micro-D cable (P/N 88001120-8), connect Pattern Out (**J1-J4**) connectors on the Pattern Card to the **Clk In (J107)** connector on each AIM.

For ease of channel identification and troubleshooting, timing connections should be made in ascending order by AIM, e.g. the first logical AIM should be connected to the first set of pixel clocks, etc.

2.2.4 PI-3100 Acquisition Interface Module, Front Panel



Figure 4: PI-3100 Acquisition Interface Module, Front

The PI-3100 houses up to four Gain/Filter stages, ADCs, and optional CDS converters. It also provides the controls and power for the associated preamplifier modules.

The front panel of the PI-3100 has the following connectors:

- Signal Input 1-4 (**J201, J204, J207, J210**, SMA)
- Preamp Control/Offset Out 1-4 (**J202, J205, J208, J211**, 15-pin D)

Each **Signal In** and **Preamp Control Out** may be cabled to a Preamplifier module.

WARNING: The PI-3103D **must** be powered off when attaching or detaching preamplifier modules. Plugging or unplugging preamps when the power is turned on will permanently damage the preamps and void the warranty.

Connect the BNC connector on each Preamplifier to your DUT or test fixture. For best noise performance, minimize the cable length from your DUT or test fixture to the input of the Preamplifier. If possible, the Preamplifier should be attached directly to your Dewar or test fixture. The total signal path from the DUT output pin to the input of the Preamplifier must not exceed 36 inches, including

any wiring internal to your Dewar and PCB trace lengths on your test fixture. Longer signal paths can result in oscillations and increased noise.

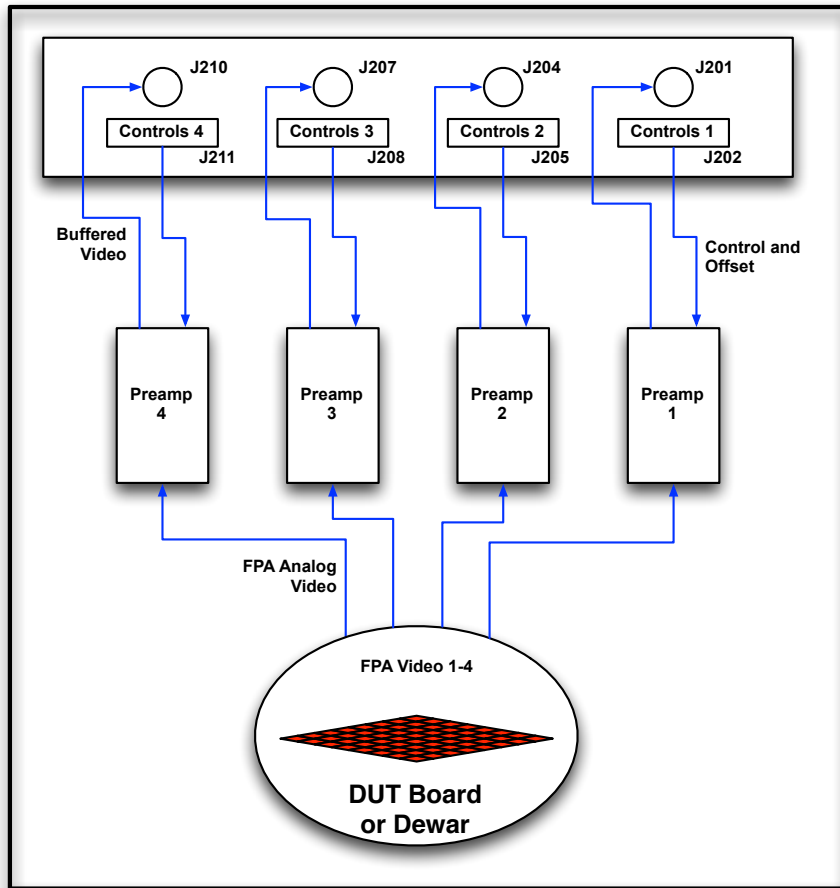


Figure 5: AIM Front Panel Connections

The three Monitor outputs may be connected to an oscilloscope with a 50 Ohm input termination. If the Monitor outputs are not properly terminated into 50 Ohms, ringing will be visible in the monitor signals. Channel selection for each monitor output is programmable in software (see below). Noise from an attached oscilloscope may couple into the analog signal. Therefore, during noise measurements or sensitive data collection procedures disconnect the cables from the Monitor outputs.

2.2.5 PI-3100 Acquisition Interface Module, Rear Panel

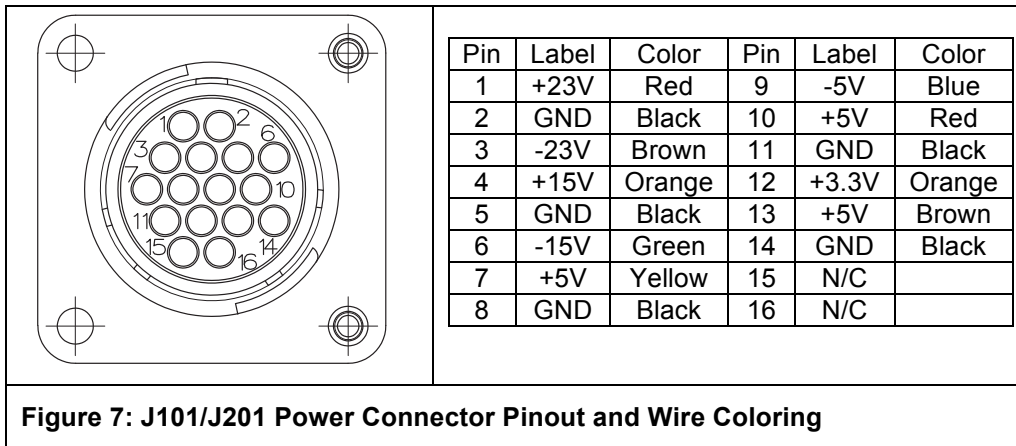


Figure 6: PI-3100 Acquisition Interface Module, Rear

The rear panel of the PI-3100 has the following connectors:

- DC power In (**J101**)
- USB In (**J104**)
- Ch 1-4 Data Out (**J103, J105, J110, J112**, 40 pin header) or Mux Out (**J113**)
- CLK In (**J107**, 9-pin Micro-D)
- Convert Monitor Out, software selectable for channels 1-4 (**J114**, BNC)
- Video Monitor Out, software selectable for channels 1-4 (**J115**, BNC)
- CDS Monitor Out, software selectable for channels 1-4 (**J116**, BNC)
- Ref Clk Out, (**J117**, BNC)

The DC Power In should be connected to an available power connector (**J101** or **J102**) on a PI-3103D Analog Power Supply using cable 88001130-8. The pinout of **J101** and **J102** on the PI-3103D is shown below:



The PI-3103D Analog Power Supply must be powered off when connecting it to the AIM.

The **Data Out** connector for each installed channel or the **Mux Out** should be cabled to an input of a **PI-41000 Digital Acquisition Card**.

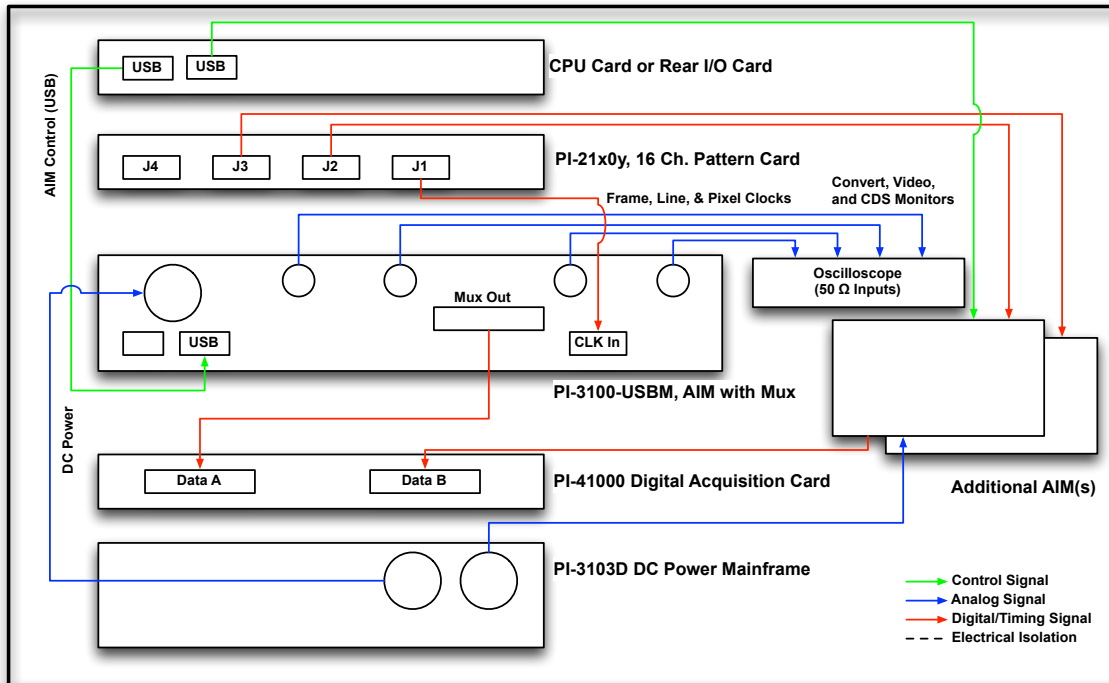


Figure 8: Rear Panel Connections, PI-3100-USB, AIM with Mux

Figure 8, above, shows typical rear-panel connections for a system with the internal multiplexer installed. The multiplexer outputs LVDS clocks and data at a maximum total data rate of 80 MHz. Any of the four ADC channels may be switched to the output port (**Mux Out**), or the card may be placed in one of 3 multiplexing modes (see software section, below). If switched to a single port, the data rate may be up to 80 MHz. If set to multiplex all active channels must have the same data rate, and the total data rate may be up to 80 MHz.

The Mux card outputs LVDS clocks and data to an input port of the PI-41000 Digital Acquisition Card

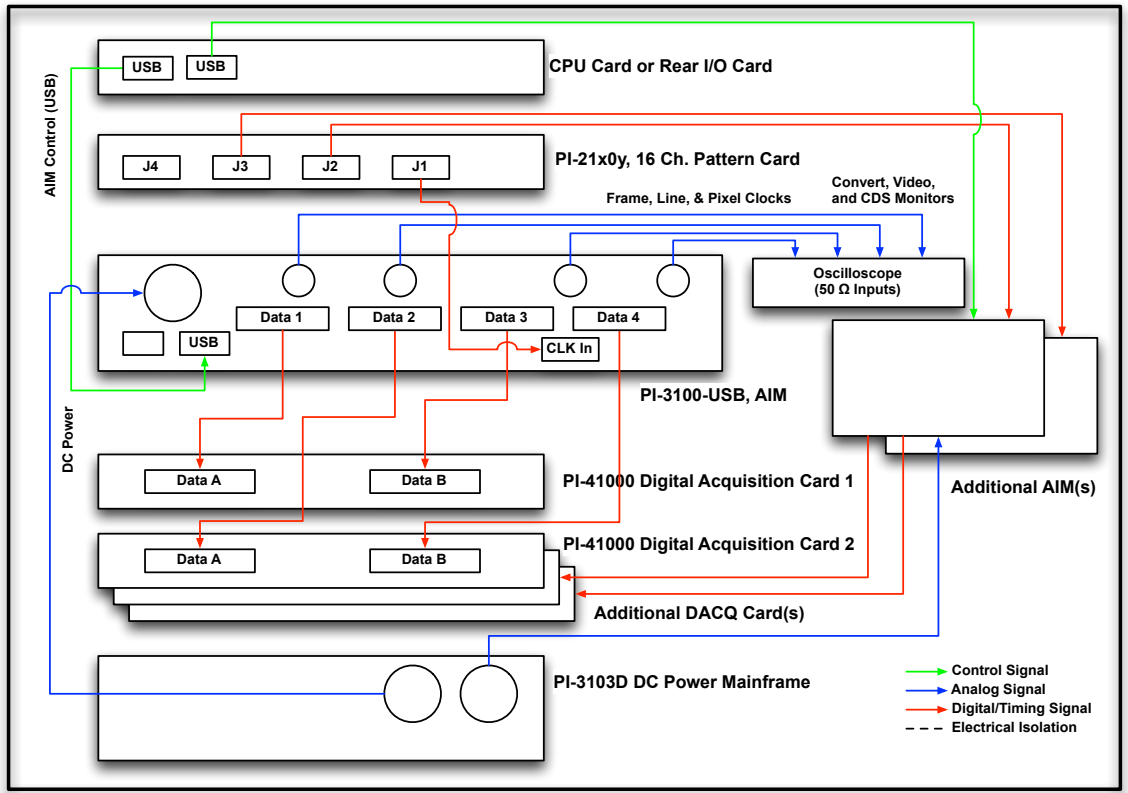


Figure 9: Rear Panel Connections, PI-3100-USB, AIM without Mux

Figure 8, above, shows typical rear-panel connections for a system without internal multiplexer. Each **Data** output delivers LVDS clocks and data to an input port of a PI-41000 Digital Acquisition Card.

2.2.6 PI-41000 Digital Acquisition Card

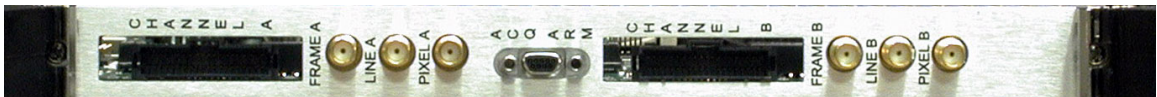


Figure 10: PI-41000 Digital Acquisition Card

The PI-41000 Digital Acquisition Card accepts LVDS clocks and data at a maximum total incoming data rate of 160 MHz (80 MHz per port), or up to 120 MHz on the A port. The PI-41000 can also accept TTL-level clocks on the SMA connectors. Each card has 256 or 512 MB of on-board SDRAM, indicated by the model number (PI-41000-256 or PI-41000-512).

Port A should be connected to the output of an AIM. If you are using only one input port on the **PI-41000 Digital Acquisition Card**, you must use **Channel A**. **Channel B** may be used in conjunction with **Channel A**, but it cannot be used by itself.

The PI-3105 provides the ability to synchronize data capture between two or more PI-41000 Digital Acquisition Cards. This is necessary for applications with many output channels or applications with very high data rates or memory depth requirements that preclude multiplexing.

If you are using more than one PI-41000 Digital Acquisition Card in your system, and you wish to synchronize data capture on all cards, then you must connect the **Remote Arm** signals. Connect the

Master Arm cable connector (marked with an **M**) on the arming cable (P/N 88001135-x) to **Acq Arm** port on the card you wish to use as the Master card. Connect **Slave 1, 2, or 3** cable connectors (marked **S**) to the **Acq Arm** ports on the cards you wish to use as Slaves. The Master Arm must be cabled to the Master card, but any Slave connector can be connected to any Slave card

2.2.6.1 Data/Clock Connectors

The DATA inputs of the PI-41000 use AMP part number 5-104069-6, Header, R/A, 20 x 2 Pos, 0.1" x 0.05", with side latches. Each of the two 40-pin micro header connectors accepts up to 16 bits of low-voltage differential signal (LVDS) data and 3 LVDS clock signals (Frame Sync, Line Sync, and Pixel Clock). The clock signals may also be input with TTL levels via the SMA connectors for each channel.

The SMA/TTL clock inputs are internally terminated with 50 Ohms; therefore an external clock source must be capable of delivering TTL thresholds into a 50 Ohm load. The data acquisition system is edge-triggered on the rising edge of the pixel clock, and the Frame and Line sync signals are normally triggered on a rising edge, however all of the external clock inputs can be set for inverted polarity in software.

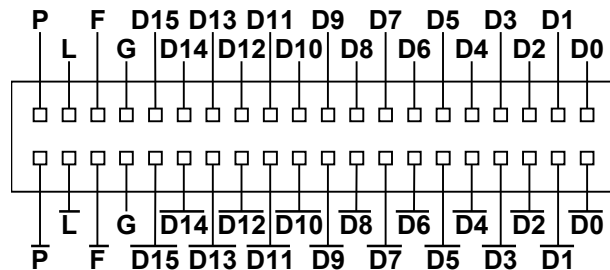


Figure 11: DATA connector, front view

The pin assignments for the Data connector are shown above, as viewed from the front of the card. D0 is the least-significant bit (LSB). The pair marked G is connected to the CompactPCI chassis ground. Note: earlier revisions of the PI-41000 may have this pair unconnected.

2.2.6.2 Acquisition Arm Connector

The Acquisition Arm connector is a Molex Micro-D and is used for synchronizing data acquisition on two or more PI-41000 cards in a system.

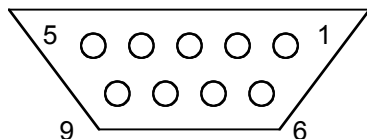


Figure 12: ACQ ARM connector, front view

The pin assignments for the Acquisition Arm connector are as follows:

- Pin 1 Arm Channel 3
- Pin 2 Arm Channel 2
- Pin 3 Arm Channel 1
- Pin 4 GND
- Pin 5 Arm In
- Pin 6 GND
- Pin 7 GND
- Pin 8 GND
- Pin 9 GND
-

This connector is used for synchronizing data capture among 2 or more PI-41000 Digital Acquisition Cards.

3. SOFTWARE

The PI-3105 is controlled by PI-Controller, Pulse Instruments' integrated application for control of focalplane test systems. PI-Controller is also used for control of the Pulse Instruments Pattern Generator, Clock Driver, and Low-Noise DC Bias cards. Please read the PI-11000 Operators Manual for an overview of PI-Controller and general operating concepts before proceeding with this manual.

3.1. Channel Assignments

PI-3105 timing and settings are programmed by channel numbers or channel names. These channels correspond to each ADC channel in the system. By default they are numbered from 1 to n, but they may also be assigned user-defined names.

In systems without ADC conversion, the ADC channels will be simulated in the hardware configuration, and all references to "ADC channels" correspond directly to digital data channels.

Channel assignments in software are determined by the DIP switches on the AIM:

- Bits 1-2: Virtual Slot Number
 - 00 = Slot 9
 - 01 = Slot 10
 - 10 = Slot 11
 - 11 = Slot 12

- Bits 3-4: AIM Number
 - 00 = AIM 1, Channels 1-4
 - 01 = AIM 2, Channels 5-8
 - 10 = AIM 3, Channels 9-12
 - 11 = AIM 4, Channels 13-16

Control of all channel-based parameters is based on these assignments. All other components are identified by CompactPCI slot number.

3.2. PI-Controller

PI-Controller is an integrated software application providing control over all hardware and data elements in a Pulse Instruments system. Information for control of pattern generator, clock driver, and DC bias supplies can be found in the PI-11000 Operators Manual, on your system's hard drive in the C:\Documentation folder or downloadable from:

<http://www.pulseinstruments.com/support>

The PI-11000 Operators Manual should be read in its entirety before continuing with this document.

This document contains information for setting up and controlling the PI-3105 Data Acquisition System and for displaying and archiving the acquired data.

3.3. Hardware Configuration

The **Hardware Configuration** window is used to verify installed hardware, add simulated hardware components, and to specify connections between components, including the routing of data from the AIMS through the multiplexers and into the data acquisition cards. If your system was shipped from the factory with all installed components, and you are running PI-Controller from the embedded CPU

board, then all required components have been configured in the default hardware configuration, and you may skip this entire section.

If you are running PI-Controller from your PC, or your system has not already been set up for data acquisition, you may need to scan your system and/or manually add simulated hardware components. **Connections** between components may also need to be specified (see **Section 3.3.2. Connecting Components**). For information on scanning your system for supported hardware, please see the PI-11000/PI-Controller Operators Manual, Hardware Configuration section. Once your system has scanned and recognized all supported hardware, return to this section of this manual.

To establish connections between components, go to **Edit: Advanced Hardware Configuration** and click on the **Connections** tab.

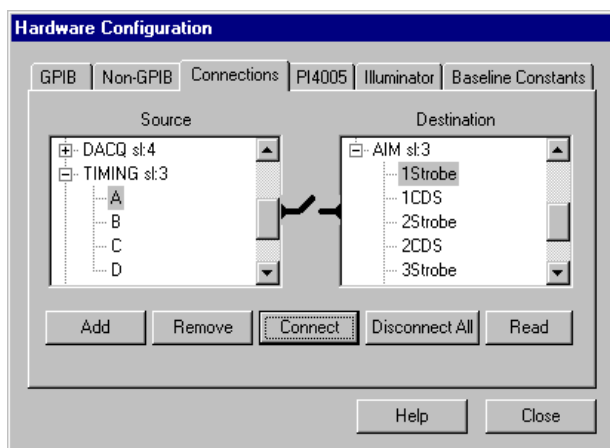


Figure 13: Advanced Hardware Configuration

The **Advanced Hardware Configuration** page contains a listing of every recognized hardware component in the system. It also allows for the specification of “connections” between hardware elements.

3.3.1. Simulated Components

Under certain conditions, some hardware elements must be manually added as simulated components in the system configuration:

- Hardware temporarily disabled or missing
- Offline editing of a PIP file
- Simulation of hardware
- Use of digital acquisition system without the ADC subsystem

The simulated status of the hardware will create a warning any time the **Hardware Configuration** is changed, and an optional warning when PI-Controller is launched or when a file is created or opened (see **Section 3.3.2.4. Confirming Simulated Components/Hardware Device Not Responding** below).

Adding simulated components needs to be done once only; therefore if these components already show up on the **Add Mnemonic** window, then you may skip this section.

To manually add a component, click the **Add** button. A list of possible items to add will appear.

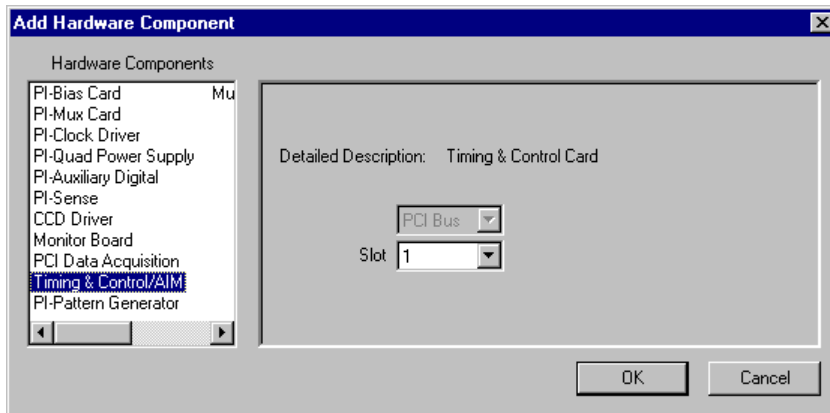


Figure 14: Add Hardware Component (Timing & Control)

For example, to add a simulated AIM, select **Timing & Control/AIM** from the left part of the dialog box, and then select an unused CompactPCI Slot on the right. Click **OK** to add the component.

3.3.2. Connecting Components

Next, your timing and data cabling must be specified. Note that your **Connection** assignments must match the physical cabling of your system.

3.3.2.1. Timing Channels

First, timing channels must be assigned to your AIM(s).

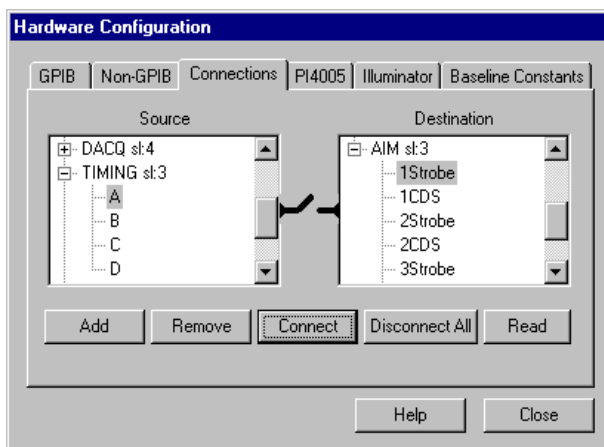


Figure 15: Connections Property Page (Timing Assignment)

On systems with the PI-3100-USB or PI-3100-USBM, this section is not applicable, as the timing assignments are made automatically, based on the virtual slot assignments from the AIM DIP switches (see Figure 3: PI-3100 Acquisition Interface Module, USB In and Address DIP Switch.)

From the left **Source** window, select a timing channel block (**A-D**) from a **Timing** card. If necessary, click the **+** sign to expand the list of available choices. Click the appropriate connector on the corresponding **AIM** in the right **Destination** window, and then click **Connect**. The broken bar will be replaced by a solid yellow bar to indicate that the connection has been made. AIMS are identified by their timing slot. For example, in **Figure 16**, below, the AIM group in slot 3 represents all possible AIMS connected to the timing card in slot 3. **1Strobe** refers to the ADC timing strobe for AIM 1, which contains channels 1-4. Similarly, **2Strobe** refers to the ADC strobes for channels 5-8, etc.

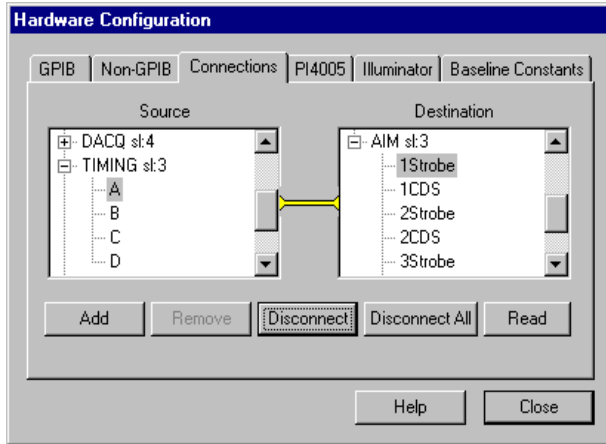


Figure 16: Connections Property Page (Timing Assignment Connected)

Command equivalent:

```
ATTACH TIMING 3 AIM 1 STR A ↵
```

Repeat this process for the **CDS** connection (if used), and then for each **AIM** until all physical timing connections are reflected in the **Connections** property page.

If a source and destination are selected that are not connectable, the connection bar between the panes will display the “un-connectable” symbol:

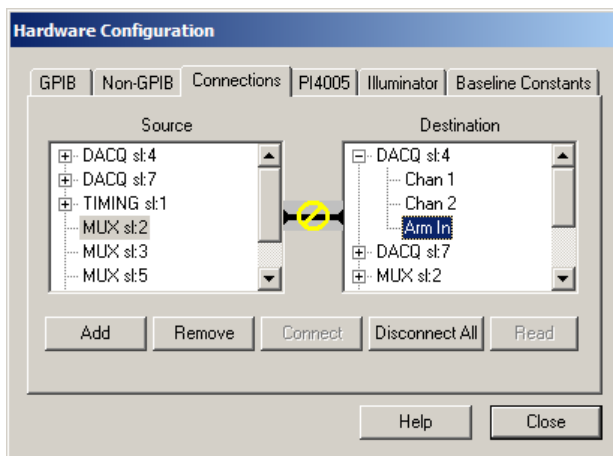


Figure 17: Connections Property Page (un-connectable components)

Note: Timing assignment must be made prior to all other connection assignments; otherwise, PI-Controller will not allow the AIM outputs to be connected to the data acquisition components.

3.3.2.2. Data Channels

Next, specify how your data channels are connected. From the **Connections** window, select an **AIM** from the **Source** window. AIMS are listed with the slot of their Timing & Control card.

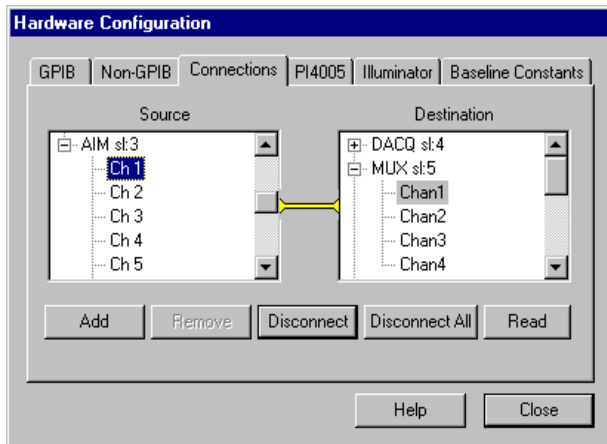


Figure 18: Connections Property Page (AIM to Mux connection)

All 16 possible data channels from AIMs connected to the Timing Card in Slot 3 will be listed here. Select an AIM output and then select the corresponding **Mux** input or **DACQ** input in the **Destination** window. Click **Connect** to establish connection. Repeat this for each AIM output in use.

If you are using a PI-41110 Multiplexer Card, you must then connect its output to the appropriate PI-41000 Digital Acquisition Card input:

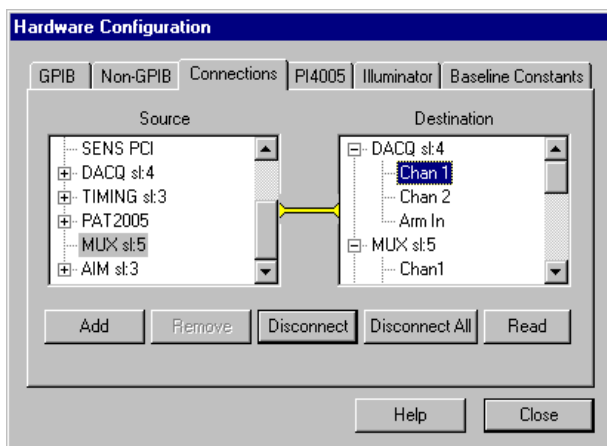


Figure 19: Connections Property Page (MUX to DACQ connection)

Command equivalent:

```
ATTACH DAQ 4, A MUX 5, 1, 2, 3, 4 ↵
```

Repeat this process until all physical data connections are reflected in the Connections property page.

3.3.2.3. Master/Slave Remote Arm

If you are using the Remote Arm feature to synchronize data capture between two or more PI-41000 Digital Acquisition Cards, you must also indicate this connection in the Connections property page:

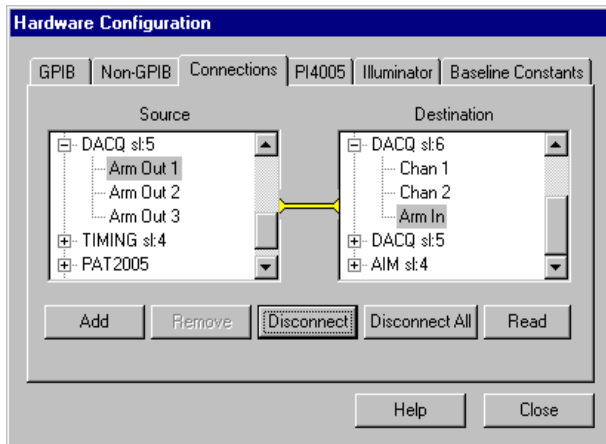


Figure 20: Connections Property Page (DACQ Arm Out to DACQ Arm In)

Select a **Master** card from the **Source** window and choose an **Arm Out** signal. Connect it to the **Arm In** on the corresponding **Slave** card in the **Destination** window. Any **Arm Out** from a **Master** may be connected to the **Arm In** on any **Slave** card.

Command equivalent:

```
ATTACH DAQ 6, A MUX 7, 9, 10, 11, 12 MASTER 5, 1 ↵
```

A Master is defined as any PI-41000 card that has at least one **Arm Out** connected to a Slave card's **Arm In**. This connection is determined by physical cabling and via the **Connections** property page, and should change only when the physical configuration of your system has changed.

Across test plans or within a test plan, a Master card may acquire data with all, some or none of its slaves. Use of Slave cards is configurable in software, and does not require changes to physical cabling or to the **Connections** settings.

A Slave is defined as any PI-41000 card that has its **Arm In** connected to the **Arm Out** of a Master card. This connection is determined by physical cabling and via the Connections property page, and should change only when the physical configuration of your system has changed.

Across test plans or within a test plan a Slave card may acquire data either with its Master or as an Independent card. Slave/Independent status is configurable in software, and does not require changes to physical cabling or to the Connections settings.

A Master and all its configured Slaves (a "**Master Group**") are triggered for acquisition synchronously, and must belong to the same FPA definition (see **Section 3.7. DEFINE mnemonic**, below).

Each Master Group and each Independent card(s) may be clocked separately, and may have different FPA definitions.

3.3.2.4. Confirming Simulated Components/Hardware Device Not Responding

Once all necessary hardware components have been added and all connections have been specified, click **Close** to close the **Hardware Configuration** dialog box.

If you have added any simulated components you will receive warnings for each simulated component.

This series of warnings will also display every time PI-Controller is launched and every time a test plan file (.pip) is created or opened, unless you check the boxes to turn off these warnings. The warnings will always appear after changes are made in the **Hardware Configuration**.

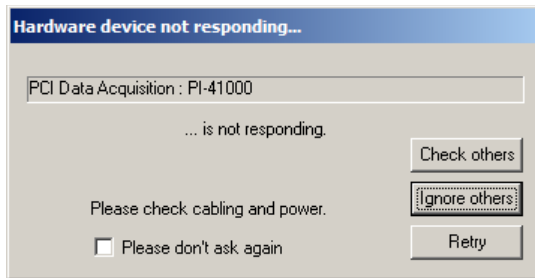


Figure 21: Hardware Not Responding (Timing & Control Card)

If this is expected behavior (e.g. you have intentionally simulated a component), you may check the **“Please don’t ask”** box to suppress future warnings for this hardware element. Note that the hardware is checked for each component in the system; therefore if you have multiple data acquisition cards you will see a warning for each.

When each warning appears, click **Check Others** to check other elements of this type (e.g. to check other Data Acquisition cards in the example above) or click **Ignore Others** to suppress further checks of this hardware type. Continue until all warnings have completed. Once the final hardware component has been checked, a list of all checked and non-responding components will appear:

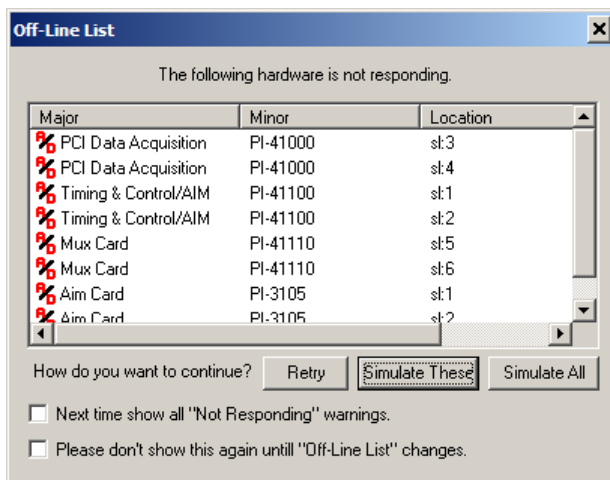


Figure 22: Simulating Hardware Components

Click **Simulate These** to allow PI-Controller to simulate these components. Click **Simulate All** to allow PI-Controller to simulate **all** hardware components in the test plan (including those that responded successfully to the hardware check). Click **Retry** to run the hardware check again.

To dismiss future listings of the Off-line list, check the **“Please don’t show again”** box. You can always view this list (and un-check the **“Please don’t show again”** box) using the **View:Off-Line** list menu item in PI-Controller.

If you have previously suppressed warnings for a hardware component, but now want to check it, check the **“Next time show all warnings”** box. All hardware components will be checked and **Hardware Not Responding** warnings will be generated next time a file is opened/created or the next time PI-Controller is launched.

3.4. Setting Up Test Plans

Once all hardware components and connections have been defined, they may be inserted into a test plan. Please see the PI-11000 Operators Manual for details on creating a test plan and inserting mnemonics.

3.5. TIMING mnemonic

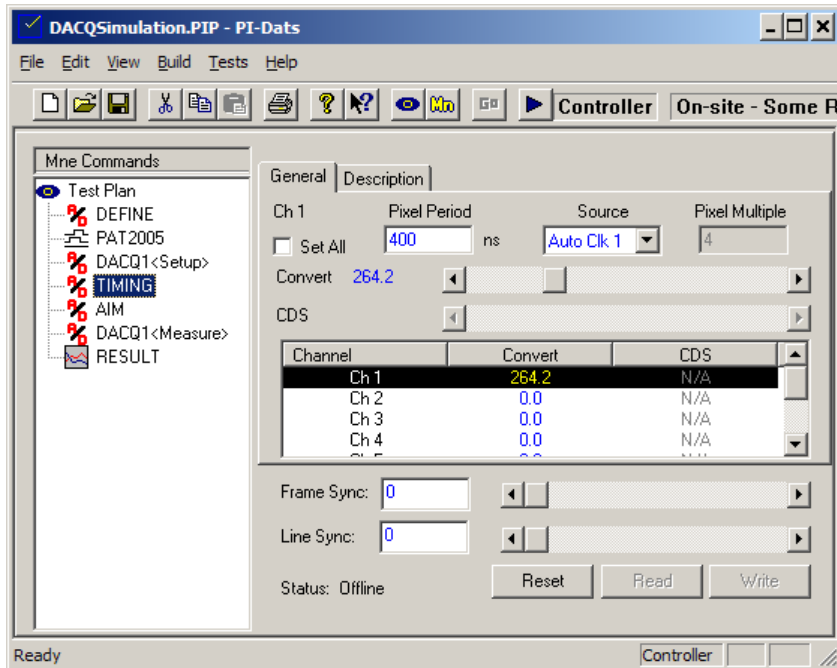


Figure 23: TIMING mnemonic

3.5.1. Overview

The **TIMING** mnemonic programs the ADC convert strobe and CDS convert strobe timing for each channel. The scrollbar controls allow real-time control of the one or all channels, and the text boxes permits numeric entry of settings.

When a scrollbar is dragged, the new timing settings are written immediately to hardware in real-time.

Settings that do not match the current state of the hardware are displayed in blue. Displayed settings can be set to match the current hardware settings by clicking the **Read** button.

When the **Write** button is clicked, all values contained in this mnemonic are written to hardware. If the hardware is currently set to some other value (e.g. the settings were previously written from another TIMING mnemonic or values were typed in numerically) it will be programmed to the setting in the list or in the text box.

3.5.2. Convert and CDS strobe timing

Before programming the **Convert** and **CDS** strobe timing, the effective pixel period for your timing generator must be specified in the **Pixel Period** box. The allowable values for **Convert** and **CDS** timing are a function of the **Pixel Period** entry. Timing values beyond the calculated range may cause the clock pulses to disappear entirely; therefore it is important to ensure that the **Pixel Period** entry is not longer than the actual pixel clock period.

The **Pixel Period** entry will also be written to the header of the acquired data header to enable time-domain analysis.

If you are using a Pulse Instruments pattern generator (e.g. PI-2005 or equivalent) also controlled by PI-Controller or PI-DATS, the **Source** mnemonic may be set to **Auto Clk**, to automatically receive the programmed master clock period from the most current pattern generator settings. Note that this may be different from the pixel clock period (e.g. a multiple thereof) on your timing generator, depending on the pattern that has been programmed. The **Pixel Multiple** setting must be set by the user to reflect this relationship (e.g. a 11001100 bit pattern will have a multiplier of 4). The **Auto Clk** source should be set to the number of the Clock Generator card providing the master clock for your Timing and Control card (typically **Auto Clk 1**).

Once programmed to **Auto**, the master clock period will be updated whenever PI-Controller writes a new master clock period to the pattern generator via the Pat mnemonic. PI-Controller will not update the **Pixel Multiple** setting.

If you are not using PI-Controller to control a Pulse Instruments pattern generator for your timing generation, the **Source** should be set to **Manual**, and the **Pixel Period** should be entered manually.

Command equivalent:

```
SET PIXEL PERIOD 400, 3 ↵
```

Note that the CDS timing entries will be available only if you have a CDS module installed on your system **and** there a Timing channel connected to it in the **Connections** property page.

To program the timing for a channel, select the channel name in the list-box and then type an entry in the appropriate column or drag the scrollbar control.

Command equivalent:

```
CHAN 1 ↵  
CONVERT STROBE 248.7 ↵  
CDS STROBE 122.3 ↵
```

If the **Set All** box is checked, the scrollbars will program all channels on a Timing card simultaneously. Otherwise, the scrollbars will program the timing only for the selected channel.

Command equivalent:

```
ALLCHAN ↵  
CONVERT STROBE 248.7 ↵  
CDS STROBE 122.3 ↵
```

Use the **Monitor** outputs (see **below**) and an oscilloscope to facilitate positioning of your timing strobes for each channel.

3.5.3. Frame Sync and Line Sync

Use the Frame Sync and Line Sync controls to program the Frame Sync and Line Sync delays. Please note that both these clocks are sampled in the acquisition system on the rising edge of each pixel clock; therefore only adjustments that are beyond a pixel period will have any effect on the output, and that adjustments will have effect only at integral multiples of the pixel period.

Command equivalent:

```
CHAN 1 ↵  
FRAME SYNC 55 LINE SYNC 69 ↵
```

If the **Pixel Period** or **Pixel Multiple** has been programmed incorrectly, it is possible to program “illegal” delays, which may cause the Timing card outputs to disappear. They delays may stop responding to programming, or else the output pulses may disappear entirely. If this happens, the Timing card and AIM hardware may be reset by using the **Reset** button. Note that programmed delays will revert to zero when the **Reset** button is clicked. To preserve your programmed information, insert a new **DEFINE** mnemonic (or copy the existing **DEFINE** mnemonic), and click **Reset** from the new mnemonic.

Command equivalent:

```
RESET 3 ↵
```

3.6. AIM mnemonic

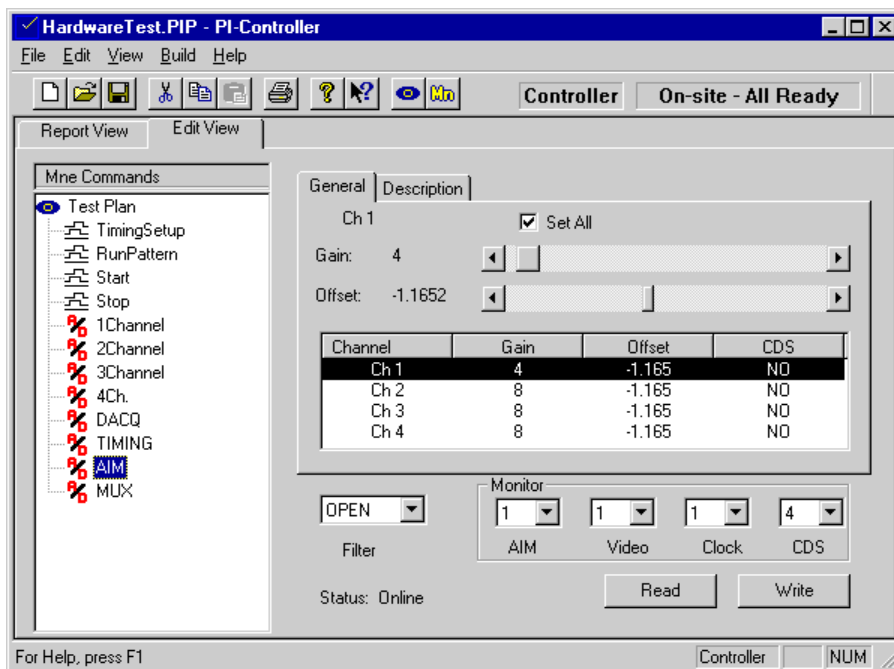


Figure 24: AIM mnemonic

3.6.1. Overview

The AIM mnemonic allows programming of the **Gain** and **Offset** for each channel, the global **Filter** setting, and the signal **Monitor** outputs for each AIM. The **CDS** mode is also programmed here.

When a scrollbar is dragged, the new timing settings are written immediately to hardware in real-time.

Settings that do not match the current state of the hardware settings are displayed in blue. Displayed settings can be set to match the hardware settings by clicking the **Read** button.

When the **Write** button is clicked, all values are written to hardware. If the hardware is currently set to some other value (e.g. the settings were previously written from another AIM mnemonic or values were typed in numerically) it will be forced to the setting in the list or in the text box.

3.6.2. Gain

To program the **Gain** for a channel, select the channel name in the list-box and then type an entry in the appropriate column or drag the scrollbar control. Gain may be set between 1x and 420x.

Command equivalent:

CHAN 1 ↵
GAIN 8 ↵

The gain stages in the preamplifiers have been optimized for low-noise performance; therefore not all gain values between 1 and 420 are available.

Available Gain Values					
1	16	64	140	208	288
2	20	72	144	216	300
3	24	80	160	220	308
4	28	84	168	224	312
5	32	96	176	240	336
6	40	100	180	252	360
7	48	112	192	260	364
8	56	120	196	264	392
12	60	128	200	280	420

Table 1: Available Gain Values

If the **Set All** box is checked, the scrollbars will program all channels on a Timing card simultaneously. Otherwise, the scrollbars will program the gain only for the selected channel.

Command equivalent:

ALLCHAN ↵
GAIN 8 ↵

3.6.3. Offset

To program the **Offset** for a channel, select the channel name in the list-box and then type an entry in the appropriate column or drag the scrollbar control. Offset may be set in the range of ± 10 V with 16 bits of resolution. Note that the offset voltage correction is applied before the gain stages; therefore the minimum resolution is 305 μ V times the current gain setting.

Command equivalent:

CHAN 1 ↵
OFFSET -4.5 ↵

If the **Set All** box is checked, the scrollbars will program all channels on a Timing card simultaneously. Otherwise, the scrollbars will program the offset only for the selected channel.

Command equivalent:

ALLCHAN ↵
OFFSET -4.5 ↵

3.6.4. CDS Mode

To set the CDS mode for each channel, select **Yes** or **No** from the **CDS** drop-menu, then click **Write**. When set to **Yes**, the data output from the AIM will be the difference between the two ADC values. When set to **No**, the data output from the AIM will be the value from the video ADC only.

Enabling CDS mode on the PI-41040 adds one bit of resolution and doubles the full-scale voltage range of the acquired data. Therefore the **DACQ** mnemonic must be configured differently to handle data from CDS mode and Normal mode. In Normal mode the PI-41040 represents values from -2.0 V to +2.0 V across 14 bits. In CDS mode two 14-bit values are subtracted, producing a full-scale range from -4.0 V to +4.0 V across 15 bits. The DACQ mnemonic settings should set to reflect these changes when CDS mode is turned on or off.

Note that the CDS setting will be enabled only if you have a CDS module installed on your system **and** there a Timing channel connected to it in the **Connections** property page.

Command equivalent:

CHAN 1 ↵
CDS TRUE ↵

3.6.5. Filter

To program the common filter for all channels in all AIMs connected to this Timing card, select a filter position from the **Filter** drop-menu, then click **Write**. The following settings are available:

Cutoff Frequency (- 3dB)
Open
10 MHz
1 MHz
100 kHz
10 kHz
Closed (No signal)

Table 2: Filter Positions

Command equivalent:

FILTERS 1 MHZ ↵

3.6.6. Monitor

Each of the three Monitor signals on each AIM may be independently switched to any of the installed channels. Use the **AIM** drop-menu to select an AIM, then use the **Video**, **Convert**, **Clock** and **CDS** drop-menus to specify the channel setting for each monitor output. Click **Write**. You will hear relays clicking on the AIM(s).

Command equivalent:

CHAN 1 ↵
VIDEO 1 CONVERT 2 CDS 2 ↵

The time relationship among the **Video**, **Clock** and **CDS** signals at the PI-3100 Monitor Out connectors is the same as the timing relationship at the ADCs.

The Monitor outputs must be connected to an oscilloscope with 50 Ohm termination at the input. If the signal is not terminated at the oscilloscope input, the monitor signals may ring, and will not accurately reflect the video and clock signals.

3.7. DEFINE mnemonic

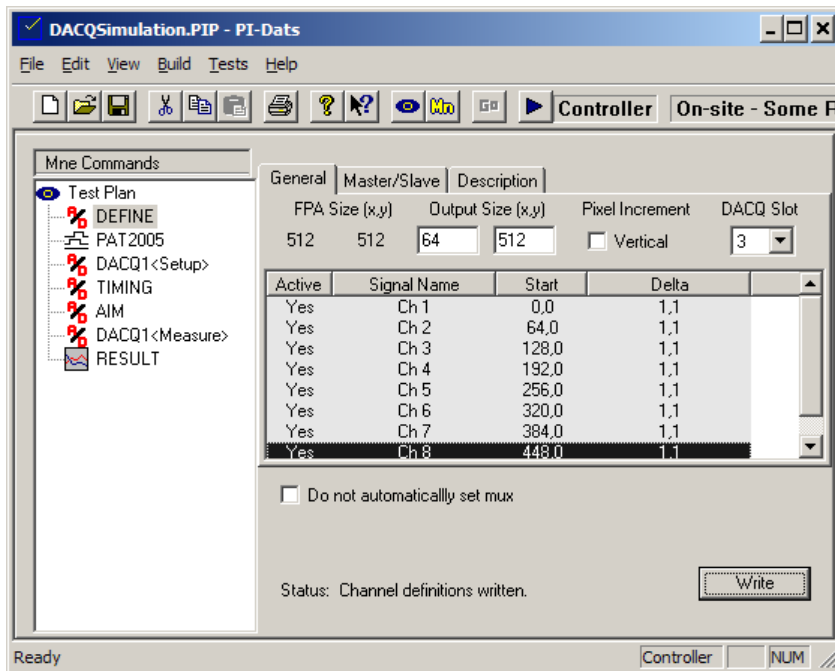


Figure 25: DEFINE mnemonic

3.7.1. Overview

The PI-3105 Multi-Channel Data Acquisition System permits synchronous ADC conversion and data acquisition from up to 32 channels. Built-in “stitching” routines can reassemble data from multiple channels into a single data array for analysis, video display, and archiving.

In order to properly acquire and display data from the device-under-test (DUT), PI-Controller needs to know about its readout characteristics.

The **DEFINE** mnemonic permits the user to define each DUT for the data acquisition system. A DUT definition consists of:

- The number of output signals (the **Active** channels)
- Number of pixels and lines per frame for each output signal (the “**Output Size**”)
- Plot direction for each signal (**Vertical** or Horizontal)
- Plot position for each signal (**Position**)
- Horizontal and vertical interval between consecutively-clocked pixels for each channel (**Delta**)
- Total size of the array (the “**FPA Size**”)

Data channels may be given user-defined names for convenience.

In test plans with multiple Master Groups or Independent cards, each Master Group or Independent card will have its own FPA definition within the **DEFINE** mnemonic.

3.7.1.1. Multiple Configurations

Multiple **DEFINE** mnemonics may reside within a single test plan file to permit switching among different readout configurations for a given device. **DEFINE** mnemonics may also be given user-defined names to assist in test planning, e.g. the name of the device or the name of a specific readout configuration.

Individual data channels may be activated or deactivated for each **DEFINE** mnemonic in the test plan. No changes to physical cabling or to the **Connections** property page are necessary for most test-planning, even as DUTs are changed or reconfigured.

The **DEFINE** mnemonic must contain definitions for all active Master Groups (see below) or Independent cards in use. If data channels are not in use, they should be turned Off (**Active: No**) in the **DEFINE** mnemonic.

For example, a large-format CCD may have various readout configurations permitting output via 1, 2, 4 or 8 output pins on the device. Different timing files and **DEFINE** mnemonics can be used for each readout scheme and selected from within a single test plan file.

3.7.1.2. Multiple Acquisition Cards

In systems with multiple acquisition cards, the cards can be set up to collect single synchronous data set, or to collect one data set per card. The synchronous option permits the PI-3105 to synchronously acquire data for up to 32 channels.

The latter configuration permits data from multiple devices to be acquired simultaneously. The devices may have different channel counts, orientations, and sizes. If your system has more than one Pattern Generator, more than one Timing and Control card and more than one AIM, the devices also may be clocked asynchronously.

Each device should be cabled to a different Master Group or Independent card.

3.7.2. Master/Slave setup

For each **DEFINE** mnemonic in a test plan, each card's Master/Slave/Independent status must first be specified.

A card may have one of three possible configurations, based both on its Connections and on the status assigned to it in the **DEFINE** mnemonic.

	Arming cable (cabling and Connections window)		
	Master	Slave	No connection
Possible configurations	<ul style="list-style-type: none"> • Master w/Slaves • Master w/o Slaves 	<ul style="list-style-type: none"> • Slave • Independent 	<ul style="list-style-type: none"> • Master w/o Slaves

Each configuration has different acquisition scenarios available to it:

- **Master with slaves**—(a Master Group) acquires data synchronously on all active channels attached to it and its configured Slaves. All Slaves must have at least one active channel, and all active channels on Master and Slaves must receive appropriate clocking. Slaves without active channels will be marked as Inactive, and do not require clock signals.
- **Master w/o slaves**—acquires data for all active channels attached to it.

- **Independent**—acquires data for all active channels attached to it
- **Slave**—acquires data on all active channels attached to it, but only when armed by a Master with active channels. Data cannot be acquired on a Slave if its Master has no active channels; it must first be promoted to **Independent**.

Click on the **DEFINE** mnemonic to select it, and then click on the **Master/Slave** tab. Each currently defined Master, Master Group or Independent card will appear in the **Master/Slave Tree** by slot number. Master cards will be indicated by a large **M** icon, Independent cards by a large **I** icon, and Slave cards by a large **S** icon:

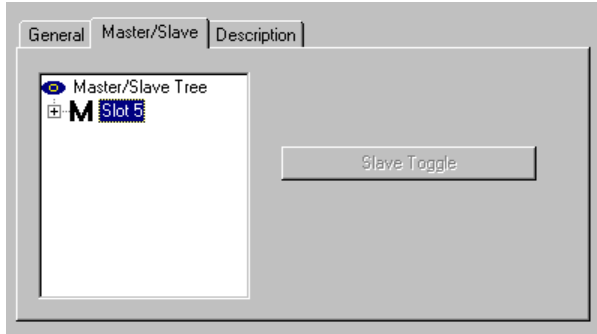


Figure 26: Master/Slave Tree with Master Group

If necessary, click on the + sign on a **Master Group** to view its configured slaves.

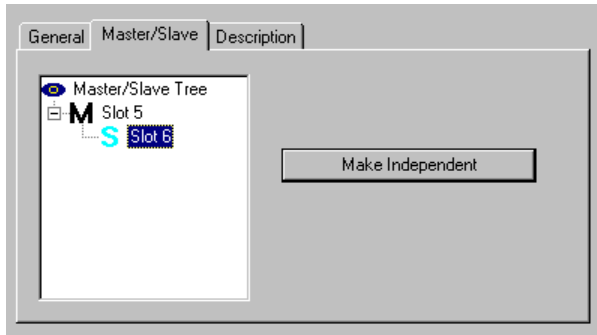


Figure 27: Master with Slave

To turn a **Slave** card into an **Independent** card, select its **Slot** number and click **Make Independent**. The card will be “promoted” in the tree and its icon changed to **I** to reflect its **Independent** status. Changes that have not yet been written to hardware will appear in blue. Click **Write** to apply the changes. All written values will turn black.

Command equivalent:

```
DAQCARD 6 ↵
SLAVE FALSE ↵
```

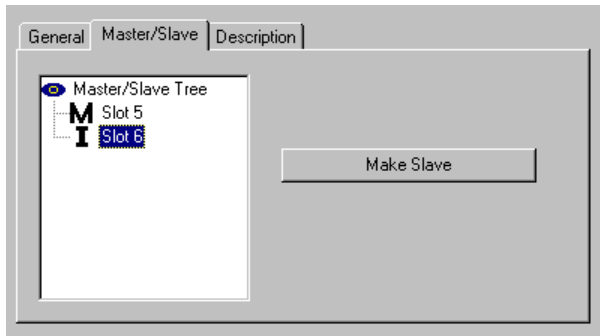


Figure 28: Master and Independent

To turn an **Independent** card into a **Slave**, select its Slot number and click **Make Slave**. The card will be “demoted” in the tree and its icon changed to an **S** to indicate its **Slave** status.

Command equivalent:

```
DAQCARD 6 ↵
SLAVE TRUE ↵
```

The number of Slaves in use by a Master card and/or a card’s Slave/Independent status may be changed across **DEFINE** mnemonics, even within a given test plan file.

3.7.3. FPA size and DACQ card selection

Click the **General** tab to view the focalplane definition. The **DEFINE** mnemonic lists all connected data channels in the system. If a data channel does not appear in the **DEFINE** mnemonic, then you must use the **Connections** property page to **Connect** it.

The **Output Size** (and therefore the **FPA Size**) must be defined for each Master Group or Independent card. A Master Group is identified by the slot number of its Master card, and an Independent card is defined by its slot number. Slaves do not appear in the **Dacq Slot** drop-menu since their settings are inherited from their Master card.

To view the **Output Size** and **FPA Size** for a given Master Group or Independent, select its slot from the **Dacq Slot** drop-menu:

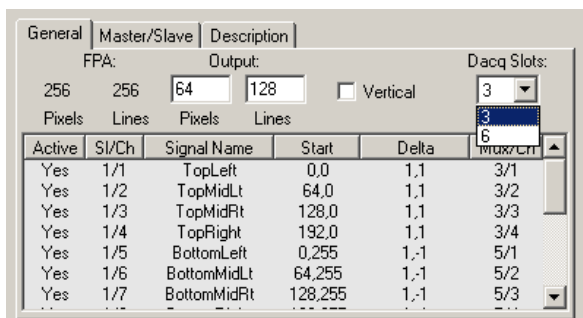


Figure 29: DACQ Slot Selection

The individual channels connected to that Master Group or Independent card will be highlighted in gray in the channel list below.

Enter the size of each output (x,y) in the **Output Size** box, check or un-check the Vertical box as appropriate (see below), then enter the output parameters of each channel (see below). Output size

and FPA size are both numbered using 1-based numbering (e.g. a VGA sensor is 640 x 480). Note that output size is specified per video output, and may not be the same as the device's pixel size if the device has multiple outputs.

3.7.4. FPA Orientation

If the **Vertical** box is checked, this indicates that each subsequent pixel clock increments in the vertical direction (y), and that each subsequent line clock increments in the horizontal (x) direction. If the **Vertical** box is un-checked this indicates that each subsequent pixel clock increments in the horizontal direction (x), and that each subsequent line clock increments in the vertical (y) direction. See the FPA definition examples, below.

3.7.5. Maximum FPA Size

The maximum FPA size supported by the PI-3105 is dependent on the manner in which the channels are connected to the Data Acquisition cards. Each data port on each card is capable of counting up to 65,536 pixels and 65,536 lines per frame. Pixels from active multiplexed channels are added together (but lines are not) for purposes of counting maximum FPA size.

For example, if an FPA is acquired into a single ADC channel connected to single port on a Data Acquisition card, it may have either dimension of up to 65,536, subject to on-board memory limitations. An FPA acquired via 4 channels multiplexed into one port of a DACQ may have only 16,384 pixels per line per output, because the pixel counts are added together.

3.7.6. Channel Parameters

Each connected data channel has the following parameters on the **General** property page:

Parameter	Description
Active	Yes =Data is collected (clocks required) and plotted. No = Channel is not used (no clocks required).
Signal Name	User-defined name for this channel.
Start	Plotted (physical) address of the first pixel clocked out of this channel. 0-based addressing (e.g. a VGA sensor "starts" at (0,0) and ends at (639, 479)
Delta	Determines the x and y increment after each pixel clock or line sync. Delta (x,y) may contain positive or negative values; if the Delta x or Delta y is negative, then the pixels will increment in the negative direction on each line or pixel clock. Use a Delta of ($\pm 1, \pm 1$) for contiguous definitions. Use values other than ± 1 for interleaved definitions (see Section 3.8. Channel Definition Examples below).

Table 3: FPA Definition Parameters

Channels not in use by any Master or Independent should be turned off (**Active = No**).

Once all settable parameters have been entered, click **Write**. Settings that are the same as the last set written to hardware will appear in black text. Settings that have changed will appear in blue. The **Status** bar will change to **Channel definitions written** unless there was an error in the definitions, in which case the **Status Bar** will report the error. If the definition was written correctly, the **FPA Size** will report the total active area of the defined array for the selected **DACQ slot**.

To define multiple arrays for multiple device testing, use the **DACQ Slot** menu to switch to each Master group and then enter its FPA and channel parameters.

All channels set to **Active = Yes** must be outputting valid clock signals (e.g. the data channel must be installed in the AIM with a timing signal routed to it and the data cable routed back to the CompactPCI mainframe). The PI-3105 uses counters to determine when the frame has been successfully collected. If an FPA definition is set for more pixels, lines or frames than are clocked into the system, then the acquisition will not complete.

Any channels not in use for this FPA definition must be set to Off (**Active = No**).

3.7.7. FPA Definition Constraints

Data channels must be activated to obey the following constraints:

- A given Mux card can have any single channel activated, or
- A given Mux card can be set to one of the following multiplex modes:
 - 2,1
 - 3,2,1
 - 4,3,2,1
- A given DACQ card must have either its A port activated or both A and B activated. The B port cannot be used by itself.
- If both A and B ports on a DACQ are activated, they must have the same number of multiplexed data channels active on each.
- A **Slave** card may be activated only if its Master also is activated. To acquire on a **Slave** card without its **Master**, the Slave must first be promoted to an **Independent**.
- All channels connected to a active on a Master Group (**Dacq Slot**, including via multiplexers and/or configured slaves) must have the same output size, and the sum of all Active and Simulated channels connected to that **Dacq Slot** must make up a contiguous rectangular area with no gaps or overlapping pixels.
- **FPA Size**, **Output Size**, and **Orientation** may be different for each Master Group (**Dacq Slot**).

To collect and analyze data from a subset of channels that violates the above constraints (e.g. a single channel connected to a “B” port), activate additional channels to satisfy the constraints, then use the Area-of-Interest feature (below) to isolate only the desired data.

3.8. Channel Definition Examples

The following diagrams will illustrate common output configurations for 4- and 8-channel devices. We will use the following conventions and definitions for describing focalplanes and plotted images:

- **Definitions:**
 - **Pixel clock:** timing signal used to strobe the ADC for each pixel on a video channel.
 - **Pixel count:** number of pixels to be acquired per line for each video output. Lines may be plotted vertically or horizontally.
 - **Line Sync:** timing signal to indicate the start of a valid line. A valid line sync must be followed by at least **pixel count** pixel clocks before the next valid line sync.
 - **Line count:** number of lines per video output to be acquired per frame.
 - **Frame Sync:** timing signal to indicate the start of a valid frame. A valid Frame Sync must be followed by at least **line count** line syncs before the next valid Frame Sync.
 - **Pixel count** and **Line count** refer to the data as it is clocked off the array, regardless of how it is plotted.
 - **Column**, **Row**, **x** and **y** refer to the data as it is plotted to the screen.
- **Conventions:**
 - Sizes (e.g. pixel count, line count, FPA size) use 1-based numbering. A VGA-sized focalplane has 640 pixels and 480 lines.
 - Plotted pixel locations are 0-based. A VGA image begins at pixel (0,0) and ends at pixel (639, 479).
 - The origin of a plot (0,0) is at the upper-left. The positive x direction is to the right, and the positive y direction is toward the bottom of the image. Plotted pixels are addressed as (x,y).
 - Pixels are sequentially numbered for each channel in the order that they are acquired (e.g. the order in which they are clocked off the device).

- In the following diagrams, a white box will indicate the plotted position of the first pixel for a channel, a gray box the second pixel, and a black box the last pixel. Numbers inside the boxes indicate the ADC channel from which the pixel was acquired.
- Pixel addresses are indicated for the first and last pixel from each channel and, if necessary, the upper-left and lower-right corners of the array.

3.8.1. Contiguous FPA Definitions

In the following examples, each video output on the device corresponds to a contiguous rectangular area on the array.

3.8.1.1. Columnar Device, Vertically Clocked from Lower Left Corner

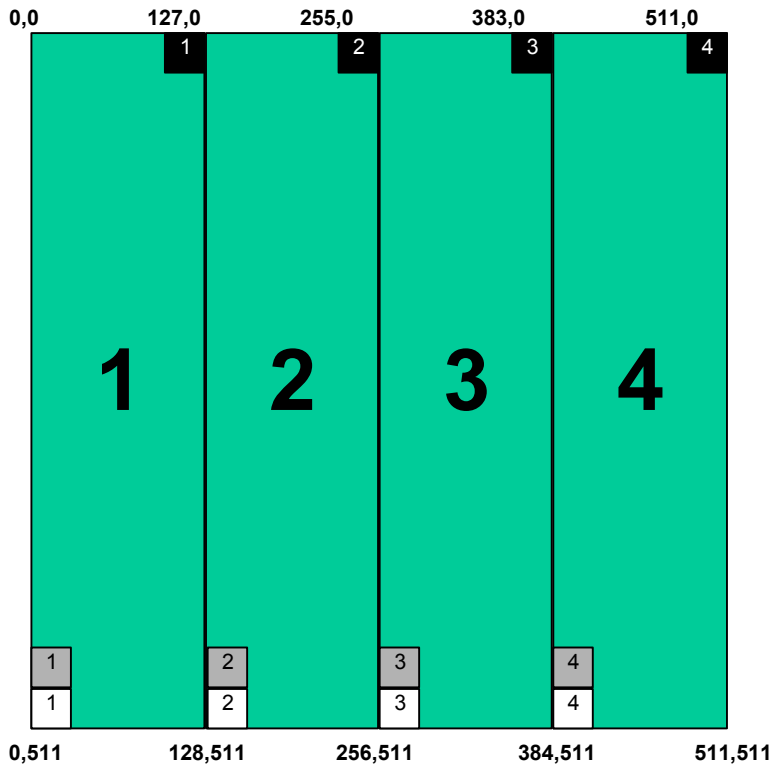


Figure 30: Contiguous FPA Definition, Vertically Clocked

In this example, the **FPA size** is 512 x 512, **Output Size** is 128 x 512, **Vertical** is checked, and the channel setup would be as follows:

Active	Signal Name	Start	Delta
Yes	Ch 1	0,511	1,-1
Yes	Ch 2	128,511	1,-1
Yes	Ch 3	256,511	1,-1
Yes	Ch 4	384,511	1,-1

Table 4: Parameters for Contiguous FPA Definition, Vertically Clocked

The number of pixel and line clocks required for each channel is determined by the **Output Size** and the **Vertical** checkbox. If **Vertical** is checked, then each pixel clock corresponds to an increment in the y direction; therefore the **y** value from the **Output Size** is the minimum number of pixel clocks required per line, and the Output Size **x** is the minimum number of line syncs required per frame.

Because the **Vertical** box is checked, each pixel clock corresponds to an increment in the vertical plotted direction. When setting up clock signals for this device, each channel should have 512 pixel clocks per line and 128 line clocks per frame.

Command equivalent:

```

DAQCARD 6 ↵
OUTPUT X128 Y512 ↵
VERTICAL TRUE ↵
SET ACTIVE CHAN 1-4 ↵
CHAN 1 ↵
POSITION 0,511,1,-1 ↵
CHAN 2 ↵
POSITION 128,511,1,-1 ↵
CHAN 3 ↵
POSITION 256,511,1,-1 ↵
CHAN 4 ↵
POSITION 384,511,1,-1 ↵

```

Note that the device output may be rotated, transposed, or flipped along either axis by changing the plot locations in the **DEFINE** mnemonic.

3.8.1.2. Columnar Device, Horizontally Clocked from Lower Left Corner

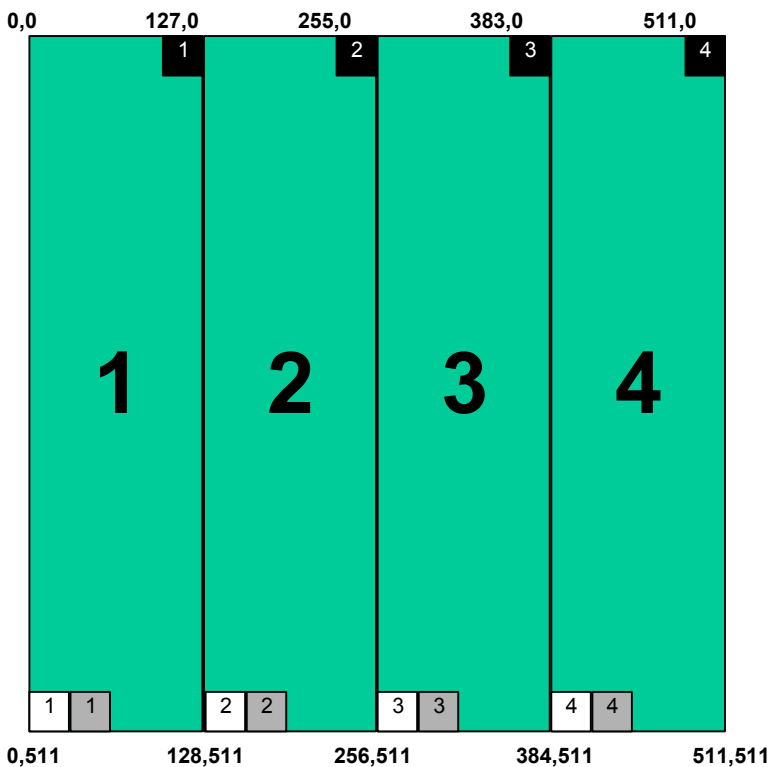


Figure 31: Contiguous FPA Definition, Horizontally Clocked

In this example, the **FPA size** is 512 x 512, **Output Size** is 128 x 512, **Vertical** is un-checked, and the channel setup would be as follows:

Active	Signal Name	Start	Delta
Yes	Ch 1	0,511	1,-1
Yes	Ch 2	128,511	1,-1
Yes	Ch 3	256,511	1,-1
Yes	Ch 4	384,511	1,-1

Table 5: Parameters for Contiguous FPA Definition, Vertically Clocked

The number of pixel and line clocks required for each channel is determined by the **Output Size** and the **Vertical** checkbox. If **Vertical** is un-checked, then each pixel clock corresponds to an increment in the x direction; therefore the **x** value from the **Output Size** is the minimum number of pixel clocks required per line, and the Output Size **y** is the minimum number of line syncs required per frame.

Because the **Vertical** box is un-checked, each pixel clock corresponds to an increment in the horizontal plotted direction. When setting up clock signals for this device, each channel should have 128 pixel clocks per line and 512 line clocks per frame.

Command equivalent:

```

DAQCARD 6 ↵
OUTPUT X128 Y512 ↵
VERTICAL FALSE ↵
SET ACTIVE CHAN 1-4 ↵
CHAN 1 ↵
POSITION 0,511,1,-1 ↵
CHAN 2 ↵
POSITION 128,511,1,-1 ↵
CHAN 3 ↵
POSITION 256,511,1,-1 ↵
CHAN 4 ↵
POSITION 384,511,1,-1 ↵

```

3.8.2. Interleaved FPA Definitions

In the following examples, each video output on the device corresponds to a non-contiguous region on the array. There is a regular interval, greater than one, in at least one direction, between sequential pixels on each channel.

3.8.2.1. Alternating Lines

Some devices are clocked out using interleaved lines. For example, in **Figure 32** below, each data channel corresponds to a series of horizontal lines across the FPA.

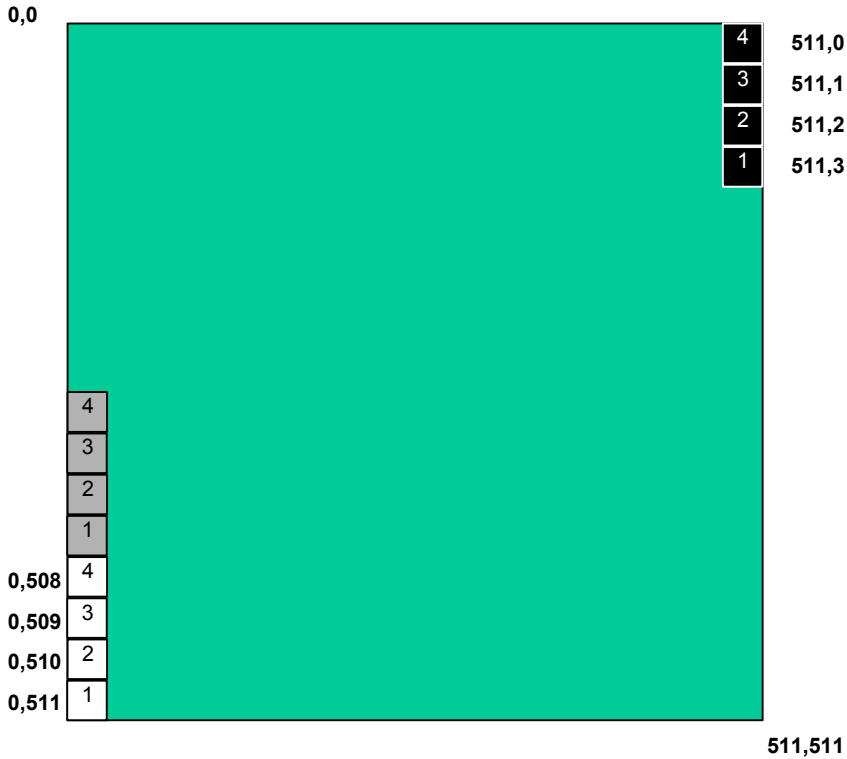


Figure 32: Interleaved FPA Definition, Alternating Lines

In this example, the **FPA size** is 512 x 512, the **Output Size** is 512 x 128, **Vertical** is checked, and the channel setup would be as follows:

Active	Signal Name	Start	Delta
Yes	Ch 1	0,511	1,-4
Yes	Ch 2	0,510	1,-4
Yes	Ch 3	0,509	1,-4
Yes	Ch 4	0,508	1,-4

Table 6: Parameters for Interleaved FPA Definition, Alternating Lines

Each channel spans the entire array. All **Delta** values are (1,-4). When setting up clock signals for this device, each channel should have 128 pixel clocks per line and 512 line clocks per frame.

Command equivalent:

```

DAQCARD 6 ↵
OUTPUT X512 Y128 ↵
VERTICAL TRUE ↵
SET ACTIVE CHAN 1-4 ↵
CHAN 1 ↵
POSITION 0,511,1,-4 ↵
CHAN 2 ↵
POSITION 0,510,1,-4 ↵
CHAN 3 ↵
POSITION 0,509,1,-4 ↵
CHAN 4 ↵
POSITION 0,508,1,-4 ↵

```


3.8.2.2. Interleaved Grid

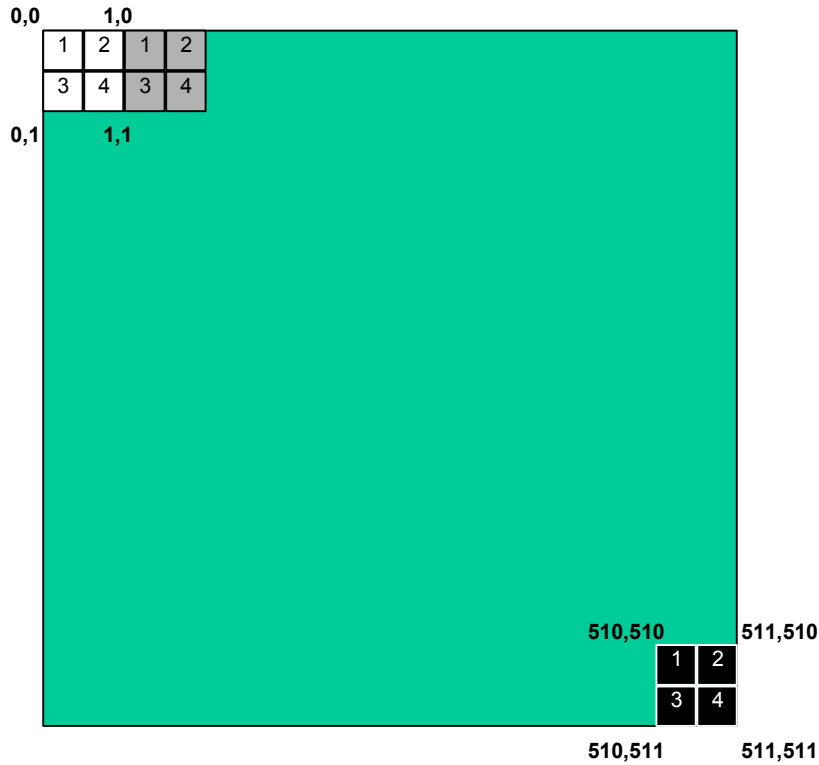


Figure 33: Interleaved FPA Definition, Interleaved Grids

In this example, the **FPA size** is 512 x 512, the **Output Size** is 256 x 256, **Vertical** is un-checked, and the channel setup would be as follows:

Active	Signal Name	Start	Delta
Yes	Ch 1	0,0	2,2
Yes	Ch 2	1,0	2,2
Yes	Ch 3	0,1	2,2
Yes	Ch 4	1,1	2,2

Table 7: Parameters for Interleaved FPA Definition, Interleaved Grid

Each channel spans the entire array. All **Delta** values are (2,2). When setting up clock signals for this device, each channel should have 256 pixel clocks per line and 256 line clocks per frame.

Command equivalent:

```
DAQCARD 6 ↵  
OUTPUT X256 Y256 ↵  
VERTICAL FALSE ↵  
SET ACTIVE CHAN 1-4 ↵  
CHAN 1 ↵  
POSITION 0,0,2,2 ↵  
CHAN 2 ↵  
POSITION 1,0,2,2 ↵  
CHAN 3 ↵  
POSITION 0,1,2,2 ↵  
CHAN 4 ↵  
POSITION 1,1,2,2 ↵
```

3.8.3. Stitching Across Multiple Data Acquisition Cards

Several applications require the use of more than one data acquisition card for a single FPA definition (e.g. a Master/Slave(s) configuration):

- Devices with more than 4 outputs
- Devices with very high data rates per output channel
- Applications with very deep acquisition memory depth requirements

FPA definition is largely the same as with smaller configurations:

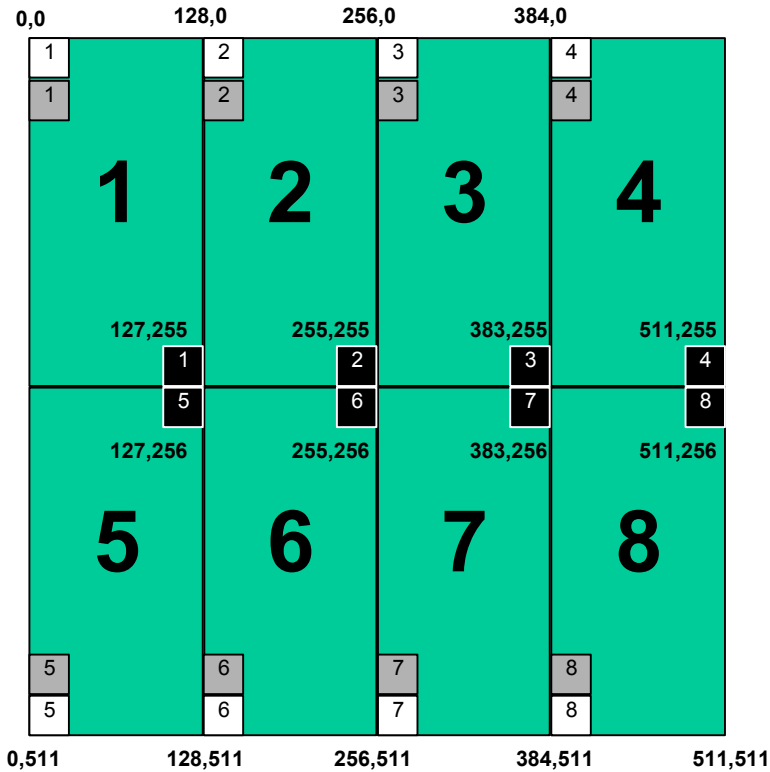


Figure 34: 8-Channel Device, Contiguous Regions

In this example, the **FPA size** is 512 x 512, the **Output Size** is 128 x 256, **Vertical** is checked, and the channel setup would be as follows:

Active	Signal Name	Start	Delta
Yes	Ch 1	0,0	1,1
Yes	Ch 2	128,0	1,1
Yes	Ch 3	256,0	1,1
Yes	Ch 4	384,0	1,1
Yes	Ch 5	0,511	1,-1
Yes	Ch 6	128,511	1,-1
Yes	Ch 7	256,511	1,-1
Yes	Ch 8	384,511	1,-1

Table 8: Parameters for 8-Channel FPA Definition

Note that all eight channels must be in a Master Group with one of the following configurations, reflected both in the physical cabling and in the **Connections** window.

- One DACQ card with two Mux Cards, fully populated
- Two DACQ cards each with two Mux cards, each with channels 1 and 2 active
- Four DACQ cards, each fully populated

Note: For the two- and four- DACQ card configurations, one DACQ card must be connected and configured as a **Master** and the remainder connected and configured as **Slaves**.

Each channel has a magnitude of 128 columns x 256 rows. When setting up clock signals for this device, each channel should have 256 pixel clocks per line and 128 line clocks per frame.

Command equivalent:

```
DAQCARD 5 ↵
OUTPUT X128 Y256 ↵
VERTICAL TRUE ↵
DAQCARD 6 ↵
OUTPUT X128 Y256 ↵
VERTICAL TRUE ↵
SLAVE TRUE ↵
DAQCARD 7 ↵
OUTPUT X128 Y256 ↵
VERTICAL TRUE ↵
SLAVE TRUE ↵
DAQCARD 8 ↵
OUTPUT X128 Y256 ↵
VERTICAL TRUE ↵
SLAVE TRUE ↵
SET ACTIVE CHAN 1-8 ↵
CHAN 1 ↵
POSITION 0,0,1,1 ↵
CHAN 2 ↵
POSITION 128,0,1,1 ↵
CHAN 3 ↵
POSITION 256,0,1,1 ↵
CHAN 4 ↵
POSITION 384,0,1,1 ↵
CHAN 5 ↵
POSITION 0,511,1,-1 ↵
CHAN 6 ↵
POSITION 128,511,1,-1 ↵
CHAN 7 ↵
POSITION 256,511,1,-1 ↵
CHAN 8 ↵
POSITION 384,511,1,-1 ↵
```

3.8.4. Acquiring Across Multiple Masters

If you have only a single PI-41000 Digital Acquisition Card in your system, or if all your PI-41000 cards will always be operated synchronously, you may ignore this section.

Several applications require the simultaneous acquisition of data from two or more devices:

- Simultaneous capture of different spectra
- Simultaneous capture from different positions or angles
- Simultaneous capture with different integration times or gain/sensitivity
- Simultaneous capture with a vertical-scanning linear array and a horizontal scanning array

This requires at least one data acquisition card per device. If the devices are clocked asynchronously with respect to each other, then at least one AIM and Timing card is required for each timing domain.

In this example, two full complements of acquisition hardware reside in the same system:

- Two Timing and Control cards
- Two AIMS, one connected to each Timing card
- 4 ADC channels and preamplifier channels installed in each AIM
- Two Data Acquisition cards, each with a Multiplexer card
- Two sensors:
 - Sensor 1 with 4 outputs, acquired through AIM 1 into DACQ3 via a multiplexer
 - Sensor 2 with 2 outputs, acquired through AIM 2 into DACQ6 via a multiplexer

Assuming that all hardware connections have been properly cabled and specified in the Connections window, the **DEFINE** mnemonic may be set up:

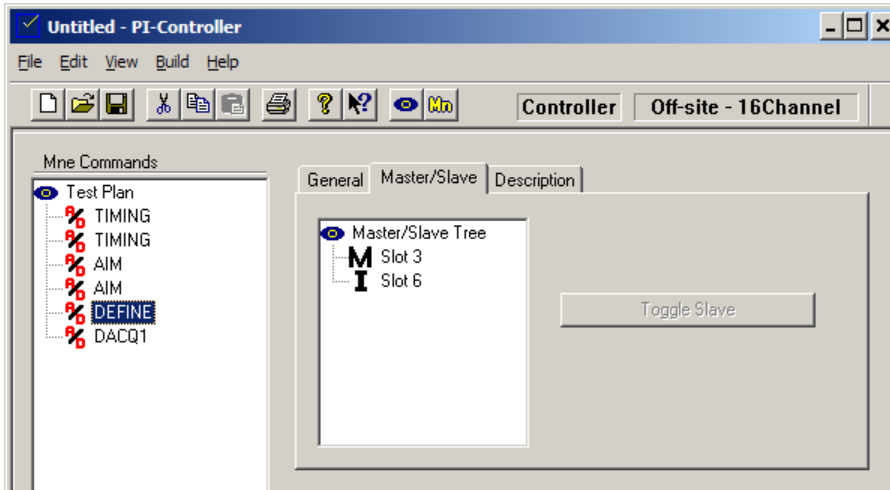


Figure 35: Multiple Device Setup, Master/Slave Configuration

Because the data being acquired by the two data acquisition cards will not be synchronous and will not be stitched into one image, there will be no slaves. If the arming cable is connected, we can choose not to use it by configuring **Slot 6** as Independent. If the arming cable is not connected, then Slot 6 will be a Master.

Command equivalent:

```
DAQCARD 6 ↵
SLAVE FALSE ↵
```

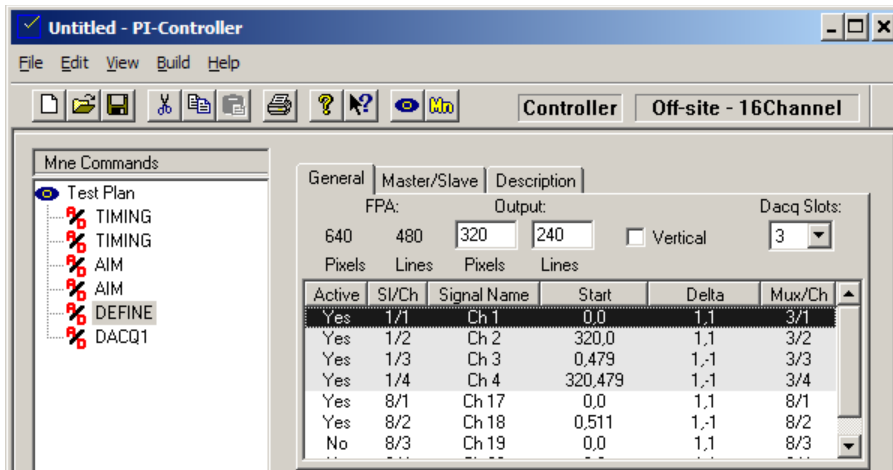


Figure 36: Multiple Device Setup, FPA 1 definition

Because we now have two separate acquisition groups, we have two FPAs to define. In this example, FPA 1 has an **FPA size** of 640 x 480, it has 4 outputs, and it will be acquired into Slot 3.

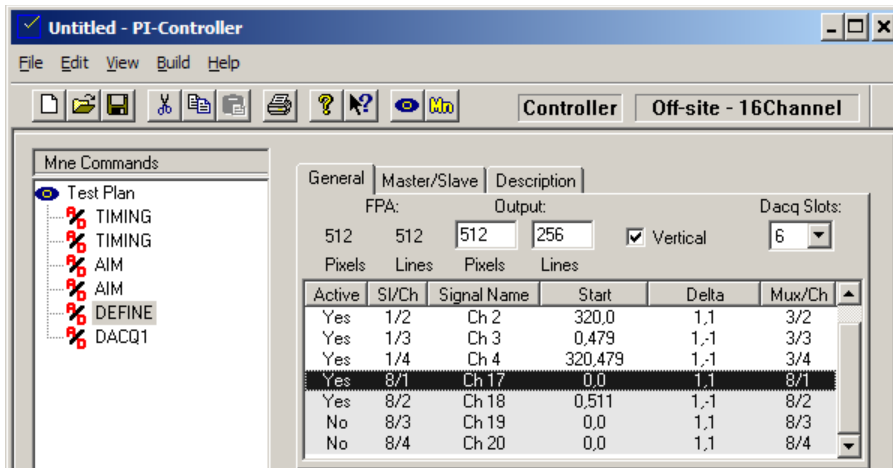


Figure 37: Multiple Device Setup, FPA 2 Definition

To set up FPA 2, we click on the **Dacq Slots** menu to bring up the **FPA size** for Slot 6. In this example, FPA 2 has a size of 512 x 512, it has 2 outputs, and it will be acquired into Slot 6. Note that, although Slot 4 has 4 channels cabled to it and connected in the **Connections** window, we only use 2 of those channels for this definition. Therefore, Channels 19 and 20 must be turned off (Active = No).

Command equivalent:

```
DAQCARD 3 ↵  
OUTPUT X320 Y240 ↵  
VERTICAL FALSE ↵  
CHAN 1 ↵  
POSITION 0,0,1,1 ↵  
CHAN 2 ↵  
POSITION 320,0,1,1 ↵  
CHAN 3 ↵  
POSITION 0,479,1,-1 ↵  
CHAN 4 ↵  
POSITION 320,479,1,-1 ↵  
DAQCARD 6 ↵  
OUTPUT X512 Y256 ↵  
VERTICAL TRUE ↵  
SLAVE FALSE ↵  
CHAN 17 ↵  
POSITION 0,0,1,1 ↵  
CHAN 18 ↵  
POSITION 0,511,1,-1 ↵  
SET ACTIVE CHAN 1-4, 17, 18 ↵
```

Note that both FPA definitions reside in the same **DEFINE** mnemonic. Use the **Dacq Slots** drop-menu to toggle the display between the **FPA Sizes** for each FPA. When the **Write** button is clicked, all FPA, card and channel settings are written to hardware, regardless of which FPA definition is currently displayed.

3.9. DACQ mnemonic

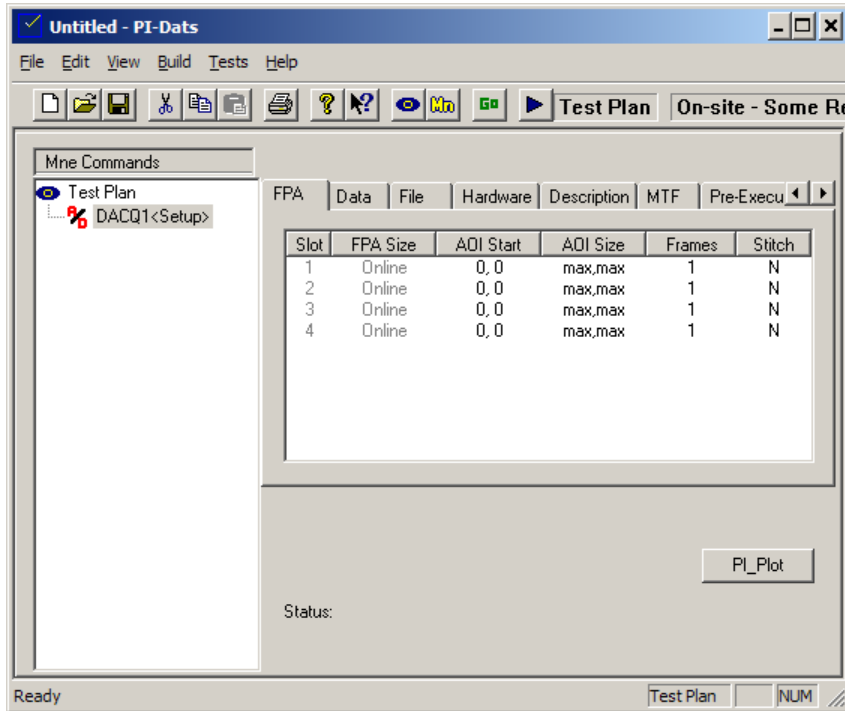


Figure 38: DACQ mnemonic

Figure 38 shows the data acquisition property page for the PI-41000 Digital Acquisition Card. This property page is used to control hardware setup and trigger data acquisition for all data acquisition cards in the system.

The **DACQ** mnemonic should not be used until the **DEFINE** mnemonic has been edited and written to hardware. If your system includes ADC components, then the **TIMING** and **AIM** mnemonics also should have been written to hardware.

The **DACQ** mnemonic contains 7 property pages and several general controls that are always visible

- **FPA property page**—Reports the **FPA Size** currently defined for each data acquisition card, and controls the amount of data to be collected (area of interest and framecount). Also controls whether data from multiple channels will be “stitched” or not.
- **Data property page**—Controls the format and type of data to be collected (bit-width, integer, floating point conversion), and whether or not to reduce the acquired data to a single frame via averaging.
- **Post-processing property page**—Not documented or supported at this time. Will be implemented in a future release to support user-defined post-processing routines via a user-provided DLL.
- **File property page**—Control the path/filename and format of the data file to be saved, when data are to be saved, and whether or not to over-write existing data files.
- **Hardware property page**—Controls the clock input settings for each data port on each card.
- **Description property page**—reports the model, memory size, and physical location for each card installed in the system.
- **Simulation property page**—Not documented or supported at this time.

3.9.1. General Controls

The general controls are always visible, regardless of which property page is currently being displayed.

- **Start Acq button**—arms all active cards in the system for data acquisition
- **PI-Plot button**—Launches PI-Plot for viewing acquired data or previously-saved data files.
- **Save File button**—Saves acquired data to a file. This button is used in Local mode only. In Remote mode, data are automatically saved to a file after every acquisition cycle.
- **Memory button**—Internal use only.
- **Status bar**—reports the current state of the acquisition cards.

3.9.1.1. Start Acq/Stop Acq button

To acquire data, click the **Start Acq** button. The **Start Acq** button will change into a **Stop** button, and the status area message will change to **Waiting for Acquisition**. All Master Groups and Independent cards with active channels will begin acquiring data upon receipt of a valid Frame Sync pulse. The acquisition time is determined by the clock rate of your device, the integration time in your timing pattern, and the size of the acquisition you have specified. There is no time-out period in PI-Controller; it will wait as long as required to acquire the programmed acquisition size. A programmable time-out period is available only in PI-DATS or when scripting the DLL directly.

If the size of the data acquisition is larger than the amount of available memory on the data acquisition cards, the Status bar will report an error instead.

If multiple Master Groups or Independents are active, the **Status** bar will report which cards are pending after each card or group completes.

Command equivalent:

STARTACQ ↵
ACQSTATUS ↵

The card must receive the programmed number of pixel clocks per line, the programmed number of line syncs per frame and the programmed number of frame syncs in order to complete the acquisition. To stop a data acquisition before it has completed, click the **Stop** button.

Command equivalent:

STOPACQ ↵

Once the specified number of frames has been acquired, DMA transfer occurs at a rate of approximately 90 MB/second, plus a small constant. DMA and data processing may take up to 5-6 seconds, depending on the size of your acquisition and the current load on the system bus and memory. Stitching and floating point conversion can take as long as DMA transfer, especially for large acquisitions.

If the size of the data acquisition is larger than the amount of available contiguous system RAM, data will be transferred instead into a small DMA buffer in RAM, and then immediately written to disk, frame by frame. If a valid filename has been specified, the data will be written to that file. If no filename has been specified, the data will be written to a temp file. Systems running 32-bit Windows cannot typically allocate more than 1.2 GB of contiguous RAM, even if the amount of physical RAM installed in the computer is much larger.

Acquired data in a temp file are available to PI-Plot only when run on the local CPCI bus. If PI-Controller is running on a different CPU, controlling the PI-3105 over GPIB or another control bus, a valid filename must be specified that resolves to the same file location from both CPUs, e.g. a network volume or drive-mapped network share.

Once data transfer has completed, the **Start Acq** button will be re-enabled.

Note that all acquisition settings on all property pages are written to the hardware only when a data acquisition is triggered. Therefore, if settings are changed while an acquisition is in progress or waiting, they will not take effect until the next acquisition is triggered. This is particularly true when an acquisition does not complete due to an incorrect setting. For example, if an acquisition is triggered with the clock source set to an input with no clocks, the acquisition system will not complete. Changing the clock source while the acquisition system is waiting will not cause the acquisition to complete. It must be stopped and then restarted for the clock source change to take effect.

3.9.1.2. PI-Plot button

Launches PI-Plot for viewing acquired data or a previously saved file. See **Section**

4. PI-Plot

3.9.1.3. Save File button

Saves acquired data to a file or set of files. When clicked, each Master or Independent card with active channels and a defined save-to filename will have its data written to disk. This button is used in Local mode only. In Remote mode, data are automatically saved to a file or files after every acquisition cycle.

Command equivalent:

```
SAVEFILE ↵
```

The data will be written to disk as a short header followed by raw data. The header of the file is in the following format.

```
struct IMGSTRUCT
{
    char Title[8];
    long x;
    long y;
    long nFrames;
    long nType; //0 = float, 1 = word
    long version;
    long lPixelSkip;
    long lLineSkip;
    long numOfBits;
    long lChannels;
    float fPixelPeriod; //in nanoseconds
    float fGain;
    float fOffset;
    long lMTFInfo;
    float fPixelPitch;
    //new items for version 4
    DWORD dwDataStart; //starting offset to the data = IMGSTRUCT
    float fVmin; //minimum input voltage to AD
    float fVmax; //maximum input voltage to AD
    BOOL bCDSenabled; //TRUE if CDS was enabled;
};
```

The raw data is stored following the header, row by row, in the order that the pixels were acquired.

If multiple frames have been acquired, the user may choose to average all frames into a single frame by choosing **Y** in the **Avg Frames** column before the acquisition is triggered.

If multiple frames of data have been acquired, but not averaged, the first pixel of Frame 2 will follow the last pixel of Channel 1.

The following are the first several lines of a sample data file, written with the **Float** data type and **ASCII** format:

PI-Asc
Pixels = 64
Lines = 64
Frames = 1
File Type = 0
Version = 4
Pixel Skip = 0
Line Skip = 0
Num of Bits = 14
Channel = 0
PixelPeriod = 250.000
Gain = 1.0
Offset = 0.000
NumMTFSegments = 0
SizeofSegment = 12
PixelPitch = 6.000
DataStart = 00315
VMin = -2.000000
VMax = 2.000000
CDSEnable = 0
0.0000
0.0146
0.0291
0.0437
0.0583
0.0728
0.0874
0.1021

3.9.2. FPA Property Page

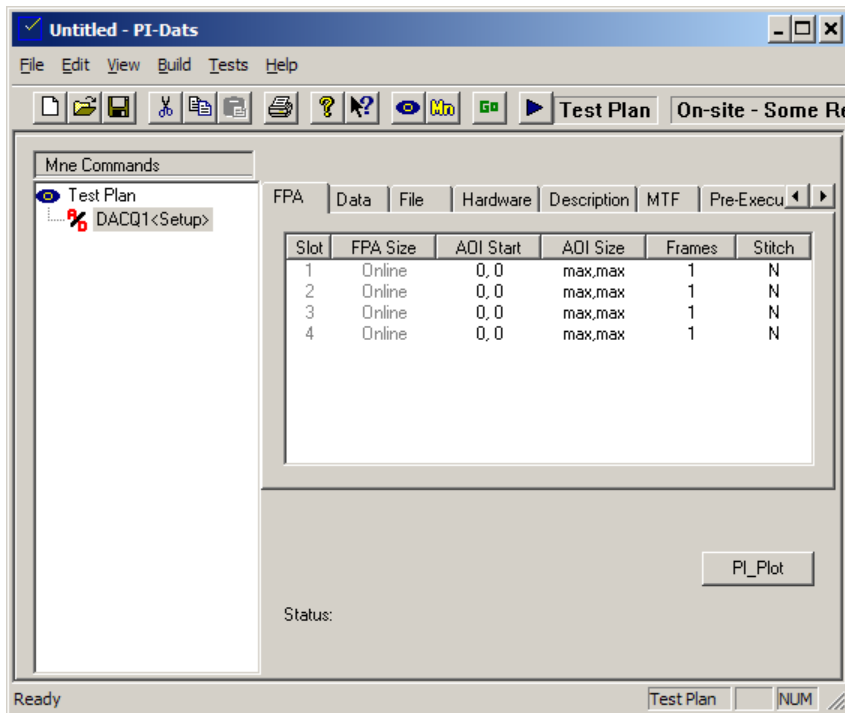


Figure 39: Data Acquisition mnemonic, FPA Property Page

The **FPA** property page displays one line of settings for each active Master or Independent card in the system. Cards currently in Slave mode are shown simply as “**Slave**,” with no visible settings. Slave cards inherit all relevant FPA settings from their Master. Inactive cards are marked with “**Inactive**.”

The card **Slot** and **FPA Size** are shown for each card. These items are informational, and cannot be edited here. To change the **FPA Size** for a card, use the **DEFINE** mnemonic to edit the active channels and/or output size and position for each channel connected to this card.

3.9.2.1. AOI Start and AOI Size

The Area-of-Interest (**AOI**) can be used to reduce the acquired frames to a rectangular subset. This can be used to isolate data from a single channel or region, reduce file size, or increase the speed of data analysis. Note that the AOI feature reduces the data set after it has been acquired in hardware, and therefore the AOI feature does not increase the maximum number of contiguous frames that can be acquired by the system.

The **AOI Start** argument specifies the first (upper-left) pixel to be retained. Pixels have zero-based numbering; therefore the first possible pixel address is 0,0. “0,0” is also the default setting for **AOI Start**.

The **AOI Size** argument specifies size of the area to be retained. The AOI Size uses one-based numbering; therefore a VGA-sized frame is 640,480. The argument “**max**” may be used in place of either or both numbers, and will resolve to the width and/or height of the FPA as defined by the last-written DEFINE mnemonic. “**max,max**” is the default setting for **AOI Size**.

If the AOI lies outside the current FPA definition, it will return the error, “**AOI Outside FPA**.”

Command equivalent:

DAQ CARD 2 ↵
AOI 100,100,max,max ↵

3.9.2.2. Frames

To set the number of frames, enter an integer in the **Frames** column for each active master. Each time the **Start Acq** button is clicked, all acquisition cards will be set up and armed. Once armed, they will start acquiring data upon receipt of the first **Frame Sync** signal, and continue until the requested number of frames, lines, and pixels has been collected. Early versions of the PI-41000 Digital Acquisition Card can acquire up to 8192 frames. Later version of the PI-41000 (Rev.1 cards with LED indicators) can acquire up to 65,536 frames, subject to memory capacity.

Command equivalent:

DAQCARD 5 ↵
FRAMECOUNT 100 ↵

3.9.2.3. Stitch

To control the Stitching function, choose **Y** or **N** in the **Stitch** column. If Stitching is set to **Y**, acquired data will be re-arranged according the **Start** and **Delta** arguments in the **DEFINE** mnemonic. If spatial location of data is not important, or if data are to be stitched by another application, stitching can be set to **N**, and acquired data will be in its raw order, determined by hardware location. Data will be interleaved, or “scrambled” in memory (see Appendix x for details) and will not be directly useful for imaging, but sequential data acquisitions can be executed at a faster rate since the processing burden on the CPU is lower.

Command equivalent:

DAQCARD 5 ↵
STITCHING TRUE ↵

3.9.3. Data Property Page

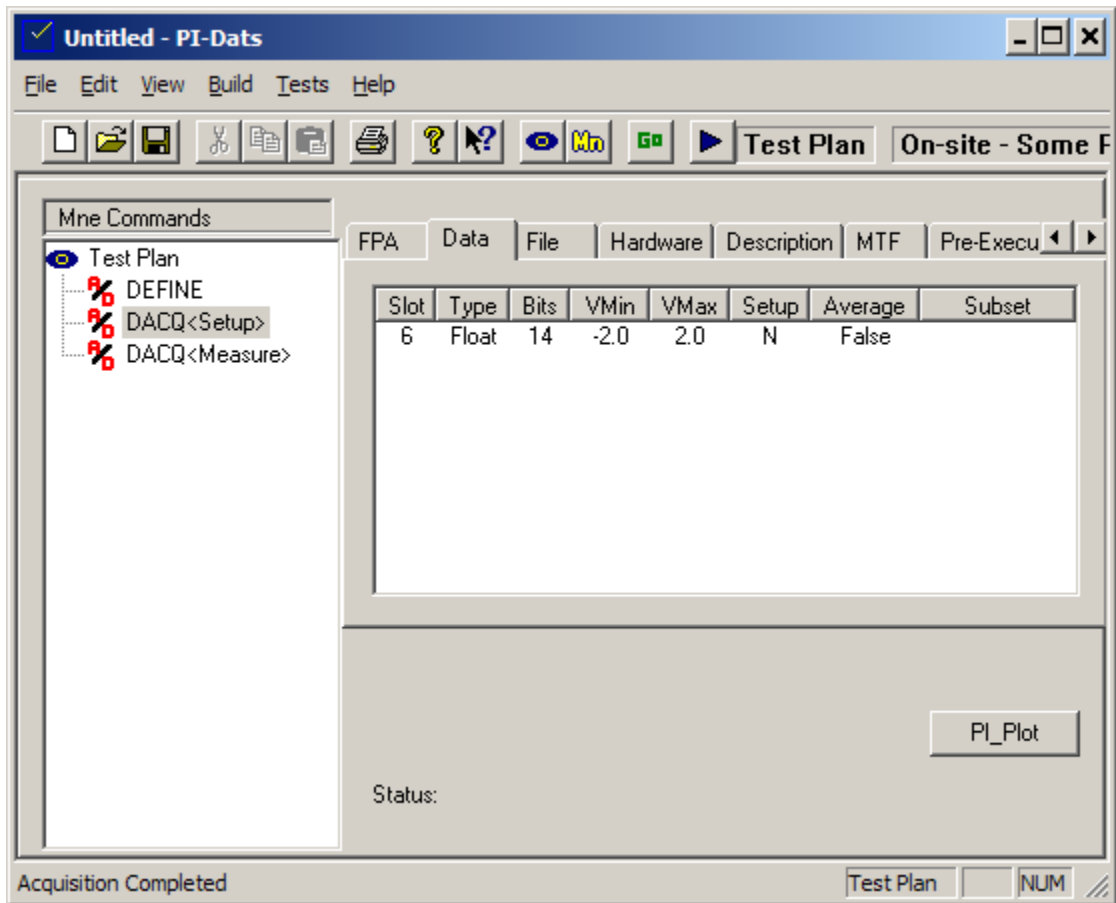


Figure 40: Data Acquisition mnemonic, Data Property Page

The **Data** property page displays one line of settings for each active Master or Independent card in the system. Cards currently in Slave mode are shown simply as “**Slave**,” with no visible settings. Slave cards inherit all relevant data settings from their Master. Inactive cards are marked with “**Inactive**.”

Use the **Data** property page to set the type of data conversion to perform.

3.9.3.1. Data Type

Select **Int** to acquire and save data as Integer data corresponding to ADC counts. Integer data are stored as right-aligned 16-bit words, with unused MSBs masked to 0.

Command equivalent:

```
DAQCARD 5 ↵  
FLOAT FALSE ↵
```

To convert the acquired data to 32-bit floating point, select **Float** before triggering the data acquisition. Acquired data will be scaled between a minimum value of **VMin** and **VMax** based on the bit-width of the data specified by the **Number of Bits** drop-menu and the current **Setup** mode setting. Please note that this doubles the amount of system memory required to analyze and process data. It will also slow down the displayed frame rate during continuous acquisition. It will not affect the

maximum number of frames that can be captured in hardware, nor will it affect the raw data capture rate into the system.

Command equivalent:

```
DAQCARD 5 ↕  
FLOAT TRUE ↕
```

3.9.3.2. Bits

Use this drop-menu to specify the bit-width of the incoming data. Acquired data less than 16 bits wide are right-aligned to the least-significant bit (e.g. the maximum value for 14-bit data is 0x3FFF). The **Bits** setting controls a hardware mask on each data acquisition card. Data above Bit **n** are masked low, to prevent non-driven data lines from corrupting the data set. **Bits** should always be set to the actual bit-width of your data. This setting is used also when acquired data are converted to Floating Point via the **Float** item in the Data **Type** drop-menu.

Command equivalent:

```
DAQCARD 5 ↕  
BITSPERPIX 14 ↕
```

3.9.3.3. VMin

Use this entry to specify the voltage corresponding to the smallest possible value from the ADC. For the PI-41040 14-bit ADC module in Normal mode (e.g. with CDS turned off) this value should be set to -2.0 V, which is also the default setting for this parameter. For the PI-41040 with CDS turned on this value should be set to -4.0 V. This value will be written to the data file header. If the Data Type is set to Float this value will also be used to perform floating-point conversion.

Command equivalent:

```
DAQCARD 5 ↕  
VMIN -2.000 ↕
```

3.9.3.4. VMax

Use this entry to specify the voltage corresponding to the largest possible value from the ADC. For the PI-41040 14-bit ADC module in Normal mode (e.g. with CDS turned off) this value should be set to 2.0 V, which is also the default setting for this parameter. For the PI-41040 with CDS turned on this value should be set to 4.0 V. This value will be written to the data file header. If the Data Type is set to Float this value will also be used to perform floating-point conversion.

Command equivalent:

```
DAQCARD 5 ↕  
VMAX 2.000 ↕
```

3.9.3.5. Setup Mode

Note: Setup mode is can not be turned off at this time. This feature will be added in a future release.

If **Setup** mode is **Y** (True), data are converted to floating point as seen at the input to the ADC, including all programmed gain and offset. Use **Setup** mode to set up your test application and ensure you are achieving the maximum possible dynamic range without clipping.

If **Setup** mode if N (False), data are converted to floating point with correction for programmed gain and offsets. This will reflect the signal at the input to the preamplifier module. Use this mode to see the signal as output by your DUT.

Command equivalent:

```
DAQCARD 5 ↵  
SETUP MODE FALSE ↵
```

3.9.3.6. Average Frames

Use the **Avg** setting to control whether and how multiple frames of acquired data are averaged into a single frame of data before saving and display. The system can be set up to average all frames, not to average any frames, or to average a subset of frames.

Command equivalent:

```
DAQCARD 5 ↵  
AVERAGE TRUE ↵
```

To average a subset of frames, choose **Subset** from the drop-menu, and then enter the first frame and the number of frames to be averaged in the Subset area. Frames are numbered from 1, and the argument “**max**” may be used in place of the second value. If the number of frames to be averaged is larger than the number of frames available for averaging (e.g. FirstFrame + NumFrames > AcquiredFrames), then the averaging routine will average out to the last frame. The default setting “**1,max**” is equivalent to choosing **Average: True**.

Command equivalent:

```
DAQCARD 5 ↵  
AVERAGE SUBSET 5, max ↵
```

3.9.4. Post Property Page

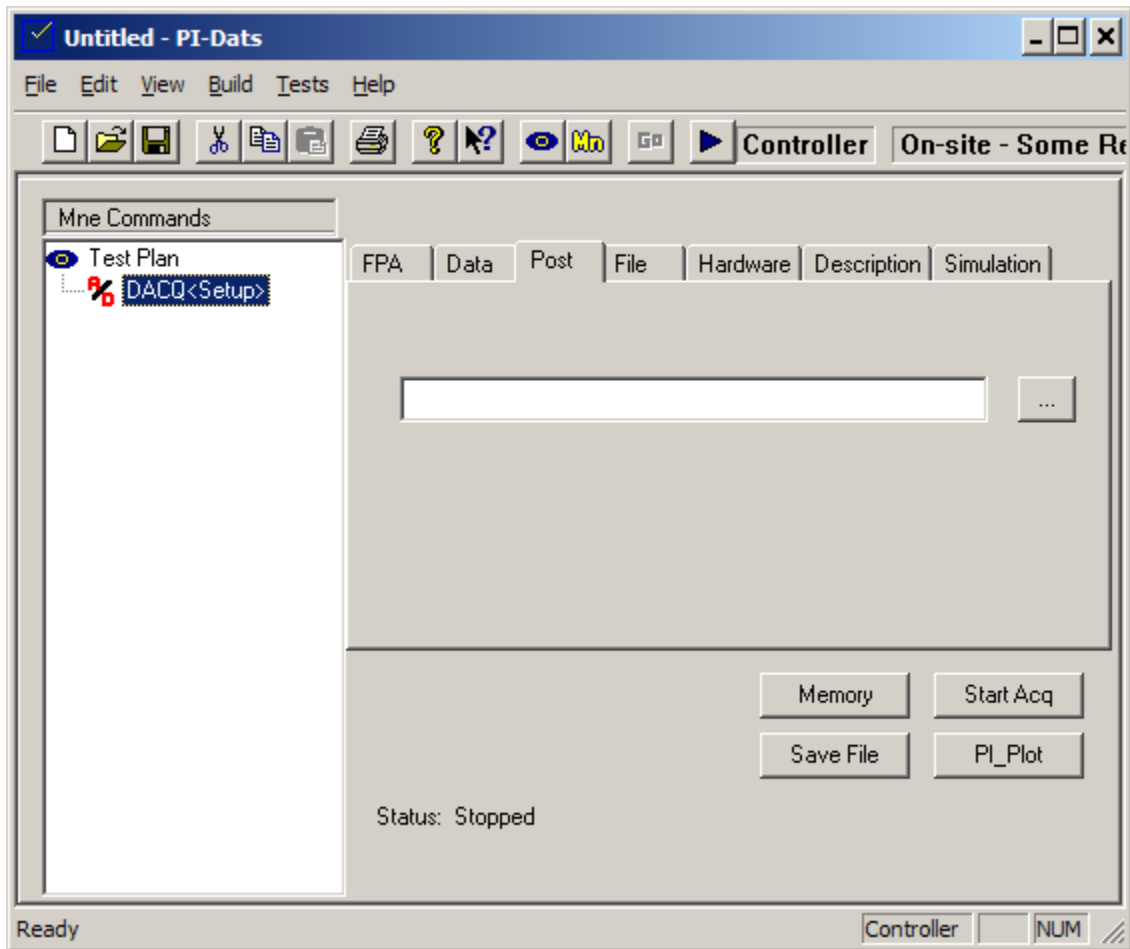


Figure 41: Data Acquisition mnemonic, Post Property Page

The **Post** property page is used to specify user-defined post-processing routines via user-provided DLLs. This topic is explored more fully in **Section 3.10. In-line Post-processing**.

3.9.5. File Property Page

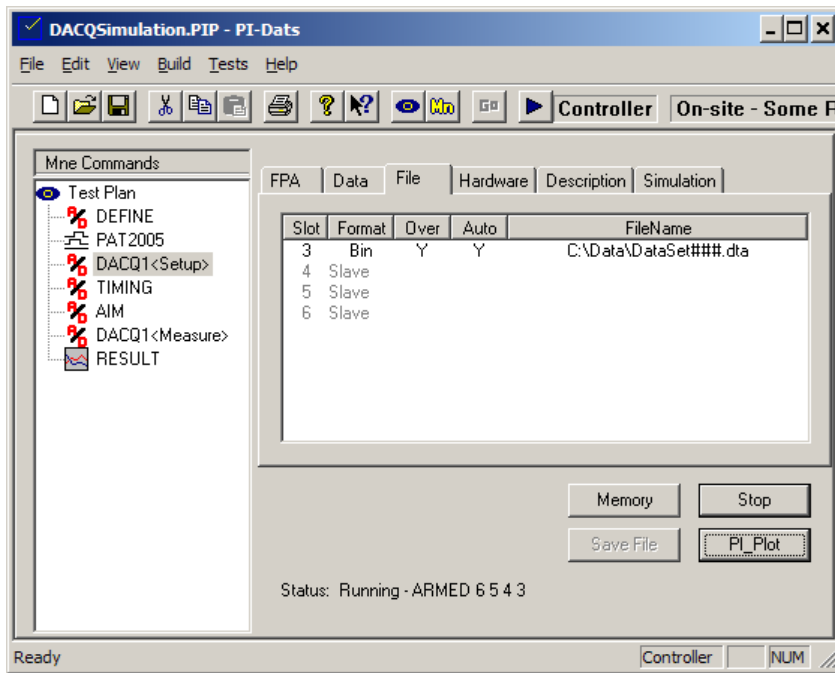


Figure 42: Data Acquisition mnemonic, File Property Page

The **File Property Page** controls where and how the acquired data will be saved.

When operated in Remote mode, data are saved to a file after every acquisition cycle. If no filename is specified in this box, a Windows Temp file will be used, and PI-Plot will not be able to open the file automatically for real-time, continuous acquisition. If a path and filename are specified, data will be saved to that file, and PI-Plot will attempt to read from it.

When operated in Local mode, data will be acquired to CPU memory whenever possible. If data are acquired into CPU memory, it will not be saved to a file unless **Save File** is clicked by the user. If insufficient CPU memory is available to hold the entire dataset, then data will be automatically saved to a file after every acquisition. If no filename is specified in the **FileName** box, a Windows Temp file will be used.

3.9.5.1. Format

If **Bin** is chosen, files will be saved in PI-DATS binary format, which can be read back into PI-DATS, PI-Plot, or a third-party application. If **Asc** is chosen, files will be saved as text with a short header. Text files should be used for troubleshooting or informational purposes only, as they are very slow to write, and do not retain the full precision of the binary format in **Float** mode.

3.9.5.2. Auto

When **Auto** is set to **True**, acquisition data will be saved to the path and file specified by the **FileName** argument (see below) each time an acquisition is complete, subject to the **Overwrite** setting (see below).

3.9.5.3. Over

When PI-Controller/PI-DATS attempts to save a file, either via the **Save File** button or the **Autosave** feature, it will use this setting to determine whether or not to overwrite an existing file with the same path and filename.

3.9.5.4. FileName

Use the Filename box to specify a path and filename for the data. By convention, PI-Plot uses a “.dta” extension, but any extension may be chosen by the user.

Note: the path and filename are relative to the Local (CompactPCI) CPU. In order for PI-Plot on the Remote computer to read the file, the path and filename must be the same as seen from the Remote computer. This is possible with a network-based URI, e.g. `\\fileserver\acquired\data.dta` or with mapped network volume, e.g. `e:\acquired\data.dta`, where the e: drive is same mapped volume on both computers.

PI-DATS and PI-Controller have an automatic incrementing filename option. If the filename argument (either fixed or using a string variable) contains # characters, those characters will be replaced by incrementing numbers each time a file is saved. The number of # characters determines how many digits the file number will have. If the # sign(s) are preceded by a number, the file index will begin with that number; otherwise it will start at zero. When the maximum file number has been reached (e.g. 999 for ###) the file index will wrap around to zero.

For example, the filename argument “C:\Data\TheFile5###.dta” will result in the following files being created on successive saves:

- C:\Data\TheFile005.dta
- C:\Data\TheFile006.dta
- C:\Data\TheFile007.dta
- etc.

until C:\Data\TheFile999.dta is reached, which would then be followed by C:\Data\TheFile000.dta.

The following should be noted when using the automatic incrementing feature:

- The filename will increment every time a file is saved, either via the **Save File** button or via the auto-save feature.
- The starting file number is set each time the filename string is sent to the DLL. This occurs each time the **DACQ Setup** mnemonic executes (in PI-DATS) or when the **Start Acq** button is pressed in PI-Controller.
- Data acquisitions can be triggered without sending a new filename string either via the DACQ Measure mnemonic in PI-DATS, or via the Go button in PI-Plot.
- The Continuous acquisition feature in PI-Plot, along with the Auto Save and this auto numbering feature, can consume very large amounts of hard drive space very quickly. Exhausting available disk space on the Windows startup volume can result in system crashes or sluggish performance.

3.9.5.5. Format

Select **Bin** to save data as 16-bit or 32-bit binary words. Select **Asc** to convert data to an ASCII representation. ASCII data sets can be 4-5 times as large as their corresponding binary representation, and are much slower to save.

Command equivalent:

```
DAQCARD 5 ↵  
SAVETOFILE \\FILESERVER\ACQUIRED\DATA.BIN, ASCII ↵
```

3.9.6. Hardware Property Page

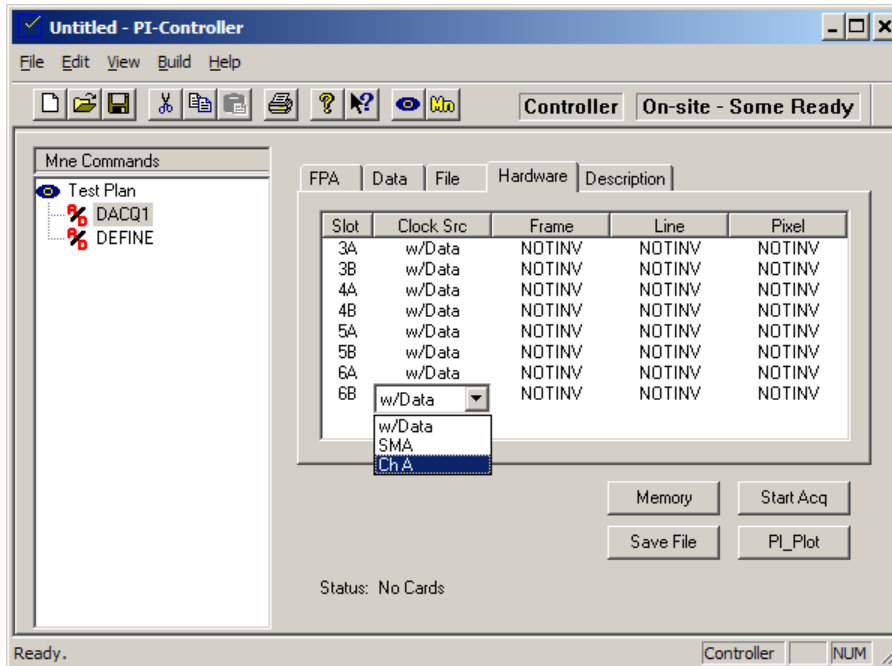


Figure 43: Data Acquisition mnemonic, Hardware Property Page

If clocking is being supplied to the Data Acquisition cards by a PI-3105 with or without multiplexer cards, this entire section can be skipped.

3.9.6.1. Clock Source Selection

The PI-41000 requires three clock signals:

- **Pixel clock**—Data are latched into the on-board memory and the pixel counter is decremented on every rising edge of the pixel clock.
- **Line sync**—The line counter is decremented and the pixel counter is reset on every valid line sync.
- **Frame sync**—The frame counter is decremented and the line counter is reset on every valid frame sync.

There are two clock inputs available for each data channel on the card. By default, the card is configured to accept LVDS clocks on the Data connectors (ribbon cable). The Hardware property page provides a drop-menu control to switch the clocks from **w/Data** to **SMA**. In the latter mode, the card accepts TTL clocking on the SMA connectors.

If you are configuring a B port, you can use the **Ch A** selection to share clocks with Channel A and reduce the number of external timing signals required:

Command equivalent:

```
DAQCARD 4 ↵  
PORT A ↵  
CLOCK SRC DATA ↵  
PORT B ↵  
CLOCK SRC DATA ↵  
DAQCARD 5 ↵  
PORT A ↵  
CLOCK SRC DATA ↵  
PORT B ↵  
CLOCK SRC SHARE ↵
```

Please see **Section 3.9.7. Timing Requirements** for important information on setting up timing correctly.

By default, the PI-41000 latches in data on the rising edge of the pixel clock. Frame and Line are treated as data, and trigger their counters when sampled High on the pixel clock. You can invert these settings by choosing the **Invert** setting on the drop-menus for **Frame**, **Line** and **Pixel**.

ALERT: The PI-3100-USB-M inverts the pixel clock relative to the Frame, Line and Data signals. Therefore the Pixel clock polarity must be set for Inv when used with the AIMS containing the multiplexer option. This setting is set by default when new mnemonics are added to a test plan, but test plans copied from previous systems may have this set to NONINV. If this setting is incorrect, multi-frame data sets will be corrupt after the first frame, and temporal noise measurements will be invalid.

Command equivalent:

```
DAQCARD 5 ↵  
PORT A ↵  
INVERT FRAME ↵  
NINVERT LINE ↵  
NINVERT PIXEL ↵
```

Your device or your timing generator must provide all the expected clocks. For example, each channel must output a number of pixel clocks and line clocks greater than or equal to the channel size defined in the **DEFINE** mnemonic, and there must be at least **Frames** frame sync signals in order to complete the acquisition.

3.9.7. Timing Requirements

The PI-41000 Digital Acquisition Card uses the incoming pixel clock(s) as its sampling strobe for data bits, frame sync and line sync. Frame sync and line sync can be treated as data bits for the purposes of timing analysis.

Note: Because Frame Sync and Line Sync are treated as data, transitions on these signals must occur within the data-valid window (e.g. sampled on the rising edge of the pixel clock).

3.9.7.1. Data Valid Window

The card is designed to sample data that is coincident with the pixel clock at the input connector. Inside the card logic the pixel clock is inverted and delayed by approximately 4 ns relative to the data bits (including Frame and Line). **Figure 44: PI-41000 Timing** shows the Clk/Data alignment assuming a pixel clock with a 50% duty cycle.

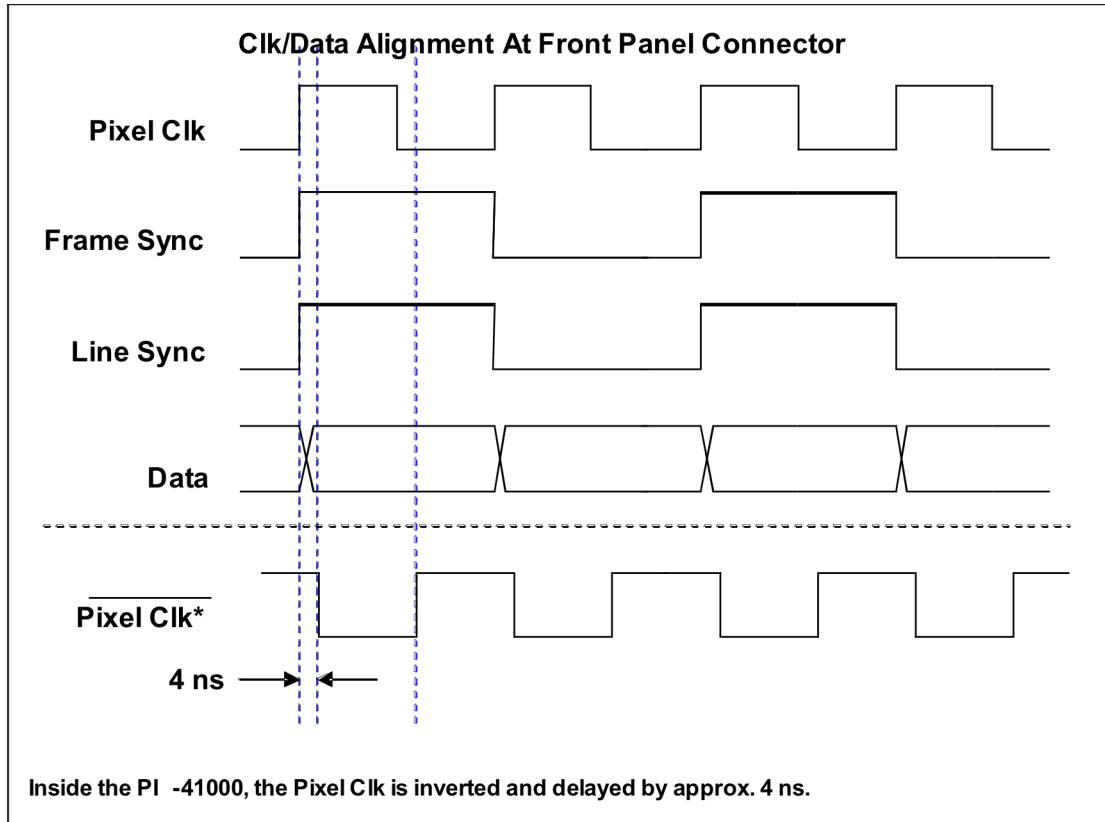


Figure 44: PI-41000 Timing Relationship

The frame sync pulse may extend beyond the clock period in which it is sampled because the PI-41000 does not begin counting frames again until the programmed numbers of line syncs and pixel clocks have been acquired.

Similarly, the line sync pulse may extend beyond the clock period in which it is sampled because the PI-41000 does not begin counting lines again until the programmed number of pixel clocks has been acquired.

3.9.7.2. Frame Sync, Line Sync, and First Valid Pixel

Receipt of a frame sync pulse resets the line counter before the line sync pulse is processed. Therefore, the frame sync pulse may be coincident with the first line sync pulse, and no vertical blanking is required between frames.

The pixel clock that samples the line sync pulse also samples the first valid data for that line, and no horizontal blanking is required between lines.

The first pixel of the second frame of data can immediately follow the last pixel of the first frame of data.

3.9.7.3. Multiplexed Data

The PI-3100-USBM with Multiplexer regenerates its own Pixel, Line, and Frame clocks based on an internal 80 MHz time-base. Therefore, it is not possible to acquire multiplexed data using external SMA clocks, since there is no way to synchronize to the timebase of the multiplexer.

3.10. In-line Post-processing

Post-processing routines are performed after completion of DMA and before display and saving of data; therefore post-processing routines will be usable for real-time image correction or real-time stitching of non-standard readout configurations. The Post-processing feature is available only in **PI-DATS**, though it will be available both in **Test Plan** mode and in **Controller** mode.

Because the algorithms are applied to the dataset before display, this feature can be used to modify “live” data during real-time display. Applications for in-line post-processing include, but are not limited to:

- Non-uniformity correction (NUC)
- Bad-pixel replacement (BPR)
- Stitching (reassembly) of readout formats not natively supported by PI
- Data reductions not otherwise provided by PI

Inline functions are called after data have been acquired into CPU memory and after the `dacq.dll` has performed the following transforms and reductions:

- standard image reassembly as programmed in the `DEFINE` mnemonic
- converting the data to float
- averaging the frames

The following operations are performed after inline functions have completed:

- Reduction to the programmed Area of Interest (AOI)*
- Saving data to file
- Passing the data to PI-Plot for display
- Passing the data to a subsequent `RESULT` mnemonic for further processing.

Versions of PI-DATS prior to 2.642 (Build 1453) performed the AOI reduction before the inline function calls, but presently the AOI reduction is called after the inline functions. Scripts and test plans written prior to version 2.642 may require editing to pass data of the correct size to the DLL. For example the **Two Point Non-Uniformity Correction (TwoPtNUC)** prior to 2.642 required the correction tables to be the same size as the final AOI, and thus changing the AOI required rebuilding of the correction tables every time. In 2.642 and after, the correction tables should be the size of the entire defined FPA, and the area of interest can be changed at any time.

In-line functions are called before data are sent to PI-Plot for display and before data are saved to file. Pre-processed data may no longer be available after the in-line function has been called; therefore if your application requires both pre- and post-processed data, your DLL should save the “raw” data to a file before it processes it or else you should post-process in the **Result** mnemonic after the raw data have been saved.

To maximize performance the dataset is passed to the in-line functions by pointer. This pointer is used throughout the data acquisition system software, therefore the same pointer must be returned by the in-line function, and no re-allocation of that memory is permitted.

3.10.1. Implementation

To implement custom in-line functions, a customer must write one or more DLLs to process data in Pulse Instruments’ documented DTA format and then return it in the same format. The data and some

metadata elements may be changed (e.g. the number of frames may be reduced), but the data header must be updated to match any changes, and the total size of the returned dataset must be the same as or smaller than the size of the input data set from the acquisition system.

Each DLL may contain one or more functions, each of which can take optional input parameters in addition to the data set.

The only required function in any user DLL is prototyped as follows:

```
extern "C" int GetFunctions(int nIndex, LPSTR strFuncProto,int nMaxStrLen);
```

A new function **GetFunctionsEx()** has been implemented to allow reading the names of the functions and the number of variables used by the function. The old **GetFunctions()** is still available for backward compatibility, and calls the new extended function. The new **GetFunctionsEx()** follows this prototype:

```
extern "C" int GetFunctionsEx(int nIndex, LPSTR strFuncProto,int nMaxStrLen, int* pNumVars);
```

It must be declared exactly as in the above declaration. This function passes to PI-DATS the names of the in-line functions the DLL exports. Users writing their own DLLs can copy and paste this function from the sample into their own DLL or just extend the functionality of the **functions.dll** sample supplied.

The user defined DLL must be placed in the “**functions**” sub directory to the Pulse Instruments application directory, by default:

```
C:\Program Files\Pulse20\Functions\
```

The user DLL must include the **InLineProcess.h** file for the structures definition. Each user function must be defined as follows:

```
extern "C" int UserFunction(void* ptrVoid).
```

The ptrVoid is cast to a **TRANSFER_DATA** structure that provides information about the acquisition data size and data type as well as a pointer to the acquisition data memory and pointers to any variable(s) that the function(s) may require. The name(s) of the function(s) can be user-defined but **GetFunctionsEx()** must report the name(s) of the function(s) exactly as the function(s) are named. Functions names are case sensitive.

Variables are passed in a linked list. Each record must contain the variable name, value, and type (e.g. “VT_R4” for a 4-byte real). A complete list of variable types is included in the table below. Each in-line function that needs a variable will have to step through the linked list to obtain the variables it needs before doing any processing. The user can add variables to the linked list if desired. If new variables are added to the linked list the memory allocated must use **malloc()** because **dacq.dll** uses **free()** to release the variables after all the inline functions have been called. Allocating memory with **new()** will cause memory leaks, so it is mandatory that memory be allocated with **malloc()**.

The contents of the data set can, of course, be modified, but if the frame size or number of frames is changed the **TRANSFER_DATA** structure must be updated with the new settings. **The resulting data set may not be larger than the input data set.** Memory for the data must not be released, reallocated or modified in any way that invalidates the pointer to memory, even if the resulting data set is smaller. If the resulting data set is smaller, updating the frame size and frame count in the header is sufficient to inform the rest of the system of how to analyze and/or display the data. The “excess” memory allocated to the data set will simply be ignored until the memory is freed by **dacq.dll**.

To assist the **GetFunctionsEx()** a new function has been added to allow getting the names and type of variables each function expects.

```
int GetVars(int nFuncIndex, int nVarIndex, LPSTR strVarName, int nMaxStrLen)
```

This **GetVars()** is called in a loop for each variable of each function. The return type is the type of variable.

To call the User DLL, a path and filename of an INI file must be passed to the **dacq.dll** during **DACQ Setup**, before an acquisition is triggered. The INI path and filename must be placed in the **Post** property page, and may be selected using the **Browse** button. The current implementation using an INI file is temporary; this will be replaced by a more elegant user interface in a future version of **PI-DATS**.

3.10.2. INI File

During the acquisition the **dacq.dll** reads the INI file to get the information about functions to call. All user DLLs to be used must be in the **functions** sub-folder since that is the only folder searched when **PI-DATS** is launched. If **PI-DATS** can't find the exported **GetFunctions()** or **GetFunctionsEx()** then the DLL is ignored and user functions will not be called.

The name of the DLL is used to permit multiple user DLLs to implement functions with the same name. For example **NUC|TheirDLL** and **NUC|MyDLL** can coexist in separate DLLs with out any conflict. The writer of **MyDLL.DLL** does not have to know the function **NUC** also exists in **TheirDLL.DLL**.

The INI file has just one section, **[INLINE]**, which must contain:

```
Enable = 0 | 1          Used to enable or disable the inline process
NumFunc = 1..N        Used to specify how many functions will be called
Function1 = "UserFunc|NameOfDLL" Name of function | Name of DLL file
Function2 = "UserFunc|NameOfDLL" Name of function | Name of DLL file
Function3 = "UserFunc|NameOfDLL" Name of function | Name of DLL file
FunctionN = "UserFunc|NameOfDLL" Name of function | Name of DLL file
NumUserData = number of user-defined data
UserDataName1 = variable name
UserDataName2 = variable name
UserDataName3 = variable name
UserDataNameN = variable name
UserDataName1 = type of data, which can be any value from the second column, below
UserData2 = Data to be passed in
```

For example, the following INI file directs PI-DATS to call the function **TwoPtNUC()** from **functions.dll** and passes in the two strings as specified.

```
[InLine]
Enable =1
NumFunc =1
Function1 =TwoPtNUC|functions.dll

NumUserData =2
UserDataName1 =Gain
UserDataName2 =Offset
UserDataName3 =STRING
UserDataNameN =STRING
UserData1 =c:\data\slopes.dta

UserData2 =c:\data\offsets.dta
```

The user DLL will then search for two strings by the **UserDataNames** **"Gain"** and **"Offset"**, and pass them to the **TwoPtNUC()** function.

Data Type	Data Type INI file entry	Description
VT_I4 = 3	INT	4 byte integer
VT_R4 = 4	FLOAT	4 byte real
VT_R8 = 5	DOUBLE	8 byte real
VT_UI2 = 18	WORD	2 byte unsigned
VT_UI4 = 19	DWORD	4 byte unsigned long
VT_UINT = 23	UINT	unsigned machine integer (4 bytes for Win32)
VT_LPSTR = 30	STRING	Null terminated strings

Table 9: Variable Types for Post-Processing

At the present time (PI-DATS version 2.647) all variable names are common across all user functions, so a variable name may only be declared once, and its declared value will be passed to all functions that use it. A variable's modified value cannot be passed back to another user function.

Once specified by executing **DACQ <Setup>** in Test Plan mode or by clicking **Start Acq** in Controller mode, the functions specified will be called each time the **DACQ <Measure>** is executed or each time the acquisition is triggered from **PI-Plot's "Go"** button. To change the DLL or functions to be used, execute **DACQ <Setup>** with a different INI file. To turn off all in-line processing, execute **DACQ <Setup>** with no INI file specified.

3.10.3. Configuration Utility

Pulse Instruments provides a utility, located at:

C:\Program Files\PULSE20\InLineFunc.exe

that will generate INI files for one or more functions from one or more DLLs. When launched, the **Inline Function INI Generator** will search **C:\Program Files\PULSE20\Functions** for valid DLLs and then present a function browser. Expand the tree below each DLL to see the available functions:

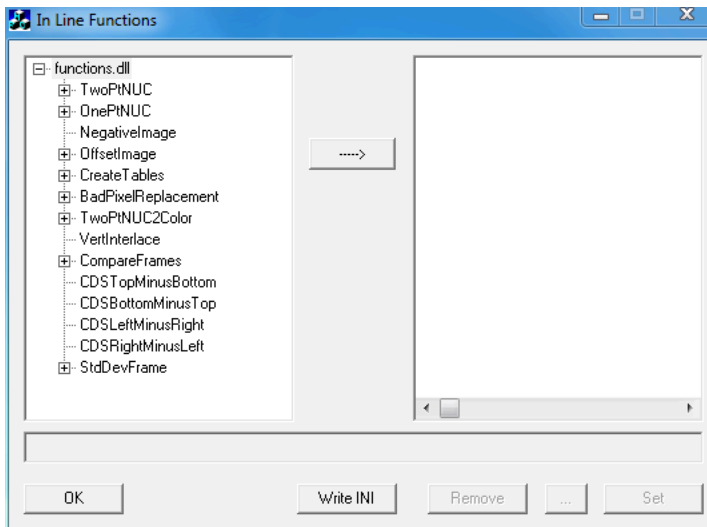


Figure 45: Configuration Utility, Function Selection

then expand the tree below each function to see what argument(s) each requires.

To build a script, select the function name and click the **Add Function** button (→). The function will be added to the list on the right side of the window.

Expand the tree below each added function to view and edit the argument(s) for each function:

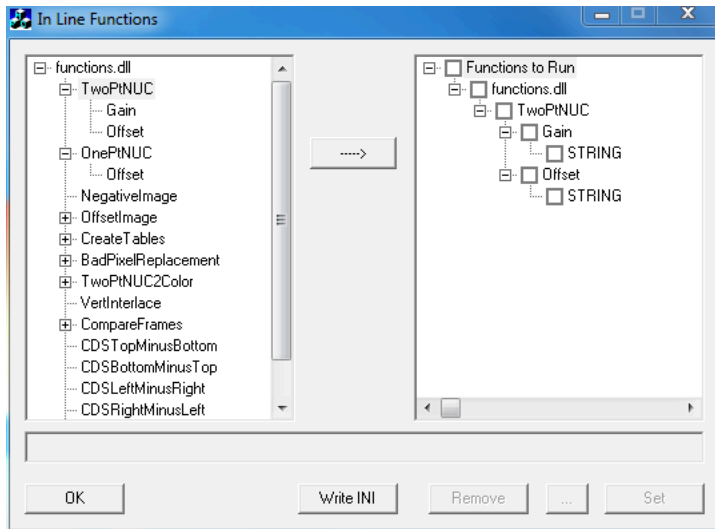


Figure 46: Configuration Utility, Arguments List

To enable a function, check the checkbox next to the function name. To edit an argument, select its value (which is a description of the argument type, by default, i.e. **STRING** in the example shown above), then edit its value in the text box below, and then click **Set**:

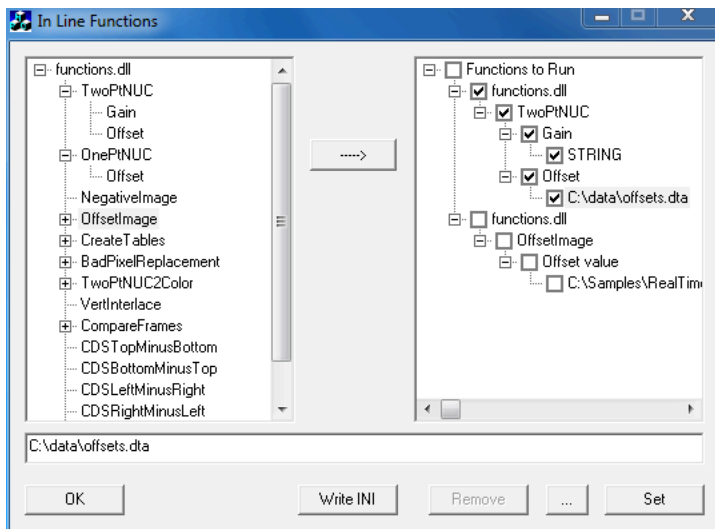


Figure 47: Configuration Utility, Argument Entry

The default value will be replaced by the entered value.

To remove a function from the list, select the name of the function in the right windows and click **Remove**.

Functions are executed in the order called by the INI file, so they should be added to the INI generator in the order they should be executed.

Once all the required arguments have been **Set** for all the desired functions, click **Write INI** to open a standard Windows file browser and save the INI file where desired.

3.10.4. Examples

A sample user DLL, INI file, and complete source code in a MSVC++ project have been supplied to demonstrate one implementation of some in-line functions, including a two-point Non-Uniformity Correction (NUC). The compiled DLL is located (as required) at:

C:\Program Files\Pulse20\Functions\Functions.dll

the MSVC++ project is located at:

C:\Program Files\Pulse20\CodeSamples\Functions

and the sample INI files are located at:

C:\Samples\RealTimeNUC

3.10.5. Support

For support with custom user functions, please contact support@pulseinstruments.com. Please email or upload to publicly-accessible server a complete source code project (Visual Studio 2010 or earlier) or at least a complete function implementation. Please include sample datasets as well. We can execute NDAs if necessary.

4. PI-PLOT

4.1. Introduction

PI-Plot is a companion application to PI-DATS/PI-Controller that provides a variety of ways to view your acquired image data graphically. PI-Plot features:

- Support for multiple, simultaneous views of the same data set, such as grayscale and histogram, with linked cursors to provide interactive, dynamic histogram equalization
- Support for multiple, simultaneous views of different data sets, such as “live” data and a reference data set
- Support for multiple-monitor systems
- Multi-threaded performance for multi-core systems

PI-Plot must be launched from the PI-DATS or PI-Controller toolbar or via the PI-Plot button on the **DACQ <Setup>** mnemonic:

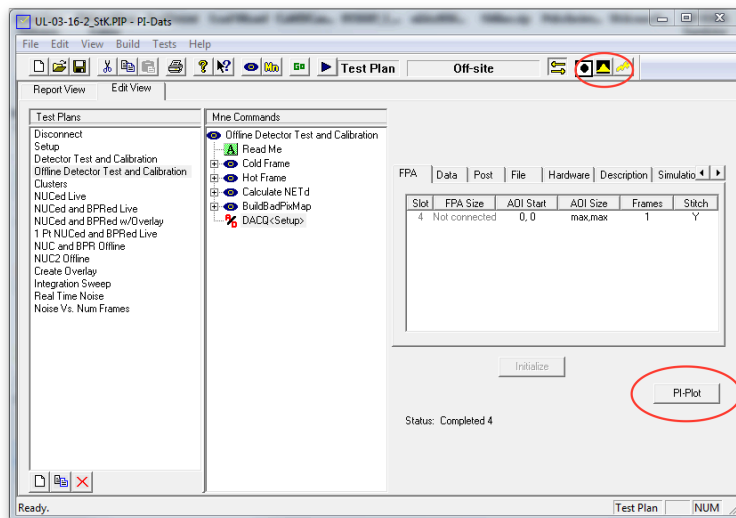


Figure 48: DACQ Mnemonic

Pressing either PI-Plot button will open the PI-Plot Main Window:

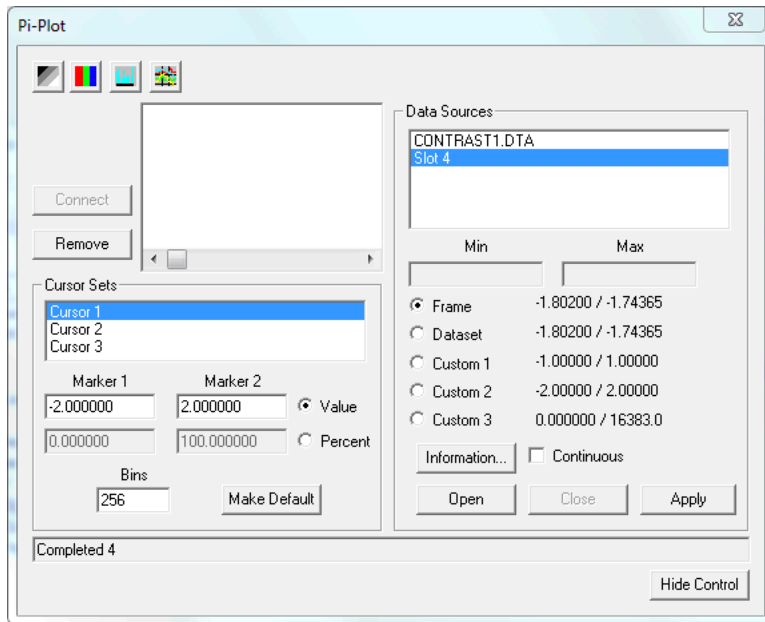


Figure 49: PI-Plot Main Window

4.2. Data Sources

The **PI-Plot Main Window** controls the opening and closing of data sources and the opening of views for those data sources. There are two types of data sources:

- Live data from the **PI-3105 Data Acquisition** subsystem. If multiple **Masters** are enabled, each **Master** can be a separate data source
- Data files, either from previously acquired data or from PI-DATS processing in PI-Result

From each type of data source, four types of views can be generated:

- Grayscale View
- False Color View
- Histogram View
- Scope View

Each data source may have multiple views opened from it, and the different views for a given data source are always frame-synchronized (e.g. always displaying the same frame number). A data source can also have multiples of the same type of view opened, such as two different grayscale views.

To open a data set from file, click the **Open Data** button. By default PI-Plot will list files with the extension **.DTA**, but it will open any properly formatted data file. Once opened from file, a data set will appear in the **Data Sources** list on the right side of the **Main Window**.

Data from the PI-3105 Data Acquisition Subsystem will automatically appear in the **Data Sources** list by **DACQ** slot number. A **DACQ** card will not appear in the **Data Sources** list until it has successfully acquired a data set.

To obtain information about a data set, either click the **Information** button or double-click the name of the data source:

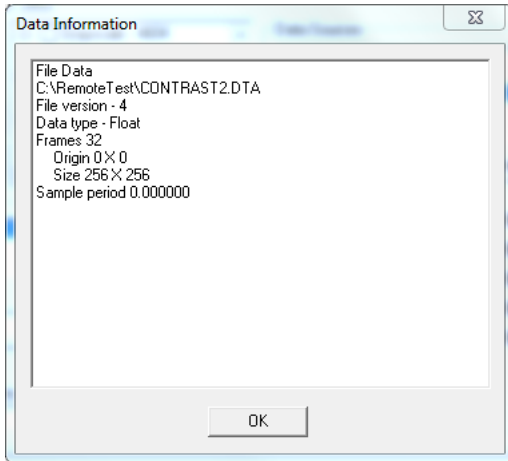


Figure 50: Data Information Dialog Box

The **Data Information** dialog box will display the source of the data (**File** or **Acquisition**), filename, number of bits, file version, data type, and array size.

To close a data set and all its associated views, select it in the **Data Sources** list and click **Close**.

4.3. Data Ranges

Every data set opened in **PI-PLOT** has an associated “**Data Range**” used for calculating histogram bins. The bin calculations happen whether or not a **Histogram View** is displayed, as the calculation is also used for the **Grayscale** and **False Color** views. By default the **Data Range** for a data set is the minimum and maximum of the present **Frame**:

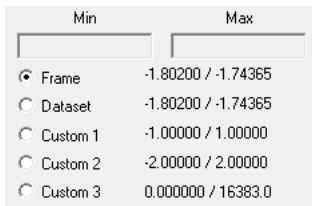


Figure 51: Data Range

Other ranges can be selected, such as the min/max of the entire **Dataset** or any of three **Custom** ranges. The three **Custom** ranges are prepopulated with commonly-used values, such as full-scale for a 14-bit ADC digital output and full-scale for common ADC voltage ranges. These may be edited by selecting them, typing new min/max values, and clicking **Apply**.

4.4. Histograms and Cursors

From the **Data Range** minimum and maximum, **PI-PLOT** then creates from 1 to 3 sets of histogram bins and associated **Cursor Pairs**:

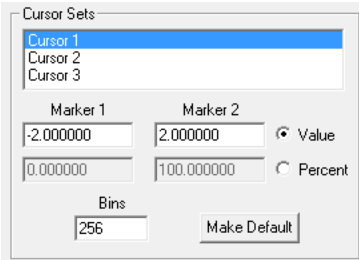


Figure 52: Cursor Pairs and Histogram Bins

All three histogram calculations share the same **Data Range**, but each histogram can have a different number of bins. The left edge of the lowest-value bin is equal to the **Data Range** minimum, and the right edge of the highest-value bin is equal to the **Data Range** maximum. The bin boundaries are then equally distributed across the number of bins.





To change the number of bins, select the **Cursor Pair**, type the desired number of **Bins**, and then click **Apply**. Note that **Apply** will also apply the entered **Cursor** positions. If the **Cursor** positions have been modified by dragging them in the **Histogram View**, the positions in the **PI-PLOT Main Window** will over-write them. To read in the present cursor positions from the **Histogram View**, double-click the **Cursor Pair**.

For each histogram calculation there is an associated **Cursor Pair** defining the black and white points for the **Grayscale** view and/or the endpoints of the **False Color** view. A **Cursor Pair** can be set by **Value** or by **Percentile** by clicking the appropriate radio button. The cursor positions can also be set by dragging them in a **Histogram View** (see below).

To set a cursor position, select the **Cursor Pair**, type the desired **Value** or **Percentile**, and then click **Apply**.

4.5. Managing Views

To open a view select a data set from the **Data Sources** list, then click one of the four View buttons:

-  Grayscale View: Pixel values are mapped to a continuous-tone image, scaled to the data set or to a set of user-defined cursor sets.
-  False Color View: Pixel values are mapped to a user-definable color table, scaled to the data set or to a set of user-defined cursor sets.
-  Histogram View: Pixel values are sorted into a user-definable set of bins, with the view displaying the number of pixels in each bin.
-  Scope View: Pixel values are plotted sequentially such as on an oscilloscope.

Multiple Views can be opened for any data set by clicking the View buttons again. PI-PLOT will display a “tree” view of each view and its associated data set(s).

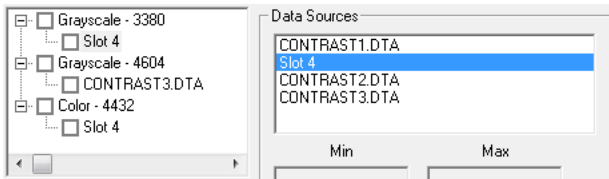



Figure 53: Views and Data Sources

A **View** can be removed by clicking its **Close** box or by checking one or more **Views** in the **View Tree** and clicking **Remove**.

A **View** can be re-assigned to a different data set by selecting it in the **View Tree**, selecting the desired data set in the **Data Sources** list, and clicking **Connect**. This preserves all the settings for that **View**.

The **PI-Plot Main Window** is tied to the **PI-DATS** or **PI-Controller** window, and therefore cannot be placed behind it. It can be hidden with the **Hide Control** button, or its visibility can be toggled with

either the **PI-PLOT** button the **DACQ** mnemonic or the **Show/Hide Control** button  on any **PI-PLOT** View.

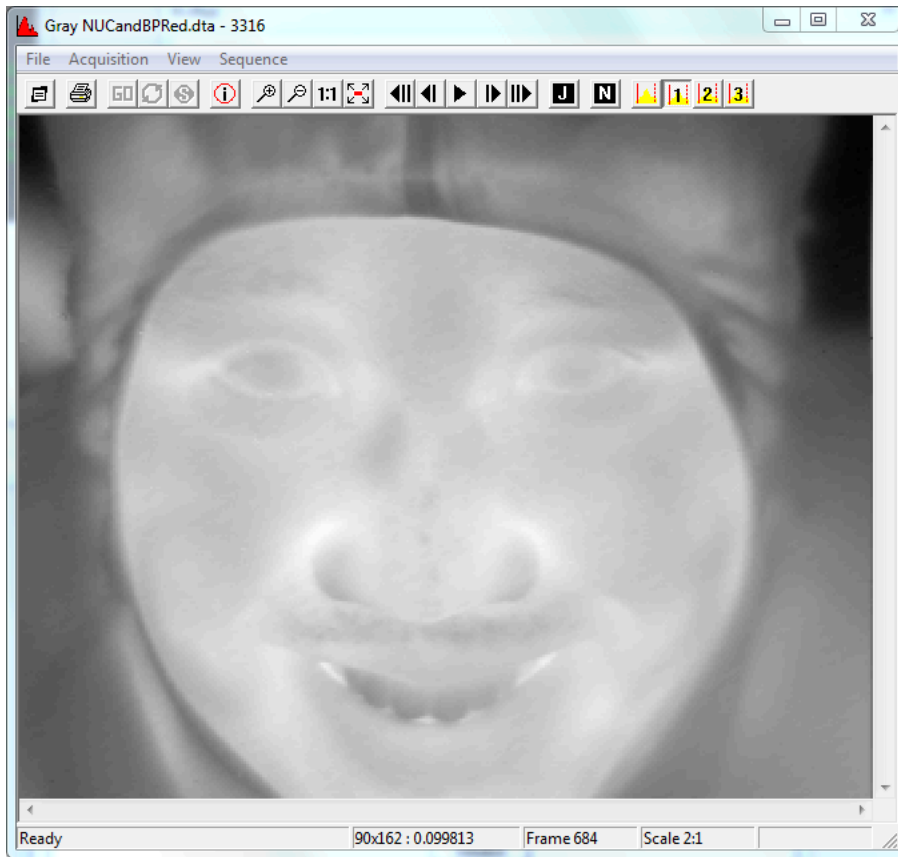


Figure 54: PI-Plot (Grayscale View)

4.6. Common Controls

All **PI-Plot** views have a set of common controls, including controls for:

- Saving data from the current view
- Printing images
- Exporting image sequences
- Jumping to a particular frame number in the data set
- Editing the properties of the view, such as zoom level, scaling, etc.

Views also have controls for acquiring new data sets.

4.6.1. Multiple Frames



Figure 55: Playback Buttons (First, Prev, Play, Next, Last)

If the displayed data set has multiple frames of data in the **Grayscale**, **False Color**, or **Histogram** views, the user may “page” through the frames by using the **Prev** and **Next** buttons on the toolbar, by selecting **Sequence: Previous** or **Sequence: Next** menu items, or by using the left and right arrow

keys on the keyboard. The user may choose the first or last frames by using the **First** and **Last** buttons on the toolbar, by selecting **Sequence: First** or **Sequence: Last** menu items, or by using the **Home** and **End** keys on the keyboard. The user may also jump to any frame number using the **Jump** button or the **Sequence: Jump** menu item.

Frames within a data set may also be played back continuously using the **Play** button.

4.6.2. Zooming




Figure 56: Zoom Buttons (In, Out, 1:1, Fit)

By default, the **Grayscale** and **False Color** views display one acquired pixel as one plotted pixel. The display may also be zoomed in or out via the **Zoom In** and **Zoom Out** buttons on the toolbar, by selecting the **View: Zoom: Zoom+** and **View: Zoom: Zoom-** menu items, or by pressing the **+** and **-** keys on the keyboard. Zoom levels are available from 1:10 to 10:1, in integer increments. Pressing **=** on the keyboard or the **1:1** button will set the display back to 1:1, and pressing **0** on the numeric keypad or the **Fit** button will zoom the image to fill the current window size. If the image is larger than the **PI-Plot** window, horizontal and/or vertical scrollbars will appear.


4.7. Acquisition Controls

PI-Plot provides basic controls for running data acquisitions. There are two buttons on the toolbar and three menu items:

4.7.1. Start

The user may start a data acquisition by click the **Go** button  or by selecting the **Start** item from the **Acquisition** menu. This is equivalent to clicking the **Start Acq** button on the **DACQ** mnemonic of **PI-DATS/PI-Controller**, except that the filename argument is not reset (see **FileName**, auto-numbering option, in the **PI-DATS** manual). When the acquisition and data transfer are complete, **PI-PLOT** will automatically update the display with the new data, at the currently selected frame number and zoom level.

4.7.2. Continuous Acquisition

PI-Plot can also display real-time video by running acquisitions continuously. Use the **Continuous** button  or select the **Continuous** item from the **Acquisition** menu to toggle this feature. When enabled, the **Continuous** button on the toolbar will retain a “depressed” appearance, and the **Continuous** item on the **Run** menu will be checked.

If the **Start/Go** button is clicked (either from **PI-Plot** or from **PI-DATS**) while **PI-Plot** is in **Continuous** mode, data acquisitions and screen updates will run continuously until interrupted by the user.

To stop continuous acquisition, click the **Continuous** button on the **PI-Plot** toolbar. **PI-DATS** will complete the current acquisition and then stop.

Note that the acquisition parameters on the **Data Acquisition** property page in **PI-DATS** will remain the same for every acquisition in **Continuous** mode. For example, if the **Frame Count** is set to 10 frames, then each click of the **Start/Go** button will collect 10 frames. If **PI-Plot** is set to **Continuous** mode, it will repeatedly collect 10 frames and update the screen after every 10th frame has been collected. To display the fastest possible frame rate, set the **Frame Count** to 1

4.8. Scope View

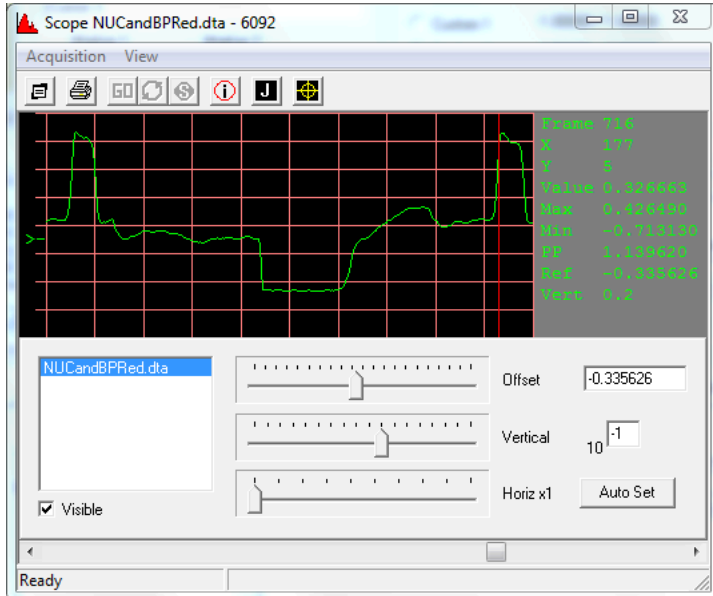


Figure 57: Scope View

In a **Scope View** (also called a “skyline” view) all acquired pixels from all frames from a data source are displayed in sequential fashion, with each pixel’s value represented by the height of the “skyline” at its position.

The information pane in the right margin of the **Scope View** displays information about the pixel under the cursor line. The cursor line may be dragged with the mouse.

The horizontal scrollbar at the very bottom of the **Scope View** scrolls through the data set.

If multiple data sets have been connected to the **Scope View**, each data set can be drawn in a different color.

Pixels are numbered in the order they are acquired; therefore the first pixel of the second frame of a 100 x 100 array would be the 10,001st pixel.

To scale the displayed data, click a data source from the **Data Sources** pane in the lower left of the **Scope View** window, then use the sliders below the display. The **Offset** slider changes the displayed DC offset of the signal. The **Gain** slider changes the displayed vertical scale. The **Horizontal** slider changes the displayed horizontal scale.

The **Scope View** can be autoscaled. Select the data set from the **Data Sources** pane, and then click **Autoscale**. The **Offset** and **Vertical** settings will be automatically chosen to center and fill the display with the data presently in view.

Different data sources can have different gains and offsets, but the same horizontal scale is used for all data sets in a **Scope View**. Different **Scope View** windows can have different horizontal scales.

Use the **Settings** button  or the **View: Properties** menu item to select colors for the **Scope View**:

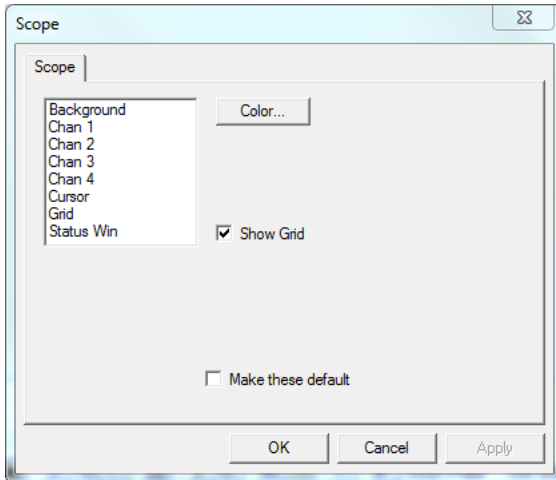


Figure 58: Scope View settings

4.9. Histogram View

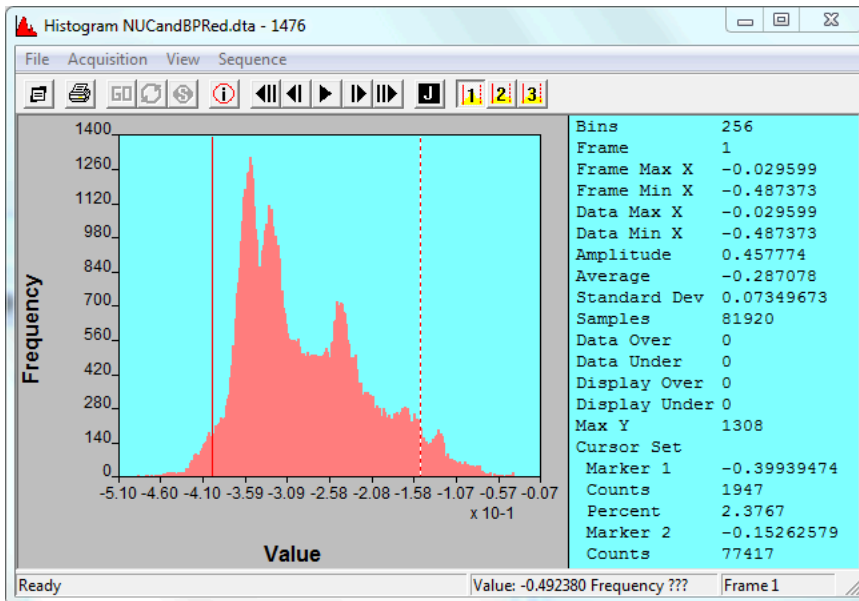



Figure 59: Histogram View

The **Histogram View** displays the frequency of acquired pixel values for each frame. The height of each bar represents the frequency (number of occurrences) of each pixel value. The **Histogram View** also displays basic statistical information (**Mean**, **Min**, **Max**, **Amplitude** and **Standard Deviation**) about each frame.

As the mouse moves over the **Histogram View**, the pixel value and pixel frequency under the pointer will be displayed in the PI-Plot status bar.

To change the settings for the **Histogram View**, Use the **Info** button  or the **View: Properties** menu item:

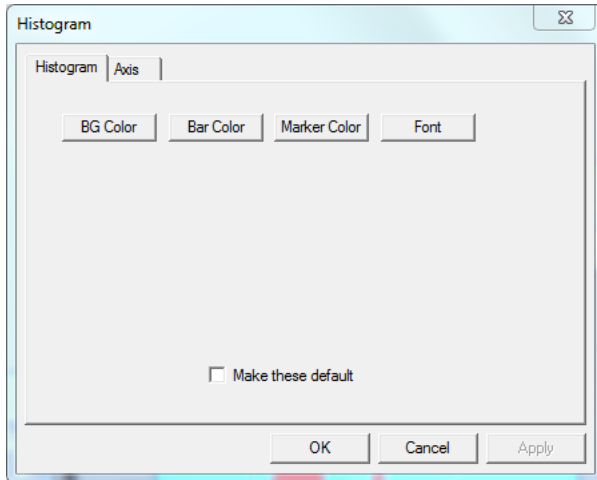


Figure 60: Histogram Settings, Histogram Property Page

The **Histogram** property page may be used to change the colors on the **Histogram View**.

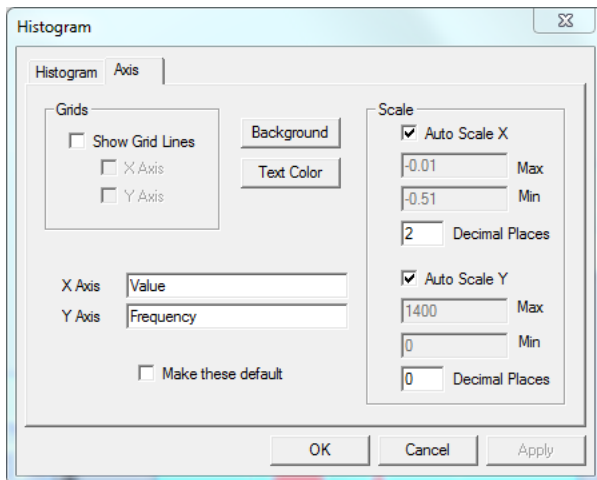


Figure 61: Histogram Settings, Axis Property Page

The **X-Axis Minimum** and **Maximum** may be used to set the horizontal scale of the **Histogram View**. The **Y-Axis Maximum** may be used to set the vertical scale. If the **Auto Scale** boxes are checked, the scales are set to display the entire data range, plus a 5% “buffer” region on either side.

Auto Scale may fail to center the histogram in the display if the number of decimal places is insufficient to discriminate between the correct minimum and maximum bin values. To correct this, increase the number of decimal places.

The **Grid** checkboxes toggle the display of the gridlines on the plot.

4.9.1. Histogram Cursors

The three sets of **Cursor Pairs** defined in the **PI-PLOT Main Window** may be used to display data frequency at specified percentiles or values. To set a **Cursor Pair**, drag the mouse in the **Histogram View**. Press the **Tab** key to toggle which cursor is being dragged. Use the **Cursor Pair** buttons



or the **View: Cursors** menu item to select which **Cursor Pair** is being displayed and dragged.

The bin value and bin count at each marker position will be displayed in the **Statistics** pane of the **Histogram View**.

The **Cursor Pairs** can also be used to control the luminosity of the **Grayscale View** and **False Color View**.

If new data are displayed in the **Histogram View**, either via a new acquisition, a file being opened, or a new frame being displayed, the **Histogram View** will update either the marker values or the marker positions, depending on whether the **Cursor Pair** is set to **Value** or **Percentile**.

4.10. Grayscale View




Figure 62: Grayscale View

In a **Grayscale View** the acquired data are displayed as an image, with pixel values represented by brightness.

4.10.1. Grayscale Luminosity

By default, the grayscale luminosity is scaled to the minimum and maximum values as specified in the **Data Range**.

The luminosity can be re-scaled to any of the three **Cursor Pairs** by either clicking the **Scale to**

Cursors button  or by choosing **Histogram Cursors** from the **View** menu and then clicking one of the three **Cursor Pair** buttons   .

For example, **Figure 62: Grayscale** shows an image from an infrared sensor with low contrast. When **Use Cursors** is clicked with the cursors set to 5% and 95%, the display changes to the following:



Figure 63: Grayscale View, scaled to cursors

As the mouse moves over the **Grayscale View**, the status bar at the bottom will display the pixel address (row, column) and the pixel value under the mouse pointer.

4.11. False Color View

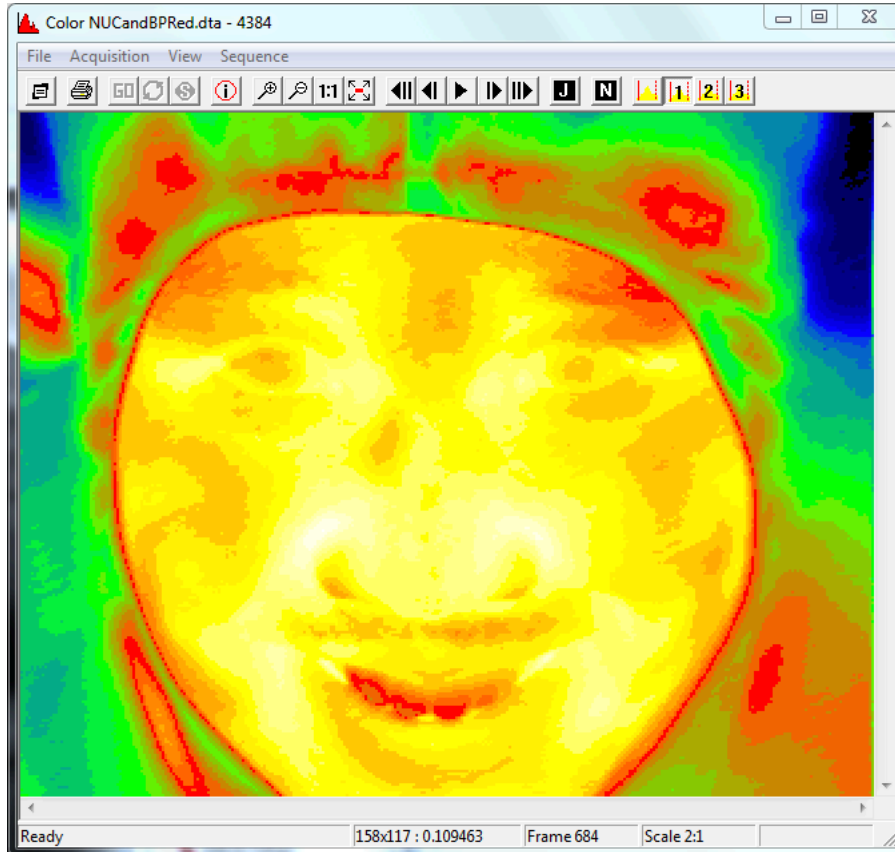



Figure 65: False Color View

In a **False Color View** the acquired data are displayed as an image, with pixel values represented by a range of colors.

4.11.1. False Color Table

By default, the color table is set for 32 colors, evenly spaced between the minimum and maximum values of the data set.

The current color table can be displayed and edited by clicking the **Info**  button or by selecting the **View: Properties** menu item.

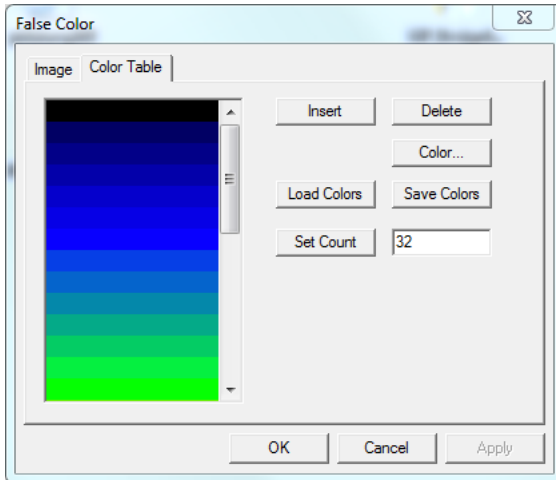



Figure 66: False Color View, Properties, Color Table

The **False Color View** can be re-scaled according to specified minimum and maximum values from the **Histogram** view either by clicking the **Use Cursors** button  or by choosing **Use Cursors** from the **View** menu. For example, Figure 67: False Color View, scaled to cursors shows an image from an infrared sensor with low contrast. When **Use Cursors** is clicked, the display changes to the following:

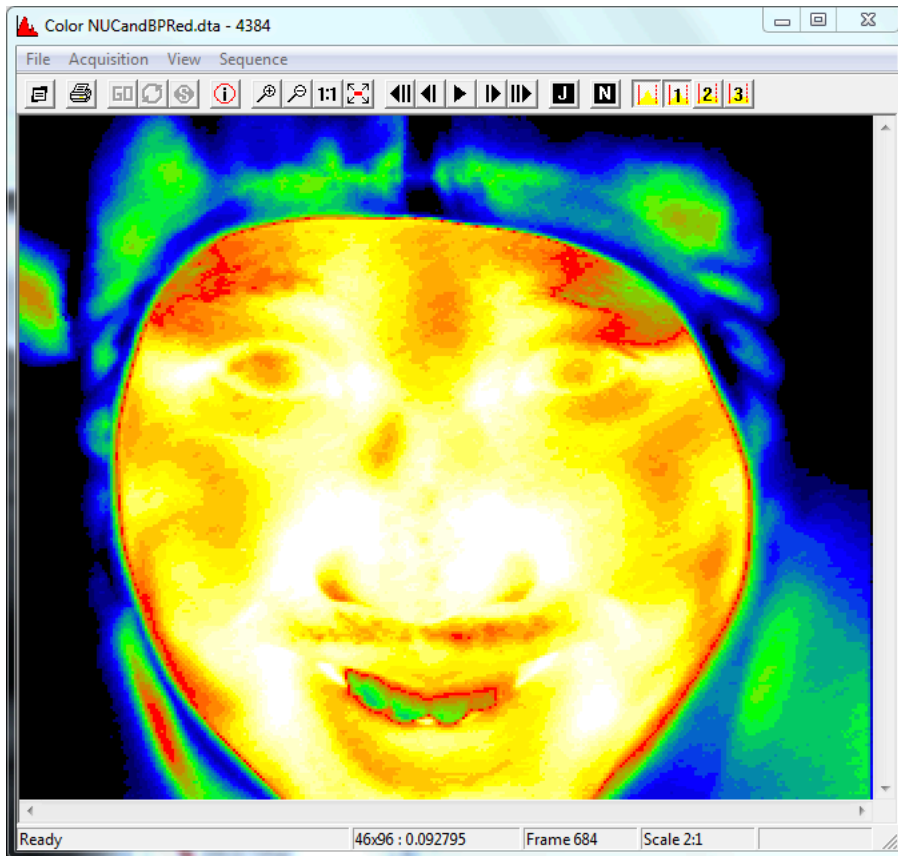


Figure 67: False Color View, scaled to cursors

As the mouse moves over the **False Color View**, the status bar at the bottom will display the pixel address (row, column) and the pixel value under the mouse pointer.

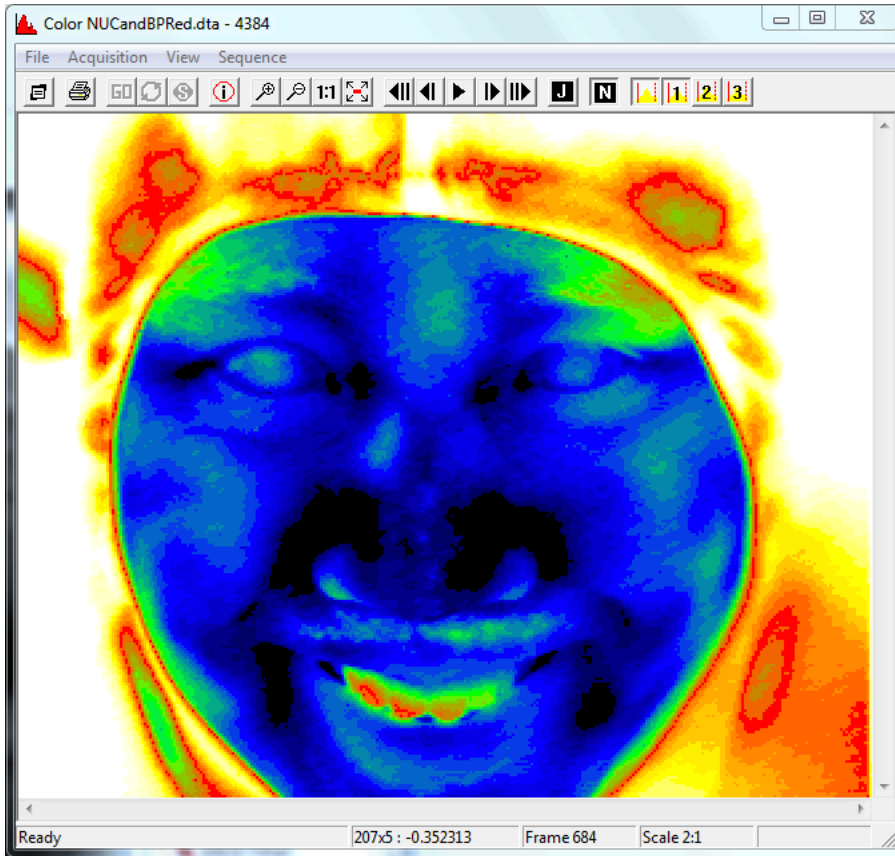


Figure 68: False Color View, Inverted.

5. SAMPLE FILES AND TUTORIAL

This section provides several simple examples of acquisition timing setup and acquisition of simulated data from the PI-3105. This section assumes basic familiarity with the PI-2005 Pattern Generator and PI-PAT software. The following pattern files are referenced in the examples below:

- SimpleFrameAIM.w25
- WelcomeToPulse4.w25

They are available either on your instrument, in the C:\SAMPLES directory, or you may download them from the Pulse Instruments website at:

<http://www.pulseinstruments.com/samples>

5.1. Quick Start—Acquisition Timing Setup

The PI-PAT pattern file shown below will provide timing for a video frame of varying sizes, from 16 x 1 up to 1024 x 1024 or more.

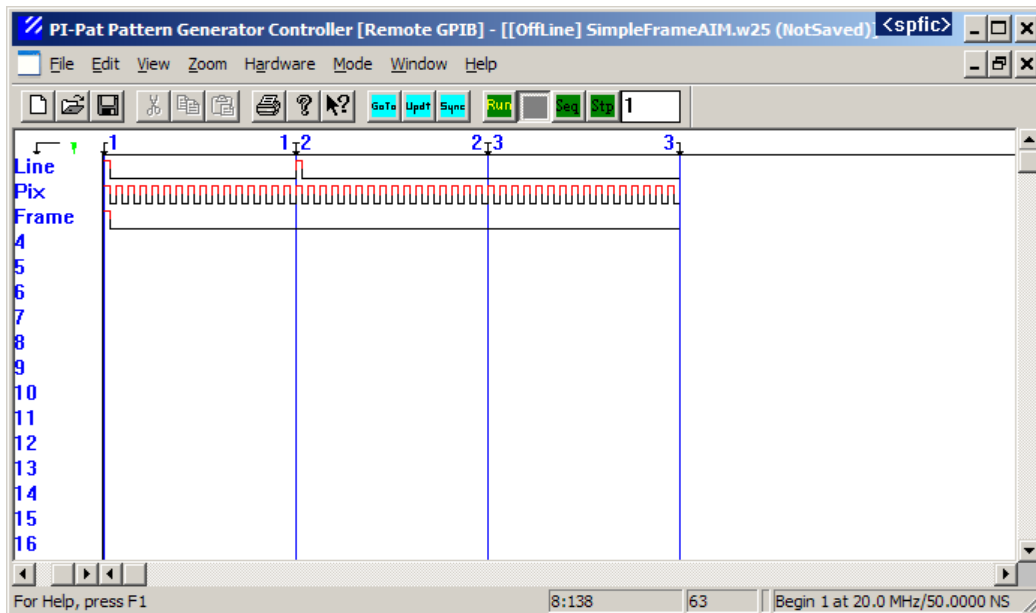


Figure 69: Sample PI-PAT file for acquisition timing

It is assumed that Channels 1-4 from the pattern generator are cabled to the CLK IN connector on the Timing and Control card. If you are connecting the pattern generator outputs directly to your PI-41000 SMA clock inputs, then ensure that the clock signals are cabled to the appropriate inputs.

Each subpattern in **Figure 69** is 32 clock periods long. The pixel clock channel has an alternating 1/0 pattern, providing one pixel clock pulse for every two clock periods of the pattern generator. Therefore each subpattern contains 16 pixel clocks. The PI-3105 with the PI-41040 ADC Modules will digitize data at up to 10 MHz; therefore this pattern can be run into the Timing and Control card at up to 20 MHz master clock rate.

The program numbered BEGIN 1 has the following instructions:

```

1 Begin 1
2 SubPattern 1 x 1

```

Subpattern 1 provides 1 frame sync, 1 line sync, and 16 pixel clocks.

Ensure that this pattern file is loaded and BEGIN 1 has been compiled and is running. Launch PI-Controller and add a **DEFINE** mnemonic, then set up an FPA definition with the following parameters:

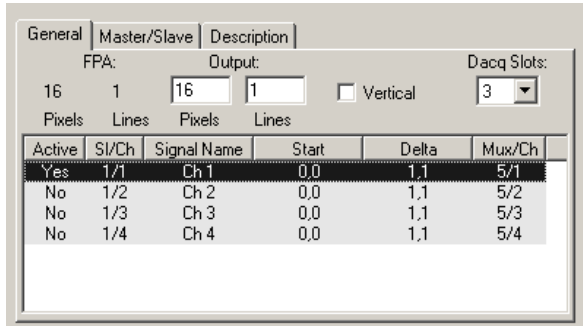


Figure 70: DEFINE setup for 16 x 1 frame

Enter the above values and click **Write**. Ensure that all other channels in the system are set to Off (**Active = No**). Next, add a **DACQ** mnemonic, set for 1 frame:

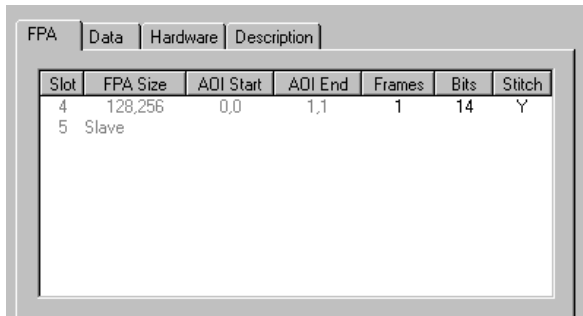


Figure 71: DACQ setup for 1 frame

Click **Start**. The acquisition should complete.

Since the PI-2005 will repeat the program indefinitely, the PI-41000 can be set up to acquire as many frames of this size as desired, up to 65,536 or up to the available memory on the card.

Subpattern 2 may be used to increase the number of lines produced by this pattern, e.g.:

```

4 Begin 2
5 SubPattern 1 x 1
6 SubPattern 2 x 15

```

When executed, this pattern will generate timing for a 16 x 16 frame. Subpattern 1 provides 1 frame sync, 1 line sync, and 16 pixel clocks. Then, each execution of Subpattern 2 will provide 1 line sync and 16 pixel clocks. Since Subpattern 2 is repeated 15 times, the entire program will generate 16 line syncs, each of which is followed by 16 pixels clocks.

Assuming this pattern file is loaded and **BEGIN 2** has been compiled and run set up a **DEFINE** mnemonic like the following:

General		Master/Slave		Description	
FPA:		Output:		Dacq Slots:	
16	16	16	16	<input type="checkbox"/> Vertical	3
Pixels	Lines	Pixels	Lines		
Active	SI/Ch	Signal Name	Start	Delta	Mux/Ch
Yes	1/1	Ch 1	0.0	1.1	5/1
No	1/2	Ch 2	0.0	1.1	5/2
No	1/3	Ch 3	0.0	1.1	5/3
No	1/4	Ch 4	0.0	1.1	5/4

Figure 72: DEFINE setup for 16 x 16 frame

Enter the above values, click **Write** then switch to the DACQ mnemonic and click **Start**. The acquisition should complete.

Subpattern 3 may be interleaved between Subpatterns 1 and 2 to increase the size of the frame again. For example, consider the following program:

```

8 Begin 3
9 SubPattern 1 x 1
10 SubPattern 3 x 15
11 SubPattern 2 x 1
12 SubPattern 3 x 15
13 Repeat From L 11 to L 12 x 255

```

When executed, this pattern will generate timing for a 256 x 256 frame. Subpattern 1 provides 1 frame sync, 1 line sync, and 16 pixel clocks. Subpattern 3 provides 15 x 16 = 240 pixel clocks, for a total of 256.

In a similar manner, instruction lines 11 and 12 provide 1 line sync and 256 pixel clocks. Since instruction lines 11 and 12 are looped 255 times, the entire program will generate 256 line syncs, each of which is accompanied by 256 pixels clocks.

Assuming this pattern file is loaded and **BEGIN 3** has been compiled and run, set up another **DEFINE** mnemonic like the following:

General		Master/Slave		Description	
FPA:		Output:		Dacq Slots:	
256	256	256	256	<input type="checkbox"/> Vertical	3
Pixels	Lines	Pixels	Lines		
Active	SI/Ch	Signal Name	Start	Delta	Mux/Ch
Yes	1/1	Ch 1	0.0	1.1	5/1
No	1/2	Ch 2	0.0	1.1	5/2
No	1/3	Ch 3	0.0	1.1	5/3
No	1/4	Ch 4	0.0	1.1	5/4

Figure 73: DEFINE setup for 256 x 256 frame

Enter the above values, click **Write** then switch to the DACQ mnemonic and click **Start**. The acquisition should complete.

If acquired through an AIM and displayed using PI-Plot, the image will show a small amount of random noise. If desired, a signal (± 2 V) may be injected into the preamplifier and viewed in PI-Plot.

Note that all 3 programs are stored in the same file and can be compiled for use at any time by using PI-PAT's **Hardware** window or the **Edit:Edit Instructions** window. Please see the PI-2005 Operators Manual for details.

The pattern file can also be called up, a program compiled and run, via the **Pat2005** mnemonic in PI-Controller. Please see the PI-11000 Operators Manual, **Pat mnemonic** section, for details.

5.2. Quick Start—Multiple Channel Acquisition

In addition to providing acquisition timing, the pattern generator cards can also generate data streams. These data streams can simulate a video signal. To connect the pattern generator to the PI-3105, use the Pattern Output cables to connect channels 5-8 to the BNC inputs of the PI-3150 Pre-amplifier modules 1 through 4. (Although the PI-2005 outputs a TTL signal of approximately 0 – 5 V into 1 M Ω , the PI-3150 can tolerate this signal level without damage. It is best if the signal is terminated into 50 Ω with a BNC tee in order to attenuate the signal to ~2.5 V amplitude).

Open the test plan file, C:\Samples>WelcomeToPulse4.pip:

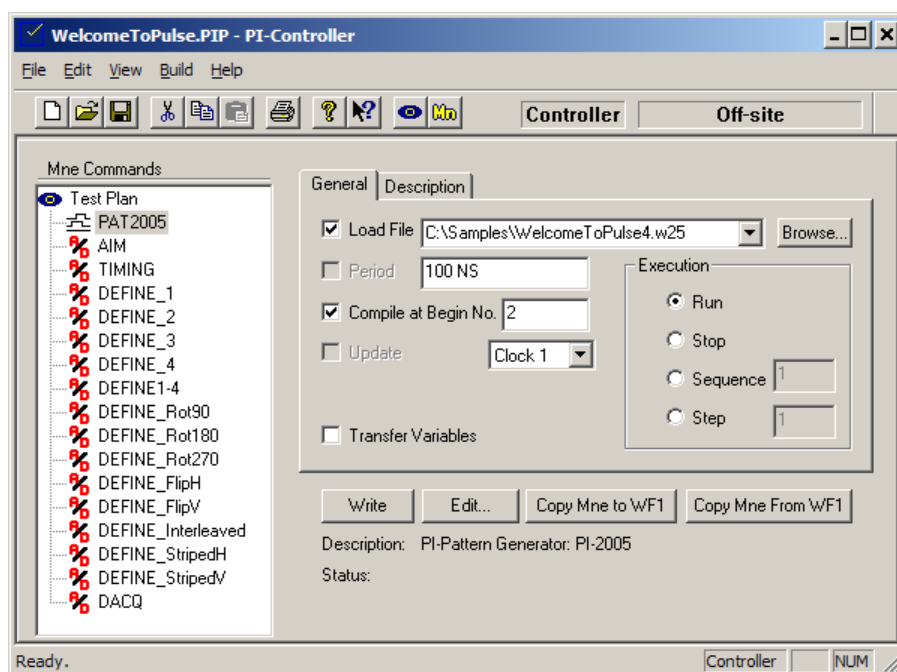


Figure 74: WelcomeToPulse4.pip, Test Plan File

Click on the PAT2005 mnemonic, as shown above, and click Write. The pattern generator will begin running, using a hidden instance of PI-PAT. Click Edit to make PI-PAT visible, then bring the PI-PAT window into the foreground.

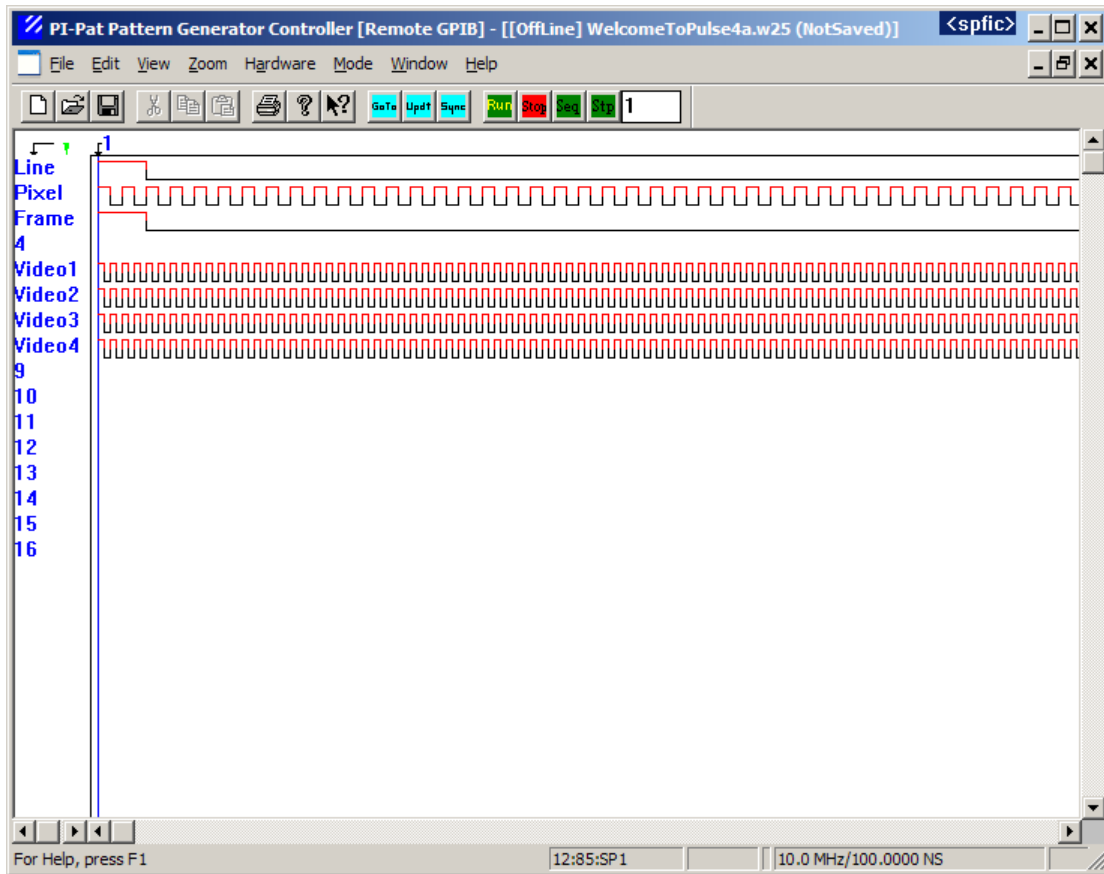


Figure 75: WelcomeToPulse4.w25, PI-PAT file with “walking” image

The PI-PAT file shown (WelcomeToPulse4.w25) below provides timing for four channels of a 256 x 256 simulated device. Each channel is 64 x 256, and the four channels are “stitched” together to produce the final image. Furthermore the simulated “video” produces a “box” of bright pixels around a dark frame and an image that “walks” up and down the frame. This pattern can be used to simulate various types of devices and to test various FPA definitions. The standard Pixel, Line and Frame clocks are on Ch. 1-3, and the simulated video streams are on Ch. 5-8.

Ensure that channels 1-3 of your pattern generator are cabled to the CLK In input of your Timing and Control card, and that all 4 channels of ADC are cabled correctly to the system.

Scroll to the right until the beginning of Subpattern 2 is visible, and then use the View menu to increase the Zoom level to 4.

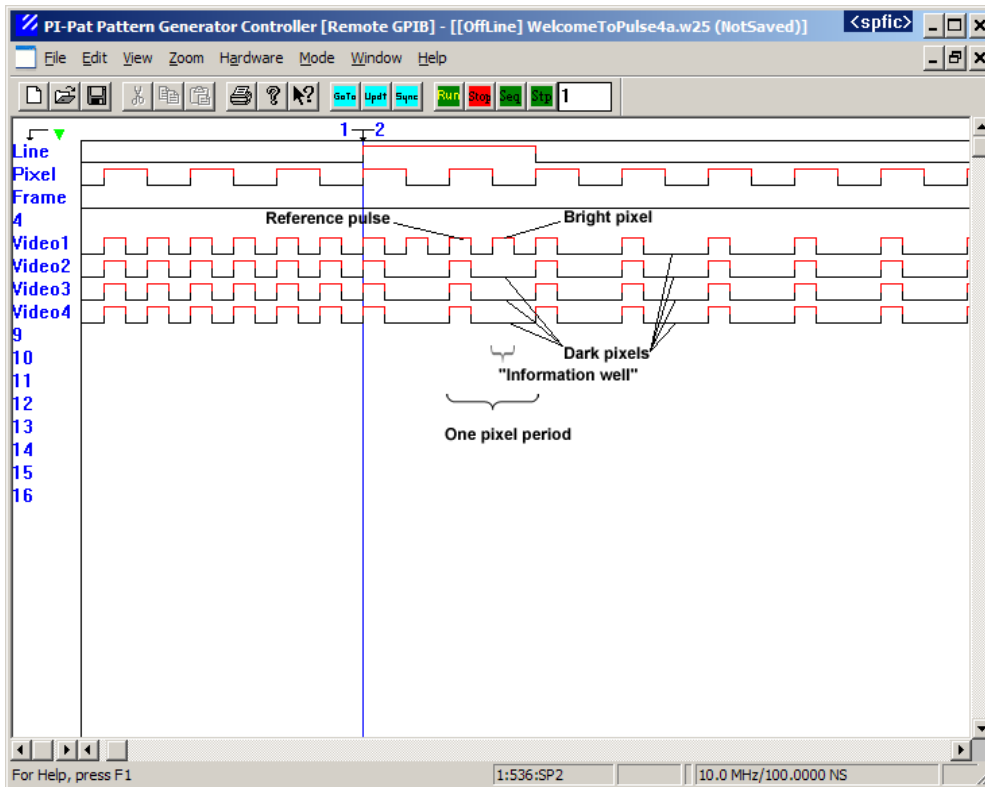


Figure 76: WelcomeToPulse4.w25, PI-PAT file, zoomed view

Each period of the pixel clock signal on channel 2 (the “pixel period”) is 4 clock periods of the Pattern Generator’s master clock. Therefore, if the PI-2005 is set for a clock period of 100 ns, the effective pixel period will be 400 ns (2.5 MHz), and each pixel period will contain 4 periods of the Master Clock.

During each pixel period, each “video” channel has a high “reference” pulse during Master Clock Period 1, and a simulated “information well” during Master Clock period 3 that can be Hi or Lo, depending on the position within the simulated image. The information well is bracketed by low levels during Master Clock periods 2 and 4. For example, in the pixel period illustrated above, Video 1 has a bright pixel during this period, and Video 2-4 have dark pixels. The simulated image has only bright or dark pixels, but the actual image acquired by the system may have a wide range of values, based on the gain, offset, and convert strobe positioning.

Click back on the PI-Controller window, then click on the **AIM** mnemonic and click the **Write** button to set the gain, offset, and monitor settings:

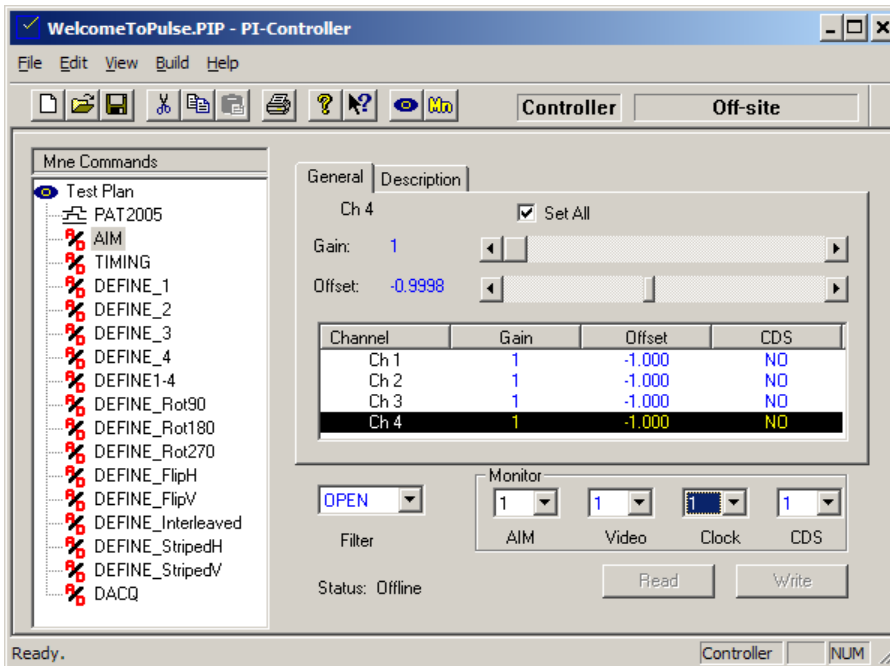


Figure 77: WelcomeToPulse4, AIM mnemonic

Click on the **Timing** mnemonic and click the **Write** button to set the Convert strobes.

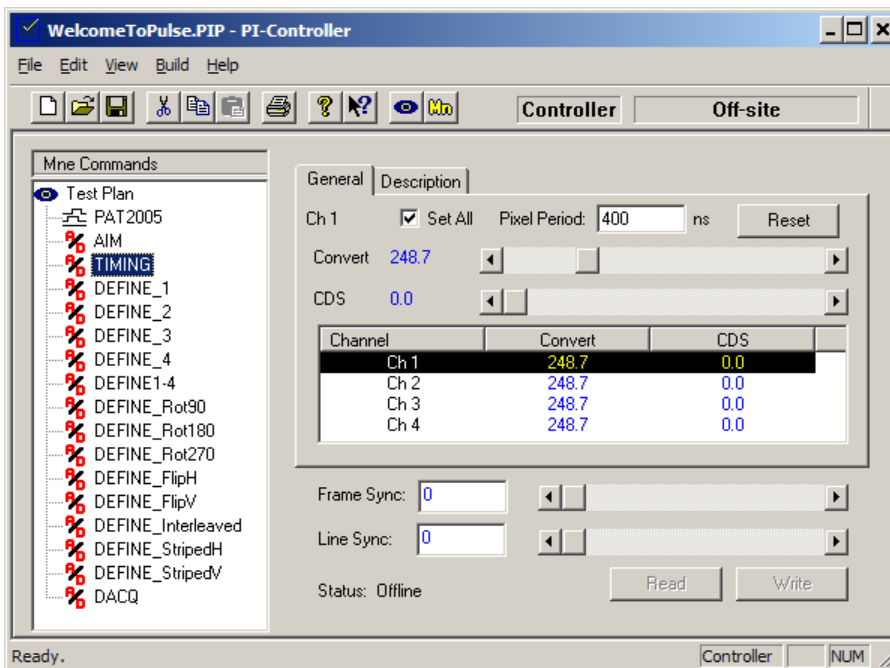


Figure 78: WelcomeToPulse4, Timing mnemonic

To view the simulated video signal on an oscilloscope, connect the Vid Mon (Video Monitor) and Conv Mon (Convert Strobe Monitor) signals to your 'scope, and trigger on the Conv Mon signal. Invert the **Vid Mon** signal. The signal is easiest to see on an analog scope. The video signal will resemble the trace below (the smaller-amplitude signal):

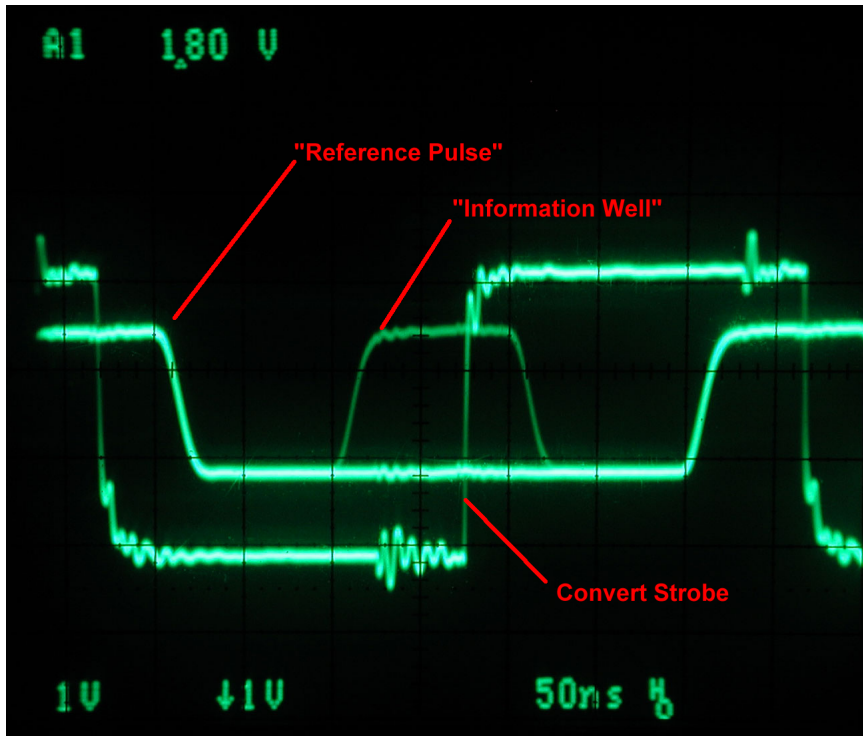


Figure 79: WelcomeToPulse4, Oscilloscope Capture

The “reference” pulse is solid because it is “on” during every pixel period. The “information well” is fainter because it is only “on” when the pixels are bright. The low periods bracketing the information well also are solid because they are constant during every pixel period.

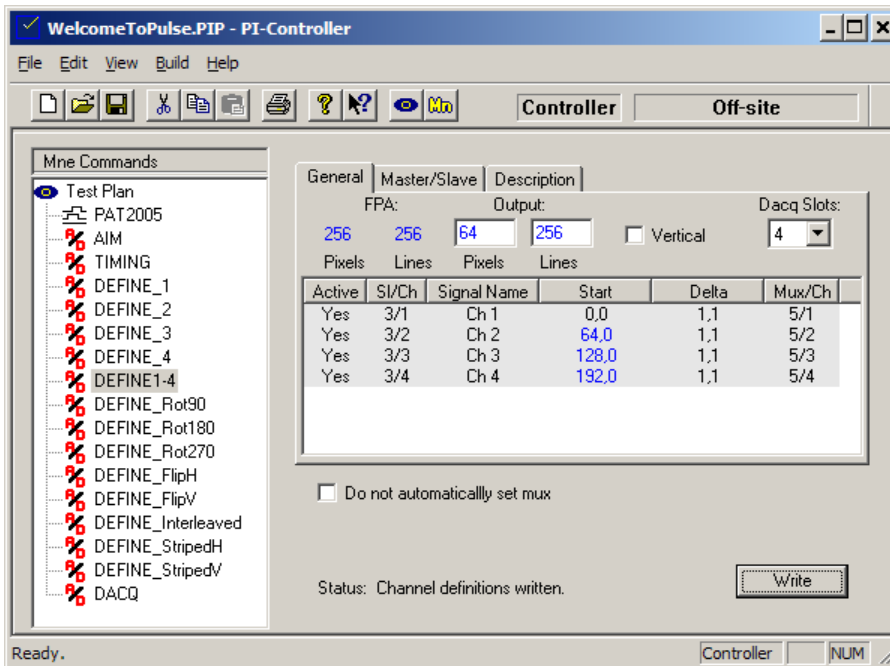


Figure 80: WelcomeToPulse4, Define Mnemonic

Click on the DEFINE1-4 mnemonic, as shown above, click **Write**, then switch to the **DACQ** mnemonic and click **Start**. The acquisition should complete. Click the **PI-Plot** button to open a PI-Plot window, and then click the **False Color Plot** button.

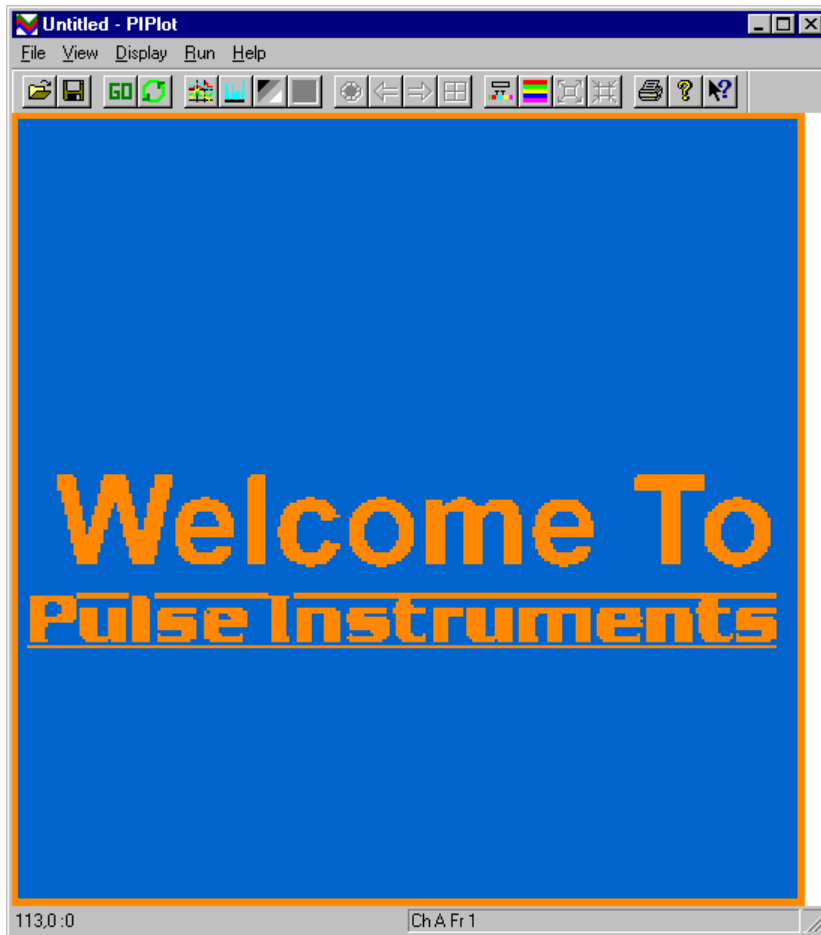


Figure 81: WelcomeToPulse4, False-Color Plot

The above screen capture shows the 256 x 256 acquired image in the False Color view. The vertical position of the “Welcome To Pulse Instruments” may be anywhere in the image.

In **False Color** or **Grayscale** view, click the **Continuous** button to activate Continuous Acquisition, use the **Run** menu to set the **Pause Time** to 0 ms, then click the **Go** button.

The “Welcome To Pulse Instruments” image will walk up and down the acquired frames. To force the image to start over from the top, switch to PI-PAT and click **Stop** and then **Start**. If you have a **PAT** mnemonic in your PI-Controller test plan, you may also use that to re-start the pattern output.

To view the effect of the Convert Strobe positioning, leave the continuous acquisition running and click on the **Timing** mnemonic. Check the “**Set all**” checkbox, and then drag the **Convert Strobe** slider to adjust the strobe position.

When viewed on your ‘scope, the rising edge of the Convert strobe will move in and out of the “information well.” As the Convert strobe moves through the information well, the amplitude of the bright pixels will change, based on the “height” of the video pulse at the position of the convert strobe. The plotted image will reflect this, either as a change in color or a change in amplitude (in ‘Scope plot mode).

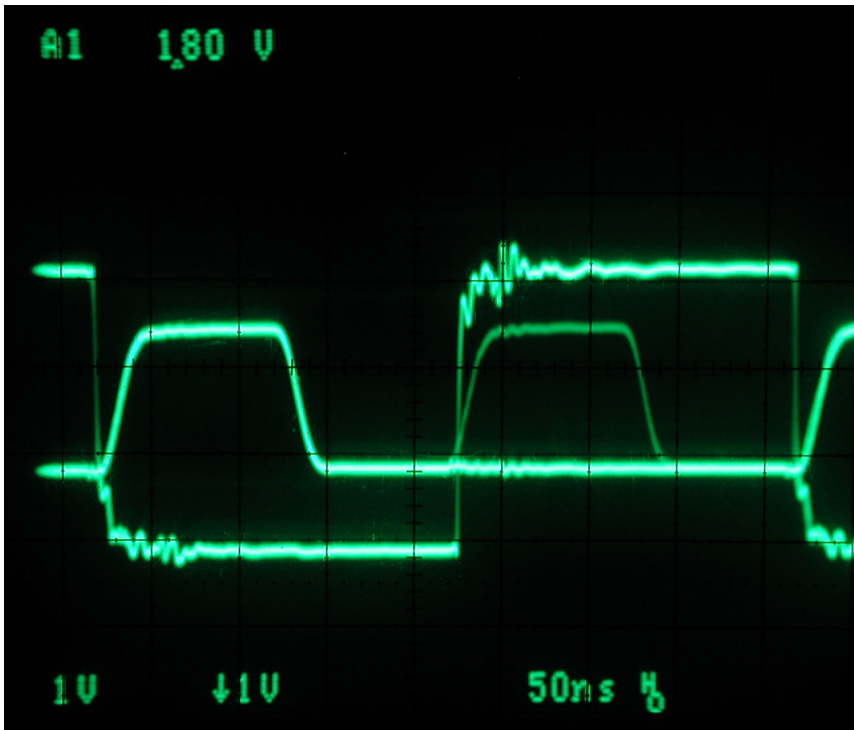


Figure 82: WelcomeToPulse4, Oscilloscope Capture, Low Amplitude

In this 'scope shot, the Convert strobe is triggering the data acquisition system just at the rising edge of the video pulse. When viewed in PI-Plot's false-color mode, the image may resemble the following:



Figure 83: WelcomeToPulse4, False Color Plot, Low Amplitude

The green color indicates a relatively low amplitude signal.

Note that some channels may not “show up” at this strobe positioning and some channels appear brighter than others because the signal amplitude at the convert strobe varies slightly from channel to channel. This is because the pattern generator simulating the video stream is a digital timing device, and the high and low levels are specified to TTL levels only. Some variation in amplitude across channels is normal.

When viewed in “Scope” plot mode, the reduced amplitude between the bright and dark areas of the image is evident:

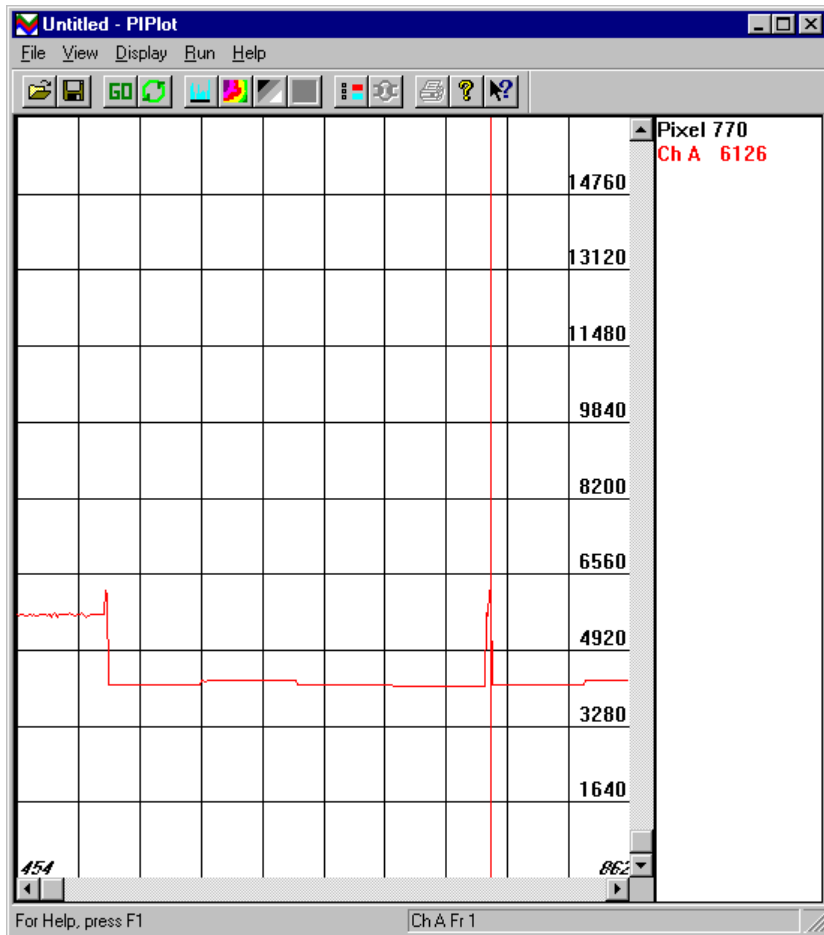


Figure 84: WelcomeToPulse4, 'Scope Plot, Low Amplitude

As the Convert Strobe moves further "into" the information well, the bright pixels become brighter:

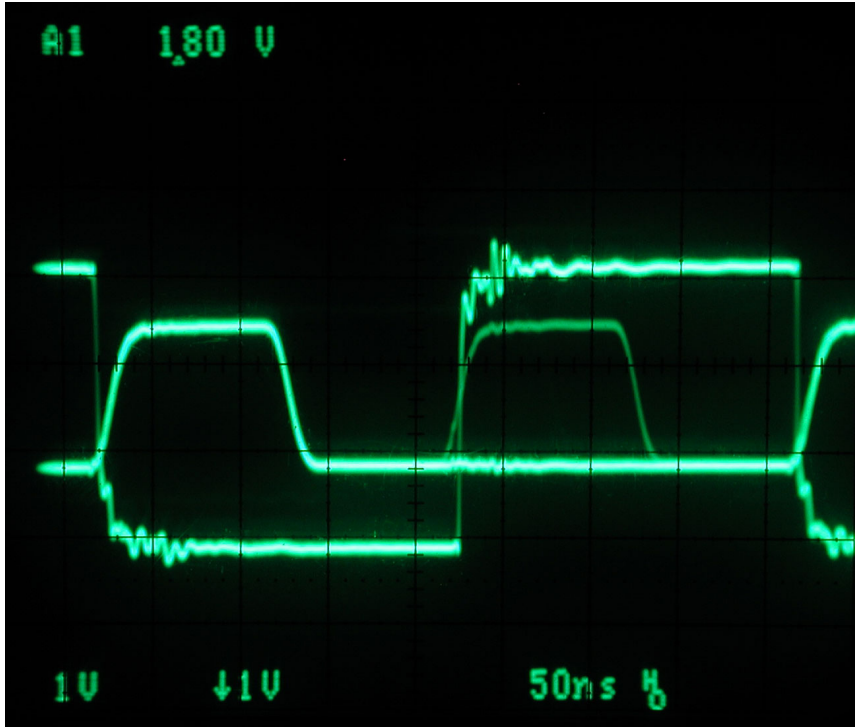


Figure 85: WelcomeToPulse4, Oscilloscope Capture, Medium Amplitude

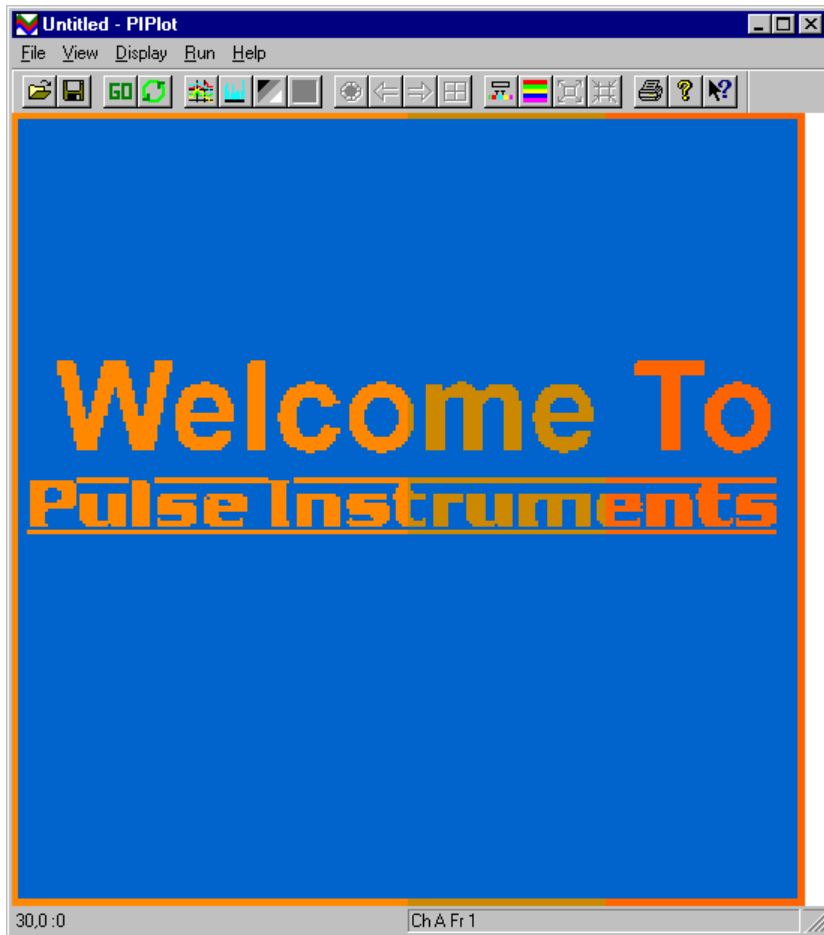


Figure 86: WelcomeToPulse4, False Color Plot, Medium Amplitude

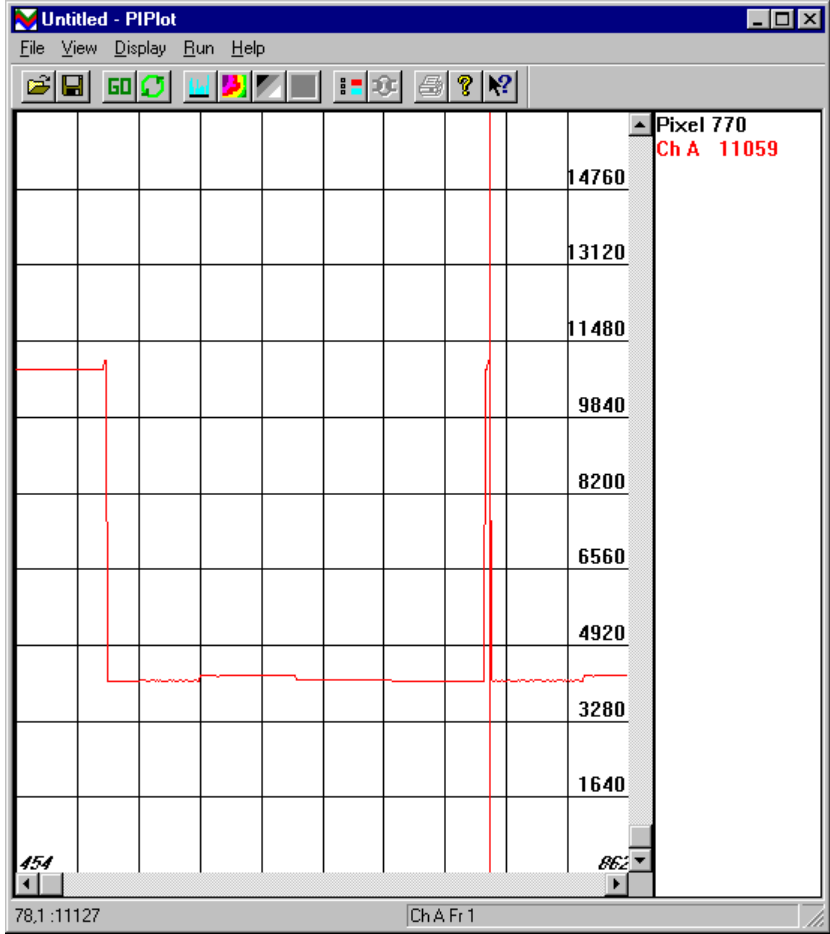


Figure 87: WelcomeToPulse4, 'Scope Plot, Medium Amplitude

At approximately 250 ns delay, the Convert Strobe is in the “fat” part of the information well, and the bright pixels’ amplitude is at its maximum:

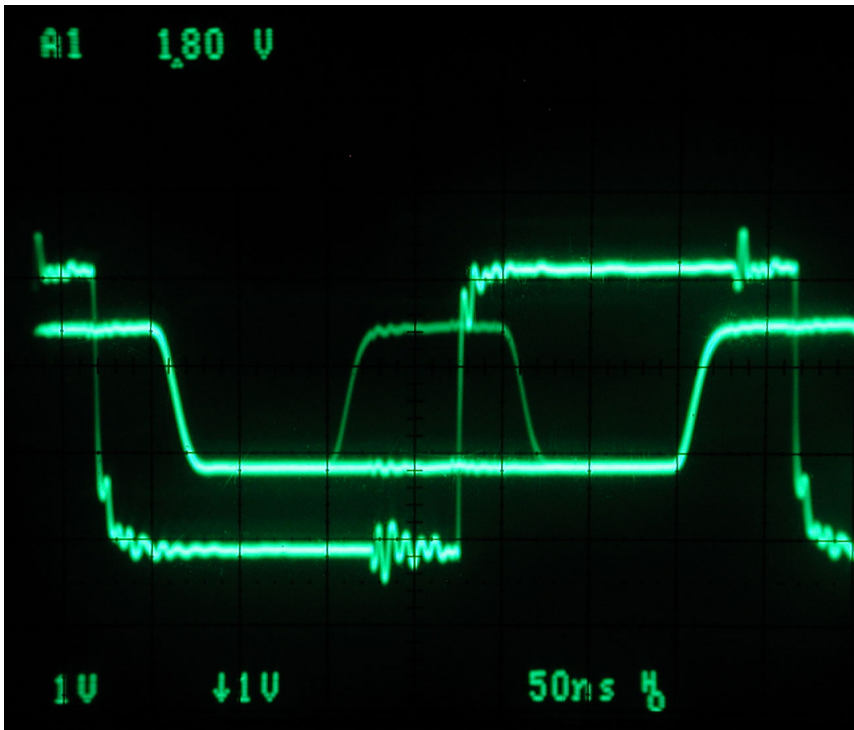


Figure 88: WelcomeToPulse4, Oscilloscope Capture, Full Amplitude

The acquired image shows a full-amplitude signal:

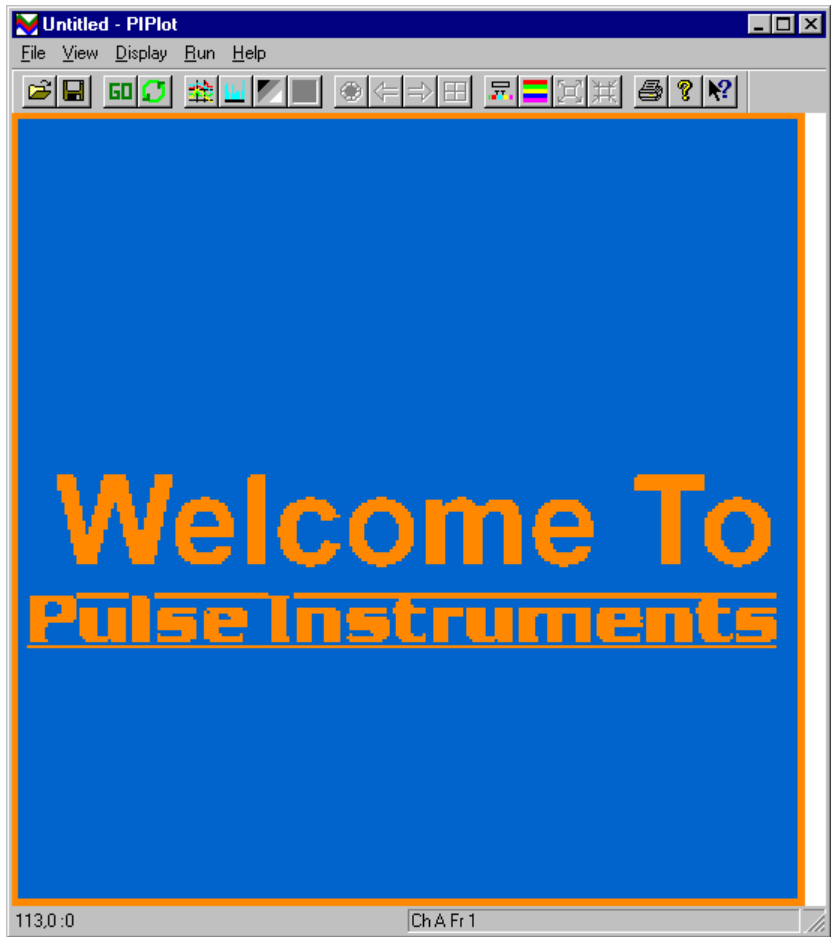


Figure 89: WelcomeToPulse4, False Color Plot, Full Amplitude

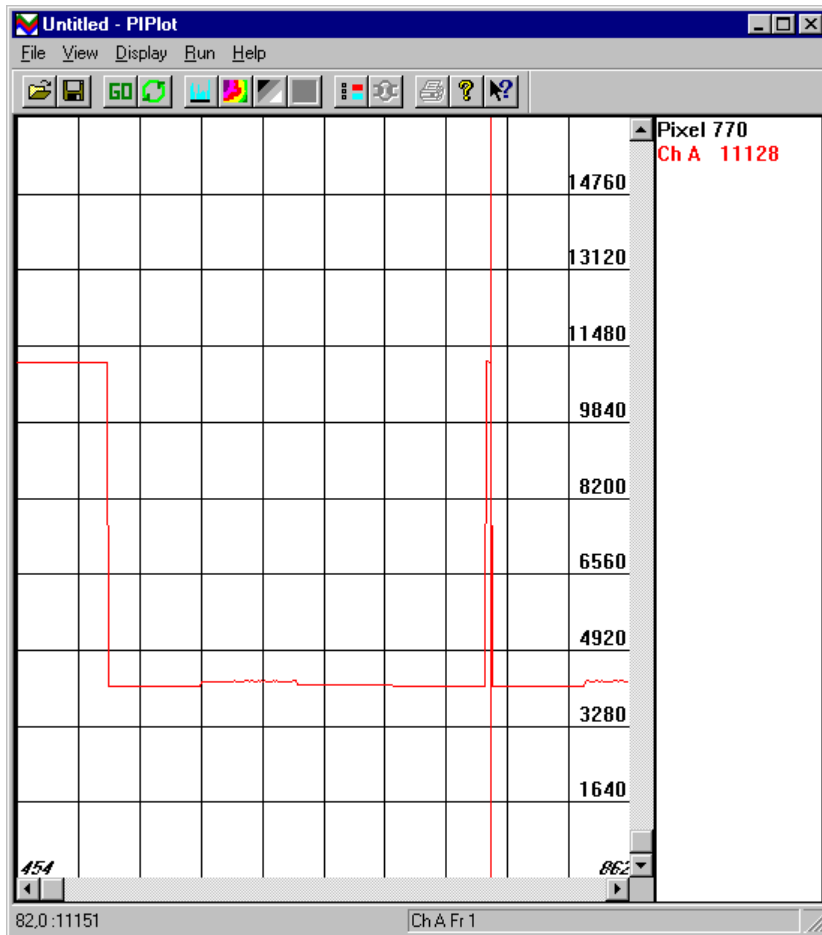


Figure 90: WelcomeToPulse4, 'Scope Plot, Full Amplitude

Once you have positioned the Convert strobe in the optimal location, you can then use the AIM mnemonic to adjust the gain and offset. Note that, because the pattern generator signal is greater than 2 V, any gain greater than 1 will saturate the ADCs.

In addition to the gain and offset, you can use the different DEFINE mnemonics to rotate or flip the image:

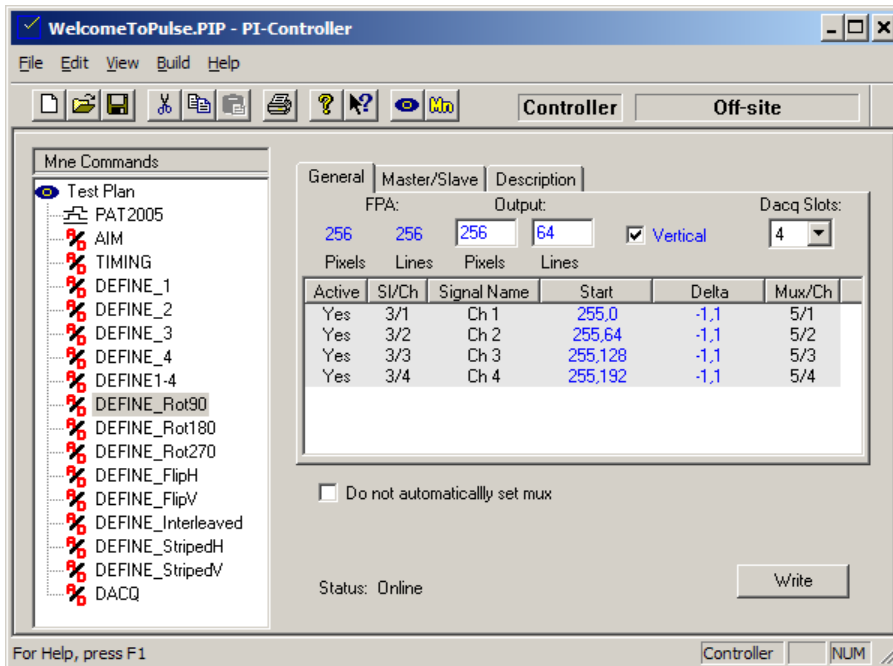


Figure 91: Define Mnemonic, Rotated 90 Degrees

Click the DEFINE_Rot90 mnemonic and click Write. The image will rotate by 90 degrees:

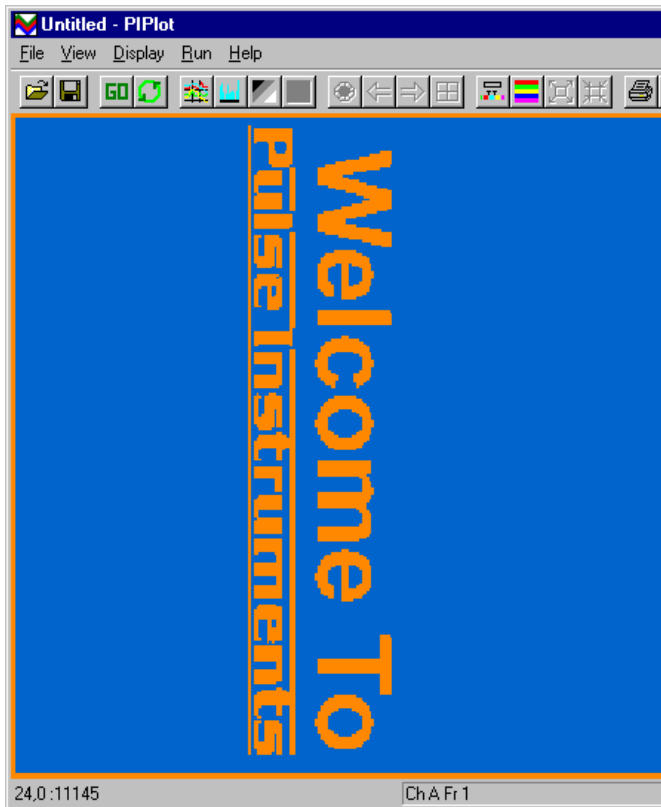


Figure 92: Image Rotated 90 Degrees

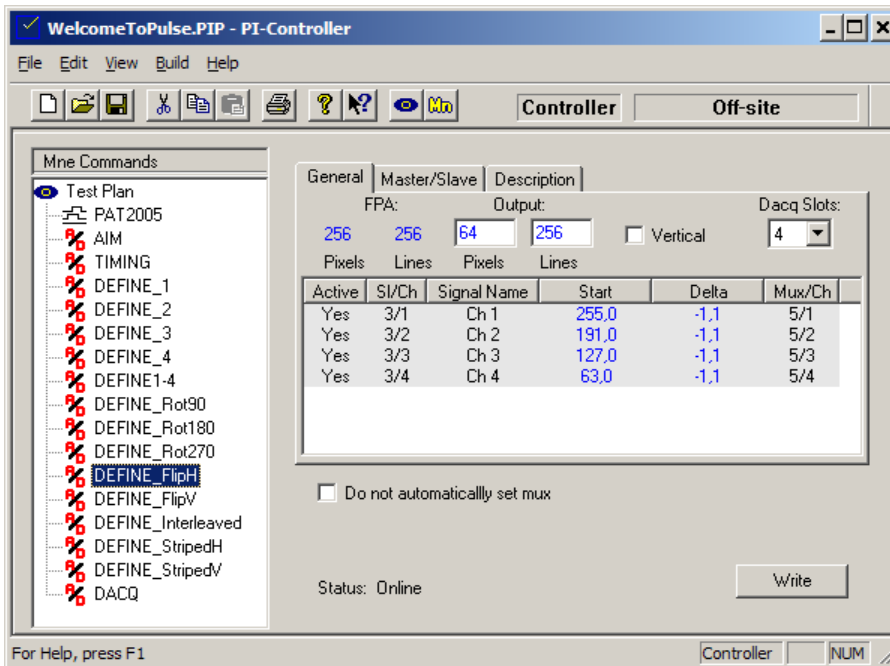


Figure 93: Define Mnemonic, Flipped Horizontally

Write in the DEFINE_FlipH mnemonic:

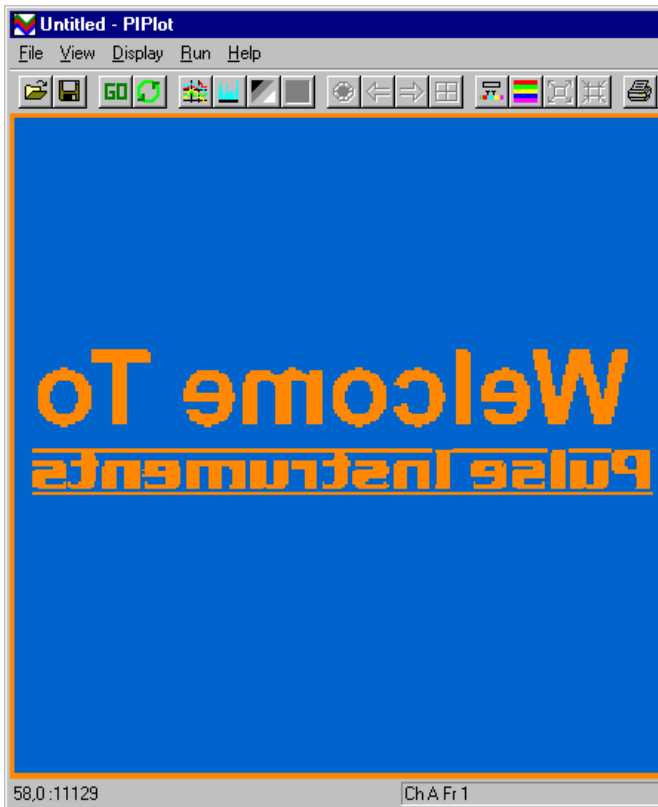


Figure 94: Image Flipped Horizontally

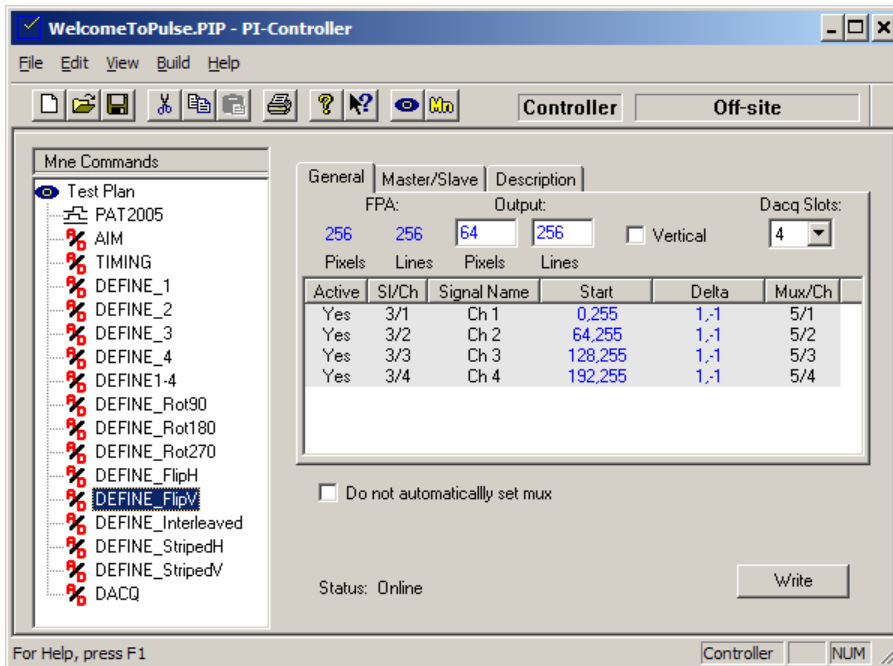


Figure 95: Define Mnemonic, Flipped Vertically

Write in the DEFINE_FlipV mnemonic:

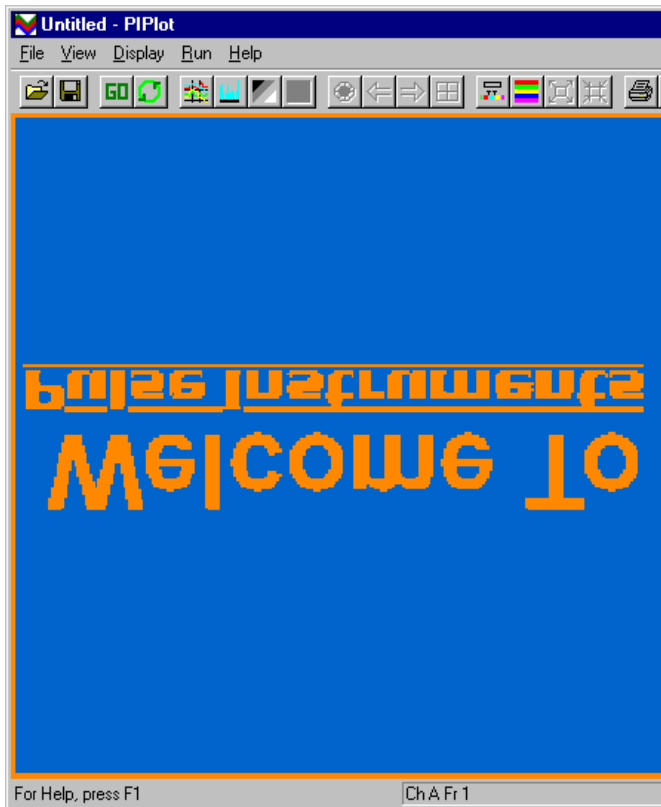


Figure 96: Image Flipped Vertically

6. TROUBLESHOOTING

6.1. Common Problems

The following is a list of commonly encountered problems and suggested actions for resolution. Please check the list of **Known Issues** (above) first, to determine whether the issue you are encountering needs a different workaround.

Problem	Description	Resolution/Workaround
FPA Noise	<ul style="list-style-type: none"> • Temporal FPA noise higher than expected 	<ul style="list-style-type: none"> • Ensure all signal cables are correct and correctly tightened. • Ensure cabling from DUT output to preamp input is no longer than 36"/1m. • Disconnect cables from AIM Monitor outputs. • Isolate preamplifier enclosures from the optical table. • Experiment with connecting or disconnecting all of the following: AIM chassis ground, analog ground, digital ground, DUT board ground(s). • Move clock, bias, and video cables away from potential sources of radiant noise (e.g. PI-3103D, AC power cables, other instrumentation, etc.) • Ensure AIM and preamps are not on top of or near radiant noise sources. • Turn off and/or disconnect other 3rd-party instrumentation in the system rack. Instruments with high current and/or switching supplies can inject significant noise into the system. If possible, place these in a separate rack with separate AC power. • Connect 3rd-party instrumentation to a different AC power circuit than the one powering the PI-3105 instruments. • Ensure that the DACQ Setup Pixel clock polarity is set correctly, e.g. INV when using the PI-3100-USBM and NONINV for all other models. • Avoid inducing CPU activity during sensitive noise experiments, especially in single-mainframe systems where the bias cards are in the same mainframe as the CPU card.
FPA Non-Uniform	<ul style="list-style-type: none"> • NUC, bad pixel replacement, or other post-processing failure 	<ul style="list-style-type: none"> • Ensure that the post-processing script(s) refer to valid DLL filenames and function names, with valid arguments. For best portability, use relative file names without full paths. • Ensure that DTA files have the same data type and pixel dimensions as acquired data.
Cannot write to Timing & Control card	<ul style="list-style-type: none"> • Timing delays cannot be changed. 	<ul style="list-style-type: none"> • Check cabling. Ensure that all ribbon cables are secured with the side latches, and that all micro-D connectors have their jackscrews fully secured. • Ensure that the PI-3103D is powered on (power switch illuminated) and that the power cable is properly connected.

Problem	Description	Resolution/Workaround
Acquisition does not complete	<ul style="list-style-type: none"> PI-Controller Status bar says ARMED 	<ul style="list-style-type: none"> If an acquisition has been triggered with no valid timing, the system may need to be restarted to reset the DACQ card. Ensure that PI-3103D has power, and that valid settings have been written to the PI-3100 AIM. Check for proper cabling, including Remote Arm cable. Ensure that Frame and Line are not reversed. The three LEDs on each active DACQ port should be blinking. Check Output Size and FPA Size in the desired Define mnemonic and ensure that the settings have been written to hardware. Check Master/Slave status in the last-written Define mnemonic. A card configured as a Slave cannot acquire unless its Master has active channels. Check Connections in Hardware Configuration window Check timing pattern setup and ensure that it meets the clocking requirements for the DEFINE mnemonic, including the Vertical checkbox. Check Line, Pixel and Frame pulse width/duty cycle and delay setup. Extend the pulse width of the Line Sync and Frame Sync signals. Ensure that Line Sync and Frame Sync signals are being sampled on the rising edge of the pixel clock. Sync or stop and re-start timing pattern
Stitching fails	<ul style="list-style-type: none"> PI-Plot image is scrambled. 	<ul style="list-style-type: none"> Check channel definitions, including Master/Slave settings Ensure that Stitch setting is set to Y for the Master card in the DACQ mnemonic.
Cannot set Connections	<ul style="list-style-type: none"> No AIM outputs appear in the Source pane of the Connections window AIM outputs appear in the Connections property page with a red stripe across the connection bar. 	<ul style="list-style-type: none"> Add the AIM/Timing connection first before adding data channel connections.
Hardware Not Responding error	<ul style="list-style-type: none"> System warns Hardware Not Responding after launching PI-Controller, after closing the Hardware Configuration window or after opening a file. 	<ul style="list-style-type: none"> Check power and cabling connections for unanticipated failures. This is normal behavior for simulated components; use the Don't warn again checkbox to suppress future warnings.

Problem	Description	Resolution/Workaround
No Video/ Wrong Video	<ul style="list-style-type: none"> Data is all zero or appears corrupted. Data has wrong amplitude or offset Data does not change from acquisition to acquisition 	<ul style="list-style-type: none"> Check all control, timing, and data connections. Check Gain, Offset, Filter and ADC settings Check timing pattern and timing delays (use the Monitor outputs to check strobe timing against video signal). Check PI-Plot's display settings (vertical scale, false-color table, or grayscale settings) to ensure that acquired values are within the displayed range Increase data pulse width (digital acquisition only). Choose correct data source in PI-Plot. Ensure that the DACQ Setup Pixel clock polarity is set correctly, e.g. INV when using the PI-3100-USBM and NONINV for all other models.
Low contrast	<ul style="list-style-type: none"> FPA video is dim or has low apparent response 	<ul style="list-style-type: none"> Use histogram control and Min/Max button to manually optimize the grayscale display Check the number of bits and Vmin/Vmax in DACQ Setup relative to the ADCs modules in use.
Monitor signals are ringing	<ul style="list-style-type: none"> Video and Clock monitor signals show ringing and overshoot 	<ul style="list-style-type: none"> Ensure your oscilloscope is set to 50 Ohm input termination, or add an external 50 Ohm shunt termination. Some clock feed-through in the monitor signal is normal, and does not affect measurement accuracy or noise performance.
Low frame rate	<ul style="list-style-type: none"> In continuous acquisition mode, there is a delay between frames 	<ul style="list-style-type: none"> Check the Pause Time entry in PI-Plot's Run menu. This value is 500 ms by default. Check the integration time in your timing pattern
DEFINE mnemonic errors	<ul style="list-style-type: none"> "Channel outside FPA" "Invalid Lattice Size" Wrong FPA size 	<ul style="list-style-type: none"> Ensure that Output Size, Start and Delta values are consistent with the state of the Vertical checkbox. Ensure that all Active channels sum to a contiguous rectangular area with no gaps or overlapping pixels. Ensure that the correct channels are set to Active Ensure that the Master/Slave configuration is correct
Spurious errors in the DEFINE mnemonic	<ul style="list-style-type: none"> Errors from the DEFINE mnemonic, despite a correct definition 	<ul style="list-style-type: none"> If additional channels are connected in the Advanced Hardware Configuration window, DEFINE mnemonics in existing PIP files (including any PIP file open at the time the connections are added) may show errors, even though they are entered correctly. This can be corrected by selecting each DACQ card from the Dacq Slots menu and clicking Write, and then by demoting/promoting or promoting/demoting any Slave cards in the Master/Slave property page. This can also be corrected by inserting a new DEFINE mnemonic into your test plan.

7. PI-3105 PROGRAMMER'S REFERENCE

7.1. Introduction

Previous sections of this PI-3105 Operators Manual are concerned with operation of the PI-3105 via Pulse Instruments PI-Controller software. The PI-3105 can also be controlled by your custom application using the PI-3105 command set. Please note that the PI-11000 Instrument Mainframe is a component of the PI-3105 Data Acquisition System. References in this and other manuals may make reference to either model number when referring to the instrument mainframe.

Custom applications can run on the CompactPCI CPU by linking into the Pulse Instruments DLL (dacq.dll). If the PI-11000 Instrument Mainframe housing your data acquisition system contains the optional IEEE-488 Card, it can also be controlled from a remote PC via commands sent over the IEEE-488 bus.

In Pulse Instruments terminology "**Local**" refers to the CompactPCI CPU, and "**Remote**" refers to an external PC, regardless of where the operator is sitting.

The following sections of the manual describe the interface and commands used to operate the PI-3105 via the IEEE-488 bus or via dacq.dll.

Commands may be sent the PI-3105 either as strings sent over the GPIB via gpib-32.dll's Send() command or via strings passed locally into dacq.dll's DacqInputMsg(). Responses may be read from the PI-3105 via a GPIB Receive() command from a remote PC or by passing a string pointer locally into DacqOutputMsg(). Operation is via either method is identical for all commands except for the following:

- GPIB "message mode" commands. The PI-11000 Instrument Mainframe has a single GPIB address, but contains instrument cards comprising three independent instrument types (Pattern Generator, Bias/Clock Drivers, and Data Acquisition) that have overlapping command sets. Three GPIB commands are used to switch communications among these instrument types. PATMSGs, BIASCLKMSGs, and DACQMSGs are used before each block of commands for each instrument type. These commands are necessary only for GPIB communications. For applications linking into Pulse Instruments DLLs on the local CPU, each set of commands can be sent to the specific DLL or application representing that instrument type; thus the message mode commands are not needed. Dacq.dll will accept the DACQMSGs command as a no-op in order to permit code re-use.
- Commands sent via GPIB must be terminated with a carriage-return and line feed (CHR\$(13)+CHR\$(10)). In hexadecimal these strings are "0D" and "0A." The dacq.dll will accept, but does not require, these termination characters.
- Handling Acquired Data: The text command STARTACQ may be sent either from a local application to dacq.dll or from a remote PC via GPIB. Once the data have been successfully acquired, the calling program may access the data in one of two ways: 1) data saved to file, or 2) pointer to the data passed back via a callback function. The file-based method can be used either custom applications running Locally or Remotely. The pointer-based method can be used only with applications running Locally and linked into dacq.dll. The data handling is controlled by the SetAcqCallBack() function, defined below. If this function is called with NULL or if it has never been called, then the dacq.dll will save the acquired data to a file automatically. If SetAcqCallBack() has been called with a valid address, then dacq.dll will call a callback function, also defined below, and pass the data to the calling application via the pointer. Information about the acquired data set can be read from the header of the saved file or by calling GetAcqDataInfo().

With the exception of PI-Plot, any action or function that can be performed from PI-Controller can also be performed by a custom application. You should already be familiar with using PI-Controller for any of the functions or features you wish to use before developing your application.

7.2. Local Operation

7.2.1. Configuring the GPIB Interface

If your PI-11000 Instrument Mainframe was ordered with the optional GPIB interface board (PI-31000), then it is configured by default as a GPIB instrument. Inbound GPIB communication with the PI-11000 is handled by PI-PAT; therefore if PI-PAT is running in Local GPIB mode, it will link to dacq.dll and prevent your application from linking to dacq.dll. To run your custom application, either quit PI-PAT or switch it to Local Only mode. Please see the PI-2005/PI-PAT manual for information on switching modes. If configured from the factory as an instrument, your PI-11000 will automatically launch PI-PAT at startup in Local GPIB mode. To disable this feature, remove the PI-PAT shortcut from C:\WINNT\Profiles\All Users\Start Menu\Programs\Startup.

When configured as an instrument, the PI-11000 runs a software package called NI-Device, from National Instruments, in order to emulate a standards-compliant GPIB instrument. If you wish to use your optional GPIB interface as a system controller (e.g. to control other equipment, such as a DVM or 'scope), then you must install the GPIB driver. To run the PI-11000 as a system controller, you must first uninstall NI-Device (via the Windows Add/Remove Programs control panel), and then install the IEEE-488 driver from the C:\Drivers folder.

If you wish to reconfigure the PI-11000 to be an instrument at a later date, you must uninstall the IEEE-488 driver and then reinstall NI-Device from the C:\Drivers folder. When reinstalling NI-Device, install only the run-time engine.

7.2.2. Linking to dacq.dll

If you are writing a custom application to run on the CompactPCI CPU, you may control the PI-3105 by linking into dacq.dll. Linking into dacq.dll requires two files:

- dacq.h
- dacq.dll

These files are available in the C:\Program Files\Pulse20\CodeSamples directory. The dacq.dll exposes four functions:

- void DacqInputMsg(char *pMsg);
- void DacqOutputMsg(char *pMsg, int nMaxBytes);
- void SetAcqCallback(ACQCALLBACKFN AcqAcquisitionCB)
- BOOL GetAcqDataInfo(int nCardAdx, ACQINFO* acqInfo)

DacqInputMsg() must be called with a pointer to a string containing a valid text command. Note that each valid command must be contained in its own string, which is different from the way commands are handled by pipci.dll, for example.

DacqOutputMsg() must be called with a pointer to a string that will hold the returned message and a maximum size for that message. Error messages and responses to queries are returned to the calling application via DacqOutputMsg(). There are no currently-defined commands that return messages larger than xx bytes.

If **SetAcqCallback()** is called with NULL or has never been called, then dacq.dll will save the acquired data to the defined filename (See "SAVE TO FILE" command, below) after every completed data acquisition cycle.

SetAcqCallBack() must be called at least once with a non-NULL value in order to use the pointer-based method for passing data from dacq.dll to the calling application. If SetAcqCallBack() is called with the address of the Callback function, then dacq.dll will call your Callback function (which you must implement in your application, per the definition below) when the acquisition is complete. To save data to a file after SetAcqCallBack() has been set, use the "SAVE FILE" command, defined below.

GetAcqDataInfo() may be called with the CompactPCI slot number of a data acquisition card and a pointer to a structure in order to get information about the data acquisition.

Your application must implement a callback function of this form:

```
void CALLBACK AcquisitionCB(int nMsg, LPCSTR strMsg, ACQINFO* acqInfo);
```

nMsg is defined as:

```
enum AcqCB {ACQ_CB_DONE,ACQ_CB_WAITING,ACQ_CB_MSG}
```

- ACQ_CB_DONE = 0, returned when the acquisition is complete (e.g. all armed cards have completed)
- ACQ_CB_WAITING = 1, returned when one or more data acquisition cards has completed acquiring, but the system is still waiting on other armed cards
- ACQ_CB_MSG = 2, reserved for error messages or a "verbose" mode to be implemented

strMsg is a status message such as "Waiting for acquisition card 2"

ACQINFO is a pointer to structure that contains information about the memory size and type of data. This pointer might be null for messages other than ACQ_CB_DONE. The structure is defined as:

```
struct ACQINFO
{
    long IPixelCount;
    long IPixelSkip;
    long ILineCount;
    long ILineSkip;
    long IPixelPass;
    long ILinePass;
    long IFrames;
    short NumBits;
    BOOL bFloatData;
    BOOL bAverageFrames;
    Int nCardAdx; //Address of the card assigned by PI usually the master
    Int nReportingAdx; //Address of the card that is reporting
    PVOID ptrData;
    float fPixPeriod;
    float fGain;
    float fOffset;
    BOOL bLocal;
    BOOL bEnableCDS;
    float fVMin;
    float fVMax;
};
```

Note that PVOID ptrData may be destroyed when STARTACQ is called with a new acquisition size. Acquisition size is defined as:

$$\text{Framecount} \times \text{Width} \times \text{Height} \times \text{BytesPerPixel}$$

BytesPerPixel is 2 for Integer data and 4 for Floating Point data. For example, a 2048 x 2048 array represents 8 Mbytes of data per frame in Integer mode, and 16 MB per frame in Floating Point mode.

If STARTACQ is called without changing the acquisition size, the pointer remains the same. If STARTACQ is called after any commands have changed the acquisition size, the old pointer is destroyed and a new pointer is created. Commands that may affect acquisition size are:

- FRAMECOUNT
- OUTPUT
- SET ACTIVE CHAN
- FLOAT
- SLAVE
- AVERAGE

Use GetAcqDataInto() or GET FPA, GET FRAMECOUNT, GET FLOAT, and GET SLAVE to check the acquisition parameters before assuming that the pointer to the data will be preserved.

The name of the callback function is not important as long as the format is adhered to.

7.3. Remote IEEE-488 Interface Setup

If your PI-11000 Instrument Mainframe was ordered with the optional GPIB interface board (PI-31000), then it is configured by default as a GPIB instrument. Inbound GPIB communication with the PI-11000 is handled by PI-PAT; therefore PI-PAT must be running, and be in Local GPIB mode, in order to receive any GPIB commands. If configured at the factory as a GPIB instrument, PI-PAT will already be configured to start up automatically, in Local GPIB mode. Please see the PI-2005/PI-PAT manual for information on switching modes.

If your PI-11000 has been re-configured as a system controller, or if you ordered your GPIB interface after your system was delivered, then you may have to install NI-Device. First, use the Windows Add/Remove Programs control panel to uninstall the National Instruments IEEE-488 driver, if present, then install NI-Device from the C:\Drivers folder. When installing NI-Device, install the run-time engine only. Once NI-Device has been installed, PI-PAT can then be run in Local GPIB mode and will handle in-bound GPIB communication.

To run the PI-11000 as a “headless” instrument (e.g. without a monitor, keyboard and mouse), PI-PAT must be configured to start up automatically, in Local GPIB mode. To enable or re-enable this feature, either run the PI-DATS installer and choose the “Local (CompactPCI)” option with the “Automatic PI-PAT startup” box checked, or else create the following shortcut:

```
“C:\Program Files\PULSE20\PI_Pat.exe” /Startup
```

in this directory:

```
C:\WINNT\Profiles\All Users\Start Menu\Programs\Startup
```

By default, each PI-11000 Instrument Mainframe with the optional IEEE-488 interface is configured with an address of 11. To change the GPIB address, please see the PI-2005 Operators Manual.

The PI-3105 GPIB syntax follows the NI-488.2 standard. There are a few exceptions, where the NI-488 standard is used, in order to ensure backward compatibility with the previous Pulse Instruments equipment.

7.4. Input/Output Protocol

7.4.1. Command Strings and File Exchange

7.4.1.1. Input to the PI-3105

Commands sent to the PI-3105 must obey the syntax specified in this document. When describing a command and its syntax, this document will use the following conventions:

All commands recognized by the PI-11000 will be printed in all capital letters.

Example: OFFSET 1.020↵

Placeholders for user-supplied values and parameters will be printed in lowercase, boldface, italicized characterized.

Example: CONVERT STROBE *ns*↵

Optional arguments are enclosed in square brackets, e.g. RESET [*card*], but the square brackets should **not** be sent as a part of the command string. Strings sent to the PI-11000 need not be in capital letters, but they will be printed here in all capitals for clarity.

Commands requiring a choice of operands or sub-commands are written with an OR character “|” between each choice. For example, the clock source for the B port of a data acquisition card may be set to one of three choices. The command is documented as:

CLOCK SRC DATA | SMA | SHARE↵

This indicates that the string “CLOCK SRC” should be followed by one of “DATA” or “SMA” or “SHARE”, e.g.

CLOCK SRC SMA↵

All command lines sent to the PI-3105 over GPIB must be terminated with a carriage return and a line feed (CHR\$(13)+CHR\$(10)). In hexadecimal these strings are "0D" and "0A." In this document, these two characters will be represented together by a "↵" at the end of each command line. Please note that, although it is displayed as a single character in this manual, both a carriage return and a line feed must be sent. The two examples above are correctly terminated.

Commands sent to the DLL via DACQInputMsg() do not require termination characters, as the end of the string is used to delimit a valid command.

7.4.1.2. Output From the PI-3105

All information sent back from the PI-3105 is in the form of strings with the exception of the acquired data, which is either written to a file or passed to the calling program via a pointer. Programmers using low-level GPIB routines should use the Receive() function instead of the ibrd() function, especially when reading the output from a GET or STATUS command.

Strings are terminated with two characters:

0D 0A

which are a carriage return and line feed (CHR\$(13)+CHR\$(10)).

7.4.1.3. Checking Status of the PI-3105

For query commands that return data (such as GET MUX CARDS or GET OFFSET commands), a GPIB Receive() or a call to DACQOutputMsg() will return the requested data.

For commands that do not return data (such as SETACTIVECHAN or STARTACQ), a GPIB Receive() or a call to DACQOutputMsg() will return the current error message (READY if there is no error). This eliminates the need to send the STATUS command after every command has been sent.

Please note that consecutive GPIB Receive() commands are permitted with the PI-3105, but consecutive GPIB ibrd() commands will result in a GPIB time-out.

7.5. Sample Application/Testing Utility (GPIBCom)

Included with your PI-11000 is an unsupported utility, **GPIBCom.exe** for sending and receiving strings from Pulse Instruments equipment or other GPIB-compatible devices. **GPIBCom** can be run Locally, on the CompactPCI CPU board, or Remotely, on a Windows PC with a National Instruments GPIB interface card. When running Locally, GPIBCom communicates either with Pulse Instruments DLLs (**dacq.dll** and **pipci.dll**) via their exposed functions or with Pulse Instruments applications (**PI-PAT**) via Automation. When running Remotely, **GPIBCom** communicates with any instrument that is capable of communicating over GPIB. In each case, this manual will refer to the instrument, DLL or application as the “target.”

Because Pulse Instruments DLLs and applications use the same command set whether run Locally or Remotely, via the Pulse Instruments GUI or a custom application, **GPIBCom** can be used to learn, test, and troubleshoot the hardware and your custom application.

7.5.1. Local Setup

GPIBCom is located in the C:\Program Files\PULSE20 folder. If you wish to run **GPIBCom** in Local mode, then the executable must remain in this folder, as it links into several Pulse Instruments DLLs that also are in this folder. A shortcut to this executable may be placed anywhere on your hard drive or desktop folder.

Launch **GPIBCom** by double-clicking on its icon or shortcut:

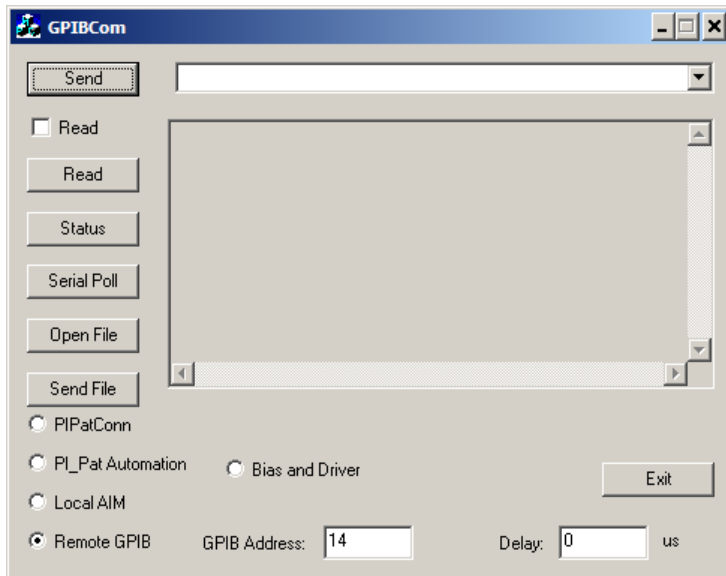


Figure 97: GPIBCom

By default, **GPIBCom** launches in Remote GPIB mode, and is set up to communicate with a GPIB-compatible device. To communicate with Pulse Instruments software on the CompactPCI CPU, click one of the following radio buttons:

- **PI_Pat Automation**—to communicate with **PI-PAT**, which controls the Pattern Generator hardware. If PI-PAT is not running when the first command is sent, PI-PAT will be launched.
- **Bias and Driver**—to communicate with **pipci.dll**, which controls the DC Bias and Clock Driver cards. If PI-PAT is running on the Local CPU, it **must** be in Local Only mode before **GPIBCom** is launched.
- **Local AIM**—to communicate with **dacq.dll**, which controls all data acquisition hardware. If PI-PAT is running on the Local CPU, it **must** be in Local Only mode before **GPIBCom** is launched.

Multiple instances of **GPIBCom** may be launched, in different modes, to communicate with different targets. Note that each DLL or application (in the case of PI-PAT) is quit and re-launched when the radio buttons are clicked in a particular instance of **GPIBCom**. Therefore, if you wish to communicate with **PI-PAT** and with **dacq.dll**, for example, it is highly recommended that you launch two instances of **GPIBCom**, rather than switching one instance of **GPIBCom** back and forth between the two modes.

Note that **GPIBCom** will link to **dacq.dll** or **pipci.dll** when used, and therefore will prevent PI-Controller from controlling the relevant hardware. If PI-Controller is launched while **GPIBCom** has locked the hardware, PI-Controller will report a “Hardware not responding” error. The instance(s) of **GPIBCom** using **dacq.dll** and/or **pipci.dll** should be quit before launching PI-Controller.

7.5.2. Remote Setup

GPIBCom may be copied from your PI-11000 to any Windows PC equipped with a National Instruments GPIB interface card and driver. No other drivers or DLLs are required for remote control. In Remote mode, please ensure that the **Remote GPIB** radio button is selected, and that the **GPIB Address** box corresponds to the GPIB address of the device you are controlling.

Multiple instances of **GPIBCom** may be launched to communicate with multiple instruments at different GPIB addresses. To communicate with multiple instrument types within a PI-11000 Instrument Mainframe, use the GPIB Message Mode commands (**PATMSGs**, **BIASCLKMSGs**, and **DACQMSGs**) to switch among different messaging modes.

When communicating with a PI-11000 Instrument Mainframe or PI-2005 Pattern Generator, ensure that **PI-PAT** is running on the Local CPU in Local GPIB mode.

7.5.3. General Operation

GPIBCom can send either single lines of text or entire text files to a target. To send a single string, set up communications per the instructions above, and then type a command in the text entry box next to the **Send** button:

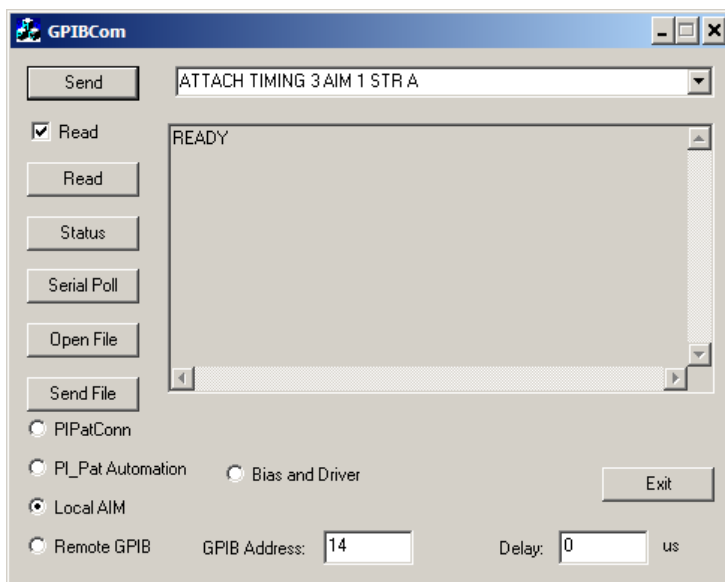


Figure 98: GPIBCom communicating with dacq.dll

Click the **Send** button to send the command. If the **Read** checkbox is checked, **GPIBCom** will immediately query the target for a response. In the case of Pulse Instruments instruments and DLLs, the response will be either an error message, a no-error message, or the response to the query.

Clicking the **Read** button causes **GPIBCom** to query the target for a response.

Clicking the **Status** button is equivalent to sending the string "**STATUS**", followed by a **Read**.

Clicking **Serial Poll** executes a standard GPIB Serial Poll and displays the result. Serial Poll functions only when talking to a Remote GPIB instrument.

GPIBCom can also send the contents of text file, either one line at a time or all at once. To send a file, click the **Open File** button. A standard Windows **Open File** dialog box will appear, allowing you to select a text file containing a command script. The Open File box also contains a checkbox to specify an optional output log. If the output log option is chosen **GPIBCom** will then prompt you for a filename to save query responses to. This must be an existing file, and **GPIBCom** will append strings to the end of it.

Once the file has opened, the first line of text from the file will appear in the text entry box. Click **Send** to send a single line of text and read the next line into **GPIBCom**. Click **Send File** to send the entire contents of the file to the target. Please note that, when **Send File** is clicked, the file contents are sent

via a series of GPIB Send() commands, not ibwrtf(). Once a file has been sent via **Send File**, the line counter resets, and clicking Send File will send the file again, from the beginning of the file.

To cancel sending of a file, click the **Close File** button.

If you wish to insert a delay between strings that are sent to the target via **Send File**, type a value in the **Delay** box.

8. PI-3105 COMMAND REFERENCE

8.1. Chart of Commands

The following tables show the commands supported by the PI-3105 Multi-Channel Data Acquisition System and each of its components. There are 3 categories of commands:

- **System commands**--System commands do not require any card or channel specifier. They can be issued at any time and will operate on the data acquisition system as a whole. System commands include commands specific to a Timing/Control or Mux card, since the slot number is embedded in these commands.
- **Card commands**—These commands are specific to a particular Data Acquisition card or port, and must be preceded by a DAQ CARD command specifying a card. Use the GET . . . CARDS or ID commands to identify cards installed in the system. Once a card is “selected” for programming, any card-specific commands that follow will apply to that card until the selection is changed.
- **Channel commands**—These commands are specific to a particular ADC channel, and must be preceded by a CHAN command specifying the number or user-defined name of the channel. Channel numbers are specified by the ATTACH TIMING commands, and channel names may be user-specified using the NAMEAD command. Use the GET NAMES commands to identify named ADC channels installed in the system. Once a channel is “selected” for programming, any channel-specific commands that follow will apply to that channel until the selection is changed

8.1.1. System Commands

Command	Short Description
; (REM)	Comment delimiter
@name	Selects a named ADC channel for programming
ACQSTATUS	Returns the status of any pending data acquisition
ACQTIME	Sets the acquisition timeout
ALLCARD	Selects all installed DAQ cards for simultaneous programming
ALLCHAN	Selects all installed ADC channels for simultaneous programming
ATTACH DAQ	Specifies data attachments from ADC channels to optional muxes and to data acquisition cards
ATTACH TIMING	Specifies timing attachments from Timing/Control card to Convert and optional CDS strobes for ADC channels
BIASCLKMSGS	Sets Bias/Clk message mode
BLOCKING	Sets the blocking/non-blocking mode for data acquisitions
CHAN	Selects an ADC channel for programming
DACQMSGS	Sets Data Acquisition message mode
DAQ CARD	Selects a Data Acquisition Card for programming
FILTERS	Sets the global filter setting
GET ACQTIME	Returns the acquisition timeout
GET ACTIVECHAN	Returns which channels in the system are active
GET ATTACHMENTS	Returns all data and timing attachments
GET BLOCKING	Returns the blocking/non-blocking mode for data acquisitions

Command	Short Description
GET CHAN	Returns the number or name of the currently selected ADC channel
GET DAQ CARDS	Returns location of all Data Acquisition cards in the system
GET FILTERS	Returns the current global filter setting
GET MUX	Returns the multiplexer or switch setting for a Mux card
GET MUX CARDS	Returns location of all Multiplexer cards in the system
GET NAMES	Returns all defined names for ADC channels in the system
GET PIXEL PERIOD	Returns the pixel period for a Timing/Control card
GET PORT	Returns the letter of the currently selected port on a data acquisition card
GET TIMING CARDS	Returns location of all Timing/Control cards in the system
GET TIMING MAXIMA	Returns the maximum delay value for each Timing/Control card in the system
GET TIMING STEPS	Returns minimum delay increment for a Timing/Control card
ID	Returns the slot locations of all installed Timing/Control, Data Acquisition, and Multiplexer cards
MUX	Sets multiplexing mode for a Mux card
NAMEAD	Applies a user-defined name to an ADC channel
PAUSE	Pauses the system for specified amount of time
PIXEL PERIOD	Sets the pixel period for a Timing/Control card
PORT	Selects a port on a data acquisition card for programming
RESET	Resets a Timing/Control card
SETACTIVECHAN	Specifies which channels in the system are active
STARTACQ	Starts all data acquisition cards that have active channels
STATUS	Returns system status and error
STOPACQ	Stops all running data acquisition cards

8.1.2. Card-Specific Commands (Including port-specific commands)

Command	Short Description
BITSPERPIX	Sets the bit-width for the selected Port on the selected Data Acquisition card.
CLOCK SRC	Sets the clock source for the selected port on the selected Data Acquisition card
FLOAT	Sets the floating-point or integer mode
FRAMECOUNT	Sets the number of frames to be collected on the selected Data Acquisition card
GET BITSPERPIX	Returns the bit-width for the selected Port on the selected Data Acquisition card
GET CLOCK SRC	Returns the clock source for the selected port on the selected Data Acquisition card
GET FLOAT	Returns the floating-point or integer mode
GET FPA	Returns the total FPA size of all ADC channels connected to the selected Data Acquisition card, including any connected and configured Slave cards.

Command	Short Description
GET FRAMECOUNT	Returns the number of frames to be collected on the selected Data Acquisition card
GET INVERT	Returns the inversion mode for pixel, line, and frame sync clocks for the selected port on the selected Data Acquisition card
GET OUTPUT	Returns the output size for all ADC channels connected to the selected Data Acquisition card
GET SAVETOFILE	Returns the path, filename and options for the file to save after each Data Acquisition card completes an acquisition
GET SETUP MODE	Returns Setup or Raw mode for floating point conversion
GET SLAVE	Returns the Slave or Independent status of a selected Data Acquisition card that is connected to a Master
GET STITCHING	Returns the stitching setting
GET VERTICAL	Returns the plot direction for the FPA defined on the selected Data Acquisition card
GET VMAX	Returns the voltage corresponding to a full-scale value from the ADCs.
GET VMIN	Returns the voltage corresponding to a zero (0x0000) value from the ADCs.
INVERT/NINVERT	Sets the inversion mode for pixel, line, and frame sync clocks for the selected port on the selected Data Acquisition card
OUTPUT	Sets the output size for all ADC channels connected to the selected Data Acquisition card
PORT	Selects a port on a Data Acquisition card for programming
SAVETOFILE	Specifies a path and filename to save the selected Data Acquisition card's data to after each acquisition
SETUP MODE	Sets Setup or Raw mode for floating point conversion
SLAVE	Sets the Slave or Independent status of a selected Data Acquisition card that is connected to a Master
STITCHING	Turns the stitching mode on or off
VERTICAL	Sets the plot direction for the FPA defined on the selected Data Acquisition card
VMAX	Sets the voltage corresponding to a full-scale value from the ADCs.
VMIN	Sets the voltage corresponding to a zero (0x0000) value from the ADCs.

8.1.3. Channel-Specific Commands

Command	Short Description
CDS	Sets the optional CDS mode for the selected ADC channel
CDS STROBE	Sets the optional CDS strobe delay for the selected ADC channel
CONVERT STROBE	Sets the convert strobe delay for the selected ADC channel
FRAME SYNC/LINE SYNC	Sets the Frame Sync and Line Sync delays for AIM containing the selected ADC channel
GAIN	Sets the gain for the selected ADC channel
GET ADC IDS	Returns 3-bit ID code for the selected ADC

Command	Short Description
GET CDS	Returns the CDS mode for the selected ADC channel
GET CDS STROBE	Returns the CDS strobe delay for the selected ADC channel
GET CONVERT STROBE	Returns the convert strobe delay for the selected ADC channel
GET GAIN	Returns the gain for the selected ADC channel
GET MONITOR	Returns the output monitor channels for the AIM containing the selected ADC channel
GET OFFSET	Returns the offset voltage for the selected ADC channel
GET POSITION	Returns the plot coordinates and increment for the selected ADC channel
GET PREAMP ID	Returns 2-bit ID code for the selected preamplifier
GET SYNC	Returns the Frame Sync and Line Sync delays for AIM containing the selected ADC channel
OFFSET	Sets the offset voltage for the selected ADC channel
POSITION	Sets the plot coordinates and increment for the selected ADC channel
VIDEO/CONVERT/CDS	Selects the output monitor channels for the AIM containing the selected ADC channel

8.2. Command Sequencing

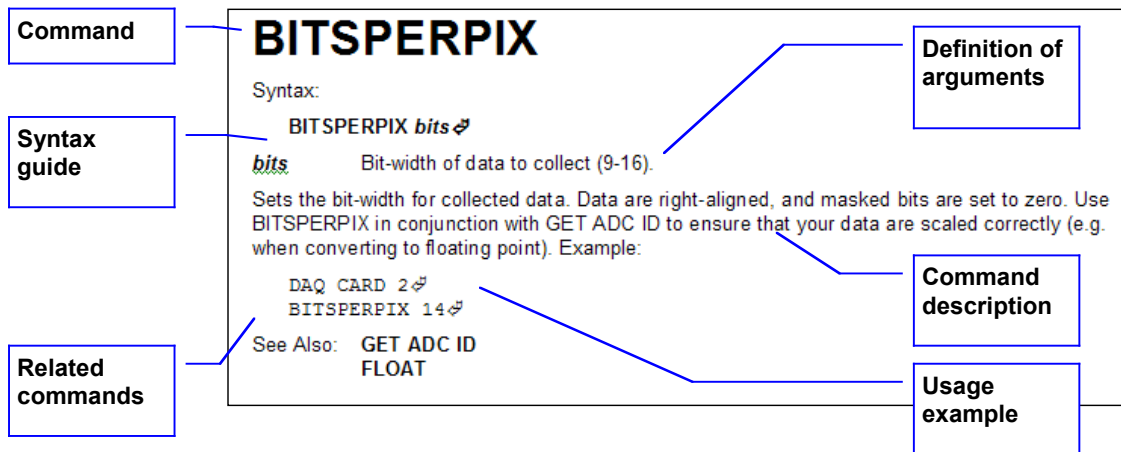
Certain commands for the PI-3105 are related to the specification of hardware connections, some are related to the setup of a particular DUT or application, and still others are related to parameters that may be varied within a test plan for a given DUT.

Furthermore, channel-specific commands have no meaning until the channel attachments have been defined. Therefore, it is recommended that your application or script perform the following functions in sequence:

- Define hardware connections and Setup
 - ATTACH TIMING commands
 - ATTACH DAQ commands
 - Activate channels using SET ACTIVE CHAN
 - Define Slave cards using the SLAVE command
 - Set up data acquisition hardware using BITS PER PIX, CLOCK SRC, and INVERT
- Define your DUT or DUT readout configuration using
 - OUTPUT, VERTICAL and POSITION
 - Check your setup using GET FPA
- Set up tests and acquire data
 - Define your data collection using FRAMECOUNT, FLOAT, VMIN, VMAX, SETUP MODE, STITCHING, and SAVE TO FILE
 - Set up your ADC strobes using PIXEL PERIOD, CONVERT, CDS, FRAMESYNC and LINESYNC
 - Set up your GAIN, FILTER, OFFSET and various MONITOR settings
 - Setup your callback function, if used
 - Acquire data using STARTACQ

8.3. Command Reference Format

Each entry in the command reference is presented in the following format:



Commands are listed in alphabetical order, and each entry includes a syntax guide, a definition of each argument, a description of what the command does, an example of usage with sample arguments, and a listing of related commands.

; (REM)

Syntax:

```
; remarks ↵
```

Remarks any text.

Allows the user to insert remarks in any command line. Once the semicolon is received, the instrument will ignore anything between it and the next carriage return and line feed. Example:

```
CONVERT STROBE 4.00; IGNORE THIS ↵
;IGNORE THIS AS WELL ↵
```

@

Syntax:

```
@label command ↵
```

label any previously defined label

command any valid channel command

Once a NAME AD command has been issued, @label can be substituted for the standard ADC channel number argument in a command. For example, if ADC channel 1 had previously been assigned the label ALPHA, then the command sequence:

```
@ALPHA ↵
CONVERT STROBE 5.00 ↵
CDS STROBE 3.00 ↵
```

would set Channel 1's Convert and CDS strobes to 5 ns and 3 ns, respectively.

See Also: **ALLCHAN**
CHAN
NAMEAD

ACQSTATUS

Syntax:

ACQSTATUS ↵

Returns the current state of the data acquisition. If cards have armed, but not completed, they will be reported as "Waiting." When all cards have completed, ACQSTATUS will return Idle. Example:

```
ACQSTATUS ↵
                ↳Waiting for 5↵
ACQSTATUS ↵
                ↳Waiting for 5↵
ACQSTATUS ↵
                ↳Waiting for 5↵
ACQSTATUS ↵
                ↳Idle
```

See Also: **ACQTIME**
BLOCKING
SAVETOFILE
STARTACQ
STOPACQ

ACQTIME

Syntax:

ACQTIME *seconds* ↵

seconds Number of seconds to wait before returning control to the user.

Sets the acquisition timeout. If Blocking is enabled when STARTACQ is received, the system/DLL will refuse any further communications or user input until either all armed cards complete or until the acquisition timeout has elapsed. If Blocking is disabled, then ACQTIME is ignored, and the user or application may use STOPACQ to terminate a pending acquisition.

If Blocking is enabled and an acquisition has not completed by the timeout period, the system/DLL will return an "Acquisition Timeout" message.

Example:

```
ACQTIME 60 ↵
```

ACQTIME may be set to 0 for infinite timeout. **WARNING:** if ACQTIME is set for 0, Blocking is enabled, and the acquisition fails to complete for any reason, the system or DLL will hang and will have to be restarted. There is no way to interrupt the system during ACQTIME while Blocking is enabled. Users desiring to use an infinite timeout period should test their applications with a finite timeout before switching to 0 timeout.

See Also: **STARTACQ**
ACQSTATUS
STOPACQ
BLOCKING
SAVETOFILE

ALLCARD

Syntax:

ALLCARD ↗
command ↗

command any valid Data Acquisition card command

Many commands can be issued to all available Data Acquisition cards by using ALLCARD in place of the standard DAQ CARD selector commands. See **Section 8.1. Chart of Commands** for a list of commands that support the ALLCARD syntax. Example:

```
ALLCARD ↗  
OUTPUT P64 L512 ↗  
VERTICAL FALSE ↗
```

See Also: **DAQ CARD**
ALLCHAN

ALLCHAN

Syntax:

ALLCHAN ↗
command ↗

command any valid command

Many commands can be issued to all available ADC channels by using ALLCHAN in place of the standard CHAN selector command. See **Section 8.1. Chart of Commands** for a list of commands that support the ALLCHAN syntax. Example:

```
ALLCHAN ↗  
CONVERT STROBE 5.00 ↗  
CDS STROBE 3.00 ↗
```

See Also: **@**
ALLCARD

AOI

Syntax:

AOI startx, starty, sizex, sizey ↗

startx horizontal address (0-based) of starting pixel

starty vertical address (0-based) of starting pixel

sizex horizontal size (1-based) of desired rectangle

sizey vertical size (1-based) desired rectangle

Reduces each acquired frame to a rectangular subset. The **startx** and **starty** arguments specify the first (upper-left) pixel to be retained. Pixels have zero-based numbering; therefore the first possible pixel address is 0,0, and the default arguments for **startx** and **starty** are 0. The **sizex** and **sizey** arguments specify the size of the area to be retained. The AOI Size uses one-based numbering; therefore a VGA-sized frame is 640,480. The argument “**max**” may be used in place of either or both numbers, and will resolve to the width and/or height of the FPA as defined by the set of FPA definition commands. “max,max” is the default setting for **sizex** and **sizey**.

If the AOI lies outside the current FPA definition, it will return the error, “AOI Outside FPA.”

Note that the AOI feature reduces the data set after it has been acquired in hardware, and therefore the AOI feature does not increase the maximum number of frames that can be acquired by the system. Example:

```
DAQ CARD 2 ↵  
AOI 0, 0, 256, 256 ↵
```

Example:

```
DAQ CARD 2 ↵  
AOI 100, 100, max, max ↵
```

See Also: **OUTPUT
POSITION
GET AOI
GET FPA**

ATTACH DAQ

```
ATTACH DAQ z, i CHAN a [MASTER w, j] ↵
```

```
ATTACH DAQ z, i MUX y, a, b, c, d [MASTER w, j] ↵
```

- z** slot number of the Digital Acquisition Card.
- y** slot number of the Multiplexer Card.
- w** slot number of the Digital Acquisition Card that is Master for slot z.
- i** data port (A or B) on the Digital Acquisition Card
- j** Arm-Out connector for Digital Acquisition Card w.
- a,b,c,d** ADC Channel number(s).

Specifies data connections between a Digital Acquisition Card port, an optional Multiplexer, and 1-4 ADC channels from the AIMS. If you are using a Timing and Control card with an AIM, you must also send the appropriate ATTACH TIMING commands.

Each AIM data output being used must eventually connect to a port on a DAQ. A connection through a Mux is optional. Any output blocks not specified are assumed to be unused. Mux channel inputs of 0 are assumed to be dummy inputs.

Example 1:

```
ATTACH DAQ 4, A CHAN 1 ↵  
ATTACH DAQ 4, B CHAN 2 ↵
```

This is a basic example, attaching Port 1 on the DAQ in slot 4 to the first ADC channel in the system, and Port 2 on the same DAQ to ADC channel 2. By definition, ADC Channel 1 is the first ADC in AIM 1. Please see **Section 3.1. Channel Assignments** for more details.

Example 2:

```
ATTACH DAQ 4, A CHAN 1 ↵
ATTACH DAQ 4, B CHAN 2 ↵
ATTACH DAQ 5, A CHAN 3 MASTER 4,1 ↵
ATTACH DAQ 5, B CHAN 4 MASTER 4,1 ↵
```

This sequence connects ADC Channels 1 and 2 to ports 1 and 2 on the DAQ cards in slot 4, and connects ADC channels 3 and 4 to the ports on DAQ 5. It also connects the Arm-Out connector #1 on DAQ in slot 4 to the Arm In connector on DAQ 5. This enables DAQ 5 to be a Slave of DAQ 4. Note that all channels connected to DAQ 5 **must** have the Master connection specified.

Example 3:

```
ATTACH DAQ 4, A MUX 3, 1, 2, 3, 4 ↵
ATTACH DAQ 4, B MUX 5, 5, 6, 7, 8 ↵
ATTACH DAQ 7, A MUX 6, 9, 10, 11, 12, MASTER 4,1 ↵
ATTACH DAQ 7, B MUX 8, 13, 14, 15, 16, MASTER 4,1 ↵
```

This sequence connects 16 ADC channels to the 4 available ports of 2 DAQ cards via 4 multiplexers. ADC channels are connected in sequence, for convenience, but this is not strictly necessary.

Example 4:

```
ATTACH DAQ 4, A MUX 3, 1, 2, 3, 0 ↵
ATTACH DAQ 4, B MUX 5, 4, 5, 6, 0 ↵
```

This sequence connects 6 ADC channels to a single DAQ cards via 2 multiplexers. The “0” arguments for the 4th port on each multiplexer indicate that these ports are not currently being used.

See Also: **ATTACH TIMING
SLAVE**

ATTACH TIMING

Syntax:

```
ATTACH TIMING w AIM x STR A [CDS B] [AIM y STR C] [AIM z STR D] ↵
```

w slot number of the Timing and Control Card.

x,y,z AIM number.

A,B,C,D Timing and Control Card output block.

Specifies timing connections between a Timing/Control Card and the Strobe and (optional) CDS inputs of 1-4 AIMS. Each AIM must have its Convert Strobe (STR) attached to a Timing Card Output. AIMS's with the optional CDS feature must also have their CDS Strobes (CDS) attached to a Timing Card output. Any output blocks not specified are assumed to be unused.

Example 1:

```
ATTACH TIMING 3 AIM 1 STR A ↵
```

This is the most basic example, attaching the first AIM in the system to its Timing card in slot 3, without the CDS option. AIM 1, by definition, is the first AIM daisy-chained onto the first Timing Card in the system. Note that this attachment also defines the ADC channel numbers, since by definition

ADC Channel 1 is the first ADC in AIM 1. Please see **Section 3.1. Channel Assignments** for more details.

Example 2:

```
ATTACH TIMING 3 AIM 1 STR A CDS B AIM 2 STR C CDS D
```

This string attaches the Convert Strobe input of AIM 1 to Timing Card 3's output block "A", and connects the CDS input of AIM 1 to output block "B". Output blocks "C" and "D" are connected to the Convert and CDS inputs of AIM 2, respectively.

```
ATTACH TIMING 3 AIM 5 STR A AIM 6 STR B AIM 7 STR C AIM 8 STR D
```

This string attaches Timing Card 3's output blocks A-D to the Convert Strobes of AIMS 5-8, respectively.

See Also: **ATTACH DAQ**

AVERAGE

Syntax:

```
AVERAGE TRUE|FALSE|SUBSET [#1, MAX|#2]
```

#1 First frame to be averaged

#2 Number of frames to average

Specifies whether to reduce the acquired dataset to a single frame of averaged data. If **AVERAGE** is set to **FALSE**, data will not be averaged and the dataset will have a number of frames, as specified by the last **FRAMECOUNT** command.

If **AVERAGE** is set to **TRUE** or **SUBSET** a single, averaged frame will be returned. If **AVERAGE** is set to **TRUE**, all acquired frames will be averaged together. If **AVERAGE** is set to **SUBSET**, then the additional arguments are required to specify the first frame to be averaged and the number of frames to be averaged.

Example 1:

```
AVERAGE TRUE
```

Example 2:

```
AVERAGE SUBSET 5,100
```

If the dataset is acquired as integer data (**FLOAT FALSE**), the average will be calculated using integer math.

BIASCLKMSGGS

Syntax:

```
BIASCLKMSGGS
```

The message mode commands (**BIASCLKMSGGS**, **DACQMSGGS** and **PATMSGGS**) are for GPIB communications only. Users linking directly into Pulse Instruments DLLs can ignore these commands.

Multiple types of Pulse Instruments card can be installed in the same CompactPCI mainframe, and certain commands, such as **STATUS**, are recognized by more than one type of card. Therefore, the PI-11000 uses "message modes" to determine which commands are intended for which type of card.

A BIASCLKMSGS must be sent before the initial block of commands intended for DC Bias or clock driver cards..

The BIASCLKMSGS command must be sent again if any commands have been subsequently sent to a different type of Pulse Instruments CompactPCI cards (such as Pattern Generator cards), using the DACQMSGS command.

Example:

```
BIASCLKMSGS ; THESE COMMANDS ARE FOR BIAS/CLK CARDS ↵
ALL LOLIMB -3.00 ↵
ALL UPLIMB 5.00 ↵
MF 0 SLOT 3 CH 2 VOLT -1.80 ↵
MF 0 SLOT 3 CH 1 VOLT 2.3 ↵
CONNECT ↵
ONLINE ↵
```

```
DACQMSGS ; THESE ARE FOR DATA ACQUISITION CARDS
BLOCKING FALSE ↵
STARTACQ ↵
```

```
BIASCLKMSGS ; THERE ARE FOR BIAS/CLK CARDS ↵
OFFLINE ↵
```

See Also: **DACQMSGS**
PATMSGS

BITSPERPIX

Syntax:

```
BITSPERPIX bits ↵
```

bits Bit-width of data to collect (9-16).

Sets the bit-width for collected data. Data are right-aligned, and masked bits are set to zero. Use BITSPERPIX in conjunction with GET ADC ID to ensure that your data are scaled correctly (e.g. when converting to floating point). Example:

```
DAQ CARD 2 ↵
BITSPERPIX 14 ↵
```

See Also: **GET ADC ID**
FLOAT

BLOCKING

Syntax:

```
BLOCKING bool ↵
```

bool TRUE or FALSE

Sets the Blocking mode for Data Acquisition to True or False. When Blocking is enabled (True), the system/DLL will refuse any communications once a data acquisition is started until all armed cards have completed acquisition or until the acquisition timeout is reached.

If Blocking is disabled (also), the system/DLL will respond to user input while an acquisition is pending. When Blocking is set to False, use ACQSTATUS to determine the status of an acquisition or STOPACQ to stop a pending acquisition. Example:

```
BLOCKING TRUE ↵
```

See Also: **STARTACQ**
ACQSTATUS
STOPACQ
ACQTIME

CDS

Syntax:

```
CDS bool ↵
```

bool TRUE or FALSE

Sets the CDS mode for the selected ADC channel to True or False. CDS requires the optional CDS features to be installed and requires a CDS Strobe connection (see ATTACH TIMING command). When CDS is enabled (True), the video will be digitized twice; once during the Convert Strobe and once during the CDS strobe, and the difference will be reported. When CDS is disabled (False), the video will be sampled only during the Convert Strobe.

Example:

```
CHAN 2 ↵  
CDS TRUE ↵
```

See Also: **CDS STROBE**
CONVERT STROBE
GET CDS
GET CDS STROBE

CDS STROBE

Syntax:

```
CDS STROBE ns ↵
```

ns CDS strobe delay, in ns

Sets the CDS Strobe delay for the selected ADC channel. CDS STROBE requires the optional CDS features to be installed and requires a CDS Strobe connection (see ATTACH TIMING command). When CDS is enabled (True), the video will be digitized twice; once during the Convert Strobe and once during the CDS strobe, and the difference will be reported. When CDS is disabled (False), the video will be sampled only during the Convert Strobe.

The CDS Strobe must be less than or equal to a value that is a function of the Pixel Period. Use GET TIMING MAXIMA to determine the current maximum programmable value. If a value is programmed greater than the maximum value, the maximum value will be used.

Example:

```
CHAN 2 ↵  
CDS STROBE 148.7 ↵
```

See Also: **CONVERT STROBE**

**GET CDS
GET CDS STROBE
GET TIMING MAXIMA**

CHAN

Syntax:

CHAN x

x ADC channel number

CHAN is used to select an individual ADC channel for programming. See Section x.xx for a list of commands that may follow CHAN. The channel selected will be maintained until another CHAN command is issued, so multiple valid commands may follow a single CHAN command. Use GET CHAN to determine which channel is currently selected for programming. Example:

```
CHAN 5
CONVERT STROBE 248.7
CDS TRUE
CDS STROBE 100.2
```

Note that ADC channel numbering has no meaning until the proper ATTACH commands have been issued.

See Also: @

**ALLCHAN
ATTACH DAQ
ATTACH TIMING
DAQ CARD
GET CHAN**

CLOCK SRC

Syntax:

CLOCK SRC DATA | SMA | SHARE

CLOCK SRC is used to set the clock input source for the selected Port on the selected Data Acquisition card. DATA specifies that the Pixel, Line and Frame clocks are expected on the Data cable with LVDS voltages. SMA specifies that clocks are expected on the SMA connectors with TTL voltages. SHARE is valid only for a B port on a card, and specifies that Port B will use clocks from Port A, thus reducing the number of timing signals required.

Example:

```
DAQ CARD 5
PORT A
CLOCK SRC SMA
PORT B
CLOCK SRC SHARE
```

See Also: **GET CLOCK SRC**

CONVERT STROBE

Syntax:

CONVERT STROBE *ns*

ns CDS strobe delay, in ns

Sets the Convert Strobe delay for the selected ADC channel. When CDS is enabled (True), the video will be digitized twice; once during the Convert Strobe and once during the CDS strobe, and the difference will be reported. When CDS is disabled (False), the video will be sampled only during the Convert Strobe.

The Convert Strobe must be less than or equal to a value that is a function of the Pixel Period. Use GET TIMING MAXIMA to determine the current maximum programmable value. If a value is programmed greater than the maximum value, the maximum value will be used.

Example:

```
CHAN 2  
CONVERT STROBE 248.7
```

See Also: **CONVERT STROBE**
GET CONVERT STROBE

DACQMSGGS

Syntax:

DACQMSGGS

The message mode commands (BIASCLKMSGGS, DACQMSGGS and PATMSGGS) are for GPIB communications only. Users linking directly into Pulse Instruments DLLs can ignore these commands.

Multiple types of Pulse Instruments card can be installed in the same CompactPCI mainframe, and certain commands, such as STATUS, are recognized by more than one type of card. Therefore, the PI-11000 uses "message modes" to determine which commands are intended for which type of card.

A DACQMSGGS must be sent before the initial block of commands intended for the data acquisition subsystem.

The DACQMSGGS command must be sent again if any commands have been subsequently sent to a different type of Pulse Instruments CompactPCI cards (such as Pattern Generator cards), using the PATMSGGS command.

Example:

```
PATMSGGS ; THESE COMMANDS ARE FOR PATGEN CARDS  
UPDATE  
COMPILE 1
```

```
DACQMSGGS ; THESE ARE FOR DATA ACQUISITION CARDS  
BLOCKING FALSE  
STARTACQ
```

```
PATMSGGS  
RUN
```

```
DACQMSGGS  
ACQSTATUS
```

If linked from a local application dacq.dll will treat DACQMSGGS as a no-op, to allow for code re-use among applications that may run locally or remotely.

See Also: **BIASCLKMSGGS**
PATMSGGS

DAQ CARD

Syntax:

DAQ CARD *x* ↵

x Data Acquisition card slot

DAQ CARD is used to select an individual Data Acquisition card for programming. See Section x.xx for a list of commands that may follow DAQ CARD. The card selected will be maintained until another DAQ CARD command is issued, so multiple valid commands may follow a single DAQ CARD command. Use GET DAQ CARD to determine which card is currently selected for programming.

Example:

```
DAQ CARD 5 ↵  
OUTPUT P256 L512 ↵  
VERTICAL FALSE ↵  
FLOAT FALSE ↵
```

See Also: **ALLCARD**
ALLCHAN
CHAN
GET DAQ CARD
PORT

FILTERS

Syntax:

FILTERS *filterstring* ↵

filterstring Filter setting (OPEN, 10 KHz, 100 KHz, 1 MHz, 10 MHz, or CLOSED)

Sets all filters in the system to the selected bandwidth or OPEN. A change in the Filters setting may take *x* milliseconds to settle. If your application does not have control over the timing of commands, use the Pause command to ensure that adequate settling time takes place before beginning your data acquisition cycle.

Example:

```
FILTERS 1MHZ ↵
```

See Also: **GET FILTERS**

FLOAT

Syntax:

FLOAT *bool* ↵

bool TRUE or FALSE, to select Floating Point or Integer mode, respectively

Sets the Floating Point mode for collected data. When set to TRUE, acquired data are converted to floating point data, based on a full-scale ADC range of +/- 2.0 V and the current BITS PER PIX setting. If SETUP MODE is TRUE, the voltage at the ADC will be reported. If SETUP MODE is false, the voltage at the ADC will be gain- and offset-corrected to reflect the voltage at the input of the preamplifier.

When set to FALSE, 16-bit integer data will be collected, with unused high bits masked to 0.

Example:

```
DAQ CARD 2 ↵  
FLOAT TRUE ↵
```

See Also: **BITSPERPIX**
GET FLOAT

FRAME COUNT

Syntax:

```
FRAME COUNT n ↵
```

n The number of frames to collect, 1 [*n* [8,192

Sets the number of frames to collect on the selected Data Acquisition card when STARTACQ is issued.

```
DAQ CARD 2 ↵  
FRAMECOUNT 100 ↵
```

If a Data Acquisition card is configured as a Slave via SLAVE TRUE, then it must be set to the same FRAME COUNT value as its Master. Example:

```
DAQ CARD 2 ↵  
FRAMECOUNT 100 ↵  
DAQ CARD 5 ↵  
SLAVE TRUE ↵  
FRAMECOUNT 100 ↵
```

See Also: **GET FRAMECOUNT**

FRAME SYNC/LINE SYNC

Syntax:

```
FRAME SYNC ns1 LINE SYNC ns2 ↵
```

ns1 The desired Frame Sync delay, in nanoseconds

ns2 The desired Line Sync delay, in nanoseconds

Sets the Frame Sync and Line Sync delays for the AIM containing the selected channel. Example:

```
CHAN 1 ↵  
FRAME SYNC 30 LINE SYNC 35 ↵
```

The Frame Sync and Line Sync values must be less than a value that is a function of the currently installed delay modules. Use GET TIMING MAXIMA to determine the current maximum programmable values. If a value is programmed greater than the maximum value, the maximum value will be used.

See Also: **GET SYNC**
GET TIMING MAXIMA

GAIN

Syntax:

GAIN x

x Gain value, 1 [x [420, selected values

Sets the Gain value for the selected ADC channel. Not all values between 1 and 420 are available. See Section x.xx for a table of selectable gain values. If an invalid gain value is specified, the system/DLL will return an error with the closest available values.

Example:

```
CHAN 2  
GAIN 8
```

See Also: **GET GAIN**

GET ACQTIME

Syntax:

GET ACQTIME

Returns the acquisition timeout period. The returned value will be a string in the same syntax used by the ACQTIME command. Example:

```
GET ACQTIME  
ACQTIME 60
```

See Also: **ACQTIME**

GET ACTIVE CHAN

Syntax:

GET ACTIVE CHAN

Returns which ADC channels in the system are active. The returned value will be a string in the same syntax used by the SETACTIVECHAN command. Example:

```
GET ACTIVE CHAN  
SETACTIVECHAN 1,2,3,4
```

See Also: **SET ACTIVE CHAN**

GET ADC IDS

Syntax:

GET ADC IDS

Returns two 3-bit codes for the Gain/Filter and ADC modules installed in the selected ADC channel or channels. Example:

```
ALLCHAN
GET ADC IDS

CHAN 1: G/F ID: 7 ADC ID: 7
CHAN 2: G/F ID: 7 ADC ID: 7
```

See Also: **GET PREAMP ID**

GET AOI

Syntax:

```
GET AOI
```

Returns the currently defined Area-of-Interest. Example:

```
DAQ CARD 5
GET AOI

AOI 0,0,256,256
```

See Also: **AOI**
GET AOI
GET FPA

GET ATTACHMENTS

Syntax:

```
GET ATTACHMENTS
```

Returns all timing and data attachments. The returned string uses the same syntax as the ATTACH DAQ and ATTACH TIMING commands. Example:

```
GET ATTACHMENTS

ATTACH TIMING 3 AIM 1 STR A
ATTACH DAQ 4, A CHAN 1
ATTACH DAQ 4, B CHAN 2
ATTACH DAQ 5, A CHAN 3 MASTER 4,1
ATTACH DAQ 5, B CHAN 4 MASTER 4,1
```

See Also: **ATTACH DAQ**
ATTACH TIMING

GET AVERAGE

Syntax:

```
GET AVERAGE
```

Returns the AVERAGE setting for the selected DACQ card. The returned string uses the same syntax as the AVERAGE command. Example:

```
DACQ CARD 4
```

GET ATTACHMENTS ↵

↵ AVERAGE SUBSET 5,100 ↵

See Also: **ATTACH DAQ**
ATTACH TIMING

GET BITSPERPIX

Syntax:

GET BITSPERPIX ↵

Returns the BITSPERPIX setting for the selected data acquisition card. The returned value will be a string in the same syntax used by the BITSPERPIX command. Example:

DAQ CARD 5 ↵

GET BITSPERPIX ↵

↵ BITSPERPIX 14 ↵

See Also: **BITSPERPIX**

GET BLOCKING

Syntax:

GET BLOCKING ↵

Returns the BLOCKING TRUE/FALSE. The returned value will be a string in the same syntax used by the BLOCKING command. Example:

GET BLOCKING ↵

↵ BLOCKING TRUE ↵

See Also: **BLOCKING**

GET CDS

Syntax:

GET CDS ↵

Returns the CDS mode for the selected ADC channel. The returned value will be a string in the same syntax used by the CDS command. Example:

CHAN 5 ↵

GET CDS ↵

↵ CDS TRUE ↵

See Also: **CDS**

GET CDS STROBE

Syntax:

GET CDS STROBE ↵

Returns the CDS STROBE setting for the selected ADC channel. The returned value will be a string in the same syntax used by the CDS STROBE command. Example:

```
CHAN 5 ↵  
GET CDS STROBE ↵  
↵CDS STROBE 150 ↵
```

See Also: **CDS STROBE**

GET CHAN

Syntax:

```
GET CHAN ↵
```

Returns the number of the currently selected ADC channel. Example:

```
GET CHAN ↵  
↵CHAN 1 ACTIVE ↵
```

See Also: **CHAN**

GET CLOCK SRC

Syntax:

```
GET CLOCK SRC ↵
```

Returns the number of the currently selected port on the selected Data Acquisition card. Example:

```
DAQ CARD 5 ↵  
PORT B ↵  
GET CLOCK SRC ↵  
↵CLOCK SRC SHARE ↵
```

See Also: **CLOCK SRC**

GET CONVERT STROBE

Syntax:

```
GET CONVERT STROBE ↵
```

Returns the CONVERT STROBE setting for the selected ADC channel. The returned value will be a string in the same syntax used by the CONVERT STROBE command. Example:

```
CHAN 5 ↵  
GET CONVERT STROBE ↵  
↵CONVERT STROBE 85 ↵
```

See Also: **CONVERT STROBE**

GET DAQ CARDS

Syntax:

```
GET DAQ CARDS ↵
```

Returns the CompactPCI slot locations of all installed Data Acquisition cards in the system. See Section x.xx for information on slot numbering.

Example:

```
GET DAQ CARDS ↵  
↵Acquisition Card Slot 5
```

See Also: **GET MUX CARDS**
GET TIMING CARDS

GET FILTERS

Syntax:

```
GET FILTERS ↵
```

Returns the current FILTERS setting. The returned value will be a string in the same syntax used by the CONVERT STROBE command. Example:

```
GET FILTERS ↵  
↵FILTERS 1 MHZ ↵
```

See Also: **FILTERS**

GET FLOAT

Syntax:

```
GET FLOAT ↵
```

Returns the Float setting (TRUE or FALSE) of the currently selected Data Acquisition card. The returned value will be a string in the same syntax used by the FLOAT command. Example:

```
DAQ CARD 5 ↵  
GET FLOAT ↵  
↵FLOAT FALSE ↵
```

See Also: **FLOAT**

GET FPA

Syntax:

```
GET FPA ↵
```

Returns the total FPA size for the selected Data Acquisition card, including any enabled Slave cards. If the current FPA setup is invalid, GET FPA will return an error. Example:

```
DAQ CARD 5 ↵  
GET FPA ↵  
↵X256 Y256 ↵  
DAQ CARD 6 ↵  
GET FPA ↵  
↵Error: Overlapping Channels ↵
```

See Also: **OUTPUT POSITION**

GET FRAMECOUNT

Syntax:

GET FRAMECOUNT ↵

Returns the FRAMECOUNT for the selected Data Acquisition card. The returned value will be a string in the same syntax used by the FRAMECOUNT command. Example:

```
DAQ CARD 5 ↵  
GET FRAMECOUNT ↵  
↵FRAMECOUNT 100 ↵
```

See Also: **FRAMECOUNT**

GET GAIN

Syntax:

GET GAIN ↵

Returns the Gain value for the selected ADC channel. The returned value will be a string in the same syntax used by the GAIN command. Example:

```
CHAN 5 ↵  
GET GAIN ↵  
↵GAIN 16 ↵
```

See Also: **GAIN**

GET INVERT

Syntax:

GET INVERT PIXEL | LINE | FRAME ↵

Returns the clock inversion settings for the selected port on the selected Data Acquisition card. The returned value will be a string in the same syntax used by the INVERT/NINVERT command. Example:

```
DAQ CARD 5 ↵  
PORT B ↵  
GET INVERT PIXEL ↵  
↵NINVERT PIXEL ↵  
GET INVERT LINE ↵  
↵NINVERT LINE ↵  
GET INVERT FRAME ↵  
↵INVERT FRAME ↵
```

See Also: **INVERT/NINVERT**

GET MONITOR

Syntax:

GET MONITOR ↵

Returns the video, convert, and CDS monitor settings for the AIM containing the selected channel. The returned value will be a string in the same syntax used by the VIDEO/CONVERT/CDS command.

Example:

```
GET MONITOR ↵  
↵AIM 1: VIDEO 1 CONVERT 1 CDS 1 ↵
```

See Also: **VIDEO**

GET MUX

Syntax:

GET MUX x ↵

x CompactPCI slot number

Returns the Multiplexing or Switch setting for the Mux card in the specified slot. The returned value will be a string in the same syntax used by the MUX command. Example:

```
GET MUX 5 ↵  
↵MUX 5 MULTIPLEX 4 FORCE 4 ↵
```

See Also: **MUX**

GET MUX CARDS

Syntax:

GET MUX CARDS ↵

Returns the CompactPCI slot locations of all installed Multiplexer cards in the system. See Section x.xx for information on slot numbering.

Example:

```
GET MUX CARDS ↵  
↵Multiplexer Card Slot 4 ↵  
↵Multiplexer Card Slot 6 ↵
```

See Also: **GET DAQ CARDS**
GET TIMING CARDS

GET NAMES

Syntax:

GET NAMES ↵

Returns a listing of all attached ADC channels in the system and their defined names. The returned values will be strings in the same syntax used by the NAMEAD command

Example:

```
GET NAMES ↵  
↵NAMEAD CHAN 1 ↵
```

See Also: **@**

NAME AD

GET OFFSET

Syntax:

GET OFFSET

Returns the programmed Offset voltage for the selected ADC channel. The returned value will be a string in the same syntax used by the OFFSET command

Example:

```
CHAN 2
GET OFFSET
      OFFSET 2.000
```

See Also: **OFFSET**

GET OUTPUT

Syntax:

GET OUTPUT

Returns the output size for the selected Data Acquisition card. The returned value will be a string in the same syntax used by the OUTPUT command.

Example:

```
DAQ CARD 5
GET OUTPUT
      OUTPUT X256 Y512
```

See Also: **OUTPUT**

GET PORT

Syntax:

GET PORT

Returns the letter of the currently selected port for Data Acquisition card programming. Example:

```
GET PORT
      PORT A
```

See Also: **PORT**

GET PIXEL PERIOD

Syntax:

GET PIXEL PERIOD y

y CompactPCI slot number

Returns the Pixel Period for the Timing/Control card in the specified slot. The returned value will be a string in the same syntax used by the SET PIXEL PERIOD command

Example:

```
GET PIXEL PERIOD 3 ↵  
↳SET PIXEL PERIOD 400, 3 ↵
```

See Also: **SET PIXEL PERIOD**

GET POSITION

Syntax:

```
GET POSITION ↵
```

Returns the plot position for the selected ADC channel. The returned value will be a string in the same syntax used by the POSITION command

Example:

```
CHAN 1 ↵  
GET POSITION ↵  
↳POSITION 0,0,1,1 ↵
```

See Also: **POSITION**

GET PREAMP ID

Syntax:

```
GET PREAMP ID ↵
```

Returns a 2-bit codes for the preamplifier module installed in the selected ADC channel or channels.

Example:

```
CHAN 2 ↵  
GET PREAMP ID ↵  
↳CHAN 2: PREAMP ID: 3 ↵
```

See Also: **GET ADC IDS**

GET SAVETOFILE

Syntax:

```
GET SAVETOFILE ↵
```

Returns the Save to File path and filename, and options for the selected Data Acquisition card. The returned value will be a string in the same syntax used by the SAVETOFILE command

Example:

```
DAQ CARD 5 ↵  
GET SAVETOFILE ↵  
↳SAVETOFILE C:\DATA1.TXT, ASCII ↵
```

See Also: **SAVETOFILE**

GET SETUP MODE

Syntax:

GET SETUP MODE

Returns the SETUP MODE setting (True or False) for the selected Data Acquisition card. The returned value will be a string in the same syntax used by the SETUP MODE command

Example:

```
DAQ CARD 5  
GET SETUP MODE  
SETUP MODE FALSE
```

See Also: **SETUP MODE**

GET SLAVE

Syntax:

GET SLAVE

Returns the SLAVE setting (True or False) for the selected Data Acquisition card. The returned value will be a string in the same syntax used by the SLAVE command

Example:

```
DAQ CARD 5  
GET SLAVE  
SLAVE FALSE
```

See Also: **SLAVE**

GET STITCHING

Syntax:

GET STITCHING

Returns the STITCHING setting (True or False) for the selected Data Acquisition card. The returned value will be a string in the same syntax used by the STITCHING command

Example:

```
DAQ CARD 5  
GET STITCHING  
STITCHING TRUE
```

See Also: **STITCHING**

GET SYNC

Syntax:

GET SYNC

Returns the Frame Sync and Line Sync delays for the AIM containing the selected ADC channel. The returned value will be a string in the same syntax used by the FRAME SYNC/LINE SYNC command

Example:

```
CHAN 3 ↵  
GET SYNC ↵  
↵FRAME SYNC 108.00 LINE SYNC 53.00↵
```

See Also: **FRAME SYNC/LINE SYNC**

GET TIMING CARDS

Syntax:

```
GET TIMING CARDS ↵
```

Returns the CompactPCI slot locations of all installed Timing/Control cards in the system. See Section x.xx for information on slot numbering.

Example:

```
GET TIMING CARDS ↵  
↵Timing Card Slot 3↵
```

See Also: **GET DAQ CARDS**
GET MUX CARDS

GET TIMING MAXIMA

Syntax:

```
GET TIMING MAXIMA ↵
```

Returns the maximum allowable delays for the Convert Strobe, CDS Strobe, Frame Sync and Line Sync. The Convert and CDS strobe maximums are based on the modules installed in the system and the currently programmed Pixel Period. The Frame and Line Sync maximums are based on the installed modules.

Example:

```
GET TIMING MAXIMA ↵  
↵AIM 1: Convert 1025 ns CDS 1025 ns Sync 1025 ns↵
```

If the optional CDS feature is not installed, the “CDS” section of the string will not be returned.

See Also: **CDS STROBE**
CONVERT STROBE
FRAME SYNC/LINE SYNC
PIXEL PERIOD

GET TIMING STEPS

Syntax:

```
GET TIMING STEPS x ↵
```

x CompactPCI slot number

Returns the coarse programmable increment, in nanoseconds, for the Convert/CDS Strobe and Frame/Line Syncs for the specified Timing/Control card.

Example:

GET TIMING STEPS 3 ↵

↳TIMING STEPS 3: 4.0 N/A N/A N/A 4.0↵

Five values will be returned, in nanoseconds. The first 4 values are for the 4 timing blocks (A-D, respectively) of the Timing/Control card. If any of the timing blocks are not populated they will return "N/A." The fifth value is for the Frame and Line Sync.

See Also: **GET TIMING MAXIMA**
PIXEL PERIOD

GET VERTICAL

Syntax:

GET VERTICAL ↵

Returns the plot direction (Vertical True or False) for the selected Data Acquisition card. The returned value will be a string in the same syntax used by the VERTICAL command

Example:

DAQ CARD 5 ↵

GET VERTICAL ↵

↳VERTICAL FALSE↵

See Also: **VERTICAL**

ID

Syntax:

ID ↵

Returns the model numbers and CompactPCI slot locations for all Data Acquisition, Multiplexer, and Timing/Control components in the system. Example:

ID ↵

```
↳ SLOT 3 PI-41100 ADC Timing and Control
Module steps: 4.00 4.00 0.00 0.00 F/L 4.00
AIM #1: GF 41010, 41010, 41010, 41010:
AD 41040, 41040, 41040, 41040:
PA 3150, 3150, 3150, 3150
SLOT 4 PI-41000 Digital Acquisition
SLOT 5 PI-41110 4:1 Digital Multiplexer ↵
```

See Also: **GET DAQ CARDS**
GET MUX CARDS
GET TIMING CARDS

INVERT/NINVERT

Syntax:

INVERT PIXEL | LINE | FRAME ↵

NINVERT PIXEL | LINE | FRAME ↵

Sets the selected port on the selected Data Acquisition card to expect inverted or non-inverted clocks on the Pixel, Line and Frame clock inputs, respectively. INVERT and NINVERT can take one, two, or three arguments. By defaults, all clock inputs are non-inverted, and are triggered on rising edges.

Example:

```
DAQ CARD 2 ↵  
PORT B ↵  
NINVERT PIXEL LINE ↵  
INVERT FRAME ↵
```

See Also: **GET INVERT**

MUX MULTIPLEX MUX SWITCH

Syntax:

MUX slot MULTIPLEX port [FORCE port1, port2, port3] ↵

MUX slot SWITCH port ↵

slot slot number of the Mux Card

port Mux port number to begin multiplexing from or switch to

portn Mux port(s) to force output from

Sets the multiplexing or switching mode for the mux card in the specified slot. There are two syntaxes for MUX.

Syntax 1:

MUX MULTIPLEX causes the multiplexer to output data for multiple ports, in descending round-robin order. Example:

```
MUX 4 MULTIPLEX 3 ↵
```

This will cause the Mux card in Slot 4 to output data from input ports 3, 2, and 1, in that order.

The optional FORCE command causes the Mux to insert dummy data from the specified ports. FORCE is used to balance inputs to a Data Acquisition card. For example, in an application with 5 ADC channels muxed through two Mux cards into Ports A and B of a Data Acquisition card, the multiplexer with only two “real” data channels would need to force output from one additional port in order to balance the data rates in the Data Acquisition card. If the “real” data are input into Ports 1 and 2 of the Mux, Port 3 can be forced on using:

```
MUX 4 MULTIPLEX 3 FORCE 3 ↵
```

There must always be at least one port with “real” data and clocking; therefore FORCE can accept *n-1* arguments when the MUX mode is set to MULTIPLEXER *n*.

Syntax 2:

In Switch mode, the multiplexer acts as a simple 4:1 switch. The input is switched to the specified port number. Example:

```
MUX 4 SWITCH 4 ↵
```

See Also: **GET MUX CARDS**

NAME AD

Syntax:

```
NAMEAD CHAN chan @name
```

chan ADC channel number

name Desired name

Defines a name for an ADC channel. Once a name has been defined for an ADC channel, the @ syntax may be used to select the channel for programming.

Example:

```
NAMEAD CHAN 1 @ALPHA  
@ALPHA  
CONVERT STROBE 148.70
```

This sequence of commands will set the Convert Strobe for ADC Channel 1 to 148.70 ns..

See Also: **@**
ALLCHAN
CHAN

OFFSET

Syntax:

```
OFFSET volts
```

volts offset voltage

Sets the programmable offset voltage for the selected channel. A change in the Offset voltage may take 10-50 milliseconds to settle. A change in Offset voltage from outside the ADC saturation range (+/- 2 V) may take up to 250 milliseconds to settle. If your application does not have control over the timing of commands, use the Pause command to ensure that adequate settling time takes place before beginning your data acquisition cycle.

Example:

```
CHAN 1  
OFFSET -2.687
```

See Also: **FLOAT**
GAIN
SETUP MODE

OUTPUT

Syntax:

```
OUTPUT Xhoriz Yvert
```

horiz number of horizontal pixels per frame per ADC channel

vert number of vertical pixels per frame per ADC channel

Sets the common Output Size for each ADC channels attached to the selected Data Acquisition card, including any active Slave cards. X and Y dimensions are 1-based, and independent of the clocking orientation. Use Vertical and Position to determine the plot location and orientation of each ADC channel's output, and use GET FPA to confirm that the total defined FPA size is correct. See Section 3.7. DEFINE mnemonic for a detailed description of FPA definition and setup.

Example:

```
DAQ CARD 5 ↵  
OUTPUT X256 Y512 ↵
```

See Also: **GET FPA**
POSITION
STITCHING
VERTICAL

PATMSGGS

Syntax:

```
DACQMSGGS ↵
```

The message mode commands (BIASCLKMSGGS, DACQMSGGS and PATMSGGS) are for GPIB communications only. Users linking directly into Pulse Instruments DLLs can ignore these commands.

Multiple types of Pulse Instruments card can be installed in the same CompactPCI mainframe, and certain commands, such as STATUS, are recognized by more than one type of card. Therefore, the PI-11000 uses "message modes" to determine which commands are intended for which type of card.

A PATMSGGS must be sent before the initial block of commands intended for the pattern generation subsystem.

The PATMSGGS command must be sent again if any commands have been subsequently sent to a different type of Pulse Instruments CompactPCI cards (such as Data Acquisition cards), using the DACQMSGGS command.

Example:

```
PATMSGGS ; THESE COMMANDS ARE FOR PATGEN CARDS ↵  
UPDATE ↵  
COMPILE 1 ↵  
  
DACQMSGGS ; THESE ARE FOR DATA ACQUISITION CARDS  
BLOCKING FALSE ↵  
STARTACQ ↵  
  
PATMSGGS ↵  
RUN ↵  
  
DACQMSGGS  
ACQSTATUS ↵
```

See Also: **BIASCLKMSGGS**
DACQMSGGS

PAUSE

Syntax:

PAUSE x

x Pause time, in microseconds

When Pause is received, the system/DLL will block for x usec before processing the next command. Use Pause to ensure that sufficient settling time occurs between programming of global offsets and filter commands and the beginning of your data acquisition cycle. Accuracy of the Pause command is not guaranteed. Example:

```
ALLCHAN  
OFFSET 4.035  
FILTERS 100 KHz  
PAUSE 25000  
STARTACQ
```

PIXEL PERIOD

Syntax:

PIXEL PERIOD x, y

x pixel period, in nanoseconds

y CompactPCI slot number of the Timing/Control card

Sets the Pixel Period for the specified Timing/Control card. The Pixel Period setting does not control hardware; rather it is used to calculate constraints for the Convert Strobe and CDS Strobe commands. The pixel period value is also written to the header of acquired data sets to enable time-domain analysis.

See Also: **GET PIXEL PERIOD**

PORT

Syntax:

PORT x

x Port on Data Acquisition card

PORT is used to select an individual Data Acquisition card port for programming. See Section x.xx for a list of commands that may follow PORT. The card selected will be maintained until another PORT command is issued, so multiple valid commands may follow a single PORT command. Use GET PORT to determine which card is currently selected for programming. Example:

```
DAQ CARD 5  
PORT B  
NINVERT LINE PIXEL  
INVERT FRAME
```

See Also: **CHAN
DAQ CARD**

POSITION

Syntax:

POSITION *x, y, dx, dy*

- x*** horizontal address of the first pixel on each frame
y vertical address of the first pixel on each frame
dx horizontal increment between two corresponding pixels from this ADC channel
dy vertical increment between two corresponding pixels from this ADC channel

Sets the plot origin and increment for pixels from the selected ADC channel. X and Y addresses are 0-based, and independent of the clocking orientation. Use Vertical to specify whether X and Y correspond to Pixel and Line clocks or Line and Pixel clocks, respectively, and use GET FPA to confirm that the total defined FPA size is correct. See **Section 3.7. DEFINE mnemonic** for a detailed description of FPA definition and setup.

Example:

```
CHAN 1  
POSITION 0,0,1,1  
CHAN 2  
POSITION 256,0,1,1
```

See Also: **GET FPA**
OUTPUT
STITCHING
VERTICAL

RESET

Syntax:

RESET [*x*]

- x*** CompactPCI slot number

Resets a Timing and Control card in the specified slot. If the slot number is omitted, all installed Timing and Control cards are reset. Parameters that are set to defaults on Reset:

- Gain
- Offset
- Filter
- Convert Strobe
- CDS Strobe
- Frame Sync
- Line Sync
- Video, Convert, and CDS Monitor Selection

RESET should be sent to all Timing/Control cards after power-up to ensure that all settings are in a known state.

RESET may also be used if the clock rate into the Timing and Control card is changed such that the Convert and/or CDS strobe delays are out of bounds and become unresponsive. Programmers should set Convert and CDS strobes to 0 ns and program a new Pixel Period before changing the clock rate.

Example:

```
RESET ↵
```

See Also: **CDS STROBE**
CONVERT STROBE
GET PIXEL PERIOD
GET TIMING MAXIMA
PIXEL PERIOD

SAVEFILE

Syntax:

```
SAVEFILE ↵
```

Used only when a 3rd-party application running on the Local (CompactPCI) CPU has linked into dacq.dll and has called SetCallback() with any value other than NULL. Causes the DLL to save acquired data to the file specified by SAVETOFILE. Data will be saved for every Master or Independent card with active channels and a defined save-to filename.

If SetCallback() has been set to NULL or has never been called, data will be saved to the specified filename after every completed data acquisition.

Example:

```
DAQ CARD 5 ↵  
SAVEFILE
```

See Also: **GET SAVETOFILE**
SAVETOFILE

SAVETOFILE

Syntax:

```
SAVETOFILE fname [, ASCII] [, AUTO][, OVERWRITE] ↵
```

fname path and filename to which to save the file

Specifies the path-name and filename to which to save acquired data from the selected Data Acquisition card after each acquisition sequence or when SAVEFILE is called. SAVEFILE will create a file if the specified file does not exist, but SAVEFILE will not create directories that do not exist. If SAVEFILE is called with a non-existent path, an error will be returned.

fname should not contain spaces, as all spaces are stripped from command strings during parsing.

If ASCII is entered on the same line, the output will be saved in ASCII text, otherwise the data will be saved as binary data.

If AUTO is entered on the same line, dacq.dll will attempt to save the acquired data to the specified path and filename each time STARTACQ or ACQMEASURE is sent. Otherwise, data will not be saved unless a SAVEFILE command is sent.

If OVERWRITE is entered on the same line, dacq.dll will overwrite an existing file with new data when saving. If OVERWRITE is omitted, dacq.dll will not overwrite an existing file.

fname may contain # characters to enable automatic file numbering. If ***fname*** contains # characters, those characters will be replaced by incrementing numbers each time a file is saved. The number of # characters determines how many digits the file number will have. If the # sign(s) are preceded by a

number, the file index will begin with that number; otherwise it will start at zero. When the maximum file number has been reached (e.g. 999 for ###) the file index will wrap around to zero.

For example, the filename argument "C:\Data\TheFile5###.dta" will result in the following files being created on successive saves:

- C:\Data\TheFile005.dta
- C:\Data\TheFile006.dta
- C:\Data\TheFile007.dta
- etc.

until C:\Data\TheFile999.dta is reached, which would then be followed by C:\Data\TheFile000.dta.

The following should be noted when using the automatic incrementing feature:

- The filename will increment every time a file is saved, either via the SAVEFILE command or via the auto-save feature.
- The starting file number is set each time the SAVETOFILE command is sent.

The Auto Save and auto numbering features can consume very large amounts of hard drive space very quickly. Exhausting available disk space on the Windows startup volume can result in system crashes or sluggish performance

Example:

```
DAQ CARD 5 ↵  
SAVETOFILE C:\DATA1.TXT, ASCII ↵
```

See Also: **GET SAVETOFILE**

SET ACTIVE CHAN

Syntax:

```
SET ACTIVE CHAN x-y, z ↵
```

x,y,z any valid set of ADC channels

Sets which ADC channels in the system are active. The set of channels may be specified either as a comma-delimited list or as ranges with a dash, and both syntaxes can be combined on the same line. For example, each of the following commands is equivalent:

```
SET ACTIVE CHAN 1-4 ↵  
SET ACTIVE CHAN 1,2,3,4 ↵  
SET ACTIVE CHAN 1-3,4 ↵
```

Subsequent issuances of SET ACTIVE CHAN will entirely replace the old set of active channels. For example, after the following sequence of commands:

```
SET ACTIVE CHAN 1 ↵  
SET ACTIVE CHAN 2 ↵  
SET ACTIVE CHAN 3 ↵
```

only channel 3 will be active.

See Also: **GET ACTIVE CHAN**

SETUP MODE

Syntax:

SETUP MODE *bool* ↵

bool TRUE or FALSE

Note: Setup mode is not implemented at this time. This feature may be added in a future release.

Turns Setup Mode on or off for Floating Point conversion of data for the selected Data Acquisition card. When FLOAT is set to TRUE, acquired data are converted to floating point data, based on a full-scale ADC range of +/- 2.0 V and the current BITS PER PIX setting. If SETUP MODE is also TRUE, the voltage at the ADC will be reported. If SETUP MODE is false, the voltage at the ADC will be gain- and offset-corrected to reflect the voltage at the input of the preamplifier.

Example:

```
DAQ CARD 2 ↵  
FLOAT TRUE ↵  
SETUP MODE TRUE ↵
```

See Also: **BITS PER PIX**
FLOAT
GET SETUP MODE

SLAVE

Syntax:

SLAVE *bool* ↵

bool TRUE or FALSE

Sets the selected Data Acquisition card to Slave mode. Requires that the selected Data Acquisition card be physically cabled to and configured for (via ATTACH DAQ) slave status.

When SLAVE is enabled (True), the Data Acquisition card will be armed synchronously with its Master, and its acquired data will be saved to a common file. If Slave is disabled (False), the selected Data Acquisition card will be armed independently and its data saved to its own file. See **Section 3.7.2. Master/Slave setup** for a detailed description of FPA definition and setup.

Example:

```
DAQ CARD 7 ↵  
SLAVE TRUE ↵
```

See Also: **ATTACH DAQ**
GET FPA
GET SLAVE
STITCHING

STARTACQ

Syntax:

STARTACQ ↵

Arms all Data Acquisition cards in the system that have active channels. If Blocking is enabled, no communication with the instrument will be possible until all armed cards have completed or until the Timeout period is reached. If Blocking is disabled, use ACQSTATUS to see the current status of the acquisition.

When the acquisition has completed, the data will be saved to a file. Example:

```
STARTACQ ↵
```

See Also: **ACQSTATUS**
ACQTIME
BLOCKING
SAVETOFILE
STOPACQ

STATUS

Syntax:

```
STATUS ↵
```

Returns the error state of the Data Acquisition subsystem or the response to the last query, if it has not already been read out.

Example:

```
DAQ CARD 7 ↵  
SLAVE TRUE ↵  
STATUS ↵
```

```
↳READY ↵
```

```
SALVE TRUE ↵  
STATUS ↵
```

```
↳Error. Unrecognized command Salve ↵
```

See Also: **ACQSTATUS**

STITCHING

Syntax:

```
STITCHING bool ↵
```

bool TRUE or FALSE

Turns Stitching mode on or off for the selected ADC channel. When Stitching is enabled, each ADC channel's pixels will be plotted in accordance with their POSITION settings. Please see Appendix A for pixel locations when Stitching is disabled. See Section 3.7. DEFINE mnemonic for a detailed description of FPA definition and setup.

Example:

```
DAQ CARD 1 ↵  
STITCHING TRUE ↵
```

See Also: **GET FPA**
OUTPUT
POSITION

VERTICAL

STOPACQ

Syntax:

STOPACQ ↗

Stops any pending data acquisition. STOPACQ can be used only if Blocking is disabled. Example:

STOPACQ ↗

See Also: **ACQSTATUS**
ACQTIME
BLOCKING
SAVETOFILE
STARTACQ

VERTICAL

Syntax:

VERTICAL *bool* ↗

bool TRUE or FALSE

Sets the clocking orientation for all ADC channels connected to the selected Data Acquisition card to Vertical (True) or Horizontal (False, default). When Vertical is set to False (the default), the X dimension corresponds to the pixel clock and the Y dimension corresponds to the Line clock. Please see Appendix A for pixel locations when Stitching is disabled. See Section 3.7. DEFINE mnemonic for a detailed description of FPA definition and setup.

Example:

DAQ CARD 1 ↗
STITCHING TRUE ↗

See Also: **GET FPA**
OUTPUT
POSITION
STITCHING

VIDEO/CONVERT/CDS

Syntax:

VIDEO *x* **CONVERT** *y* [**CDS** *z*] ↗

Sets the video, convert, and (optional) CDS output monitor for the AIM housing the selected channel. The video, convert, and CDS monitors can be set to different channels, either within or among AIMS. Example:

CHAN 1 ↗
VIDEO 1 CONVERT 2 CDS 2 ↗
CHAN 5 ↗
VIDEO 2 CONVERT 3 ↗

See Also: **GET MONITOR**

VMAX

Syntax:

VMAX *volts* ↗

volts maximum voltage level from ADC

Specifies the voltage value corresponding to the maximum (full-scale) value from the ADCs for the specified Master group. This value will be written to the data file header. If the Data Type is set to Float this value will also be used to perform floating-point conversion.

Example:

```
DAQ CARD 1 ↗  
VMAX 2.000 ↗
```

See Also: **BITSPERPIX**
FLOAT
VMIN

VMIN

Syntax:

VMIN *volts* ↗

volts minimum voltage level from ADC

Specifies the voltage value corresponding to the minimum (0x0000) value from the ADCs for the specified Master group. This value will be written to the data file header. If the Data Type is set to Float this value will also be used to perform floating-point conversion.

Example:

```
DAQ CARD 1 ↗  
VMIN -2.000 ↗
```

See Also: **BITSPERPIX**
FLOAT
VMAX

9. INDEX

- (REM) command, 123
- @ command, 123
- ACQSTATUS command, 124
- ACQTIME command, 124
- acquired data, 110
- Active, 31
- ADC counts, 51
- AIM controls, 6
- AIM mnemonic, 24
- ALL command, 147
- ALLCARD command, 125
- ALLCHAN command, 125
- AOI (Area of Interest), 49
- AOI command, 125
- Area of Interest (AOI), 49
- Arm cable, 12
- ATTACH command, 126, 127
- auto numbering, 56
- auto save, 55
- average, 53
- AVERAGE command, 128
- bandwidth, 26
- BIASCLKMSGS command, 128, 129, 132, 149
- bit width, 26, 52
- BITSPERPIX command, 129
- black point, 77
- BLOCKING command, 129
- callback function, 110, 111
- Carriage return, 114, 123
- CDS command, 130
- CDS mode, 26
- CDS STROBE command, 130
- CDSTrig In, 7
- CHAN command, 131
- channel numbering, 15
- chassis, 5
- CLK In, 5
- CLK/DATA alignment, 58
- clock inputs, 5
- clock source, 57
- CLOCK SRC command, 131
- color plot, 80
- Command Reference, 119
- command sequencing, 122
- Command strings, 114
- CompactPCI, 128, 129, 132, 149
- continuous acquisition, 73
- Controls, 6
- Conv Trig In, 7
- Convert and CDS strobe timing, 22
- CONVERT STROBE command, 131
- CPU board, 5
- custom applications, 110
- DACQ mnemonic, 44
- DACQ Setup, 44
- Dacq Slot, 30
- DACQMSGS command, 132
- DAQ CARD command, 133
- Data cables, 9, 12
- data files, 56
- data format, 47, 55, 56
- data post-processing, 45, 54
- Data property page, 45, 51
- data type, 51
- data valid, 58
- DEFINE mnemonic, 27
- Delta, 31
- Description property page, 45
- DLL, 110
- external clocking, 57
- false color plot, 80
- false color table, 80
- File exchange, 114
- file name, 56
- File property page, 45, 55
- filter, 26
- FILTERS command, 133
- first valid pixel, 59
- FLOAT command, 133
- floating point, 51, 52
- FPA is not a rectangle error, 32
- FPA property page, 45, 49
- FPA size, 30, 31
- FRAME COUNT command, 134
- Frame In, 7
- frame sync, 23, 57
- FRAME SYNC command, 134
- frames, 50
- frequency plot, 75
- full scale voltage, 52
- gain, 25
- GAIN command, 135
- GET ACQTIME command, 135
- GET ACTIVE CHAN command, 135
- GET ADC IDS command, 135
- GET AOI command, 136
- GET ATTACHMENTS command, 136
- GET AVERAGE command, 136
- GET BITSPERPIX command, 137
- GET BLOCKING command, 137
- GET CDS command, 137
- GET CDS STROBE command, 137
- GET CHAN command, 138
- GET CLOCK SRC command, 138
- GET CONVERT STROBE command, 138
- GET DAQ CARDS command, 138
- GET FILTERS command, 139

GET FLOAT command, 139
 GET FPA command, 139
 GET FRAMECOUNT command, 140
 GET GAIN command, 140
 GET INVERT command, 140
 GET MONITOR command, 140
 GET MUX CARDS command, 141
 GET MUX command, 141
 GET NAMES command, 141
 GET OFFSET command, 142
 GET OUTPUT command, 142
 GET PIXEL PERIOD command, 142
 GET PORT command, 142
 GET POSITION command, 143
 GET PREAMP ID command, 143
 GET SAVETOFILE command, 143
 GET SETUP MODE command, 144
 GET SLAVE command, 144
 GET STITCHING command, 144
 GET SYNC command, 144
 GET TIMING CARDS command, 145
 GET TIMING MAXIMA command, 145
 GET TIMING STEPS command, 145
 GET VERTICAL command, 146
 Go button, 73
 GPIB, 110
 GPIB interface configuration, 111
 GPIBCom.exe, 115
 grayscale correction, 77
 grayscale plot, 77
 hardware configuration, 15
 hardware connections, 17
 hardware not responding error, 20, 116
 Hardware property page, 45, 57
 histogram correction, 76
 histogram cursors, 76
 histogram markers, 76
 histogram plot, 75
 horizontal blanking, 59
 ID command, 146
 IEEE-488, 110
 protocol, 113

 image reassembly, 50
 independent, 29
 independent card, 28
 Input syntax, 114
 INPUT/OUTPUT PROTOCOL, 114
 integer data, 51
 Intersecting Channels error, 32
 Invalid Lattice Size error, 32
 INVERT command, 146
 known issues, 3
 Line feed, 114, 123
 Line In, 7
 line sync, 23, 57

 LINE SYNC command, 134
 Line termination, 114
 linking to dacq.dll, 111
 mainframe, 5
 master/slave setup, 28
 maximum cable length, 8
 Memory button, 45
 message mode, 128, 132, 149
 monitor signals, 8, 26
 MUX command, 147
 NAME AD command, 148
 NINVERT command, 146
 No Active channel defined error, 32
 number of bits, 52
 number of frames, 50
 offset, 25
 OFFSET command, 148
 oscilloscope plot, 74
 OUTPUT command, 148
 output size, 30
 Output syntax, 114, 115
 over-writing files, 55
PATMSGs COMMAND, 129, 132, 133, 149
 PAUSE command, 150
 PI-3100 Acquisition Interface Module, 7
 PI-3150 connections, 7
 PI-41000 Digital Acquisition Card, 11
 PI-PLOT, 67
 PI-PLOT button, 45, 46
 pixel clock, 57
 pixel period, 22
 PIXEL PERIOD command, 150
 pointer to data, 110
 PORT command, 150
 POSITION command, 151
 Post property page, 54, 60
 post-processing property page, 45, 54
 Preamplifier connections, 7
 prev/next frame, 72, 73
 Programmers Reference, 110
 real-time imaging, 73
 reassembly, 50
 RESET command, 151
 ringing, 27
 SAMPLE TESTING UTILITY, 115
 Save File button, 45, 47
 SAVETOFILE command, 152
 SAVETOFILE command, 152
 scope plot, 74
 SET ACTIVE CHAN command, 153
 setup and hold, 58
 Setup mode, 52
 SETUP MODE command, 154
 Signal Name, 31
 simulated components, 16
 Simulation property page, 45

- skyline plot, 74
- SLAVE command, 154
- Slave used without Master error, 32
- SMA connectors, 57
- Start, 31
- Start Acq button, 45
- STARTACQ command, 154
- starting acquisition, 73
- STATUS command, 155
- stitching, 50
- STITCHING command, 155
- Stop Acq button, 45
- STOPACQ command, 156
- tiling, 50
- Timing cables, 7
- timing inputs, 5
- TIMING mnemonic, 22
- Troubleshooting, 107
- Tutorial, 85
- vertical blanking, 59
- vertical checkbox, 31
- VERTICAL command, 156
- VIDEO/CONVERT/CDS command, 156
- VMax, 52
- VMAX command, 157
- VMin, 52
- VMIN command, 157
- white point, 77
- workarounds, 3
- zooming in/out, 73