ARTISAN®
TECHNOLOGY GROUP

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

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

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

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

**Custom engineering**
so your equipment works exactly as you specify.

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

**Expert team | Trust guarantee | 100% satisfaction**

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

Find the *Teradyne / LitePoint / ZTEC Instruments ZT4611* at our website: *Click HERE*



# EPICS Interface

M-Class Oscilloscopes
LXI Models ZT4611, ZT4612,
ZT4211, ZT4212,
ZT4421, ZT4422,
ZT4431, ZT4432,
ZT4441, ZT4442

User's Manual: 0004-000067
Revision 2a

December 7, 2009



Downloaded from www.Manualslib.com manuals search engine

# Contact


| | |
|---|---|
| ZTEC Instruments<br>7715 Tiburon Street NE<br>Albuquerque, NM 87109 | Telephone: (505) 342-0132<br>Fax: (505) 342-0222<br>Web Site: www.ztecinstruments.com |


ZTEC Instruments, Inc. welcomes your comments on this manual. All manuals are thoroughly reviewed before distribution. We are, however, grateful for any comments from our users which will further help to improve the content and quality of our documents.

# Copyright


**Copyright 2006 by ZTEC Instruments**

Printed in the United States of America.

**All rights reserved under copyright laws of the United States and other countries.**

All technical data and computer software documentation contained herein is proprietary and confidential to ZTEC Instruments, Inc. or its licensor. The reproduction and/or transmission of this publication in whole or in part by any means, electronic or mechanical, is prohibited without the prior written consent of ZTEC Instruments, Inc.

ZTEC and the ZTEC logo are registered trademarks of ZTEC Instruments.

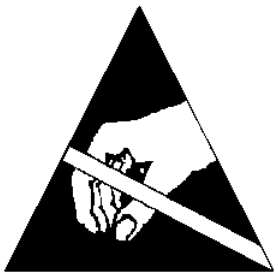
ZTEC Instruments has attempted throughout this publication to distinguish proprietary trademarks from descriptive terms by following the capitalization style used by the manufacturer. Product names listed are trademarks of their respective manufacturers. Company names listed are trademarks or trade names of their respective companies.

The material in this manual is for informational purposes only and is subject to change without notice. ZTEC Instruments, Inc. assumes no responsibility for any error or for consequential damages that may result from the use or misinterpretation of any of the procedures in this publication.





Downloaded from www.Manualslib.com manuals search engine



# Handling Precautions for Electronic Devices Subject to Damage by Static Electricity

This instrument is susceptible to Electronic Static Discharge (ESD) damage. When transporting, place the instrument or module in conductive (anti-static) envelopes or carriers. Open only at an ESD-approved work surface. An ESD safe work surface is defined as follows:

- The work surface must be conductive and reliably connected to an earth ground with a safety resistance of approximately 250 kilohms.

- The surface must NOT be metal. A resistance of 30–300 kilohms per square inch is suggested.

Ground the frame of any line-powered equipment, chassis, test instruments, lamps, soldering irons, etc., directly to the earth ground. To avoid shorting out the safety resistance, ensure that the grounded equipment has rubber feet or other means of insulation from the work surface.

Avoid placing tools or electrical parts on insulators. Do NOT use any hand tool that can generate a static charge, such as a non-conductive plunger-type solder sucker. Use a conductive strap or cable with a wrist cuff to reliably ground to the work surface. The cuff must make electrical contact directly with the skin; do NOT wear it over clothing.

**Note**: Resistance between the skin and the work surface is typically 250 kilohms to 1 megohm using a commercially-available personnel grounding device.

Avoid circumstances that are likely to produce static charges, such as wearing clothes of synthetic material, sitting on a plastic-covered stool (especially when wearing woolen material), combing the hair, or making extensive pencil erasures. These circumstances are most significant when the air is dry.

When testing static sensitive devices, ensure DC power is ON before, during, and after application of test signals. Ensure all pertinent voltages are switched OFF while circuit boards or components are removed or inserted.



Downloaded from www.Manualslib.com manuals search engine

# Revision History

| Rev | Date | Section | Description |
|---|---|---|---|
| 1 | 1-4-08 | All | Initial Release |
| 1a | 4-10-08 | All | Updated PV list, expanded introduction, added upload waveform information |
| 1b | 9-15-08 | All | Added new PVs: gating, initiate continuous, webLXI interface |
| 2 | 9-10-09 | All | All Sections revised, new sections added.<br>Added new PVs: operation complete enable, operation complete query, segment view, segment mode, segment count, accessory id, memory clear, undo, reference position,<br>Created Status Register Section<br>Renamed and Moved some previous 'Util' PVs to 'Status' PVs:<br>UtilStatus → Status (fixed instr error also)<br>UtilStandard → StatusStandard (fixed instr error also)<br>UtilFreqFault → StatusFreq<br>UtilTestResult → StatusTest<br>UtilOperation → StatusOper<br>UtilQuesReg → StatusQues<br>UtilCalResult → StatusCal<br>Fixed record type on above Status PVs, now mbb*Direct<br>Added new PVs: StatusTestDig*N*<br>Added new Status PVs: register enables and conditions<br>Moved set/getEnvView and set/getAverView to mbbo/i<br>(Changed default NELM size to 1000)<br>(Added SCAN field to all PVs)<br>(Fixed ranges to account for multiple instrument series)<br>Added missing entry for Calc*N*GateMethod |
| 2a | 12-7-09 | Process Variables, Capturing | Added Auto Load PV |



Downloaded from www.Manualslib.com manuals search engine

# Table of Contents


**Introduction** ..... **6**
- Instrument Discovery ..... 6
- Versions ..... 6
- Instrument Information ..... 6
- EPICS Information ..... 6
- Installed Files ..... 7

**Configuration** ..... **8**
- Instrument Configuration ..... 8
- Database Configuration ..... 9
  - PV Database File ..... 9
  - Database Template Files ..... 10
  - Database Definition File ..... 10
  - Changing PVs ..... 10
  - PV Naming Conventions ..... 10
  - Set/Get PVs ..... 11
  - FLNKs & Fanout PVs ..... 11
  - Viewing the Current PV Database File ..... 11
  - Uploading a PV Database File ..... 12
- Channel Access Interface ..... 13
- EDM & MEDM Interfaces ..... 14

**Functionality** ..... **15**
- Updating ..... 15
- Interfaces ..... 15
- Capturing ..... 16
  - Continuous ..... 16
  - Single ..... 16
- Uploading Waveforms ..... 17
- Instrument Use Example ..... 18

**Process Variables** ..... **19**
- Process Variable Table ..... 19
  - Input Process Variables ..... 20
  - Horizontal Process Variables ..... 21
  - Acquisition Process Variables ..... 22
  - Trigger Process Variables ..... 23
  - Advanced Trigger Process Variables ..... 24
  - Arm Process Variables ..... 26
  - Output Process Variables ..... 27
  - Calculate Process Variables ..... 28
  - Advanced Calculate Process Variables ..... 30
  - Reference Process Variables ..... 34
  - Measure Process Variables ..... 35
  - Operate Process Variables ..... 38
  - Waveform Process Variables ..... 39
  - Utility Process Variables ..... 40
  - Status Register Variables ..... 44




Downloaded from www.Manualslib.com manuals search engine

# Introduction



This manual provides detailed information on the ZTEC® EPICS interface and functionality. Printed LXI Quick Start and EPICS Getting Started guides, included with an instrument, may provide enough information to get an instrument up and running quickly. Reference this manual or visit http://www.ztecinstruments.com/support for additional technical support.

## Instrument Discovery

Interfacing to an EPICS ZTEC® M-Class Scope involves LAN discovery to identify the IP address of the desired instrument. The IP address is necessary to access network and EPICS configuration settings. The IP address can be configured or identified in one of the following ways:

1. With a DHCP server, the instrument will be dynamically assigned an IP address.
2. Without a DHCP server, the instrument will use auto-IP procedures to find a non-allocated IP address.
3. To determine the instrument's IP address, consult the network manager or use any LXI compatible discovery tool, such as the provided ZFind™ Utility. A monitor, connected to the instrument's rear VGA port during startup, will also display the IP address during boot.
4. After discovery, the ZTEC® *webLXI* interface enables the network addresses to be altered, made static, and/or given an alias.

## Versions

The ZTEC® EPICS interface was developed using EPICS version 3.14.9 and Channel Access Client version 4.11.

## Instrument Information

For more information on the instrument such as descriptions of functionality, specifications, and default states please refer to the M-Class Digital Storage Oscilloscope Instrument Manual.

## EPICS Information

Visit the Experimental Physics Industrial Control System (EPICS) Home page at http://www.aps.anl.gov/epics/ for information, detailed documentation and tutorials and other downloads for the EPICS and Channel Access systems.



Downloaded from www.Manualslib.com manuals search engine

# Installed Files

Both Windows and Linux versions of the ZTEC® M-Class ZScope® installers provide EPICS files in the following default installation directories (EPICS Installation):

| | |
|---|---|
| Windows: | C:\Program Files\ZTEC Instruments\Mclass\Scope\EPICS |
| Linux: | /usr/local/share/ZScopeM/EPICS |



Downloaded from www.Manualslib.com manuals search engine

# Configuration



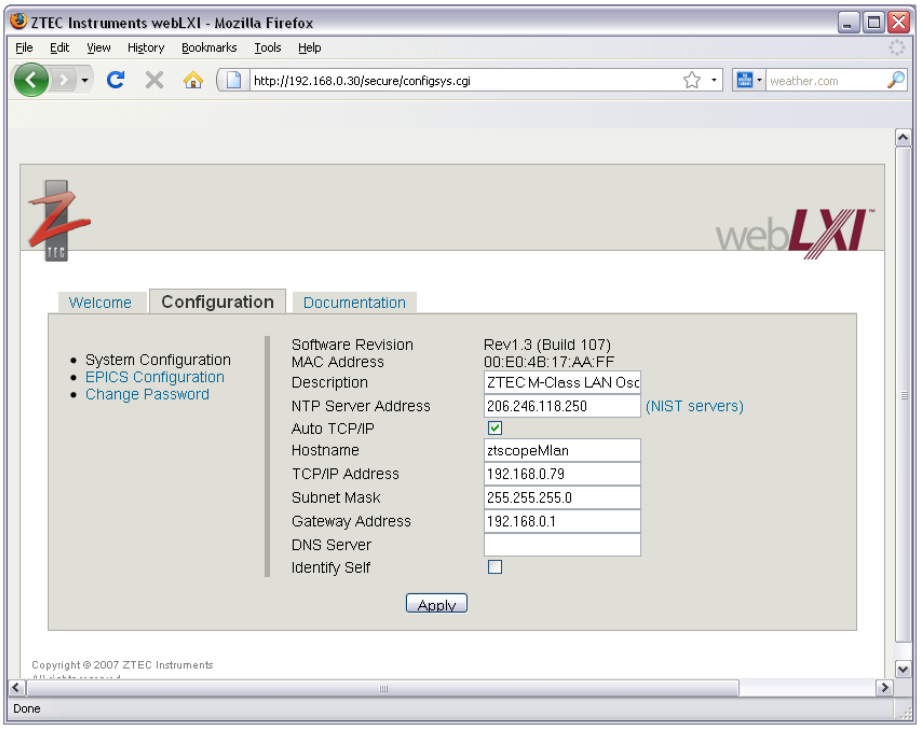

## Instrument Configuration

The instrument's Network and Channel Access settings may be changed using the ZTEC® *webLXI* interface, accessed through a web browser by navigating to the IP address of the instrument.

*webLXI* presents three main pages, or tabs, of functionality. The initial page is a Welcome tab which presents a set of instrument information available to all *webLXI* users. Network configuration is done through the System Configuration page, located on the password protected Configuration tab. EPICS configuration is also done from the Configuration tab. The final tab is Documentation, which presents information about ZTEC® and various hyperlinks.

The actions following actions should facilitate communication, but more detail on the *webLXI* interface can be found in the *ZFind User's Guide*.

Go to the Configuration → System Configuration page, change the network values as desired and press the Apply button. Browse to the new IP address if changed.

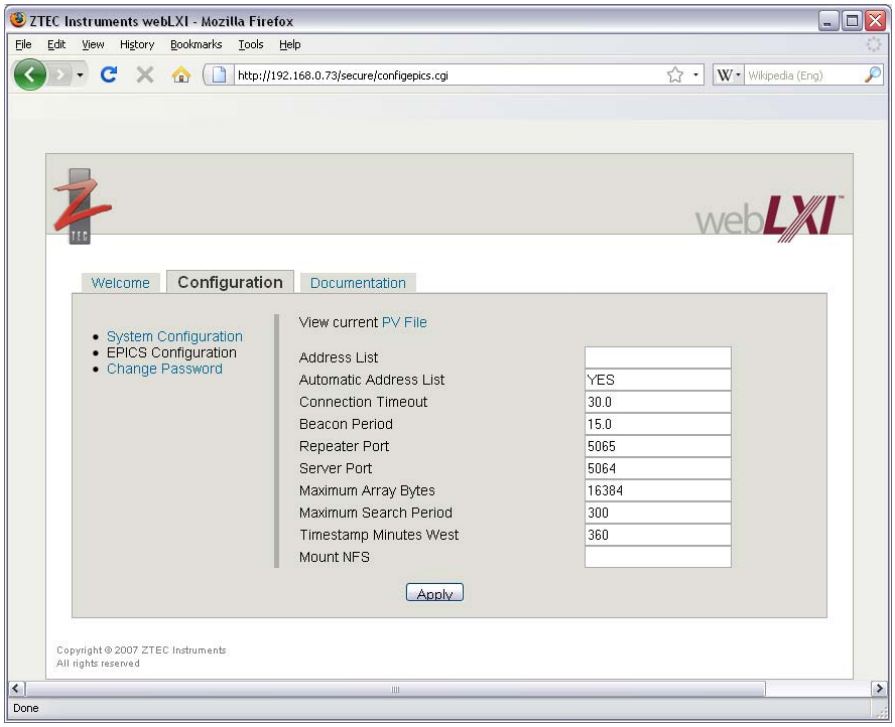



*webLXI* System Configuration



Downloaded from www.Manualslib.com manuals search engine

Go to the Configuration → EPICS Configuration page to adjust and apply the Channel Access values. Power cycle the instrument if any changes are made.




*webLXI* EPICS Configuration

# Database Configuration

## PV Database File

The PV database file defines the capabilities of the instrument accessible through EPICS. It contains a combined list of Process Variables (PVs), arranged by record type, along with their SCPI commands, initial values, ranges, forward links, and other record fields which vary by type.

The installer provides two versions of the default PV file, ztscopeM.pv, which are installed in the following locations by default:

Windows: C:\Program Files\ZTEC Instruments\Mclass\Scope\EPICS\dat\
[ztscopeM-ch2 | ztscopeM-ch4]\ztscopeM.pv

Linux: /usr/local/share/ZScopeM/EPICS/dat/
[ztscopeM-ch2 | ztscopeM-ch4]/ztscopeM.pv

There is a two-channel version, which includes ZTEC®'s common PVs as well as two-channel-only PVs. The four-channel version has the common and four-channel-only PVs.

The version of this PV database file matching the instrument's number of channels should be copied and used as a basis for modification; the name of this file is arbitrary and, as such, may be renamed to fit its description. When creating a PV File, note that ranges and other values listed in the default PV file may be instrument-generic. Refer to



Downloaded from www.Manualslib.com manuals search engine

the Instrument Manual for values that match your specific instrument. The default PV file comes preloaded on the instrument and provides most instrument functionality.

Each PV is an instance of a record, which has a type and a set of fields.
Each PV within the database file references a template file. The template file matches its type and also references a pattern to define the order of each field from the template or other fields common to all records. See EPICS documentation for further details on records.

Example contents of a PV database file, excerpted from ztscopeM.pv:

A set of Binary Output PVs, setInp1Enable and setInp2Enable:

```
file "$(EPICS_PV_PATH)/db/ztecbo.template" {
     pattern {NAME, SCPI, ONAM, ZNAM, SCAN, FLNK}
         {ztec:setInp1Enable, "INP1", "Enabled", "Disabled",
             "Passive", "ztec:getInp1Enable"}
         {ztec:setInp2Enable, "INP2", "Enabled", "Disabled",
             "Passive", "ztec:getInp2Enable"}
         ...
}
```

## Database Template Files

The database template files define the fields within records of a type defined in the database definition file. The zipped set of ZTEC®'s .template files are included in the EPICS Installation *dat* directory. These can be used for reference, but cannot be modified on the instrument.

## Database Definition File

The database definition file defines the record types, devices, and behaviors for the database. See the EPICS documentation for reference and examples. A "ztec" device type has been added and all PVs on the instrument will have this type. This cannot be modified on the instrument.

## Changing PVs

All PVs can be safely removed or renamed, as long as any forward links and fanouts referencing a PV are also changed. Additional PVs may be added to the database file by following the provided template format and uploading the file as described below. Additional instrument functionality can be exposed by creating a PV associated with an instrument SCPI call. This is done by inserting the SCPI string into the SCPI section of the PV. See the instrument manual for a list of supported SCPI calls. The ztec:UtilSCPISend and ztec:UtilSCPIRecv PVs may also be used to send and receive SCPI strings without adding new PVs.

## PV Naming Conventions

The EPICS naming convention requires the ability to add a prefix to the process variables. ZTEC® PVs have "ztec:" as a default prefix. The prefixes/names can be changed by editing and uploading the default PV file. If unique names are required, such as in the case of having two or more ZTEC® EPICS oscilloscopes, changing the PV prefixes to separate identifiers will clear up any ambiguity about which instrument is



Downloaded from www.Manualslib.com manuals search engine

being communicated with. Name changes may affect links and fanouts; to prevent loss of functionality make sure any FLNK names are also changed.

## Set/Get PVs

The “set” PVs allow the user to change an instrument’s state. The “get” PVs allow the user to read the value without changing the instrument’s state. “get” PVs are also useful for keeping the database up to date by scanning using the .SCAN parameter, and can be chained to set PVs so they will notify the user of state changes. Setting a SCAN parameter other than *Passive* on a set PV is not recommended as it will change the parameter value.

For all read/write attributes, both a set and a get PV are provided. Default names for these PVs begin with either ”set” or “get” to avoid identical names on multiple PVs. Attributes which are only read or only write have a single PV with neither a “get” nor a “set” in the default name.

## FLNKs & Fanout PVs

Forward links chain PVs together so when a PV is processed, the PV named in its FLNK field will also process. Normally, there is only one FLNK field in a PV. The instrument’s default PV database comes with a set of FLNKs to ensure “get” PVs are processed when the matching “set” PV is processed.

Fanout PVs consist of sets of forward links to other PVs. Fanouts are useful when a series of commands needs to be run in order, repeatedly. Instead of creating a long chain of forward links, fanouts provide the same functionality while maintaining a legible, traceable PV file. A fanout PV can be an entry in any FLNK field, the same as other PVs can. Their FLNKs can also link to other fanout PVs. The instrument’s default PV database comes with a set of fanouts which allow for standard instrument coercion.

Example fanout from ztscopeM.pv:

```
file "$(EPICS_PV_PATH)/db/ztecfanout.template" {
      pattern {NAME, SELM, LNK1, LNK2, LNK3, LNK4, LNK5, LNK6, SCAN}
            {ztec:SweepFanout, All,
                  "ztec:getHorzPoints",
                  "ztec:HorzRate",
                  "ztec:HorzInterval",
                  "ztec:getHorzTime",
                  "",
                  "",
                  "Passive"}
      ...
}
```

## Viewing the Current PV Database File

The current PV database file can be checked by using the webLXI interface, discussed in the Instrument Configuration section above and in the *ZFind User’s Guide*.

Click on the “PV File” link on the EPICS Configuration webLXI page, or navigate a browser straight to *http://<your instrument’s ip address>/ztec.pv*. Note whatever the name of the uploaded PV database file, it will be reached with this “/ztec.pv” path extension.



Downloaded from www.Manualslib.com manuals search engine

Through a browser's File menu, the *…/ztec.pv* page can be saved as a text file to then be modified and uploaded, or just backed up. The name of this file does not matter in regards to uploading, but the *.pv* extension is suggested, along with a relevant filename, in order to better track its contents.

## Uploading a PV Database File

There are two methods available to update the PV Database file on the instrument. New database configuration files can be saved to the instrument using the "Load PV Database File" feature in the ZFind™ (located in the *SFP/bin* directory of a ZScopeM® installation) or through the command-line-based ZPVFlash utility (located within the *bin* directory of the EPICS Installation).

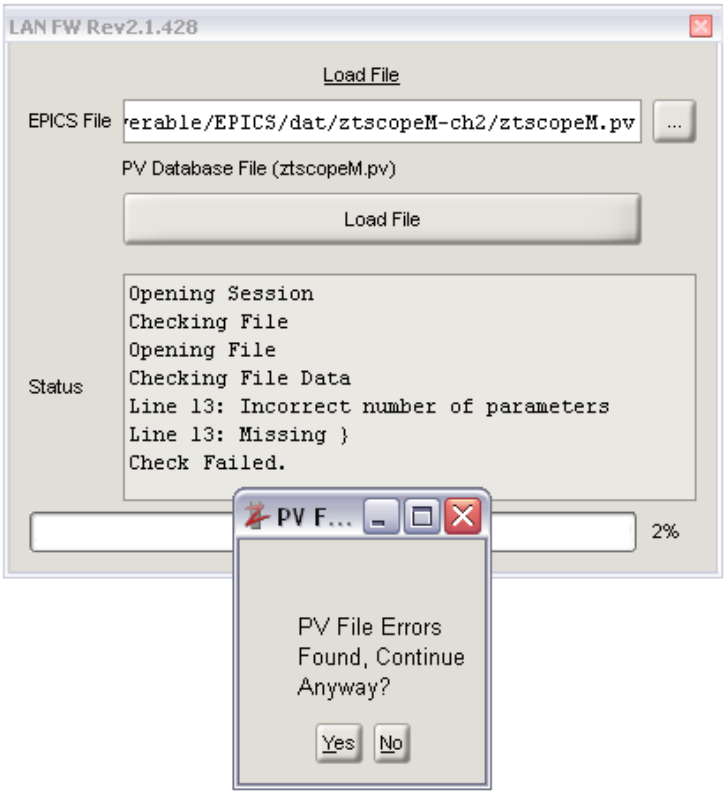
To upload a PV Database file using the ZTEC® ZFind™ utility,

- Click on the desired instrument.
- Click the EPICS icon in the toolbar (see the ZFind Users Guide for how to add an instrument). This will bring up an interface to upload a file.
- Browse to the new PV database file.
- click Load File to start the upload.
    - The instrument will restart itself during this process, once the file has finished uploading.

During upload, ZFind™ will check the PV file for errors. If an error is detected an additional popup will appear with the option to either exit upload or continue. This is necessary because uploading an invalid PV file will stop EPICS from booting and the checker may not be able to compensate for drastic changes to the default PV file. Detected errors are reported with their line number in the PV file so they may be tracked down and corrected.



Downloaded from www.Manualslib.com manuals search engine




PV File Upload Interface

Once The PV file is loaded it can be reviewed by using the link on the EPICS Configuration webLXI page, or going to http://<your instrument's ip address>/ztec.pv. If the new PV file is not visible, make sure the browser is not caching an old page by clearing the cache or temporary internet files, and then try to access the page again.

Since full access is allowed to the PVs, it is possible to upload a non-functional file. If this occurs, EPICS functionality will become fully or partially disabled, depending on the type of error(s). PV files can be invalidated in many ways, such as: missing brackets or quotations, incorrect number of fields, or FLNKs to non-existent PVs. In order to verify the upload was successful, attempt to access the instrument through Channel Access. If this fails, upload a valid PV file, such as the default ztscopeM.pv.

# Channel Access Interface

Channel Access is EPICS's default command-line and code-based interface, used to view and modify process variables. ZTEC® has provided an example configuration script that can be used as a starting point if Channel Access is the preferred interface to EPICS. This script, *epics_ca_config.sh,* can be found in the EPICS Installation *script* directory. Note this is a bash script, so it is necessary to be running Cygwin if using a Windows environment, or make command or syntax modifications if using another UNIX or Linux shell.



Downloaded from www.Manualslib.com manuals search engine

# EDM & MEDM Interfaces

The Extensible Display Manager (EDM) and the Motif Editor and Display Manager (MEDM) are common graphical extensions to Channel Access and EPICS. ZTEC® has provided an example display panel for EDM and for MEDM. The panel launcher scripts can be found in the EPICS Installation *script* directories. The configuration files for the panels are found in the EPICS Installation *dat* directory. Using the provided scripts it is possible to use these applications even if the PV prefixes have been altered. For more extensive PV changes, the provided panels must be modified.

The provided panels, while fully functional, are intended as a starting point or example application and will need to be edited to adjust for the desired functionality. In the EDM folder, modify the *.edl files, or in the MEDM folder, modify the *.adl files to edit the panels.



Downloaded from www.Manualslib.com manuals search engine

# Functionality



## Updating

ZTEC® set PVs have the option of updating upon instrument state coercion. This option is controlled by the setOutCoerce PV, which is off by default. If setOutCoerce is off when a set PV is changed, the user should poll the corresponding get PV to verify the new value. However, if setOutCoerce is set to on, set PVs will update their own value based on the current instrument state during processing.

For example, with a ZT4611, when setting an invalid input range of 3, the value will be coerced to 2. In Channel Access you would see:

with setOutCoerce Off/default:
$: caput setInp1Range 3
Old 2
New 3
with setOutCoerce On:
$: caput setInp1Range 3
Old 2
New 2
A *caget getInp1Range 1* will receive the value of 2 in either case.

There are five ways to process or trigger updates on a PV:

- Providing a forward link (FLNK) to the PV within another PV will cause the FLNKed PV to process after the first PV is processed.
- Adding the PV to a fanout PV and processing the fanout (directly, from another fanout, or from a FLNK) will process each PV in the fanout group in order.
- Sending *caput ztec:PvName value* through Channel Access (When sending input PVs through *caput*, any value will work).
- Sending *caput ztec:PvName.PROC 1* through Channel Access.
- Setting the SCAN field of the PV to an appropriate interval, as discussed in the EPICS documentation. By default the SCAN field on all PVs is set to “Passive”, so only the previous update methods are in effect. Scanning is not recommended for output PVs.

## Interfaces

ZTEC® LXI instruments are accessible through several interfaces: drivers, GUIs and EPICS. Only one non-Channel Access (CA) interface, such as the drivers or ZScope®, may access the instrument at a time. However, connections to the EPICS database are not restricted. MEDM and EDM GUIs use the EPICS database and as such may have multiple instances. Note all interfaces are independent and may cause undesirable interactions if used simultaneously. Parameters changed through non-CA interfaces will not cause PV values to update. If non-CA interfaces have been used, it is recommended to call *UtilRefresh*. *UtilRefresh* will cause all PVs to update with the current instrument parameter values. Capturing waveforms through multiple interfaces



Downloaded from www.Manualslib.com manuals search engine

is not recommended. Note that PVs with scan rates will continue to access the instrument even if they are not being used. Ensure that all scan rates are disabled before using a non-CA interface.

# Capturing

Before capturing waveforms it is best to configure the instrument as needed (see Instrument Use Example and the Waveform Process Variables Table). An important PV to configure is setOpInitCont.

Capture is not designed for use in multiple interfaces. Multiple EPICS interfaces are fine since they are accessing the same PVs, but automated code and the provided GUIs have a separate interface, see *Interfaces* above. Accessing waveforms from EPICS and a non-CA interface simultaneously may cause slower updates or incorrect data downloads as both interfaces will be attempting to access the same buffers.

## Continuous

If setOpInitCont is set to initiate continuously, the instrument will capture waveforms continuously. In this mode, to capture, set a 1 to OpInitiate, and to stop capturing, set a 1 to OpAbort. While the instrument is running continuously, downloaded waveforms are only available at 1000 points or less. Larger captures are allowed, but the data is decimated on read. Continuous initiation will occur as fast as possible, given the acquisition settings, and it may not be possible to download all waveforms before a new capture is done. This mode is recommended for Limit and Mask testing.

When running continuously, all waveforms are returned as a single buffer. This allows all the waveforms to come from the same capture, even though multiple captures may occur during the time it takes to perform the waveform download. To download the next waveform capture, process the OpAutoLoad PV. By default, this PV has a fanout to Input and Calc scaled waveform PVs. Download speed may be increased by removing any unused channels from the OpAutoLoad fanout. Recommended use is to set a SCAN on this PV for continual waveform updates during continuous acquisition.

## Single

If setOpInitCont is set to single, setting a 1 to OpInitiate will cause a single capture event followed by the instrument returning to an idle state. Setting the setOpInitCont PV to capture single waveforms and then setting the .SCAN field on OpInitiate is an alternate method to capture repeatedly while still allowing full length downloads. This method may be more desirable to avoid missing data downloads.



Downloaded from www.Manualslib.com manuals search engine

# Uploading Waveforms

Waveforms can be uploaded as an array of type DBR_DOUBLE via Channel Access using the Calc*N*Upload, Calc*N*ScaledUpload, Ref*N*Upload and Ref*N*ScaledUpload PVs (see section “Reference Process Variables”). In order to correctly store the uploaded waveform data, some waveform preamble information must be included. This is done by storing the needed values in the first four elements of the array. Note the array size transferred via Channel Access must be adjusted accordingly. Here is a simple example of a function using the Channel Access C function calls:

```
void upload_waveform(long *wf_buff, long type, unsigned long points,
                     double time_int, double, volt_int)
{
   int idx;
   double *upload_buff;

   upload_buff=(double *)calloc(points+4, sizeof(f64));
   upload_buff[0]=(double)type;
   upload_buff[1]=(double)points;
   upload_buff[2]=(double)time_int;
   upload_buff[3]=(double)volt_int;
   for (idx=0;idx<points;idx++)
   {
          upload_buff[idx+4]=(f64)(wf_buff[idx]);
   }

   ca_context_create(ca_disable_preemptive_callback);
   ca_create_channel("ztec:Ref1Upload",NULL,NULL,10,&chan_id);
   ca_pend_io(5.0);
   ca_array_put(DBR_DOUBLE, points+4, chan_id, (void *)upload_buff);
   ca_pend_io(5.0);
   ca_clear_channel(chan_id);
   ca_pend_io(5.0);
   free(upload_buff);
   ca_context_destroy();
}
```

This function would upload a long integer waveform data array named *points*, with waveform preamble data:

- waveform type = *type*,
- number of points = *points*,
- time interval = *time_int*,
- voltage interval = *volt_int.*

For more information on waveform preamble, please refer to the M-Class Digital Storage Oscilloscope Instrument Manual. For information on programming using the Channel Access C function calls, refer to the Channel Access documentation (http://www.aps.anl.gov/epics/).



Downloaded from www.Manualslib.com manuals search engine

# Instrument Use Example

This set up assumes a system with the EPICS environment installed.

1. Limit your instrument communication, there are several possible options:
    a. Upload a new PV file as described above. Changing the PV prefixes will give the instrument a unique set of PVs.
    b. Set up your local Channel Access environment variables. If you set EPICS_CA_AUTO_ADDR_LIST to “no” and EPICS_CA_ADDR_LIST to the IP address of the instrument, this will be the only instrument seen by the local system. Set EPICS_CA_MAX_ARRAY_BYTES to larger than or equal to the configured waveform download size. Valid values must be set on both the client and server side.
    c. Set up the instrument’s channel access environment variables. This will limit which systems the instrument will communicate with. The environment variables are set through *webLXI* as described above. The environment variables are the same whether they are on your local system or on the instrument.
2. Open a channel access command window
3. Test instrument communication with *caget ztec:UtilID*. If you uploaded a different PV file, be sure to use the proper prefix.
4. To set up the instrument for a test, it is suggested to use the following process:
    a. set Input PVs for all desired channels
    b. set Acquisition PVs
    c. set Horizontal PVs
    d. set Trigger PVs
        i. set Advanced Trigger PVs if required
        ii. set Arm PVs for gating triggering if required
    e. set Output PVs for output signal control if required
    f. set Calculate PVs for waveform math if required
    g. set Reference PVs for waveform storage if required
    h. set Measure PVs to be performed upon acquired waveforms if required
    i. control waveform capture with Operate PVs
    j. read acquired waveforms from Waveform PVs
    k. perform miscellaneous functions with Utility PVs if required



Downloaded from www.Manualslib.com manuals search engine

# Process Variables




## Process Variable Table

The following tables list the process variables for the oscilloscope, grouped according to instrument functionality.



Downloaded from www.Manualslib.com manuals search engine

## Input Process Variables

The following table lists the PVs for the input or vertical (voltage-axis) settings of the input channels. These PVs are duplicated for each input channel. In the following table, *N* is substituted with the appropriate channel number (for example: Inp1Enable). The ZT4xx1 has two channels (*N* = 1 to 2) and the ZT4xx2 has four channels (*N* = 1 to 4).

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setInp*N*Enable<br>getInp*N*Enable | bo<br>bi | 0 or 1 | Input channel enable for selected channel:<br>0 = disable<br>1 = enable |
| setInp*N*Range<br>getInp*N*Range | ao<br>ai | Varies by model, See Instrument Manual | Input range in Vpp for selected channel, Input range is limited to 10Vpp for 50 Ohm impedance |
| setInp*N*Delay<br>getInp*N*Delay | ao<br>ai | 0.0 to 10.0 µs | Sets or queries the input channel-to-channel delay (skew). If all channels have a delay applied to them the instrument will zero out the smallest delay and apply the difference to the other channels. |
| setInp*N*Offset<br>getInp*N*Offset | ao<br>ai | Varies by model, See Instrument Manual | Sets or queries the input offset in Volts for selected channel, Input offset is limited according to range and impedance settings (see specifications). |
| setInp*N*Pos<br>getInp*N*Pos | ao<br>ai | 0.0 to 1.0 | Sets or queries the relative DC voltage offset for the input channel represented at the vertical zero for the selected channel. Default waveform position is 0.5 (50%). The limits upon the input voltage offset setting are dependent upon the input voltage range. |
| setInp*N*ProtState<br>getInp*N*ProtState | bo<br>bi | 0 or 1 | Sets or queries the specified input channel voltage protection state.<br>0 = Input channel voltage protection OFF<br>1 = Input channel voltage protection ON |
| setInp*N*Imped<br>getInp*N*Imped | ao<br>ai | 50 or 1e6 | Input impedance for selected channel:<br>50 = 50 Ohm (low)<br>1e6 = 1 MOhm (high) |
| setInp*N*Couple<br>getInp*N*Couple | mbbo<br>mbbi | Discrete | Input coupling for selected channel:<br>“AC” = AC Coupling<br>“DC” = DC Coupling |
| setInp*N*Filter<br>getInp*N*Filter | bo<br>bi | 0 or 1 | Input lowpass filter enable for selected channel:<br>0 = bypass<br>1 = filter enable (20 MHz) |
| setInp*N*Atten<br>getInp*N*Atten | ao<br>ai | 0.9 to 1000 | Input attenuation for selected channel, typically used for probe or cable loss correction, input range and input offset are scaled accordingly, 1.0 nominal, example: use 10.0 for 10:1 probe |



Downloaded from www.Manualslib.com manuals search engine

## Horizontal Process Variables

The following table lists the PVs for the horizontal (time-axis) settings for the waveform acquisition. These PVs configure the common horizontal settings of all input channels.

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setHorzPoints<br>getHorzPoints | longout<br>longin | 10 to max, Varies by model, See Instrument Manual | Number of points in waveform |
| setHorzTime<br>getHorzTime | ao<br>ai | Varies by model, See Instrument Manual | Waveform sweep time (acquisition window) |
| setHorzRef<br>getHorzRef | ao<br>ai | 0.0 to 1.0 | Trigger location in acquisition window:<br>0.0 to 1.0, where 0.0 corresponds to left-most trigger location in acquisition window and 1.0, the right-most. |
| setHorzOffset<br>getHorzOffset | ao<br>ai | -pre-trigger time to 100 | Time offset in seconds for trigger from selected trigger location offset reference:<br>Pre-trigger: 0 to 100% of acquisition time,<br>Post-trigger: 0 to 100 seconds |
| HorzRate | ai | Varies by model, See Instrument Manual | Read back for sample rate setting, based on the configured sweep time and sweep points. |
| HorzInterval | ai | Varies by model, See Instrument Manual | Readback for sample interval setting, equal to 1/sample rate. |
| setHorzRefOsc<br>getHorzRefOsc | mbbo<br>mbbi | Discrete | Sets or queries the source for the reference clock providing the instrument time base.<br>“INT” = internal<br>“EXT” = external input<br>“CLK10” = clock 10 / backplane |
| HorzRefFreq | ai | 9,999,000 to 10,001,000 | Reference oscillator frequency relative to internal reference frequency upon power-on self test |



Downloaded from www.Manualslib.com manuals search engine

## Acquisition Process Variables

The following table lists the PVs for the waveform acquisition settings. These PVs configure the common acquisition settings of all input channels.

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setAcqType<br>getAcqType | mbbo<br>mbbi | Discrete | Waveform acquisition type:<br>"NORM" = normal<br>"AVER" = average<br>"ENV" = envelope<br>"PDET" = peak detect<br>"HRES" = high resolution<br>"FAST" = fast acquisition<br>"ETIM" = equivalent-time |
| setAcqCount<br>getAcqCount | longout<br>longin | 2 to 65536 in powers of 2 | Waveform count for multiple-capture acquisition types of average, envelope, equivalent-time, and fast acquisition |
| setEnvView<br>getEnvView | mbbo<br>mbbi | Discrete | Envelope view select:<br>"MIN" = minimum<br>"MAX" = maximum |
| setAverView<br>getAverView | mbbo<br>mbbi | Discrete | Average view select:<br>"AVER" = average<br>"SEGM" = segments |
| SegCount | longin | variable, >=0 | Queries the number of historical segments captured. This is only valid for History mode. |
| setSegView<br>getSegView | longin<br>longout | 0 to SegCount | Sets or queries the current segment view. This is only valid for History mode. The viewed segment is the segment returned in waveform downloads and the segment used for measurements and calculations. |
| setSegMode<br>getSegMode | mbbo<br>mbbi | Discrete | Segment mode select:<br>"HIST" = history mode<br>"ZOOM" = zoom mode |



Downloaded from www.Manualslib.com manuals search engine

## Trigger Process Variables

The following table lists the PVs for the trigger settings used to configure the trigger for the waveform acquisition synchronization. There are individual trigger level settings for each of the input channels, with *N* designating the appropriate channel number (for example: TrigLevInp1). The ZT4xx1 has two channels (*N* = 1 to 2) and the ZT4xx2 has four channels (*N* = 1 to 4).

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setTrigSource<br>getTrigSource | mbbo<br>mbbi | Discrete | Trigger source:<br>"TTLT0" = TTLTRG0 (external header)<br>"TTLT1" = TTLTRG1 (external header)<br>"TTLT2" = TTLTRG2 (external header)<br>"TTLT3" = TTLTRG3 (external header)<br>"TTLT4" = TTLTRG4 (external header)<br>"TTLT5" = TTLTRG5 (external header)<br>"TTLT6" = TTLTRG6 (external header)<br>"TTLT7" = TTLTRG7 (external header)<br>"INP1" = input channel 1<br>"INP2" = input channel 2<br>"INP3" = input channel 3 (ZT4xx2)<br>"INP4" = input channel 4 (ZT4xx2)<br>"EXT" = external input<br>"MAN" = software only<br>"PATT" = pattern |
| setTrigType<br>getTrigType | mbbo<br>mbbi | Discrete | Trigger type:<br>"EDGE" = edge<br>"IN" = pulse width in<br>"OUT" = pulse width out<br>"LTH" = pulse width less than<br>"GTH" = pulse width greater than<br>"EQU" = pulse width equal<br>"NEQ" = pulse width not equal<br>"GLIT" = glitch<br>"VID" = video |
| setTrigMode<br>getTrigMode | mbbo<br>mbbi | Discrete | Trigger mode:<br>"AUTO" = auto-trigger<br>"NORM" = normal trigger |
| setTrigSlope<br>getTrigSlope | mbbo<br>mbbi | Discrete | Trigger slope or polarity:<br>"NEG" = negative<br>"POS" = positive |
| setTrigLevInp*N*<br>getTrigLevInp*N* | ao<br>ai | (offset –range/2) to (offset + range/2) | Trigger level in Volts for an input channel.<br>The trigger level must be within the vertical range and offset settings for the corresponding input channel |
| setTrigLevExt<br>getTrigLevExt | ao<br>ai | -2.0 to 2.0 | Trigger level in Volts for the external input trigger |
| setTrigImpedExt<br>getTrigImpedExt | ao<br>ai | 50 or 1e6 | Input impedance for external input trigger:<br>50 = 50 Ohm<br>1e6 = 1 MOhm |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setTrigHoldoff<br>getTrigHoldoff | ao<br>ai | 0 to 100 | Holdoff time in seconds to wait before detecting trigger |
| setTrigCount<br>getTrigCount | longout<br>longin | 1 to 65535 | Trigger event count to qualify trigger |

## Advanced Trigger Process Variables

The following table lists the PVs for the advanced trigger settings. These PVs configure advanced trigger functions including trigger B, pulse width triggering, glitch triggering, pattern triggering, and video triggering. Consult the M-Class DSO Instrument Manual for complete details on advanced triggering options.

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setTrigBEnable<br>getTrigBEnable | bo<br>bi | 0 or 1 | Trigger B enable:<br>0 = disable<br>1 = enable |
| setTrigBSource<br>getTrigBSource | mbbo<br>mbbi | Discrete | Trigger B source:<br>“TTLT0” = TTLTRG0 (external header)<br>“TTLT1” = TTLTRG1 (external header)<br>“TTLT2” = TTLTRG2 (external header)<br>“TTLT3” = TTLTRG3 (external header)<br>“TTLT4” = TTLTRG4 (external header)<br>“TTLT5” = TTLTRG5 (external header)<br>“TTLT6” = TTLTRG6 (external header)<br>“TTLT7” = TTLTRG7 (external header)<br>“INP1” = input channel 1<br>“INP2” = input channel 2<br>“INP3” = input channel 3 (ZT4xx2)<br>“INP4” = input channel 4 (ZT4xx2)<br>“EXT” = external input<br>“MAN” = software only<br>“PATT” = pattern |
| setTrigBSlope<br>getTrigBSlope | mbbo<br>mbbi | Discrete | Trigger slope or polarity:<br>“NEG” = negative<br>“POS” = positive |
| setTrigBHoldoff<br>getTrigBHoldoff | ao<br>ai | 0 to 100 | Holdoff time in seconds to wait before detecting trigger B |
| setTrigBCount<br>getTrigBCount | longout<br>longin | 1 to 65535 | Trigger B event count to qualify trigger B |
| setTrigPWLower<br>getTrigPWLower | ao<br>ai | 10e-9 to 500e-3 | Lower limit for pulse width triggering time in seconds:<br>10 ns to 500 ms, 5 ns resolution |
| setTrigPWUpper<br>getTrigPWUpper | ao<br>ai | 10e-9 to 500e-3 | Upper limit for pulse width triggering time in seconds:<br>10 ns to 500 ms, 5 ns resolution |
| setTrigGlitchLimit<br>getTrigGlitchLimit | ao<br>ai | 10e-9 to 500e-3 | Limit for glitch triggering time in seconds:<br>10 ns to 500 ms, 5 ns resolution |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setTrigPattMask<br>getTrigPattMask | longout<br>longin | $0000_{16}$ to $1FFF_{16}$ | Pattern trigger mask for defining pattern match criteria (0 = ignore, 1 = match required), used in conjunction with set/getTrigPattTruth:<br>Bit 0 = input channel 1<br>Bit 1 = input channel 2<br>Bit 2 = input channel 3 (ZT4xx2)<br>Bit 3 = input channel 4 (ZT4xx2)<br>Bit 4 = external input<br>Bit 5 = TTLTRG0 (external header)<br>Bit 6 = TTLTRG1 (external header)<br>Bit 7 = TTLTRG2 (external header)<br>Bit 8 = TTLTRG3 (external header)<br>Bit 9 = TTLTRG4 (external header)<br>Bit 10 = TTLTRG5 (external header)<br>Bit 11 = TTLTRG6 (external header)<br>Bit 12 = TTLTRG7 (external header) |
| setTrigPattTruth<br>getTrigPattTruth | longout<br>longin | $0000_{16}$ to $1FFF_{16}$ | Pattern trigger truth for defining pattern match criteria (0 = low, 1 = high), used in conjunction with set/getTrigPattMask:<br>Bit 0 = input channel 1<br>Bit 1 = input channel 2<br>Bit 2 = input channel 3 (ZT4xx2)<br>Bit 3 = input channel 4 (ZT4xx2)<br>Bit 4 = external input<br>Bit 5 = TTLTRG0 (external header)<br>Bit 6 = TTLTRG1 (external header)<br>Bit 7 = TTLTRG2 (external header)<br>Bit 8 = TTLTRG3 (external header)<br>Bit 9 = TTLTRG4 (external header)<br>Bit 10 = TTLTRG5 (external header)<br>Bit 11 = TTLTRG6 (external header)<br>Bit 12 = TTLTRG7 (external header) |
| setTrigVidStan<br>getTrigVidStan | mbbo<br>mbbi | Discrete | Video trigger standard:<br>“PAL” = PAL<br>“NTSC” = NTSC<br>“SEC” = SECAM |
| setTrigVidField<br>getTrigVidField | longout<br>longin | 1 or 2 | Video trigger field:<br>1 = field 1<br>2 = field 2 |
| setTrigVidLine<br>getTrigVidLine | longout<br>longin | 1 to 625 | Video trigger line, depends upon TrigVidStan and TrigVidField:<br>For NTSC:<br>field 1, line = 1 to 263,<br>field 2, line = 1 to 262<br>For PAL or SECAM;<br>field 1, line = 1 to 313<br>filed 2, line = 314 to 625 |



Downloaded from www.Manualslib.com manuals search engine

## Arm Process Variables

The following table lists the PVs for the arm settings used to configure the arm-qualified trigger for the waveform acquisition synchronization. When using the input channels or external input as the arm source, the arm threshold level is set using the corresponding trigger PV.

| Process Variable | Type | Values | Description |
|---|---|---|---|
| ArmState | bi | 0 or 1 | The instrument arm state. |
| setArmSource<br>getArmSource | mbbo<br>mbbi | Discrete | Arm source:<br>“IMM” = immediate (bypass arm)<br>“TTLT0” = TTLTRG0 (external header)<br>“TTLT1” = TTLTRG1 (external header)<br>“TTLT2” = TTLTRG2 (external header)<br>“TTLT3” = TTLTRG3 (external header)<br>“TTLT4” = TTLTRG4 (external header)<br>“TTLT5” = TTLTRG5 (external header)<br>“TTLT6” = TTLTRG6 (external header)<br>“TTLT7” = TTLTRG7 (external header)<br>“INP1” = input channel 1<br>“INP2” = input channel 2<br>“INP3” = input channel 3 (ZT4xx2)<br>“INP4” = input channel 4 (ZT4xx2)<br>“EXT” = external input<br>“MAN” = software only<br>“PATT” = pattern |
| setArmPolarity<br>getArmPolarity | mbbo<br>mbbi | Discrete | Trigger slope or polarity:<br>“NEG” = negative<br>“POS” = positive |



Downloaded from www.Manualslib.com manuals search engine

## Output Process Variables

The following table lists the PVs for the output settings used to configure the signal outputs. These outputs include the external output on the front panel and TTLTRG*N* outputs on the external trigger header. The trigger output PVs are duplicated for each TTLTRG*N* output. In the following table, *N* is substituted with the appropriate trigger number (for example: OutTrig1Enable).

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setOutExtEnable<br>getOutExtEnable | bo<br>bi | 0 or 1 | External output enable:<br>0 = disable<br>1 = enable |
| setOutExtPolarity<br>getOutExtPolarity | mbbo<br>mbbi | Discrete | Output level or pulse polarity:<br>“NEG” = negative<br>“POS” = positive |
| setOutExtSource<br>getOutExtSource | mbbo<br>mbbi | Discrete | External output source:<br>“ARM” = arm event<br>“TRIG” = trigger complete event<br>“ATR” = trigger A event<br>“BTR” = trigger B event<br>“CAPT” = capture complete event<br>“OPC” = operation complete event<br>“CONS” = constant state<br>“PULS” = programmable pulse<br>“REF” = 10 MHz reference<br>“CLOC” = programmable clock<br>“LIM” = passed limit test<br>“MSS” = master summary status event<br>“PROB” = probe compensation output |
| setOutExtPulsePer<br>getOutExtPulsePer | ao<br>ai | 26.67e-9 to 100 | External output programmable pulse or clock period in seconds:<br>26.667 ns to 100 seconds |
| setOutLXI*N*Mode<br>getOutLXI*N*Mode | mbbo<br>mbbi | Discrete | Sets or queries the unit LXI output mode for the given line, *N*.<br>“WIR” = Wired Or<br>“DRIV” = Driven<br>“BIAS” = Bias |
| setOutputEventTime<br>getOutputEventTime | ao<br>ai | 50e-9 to 163e-3 | External output programmable event driven pulse width in seconds: 50 ns to 163 ms |
| setOutTrig*N*Enable<br>getOutTrig*N*Enable | bo<br>bi | 0 or 1 | Output enable for selected trigger:<br>0 = disable<br>1 = enable |
| setOutTrig*N*Polarity<br>getOutTrig*N*Polarity | mbbo<br>mbbi | Discrete | Output level or pulse polarity:<br>“NEG” = negative<br>“POS” = positive |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setOutTrig*N*Source<br>getOutTrig*N*Source | mbbo<br>mbbi | Discrete | Output source for selected trigger:<br>“ARM = arm event<br>“TRIG” = trigger complete event<br>“ATR” = trigger A event<br>“BTR” = trigger B event<br>“CAPT” = capture complete event<br>“OPC” = operation complete event<br>“CONS” = constant state<br>“MSS” = master summary status event |

## Calculate Process Variables

The following table lists the PVs for the calculate channel settings used to configure the waveform calculations or math. These PVs are duplicated for each calculate channel. In the following table, *N* is substituted with the appropriate channel number from 1 to 4 (for example: Calc1Enable). Range and offset variables need to be updated after a capture.

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setCalc*N*Enable<br>getCalc*N*Enable | bo<br>bi | 0 or 1 | Channel enable for selected calculate channel:<br>0 = disable<br>1 = enable |
| Calc*N*Imm | bo | 0 or 1 | Causes a calculation channel to immediately update:<br>0 = disable<br>1 = enable |
| setCalc*N*Offset<br>getCalc*N*Offset | ao<br>ai | -inf to +inf | Offset in Volts for selected calculate channel, Calculate offset is set automatically when the calculate operation is first selected |
| setCalc*N*Op<br>getCalc*N*Op | mbbo<br>mbbi | Discrete | Operation for selected calculate channel:<br>“ADD” = add<br>“AVAL” = absolute value<br>“COPY” = copy<br>“DER” = derivative<br>“INT” = integrate<br>“INV” = invert<br>“MULT” = multiply<br>“SUBT” = subtract<br>“LIM” = limit test<br>“FTR” = FFT frequency transform<br>“TTR” = time transform(smooth)<br>“MEAS” = measurement statistics<br>“HIST” = histogram<br>“NOOP” = no operation; static channel |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setCalc*N*Pos<br>getCalc*N*Pos | ao<br>ai | 0.0 to 1.0 | Sets or queries the relative DC voltage offset for the Calculate Channel represented at the vertical zero for the selected channel. Default voltage position is 0.5 (50%).<br><br>Changing a Calculate Channel's position resets the calculation. Processing Calc*N*Imm will recalculate using the new offset. |
| setCalc*N*Range<br>getCalc*N*Range | ao<br>ai | -inf to +inf | Range in Vpp for selected calculate channel, Calculate range is set automatically when the calculate operation is first selected |
| setCalc*N*Source1<br>getCalc*N*Source1 | mbbo<br>mbbi | Discrete | Source 1 for selected calculate channel:<br>"INP1" = input channel 1<br>"INP2" = input channel 2<br>"INP3" = input channel 3 (4 channel only)<br>"INP4" = input channel 4 (4 channel only)<br>"CALC1" = calculate channel 1<br>"CALC2" = calculate channel 2<br>"CALC3" = calculate channel 3<br>"CALC4" = calculate channel 4<br>"REF1" = reference channel 1<br>"REF2" = reference channel 2<br>"REF3" = reference channel 3<br>"REF4" = reference channel 4 |
| setCalc*N*Source2<br>getCalc*N*Source2 | mbbo<br>mbbi | Discrete | Source 2 for selected calculate channel,<br>Only required for add, subtract, and multiply calculate operations:<br>"INP1" = input channel 1<br>"INP2" = input channel 2<br>"INP3" = input channel 3 (ZT4xx2)<br>"INP4" = input channel 4 (ZT4xx2)<br>"CALC1" = calculate channel 1<br>"CALC2" = calculate channel 2<br>"CALC3" = calculate channel 3<br>"CALC4" = calculate channel 4<br>"REF1" = reference channel 1<br>"REF2" = reference channel 2<br>"REF3" = reference channel 3<br>"REF4" = reference channel 4 |
| Calc*N*Upload | aao | Array | Upload a signed short integer waveform data array in codes into the selected reference channel. |
| Calc*N*ScaledUpload | aao | Array | Upload a signed short integer waveform data array in Volts into the selected reference channel. |



Downloaded from www.Manualslib.com manuals search engine

## Advanced Calculate Process Variables

The following table lists the PVs for the advanced calculate channel settings used to configure specific calculations or math. These PVs are duplicated for each calculate channel. In the following table, *N* is substituted with the appropriate channel number from 1 to 4 (for example: Calc1FFTWin).

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setCalc*N*FFTWin<br>getCalc*N*FFTWin | mbbo<br>mbbi | Discrete | FFT window for selected calculate channel:<br>"RECT" = rectangular<br>"HAMM" = hamming<br>"HANN" = hanning<br>"BLAC" = blackman<br>"FLAT" = flattop |
| setCalc*N*FFTFormat<br>getCalc*N*FFTFormat | mbbo<br>mbbi | Discrete | FFT format for selected calculate channel:<br>"MLIN" = magnitude linear<br>"MLOG" = magnitude logarithmic<br>"PHAS" = phase<br>"REAL" = real<br>"IMAG" = imaginary |
| setCalc*N*FiltPoints<br>getCalc*N*FiltPoints | longout<br>longin | 2 to 40 | Digital lowpass IIR filter smoothing points for selected calculate channel |
| Calc*N*LimitFail | bi | 0 or 1 | Calculation limit test results:<br>0 = no failures<br>1 = a failure has occurred |
| setCalc*N*LimitMeas<br>getCalc*N*LimitMeas | stringout<br>stringin | string | The measurement to check during a limit test.<br>"MASK" = Mask Test<br>"AC" = AC RMS<br>"AMPL" = Amplitude<br>"AVER" = Average<br>"CAV" = Cycle Average<br>"CFR" = Cycle Frequency<br>"CPER" = Cycle Period<br>"CRMS" = Cycle RMS<br>"DC" = DC RMS<br>"ENOB" = Effective number of bits (FFT)<br>"FEDG" = Number of Falling Edges<br>"FOV" = Falling edge overshoot<br>"FPR" = Falling edge preshoot<br>"FREQ" = Frequency<br>"FTCR" = Falling edge crossing time<br>"FTIM" = Fall time<br>"HIGH" = High<br>"LOW" = Low<br>"MAX" = Maximum<br>"MIN" = Minimum<br>"MID" = Mid<br>"NDUT" = Negative duty cycle<br>"NWID" = Negative pulse width<br>"PDUT" = Positive duty cycle |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| | | | "PER" = Period<br>"PHAS" = Phase<br>"PTP" = Peak-to-peak<br>"PWID" = Positive width<br>"REDG" = Number of Rising Edges<br>"ROV" = Rising edge overshoot<br>"RPR" = Rising edge preshoot<br>"RTCR" = Rising edge crossing time<br>"RTIM" = Rise time<br>"SDEV" = Standard deviation<br>"SFDR" = Spurious-free dynamic range (FFT)<br>"SNDR" = Signal-to-noise+distortion (FFT)<br>"SNR" = Signal-to-noise (FFT)<br>"THD" = Total harmonic distortion (FFT)<br>"TMAX" = Time of maximum<br>"TMIN" = Time of minimum |
| setCalc*N*LimitLower<br>getCalc*N*LimitLower | ao<br>ai | -inf to +inf | The lower boundary for limit tests |
| setCalc*N*LimitUpper<br>getCalc*N*LimitUpper | ao<br>ai | -inf to +inf | The upper boundary for limit tests |
| setCalcLimitNumber<br>getCalcLimitNumber | longout<br>longin | 0 to 2147483647 | Sets or gets the number of limit test failures until limit test aborts. 0 runs the limit test until aborted, n stops the limit test upon the $n^{th}$ failure. |
| Calc*N*LimitClear | bo | 0 or 1 | Clears limit test statistics; this will also reset the count for limit number abort. A transition from 0 to 1 causes the clear, after which the value returns to 0 |
| setCalc*N*MaskLower<br>getCalc*N*MaskLower | mbbo<br>mbbi | Discrete | The lower boundary for mask tests:<br>"CALC1" = calculate channel 1<br>"CALC2" = calculate channel 2<br>"CALC3" = calculate channel 3<br>"CALC4" = calculate channel 4<br>"REF1" = reference channel 1<br>"REF2" = reference channel 2<br>"REF3" = reference channel 3<br>"REF4" = reference channel 4 |
| setCalc*N*MaskUpper<br>getCalc*N*MaskUpper | mbbo<br>mbbi | Discrete | The upper boundary for mask tests:<br>"CALC1" = calculate channel 1<br>"CALC2" = calculate channel 2<br>"CALC3" = calculate channel 3<br>"CALC4" = calculate channel 4<br>"REF1" = reference channel 1<br>"REF2" = reference channel 2<br>"REF3" = reference channel 3<br>"REF4" = reference channel 4 |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| Calc*N*MaskGen | bo | 0 or 1 | Generates reference waveforms for mask testing based on calculation source. |
| setCalcMaskVertOffset<br>getCalcMaskVertOffset | ao<br>ai | 0 to full scale | Sets the vertical offset in volts used in generate reference masks |
| setCalcMaskHorzOffset<br>getCalcMaskHorzOffset | ao<br>ai | 0 to sweep time | Sets the horizontal offset in seconds used in generate reference masks |
| setCalc*N*MeasType<br>getCalc*N*MeasType | stringout<br>stringin | string | The type of measurement to store in a calculation channel:<br>“AC” = AC RMS<br>“AMPL” = Amplitude<br>“AVER” = Average<br>“CAV” = Cycle average<br>“CFR” = Cycle frequency<br>“CPER” = Cycle period<br>“CRMS” = Cycle RMS<br>“DC” = DC RMS<br>“ENOB” = Effective number of bits (FFT)<br>“FEDG” = Number of Falling Edges<br>“FOV” = Falling edge overshoot<br>“FPR” = Falling edge preshoot<br>“FREQ” = Frequency<br>“FTCR” = Falling edge crossing time<br>“FTIM” = Fall time<br>“HIGH” = High<br>“LOW” = Low<br>“MAX” = Maximum<br>“MIN” = Minimum<br>“MID” = Mid<br>“NDUT” = Negative duty cycle<br>“NWID” = Negative pulse width<br>“PDUT” = Positive duty cycle<br>“PER” = Period<br>“PHAS” = Phase<br>“PTP” = Peak-to-peak<br>“PWID” = Positive width<br>“REDG” = Number of Rising Edges<br>“ROV” = Rising edge overshoot<br>“RPR” = Rising edge preshoot<br>“RTCR” = Rising edge crossing time<br>“RTIM” = Rise time<br>“SDEV” = Standard deviation<br>“SFDR” = Spurious-free dynamic range (FFT)<br>“SNDR” = Signal-to-noise+distortion (FFT)<br>“SNR” = Signal-to-noise (FFT only)<br>“THD” = Total harmonic distortion (FFT)<br>“TMAX” = Time of maximum<br>“TMIN” = Time of minimum |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| Calc*N*MeasClear | bo | 0 or 1 | Clears measurement history. A transition from 0 to 1 causes the clear, after which the value returns to 0 |
| setCalc*N*GateTimeStart<br>getCalc*N*GateTimeStart | ao<br>ai | 0 to Maximum waveform length | Sets or gets the start time for the mask test gate, if setting the gates by time. |
| setCalc*N*GateTimeStop<br>getCalc*N*GateTimeStop | ao<br>ai | 0 to Maximum waveform length | Sets or gets the stop time for the mask test gate, if setting the gates by time. |
| setCalc*N*GateFreqStart<br>getCalc*N*GateFreqStart | ao<br>ai | 0 to Maximum frequency | Sets or gets the start frequency for the mask test gate, if setting the gates by frequency. |
| setCalc*N*GateFreqStop<br>getCalc*N*GateFreqStop | ao<br>ai | 0 to Maximum frequency | Sets or gets the stop frequency for the mask test gate, if setting the gates by frequency. |
| setCalc*N*GatePointStart<br>getCalc*N*GatePointStart | ao<br>ai | 0 to Maximum waveform points | Sets or gets the start point for the mask test gate, if setting the gates by number of samples. |
| setCalc*N*GatePointStop<br>getCalc*N*GatePointStop | ao<br>ai | 0 to Maximum waveform points | Sets or gets the stop point for the mask test gate, if setting the gates by number of samples. |
| Calc*N*GateCursor | bo | 0 or 1 | Sets the mask test gate to the current cursor time values. A transition from 0 to 1 causes the set, after which the value returns to 0. |
| Calc*N*LimRepMax | ai | -inf to +inf | Gets the maximum value of the waveform reported from the calculation limit test. |
| Calc*N*LimRepMin | ai | -inf to +inf | Gets the minimum value of the waveform reported from the calculation limit test. |
| Calc*N*LimRepAvg | ai | -inf to +inf | Gets the average value of the waveform reported from the calculation limit test. |
| Calc*N*LimRepSdev | ai | -inf to +inf | Gets the standard deviation of the waveform reported from the calculation limit test. |
| Calc*N*LimRepLast | ai | -inf to +inf | Gets the last value of the waveform reported from the calculation limit test. |
| Calc*N*LimRepCount | longin | 0 to 2147483647 | Gets the number of waveforms captured in the calculation limit test. |
| Calc*N*LimRepFail | longin | 0 to 2147483647 | Gets the number of waveform capture failures in the calculation limit test. |



Downloaded from www.Manualslib.com manuals search engine

## Reference Process Variables

The following table lists the PVs for the reference channel settings used to configure the waveform storage channels. These PVs are duplicated for each reference channel. In the following table, *N* is substituted with the appropriate channel number from 1 to 4 (for example: Ref1Enable).

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setRef*N*Enable<br>getRef*N*Enable | bo<br>bi | 0 or 1 | Channel enable for selected reference channel:<br>0 = disable<br>1 = enable |
| setRef*N*Range<br>getRef*N*Range | ao<br>ai | -inf to +inf | Range in Vpp for selected reference channel,<br>Reference range is set automatically when the reference waveform is stored |
| setRef*N*Offset<br>getRef*N*Offset | ao<br>ai | -inf to +inf | Offset in Volts for selected reference channel,<br>Reference offset is set automatically when the reference waveform is stored |
| setRef*N*Pos<br>getRef*N*Pos | ao<br>ai | 0.0 to 1.0 | Offset as a fraction of full scale for selected reference channel,<br>Reference offset is set automatically when the reference waveform is stored |
| Ref*N*Store | mbbo | Discrete | Store a waveform from the selected channel to the reference channel. Source channels:<br>“INP1” = input channel 1<br>“INP2” = input channel 2<br>“INP3” = input channel 3 (ZT4xx2)<br>“INP4” = input channel 4 (ZT4xx2)<br>“CALC1” = calculate channel 1<br>“CALC2” = calculate channel 2<br>“CALC3” = calculate channel 3<br>“CALC4” = calculate channel 4<br>“REF1” = reference channel 1<br>“REF2” = reference channel 2<br>“REF3” = reference channel 3<br>“REF4” = reference channel 4 |
| Ref*N*Upload | aao | Array | Upload a signed short integer waveform data array in codes into the selected reference channel. |
| Ref*N*ScaledUpload | aao | Array | Upload a floating point scaled waveform data array in Volts into the selected scaled reference channel. |



Downloaded from www.Manualslib.com manuals search engine

## Measure Process Variables

The following table lists the PVs for the Measure functionality. Up to 32 measurements are performed automatically after each waveform acquisition, and the results updated in the corresponding measurement PVs. In the following table, *N* is substituted with the appropriate measurement number from 1 to 32 (for example: Meas1Enable), except where stated otherwise. The corresponding measurements can also be viewed through ZScope™ or the ztscopeM drivers as four lists of eight measurements.

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setMeasNMax<br>getMeasNMax | longout<br>longin | 1 to 100 | Sets or gets the maximum to use for measurements. There are 100 maxima possible counted in order of occurrence in time. Default is the first maxima. NMax is used maximum and time of maximum measurements. |
| setMeasGateTimeStart<br>getMeasGateTimeStart | ao<br>ai | 0 to Maximum waveform length | Sets or gets the start measurement gate in seconds, if setting the gates by time. |
| setMeasGateTimeStop<br>getMeasGateTimeStop | ao<br>ai | 0 to Maximum waveform length | Sets or gets the start measurement gate in seconds, if setting the gates by time. |
| setMeasGateFreqStart<br>getMeasGateFreqStart | ao<br>ai | 0 to ½ of the current Sample Frequency | Sets or gets the start measurement gate in Hertz, if setting the gates by frequency. |
| setMeasGateFreqStop<br>getMeasGateFreqStop | ao<br>ai | 0 to ½ of the current Sample Frequency | Sets or gets the stop measurement gate in Hertz, if setting the gates by frequency. |
| setMeasGatePointStart<br>getMeasGatePointStart | longout<br>longin | 0 to Maximum waveform points | Sets or gets the start measurement gate in points, if setting the gates by the number of samples. |
| setMeasGatePointStop<br>getMeasGatePointStop | longout<br>longin | 0 to Maximum waveform points | Sets or gets the stop measurement gate in points, if setting the gates by the number of samples. |
| setMeasCurs*N*X<br>getMeasCurs*N*X | ao<br>ai | -Inf to +Inf | Sets or queries the selected cursor's position on the x axis. Units depend upon the cursor source waveform type. For normal time based waveforms, the x axis is time. For FFTs the x axis is frequency. |
| setMeasCurs*N*Y<br>getMeasCurs*N*Y | ao<br>ai | -Inf to +Inf | Sets or queries the selected cursor's position on the y axis. Units depend upon the cursor source waveform type. For normal time based waveforms, the y axis is voltage. |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| MeasCursXDelta | ai | -Inf to +Inf | Queries the difference between the cursor positions on the x axis. Units depend upon the cursor source waveform type. For normal time based waveforms, the x axis is time. For FFTs the x axis is frequency. |
| MeasCursYDelta | ai | -Inf to +Inf | Queries the difference between the cursor positions on the y axis. Units depend upon the cursor source waveform type. For normal time based waveforms, the y axis is voltage. |
| setMeasErrState<br>getMeasErrState | bo<br>bi | 0 or 1 | Determines if the measurement errors are returned. If enabled, measurement errors are returned at 9.99e36. If disabled, measurement errors are returned as the status byte.<br>0 = Off<br>1 = On |
| setMeasList*N*Enable<br>getMeasList*N*Enable | bo<br>bi | 0 or 1 | Sets or queries the state of a list of measurements. A disabled measurement list retains its settings. *N* is 1 to 4. |
| MeasList*N*Clear | bo | 0 or 1 | Set to clear all eight measurements from the measurement list (*N* is 1 -4). Idle value is 0. After setting to 1, value returns to idle. |
| Meas*N*Clear | bo | 0 or 1 | Set to clear one measurement (*N* is 1-32). Idle value is 0. After setting to 1, value returns to idle. |
| setMeas*N*Source<br>getMeas*N*Source | mbbo<br>mbbi | Discrete | Source for selected measurement number(*N* is 1 - 32):<br>“INP1” = input channel 1<br>“INP2” = input channel 2<br>“INP3” = input channel 3 (ZT4xx2)<br>“INP4” = input channel 4 (ZT4xx2)<br>“CALC1” = calculate channel 1<br>“CALC2” = calculate channel 2<br>“CALC3” = calculate channel 3<br>“CALC4” = calculate channel 4<br>“REF1” = reference channel 1<br>“REF2” = reference channel 2<br>“REF3” = reference channel 3<br>“REF4” = reference channel 4 |
| setMeas*N*Type<br>getMeas*N*Type | stringout<br>stringin | string | Type for selected measurement number (*N* is 1 - 32):<br>“NONE” = none selected<br>“AC” = AC RMS<br>“AMPL” = Amplitude<br>“AVER” = Average<br>“CAV” = Cycle average<br>“CFR” = Cycle frequency<br>“CPER” = Cycle period<br>“CRMS” = Cycle RMS<br>“DC” = DC RMS |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| | | | "ENOB" = Effective number of bits (FFT)<br>"FEDG" = Number of Falling Edges<br>"FOV" = Falling edge overshoot<br>"FPR" = Falling edge preshoot<br>"FREQ" = Frequency<br>"FTCR" = Falling edge crossing time<br>"FTIM" = Fall time<br>"HIGH" = High<br>"LOW" = Low<br>"MAX" = Maximum<br>"MIN" = Minimum<br>"MID" = Mid<br>"NDUT" = Negative duty cycle<br>"NWID" = Negative pulse width<br>"PDUT" = Positive duty cycle<br>"PER" = Period<br>"PHAS" = Phase<br>"PTP" = Peak-to-peak<br>"PWID" = Positive width<br>"REDG" = Number of Rising Edges<br>"ROV" = Rising edge overshoot<br>"RPR" = Rising edge preshoot<br>"RTCR" = Rising edge crossing time<br>"RTIM" = Rise time<br>"SDEV" = Standard deviation<br>"SFDR" = Spurious-free dynamic range (FFT)<br>"SNDR" = Signal-to-noise+distortion (FFT)<br>"SNR" = Signal-to-noise (FFT)<br>"THD" = Total harmonic distortion (FFT)<br>"TMAX" = Time of maximum<br>"TMIN" = Time of minimum |
| Meas*N*Result | ai | -Inf to +Inf | Measurement result in appropriate units |
| Meas*N*Status | longin | $0000_{16}$ to $0F03_{16}$ | Status of selected measurement number:<br>$0000_{16}$ = measurement valid<br>Bit 0 = invalid waveform<br>Bit 1 = edge not found<br>Bit 8 = over-voltage maximum<br>Bit 9 = over-voltage minimum<br>Bit 10 = low signal level<br>Bit 11 = measurement command invalid |



Downloaded from www.Manualslib.com manuals search engine

## Operate Process Variables

The following table lists the PVs for the instrument operation involving the controls for waveform acquisition.

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setOpInitCont<br>getOpInitCont | bo<br>bi | 0 or 1 | The instrument's continuous acquisition mode:<br>0 = single capture<br>1 = continuous capture |
| OpInitiate | bo | 0 or 1 | Initiate waveform acquisition. |
| OpAbort | bo | 0 or 1 | Aborts waveform acquisition. |
| OpForceCap | bo | 0 or 1 | Force waveform capture (all triggers):<br>0 = no operation<br>1 = force capture<br>A transition from 0 to 1 causes the waveform acquisition to complete, after which the value returns to 0 |
| OpAutoLoad | bo | 0 or 1 | Download new waveforms during continuous acquisition.<br>0 = no operation<br>1 = download<br>A transition from 0 to 1 causes the download, after which the value returns to 0 |
| OpArm | bo | 0 or 1 | Manual arm for software arm source:<br>0 = disarm<br>1 = arm<br>This only applies when the arm source is set to software |
| OpAutoSetup | bo | 0 or 1 | Automatically setup range, offset, trigger level for enabled channels:<br>0 = no operation<br>1 = force auto setup<br>A transition from 0 to 1 causes the auto setup, after which the value returns to 0 |
| OpCompleteEnable | bo | 0 or 1 | Enable the operation complete flag to be set when all pending operations have completed.<br>0 = operation complete register is disabled<br>1 = operation complete register is enabled until pending operations complete, then is disabled again.<br>This enables the operation complete bit, which can be read in Status as well as OPC driven outputs. |
| getOpComplete | bi | 0 or 1 | Check to see if all pending operations have completed.<br>0 = all pending operations have not completed<br>1 = all pending operations have completed. |



Downloaded from www.Manualslib.com manuals search engine

## Waveform Process Variables

The following table lists the PVs for the acquired waveforms. In the following table, *N* is substituted with the appropriate channel number (for example: Inp1Wave). The ZT4xx1 has two input channels and the ZT4xx2 has four input channels. There are four calculate channels and four reference channels. Waveform PVs can only be read (downloaded) from the instrument.

The size of the waveform allowed to be transmitted via channel access is limited by EPICS_CA_MAX_ARRAY_BYTES and by the PVs number of elements (NELM). Note the maximum array bytes setting is in bytes, but the waveform PVs in EPICS use points to define their size. Make sure each waveform PV's values fit the following:

If FTVL is LONG then: (NELM x 4) ≤ EPICS_CA_MAX_ARRAY_BYTES

If FTVL is DOUBLE then: (NELM x 8) ≤ EPICS_CA_MAX_ARRAY_BYTES

Scaled Wave and Time PVs are doubles, so they require 8 bytes per point. Wave PVs are long integers, so they require 4 bytes per point. The NELM size is in points, so it should be the same for all waveforms regardless of FTVL. This can be changed in the PV file: set the maximum values as required and then upload the PV file to the instrument. Default values are 1000 points for all waveform PVs, and the default EPICS_CA_MAX_ARRAY_BYTES is 16384.

If NELM is less than the number of points captured by the instrument, a valid decimated waveform will be returned. Decimation occurs in steps of 1, 2 and 5. Example: if Inp1Wave has a NELM of 1000 and the instrument captures 1,500 points, every other point will be returned for a total of 750 points. The remaining 250 points will be zeroes. Note continuous acquisition mode waveforms are always decimated to 1000 points.

| Process Variable | Type | Values | Description |
|---|---|---|---|
| Inp*N*WavePoints | longin | 10 to 65535 | The number of actual data points stored in Inp*N*Wave, Inp*N*ScaledWave and InpScaledTime. |
| Inp*N*Wave | waveform | -2147483648 to 2147483647 | Signed short integer waveform data in codes for selected input channel |
| Inp*N*ScaledWave | waveform | -inf to +inf | Floating point scaled waveform data in Volts for selected input channel |
| InpScaledTime | waveform | 0 to 100 | Floating point scaled time (X-axis) waveform data in seconds for all input channel |
| Inp*N*Timestamp | ai | 0.0 to 0.9999999 | Relative timestamp in seconds for acquired waveform, 1 second wrap period, 100 ns resolution |
| Calc*N*WavePoints | longin | 10 to 65535 | The number of actual data points stored in Calc*N*Wave, Calc*N*ScaledWave and Calc*N*ScaledTime. |
| Calc*N*Wave | waveform | -2147483648 to 2147483647 | Signed short integer waveform data in codes for selected calculate channel |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| Calc*N*ScaledWave | waveform | -inf to +inf | Floating point scaled waveform data in Volts for selected calculate channel |
| Calc*N*ScaledTime | waveform | -inf to +inf | Floating point scaled time (X-axis) waveform data for selected calculate channel |
| Ref*N*WavePoints | longin | 10 to 32768 | The number of actual data points stored in Ref*N*Wave, Ref*N*ScaledWave and Ref*N*ScaledTime. |
| Ref*N*Wave | waveform | -2147483648 to 2147483647 | Signed short integer waveform data in codes for selected reference channel |
| Ref*N*ScaledWave | waveform | -inf to +inf | Floating point scaled waveform data in Volts for selected reference channel |
| Ref*N*ScaledTime | waveform | -inf to +inf | Floating point scaled time (X-axis) waveform data for selected reference channel |

## Utility Process Variables

The following table lists the PVs for the miscellaneous instrument utilities. For the save and recall operations in the following table, *N* is substituted with the appropriate state number from 1 to 30 (for example: UtilSave1).

| Process Variable | Type | Values | Description |
|---|---|---|---|
| UtilID | stringin | ID string | Identification string of following format: "ZTEC,model_number,serial_number,version" |
| UtilClearStatus | bo | 0 or 1 | Clears all event registers, the request for OPC flag, and all status queues (except the response queue).<br>0 = no operation<br>1 = reset<br>A transition from 0 to 1 causes the instrument reset, after which the value returns to 0. |
| UtilReset | bo | 0 or 1 | Instrument reset:<br>0 = no operation<br>1 = reset<br>A transition from 0 to 1 causes the instrument reset, after which the value returns to 0. |
| UtilSave | longout | 1 to 30 | Instrument state save:<br>Causes the instrument state to be saved to one of thirty one locations in non-volatile memory. |
| UtilRecall | longout | 1 to 30 | Instrument state recall:<br>Causes the instrument state to be recalled from one of thirty one locations in non-volatile memory. |
| UtilErrCount | longin | 0 to 32 | Number of errors in error log,<br>0 = no errors |
| UtilErrNext | longin | 0 to -32767 | Current error code, processing clears current error and retrieves next error. |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| UtilErrReport | stringin | command string | Returns last command to cause an error.<br><br>**Note:** This functionality is only valid for the ZT44xx |
| UtilErr | stringin | error string | Error string explaining current error code |
| UtilTemp | ai | 0.0 to 100.0 | Internal temperature in degrees C |
| UtilMem | longin | Varies by model, see instrument manual | Waveform capture memory in samples per channel, varies by installed memory option |
| UtilMemClear | bo | 0 or 1 | Clears all non-volatile system memory. A transition from 0 to 1 causes the clear, after which the value returns to 0. |
| UtilSelfCal | longin | 0,1 or 2 | Initiates and returns the result of the unit self-calibration process. The internal calibration determines the zero DC offset, the DC offset adjust scale factor, and the ADC balance for all input range settings for all input channels. The internal calibration process can take several minutes to complete. The instrument is reset upon completion of the calibration process.<br><br>**Note:** The input channels must be disconnected or be driven with 0.0 VDC before starting the calibration.<br><br>The timeout value should be set to infinite before starting the calibration, and reset to the default value when completed.<br><br>**Note:** Do not interrupt the instrument during calibration or the calibration tables could be corrupted.<br><br>0 Pass<br>1 Fail (Did not converge)<br>2 Corrupt |
| setUtilSelfID<br>getUtilSelfID | bo<br>bi | 0 or 1 | Instrument HST LED state:<br>0 = LAN status<br>1 = blink<br>Allows instrument identification. |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| UtilSelfTest | mbbiDirect | $0000_{16}$ to $0375_{16}$ | Initiates an instrument self test and returns the test results as a 16-bit code. The self test is initiated on instrument power up.<br>Bit 0 = baseboard test failed<br>Bit 1 = unused<br>Bit 2 = ROM test failed<br>Bit 3 = unused<br>Bit 4 = reference oscillator test failed<br>Bit 5 = SDRAM test failed<br>Bit 6 = flash memory test failed<br>Bit 7 = unused<br>Bit 8 = digitizer submodule 1 test failed<br>Bit 9 = digitizer submodule 2 test failed (ZT4xx2)<br>Bit 10 -15 = unused |
| UtilTestCountBB | longin | integer | Returns the number of failure reports from last baseboard self test. 0 to 9 failures.<br><br>**Note:** This functionality is only valid for the ZT44xx |
| UtilTestCountSM1 | longin | integer | Returns the number of failure reports from last baseboard self test. 0 to 9 failures.<br><br>**Note:** This functionality is only valid for the ZT44xx |
| UtilTestCountSM2 | longin | integer | ZT4xx2 Only. Returns the number of failure reports from last submodule2 self test. 0 to 19 (ZT4610), 13(ZT4400), 10(ZT4210) failures.<br><br>**Note:** This functionality is only valid for the ZT44xx |
| UtilTestReportBB | stringin | string | Returns last baseboard command that caused a failure.<br><br>**Note:** This functionality is only valid for the ZT44xx |
| UtilTestReportSM1 | stringin | string | Returns last submodule1 command that caused a failure.<br><br>**Note:** This functionality is only valid for the ZT44xx |
| UtilTestReportSM2 | stringin | string | ZT4xx2 Only. Returns last submodule2 command that caused a failure.<br><br>**Note:** This functionality is only valid for the ZT44xx |
| UtilLockState | bi | 0 or 1 | Queries the instrument's lock state.<br>0 = unlocked, full control available through EPICS interface<br>1 = locked, EPICS interface is read-only |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| UtilSCPISend | stringout | string | Sends the entered SCPI string to the instrument. If sending a query, call UtilSCPIRecv immediately after this. To ensure a proper response from the UtilSCPIRecv, make sure there are no other queries occurring at the same time, whether through another PV's Scan Rate, or any other instrument interface. If so, UtilSCPIRecv will return 0. Note that most PVs will send a query during processing, even if they are output PVs. |
| UtilSCPIRecv | stringin | string | Receives the current instrument string output buffer. Use after sending a query via UtilSCPISend. (See additional notes there). |
| setOutCoerce<br>getOutCoerce | bo<br>bi | 0 or 1 | Selects whether set parameters update their values based on instrument coercion. |
| setRestoreState<br>getRestoreState | longout<br>longin | 0 to 31 | Sets or gets the location to load the instrument state from on initialization. 0 location restores the instrument to reset conditions. Locations can be saved using the UtilSave*N* PVs. |
| UtilRefresh | bo | 0 or 1 | Manual trigger to update all PVs. |
| UtilUndo | bo | 0 or 1 | Reverses instrument reset, auto setup or state recall events. |
| UtilAcc*N*ID | stringin | ID string | Accessory Identification string with format: "ZTEC,model_number,serial_number,version" |



Downloaded from www.Manualslib.com manuals search engine

## Status Register Variables

The following table lists the PVs associated with the various instrument status registers. For the Digitizer Test Register PVs in the following table, *N* is substituted with either 1 or 2 (for 4 channel instruments only) (for example: getStatusTestDig1Enable).

| Process Variable | Type | Values | Description |
|---|---|---|---|
| Status | mbbiDirect | $00_{16}$ to $FF_{16}$ | Returns the latched instrument status:<br>1 = latched event<br>Bit 0 = unused<br>Bit 1 = unused<br>Bit 2 = error log not empty<br>Bit 3 = questionable event<br>Bit 4 = message available<br>Bit 5 = standard event<br>Bit 6 = master summary status<br>Bit 7 = operation event |
| setStatusEnable<br>getStatusEnable | mbboDirect<br>mbbiDirect | $00_{16}$ to $FF_{16}$ | Status register bit enable mask<br>Bit 0 = unused<br>Bit 1 = unused<br>Bit 2 = error log not empty<br>Bit 3 = questionable event<br>Bit 4 = message available<br>Bit 5 = standard event<br>Bit 6 = master summary status<br>Bit 7 = operation event |
| StatusStandard | mbbiDirect | $00_{16}$ to $FF_{16}$ | Returns the latched standard register events:<br>1 = latched event<br>Bit 0 = operation complete<br>Bit 1 = request control<br>Bit 2 = query error<br>Bit 3 = device dependent error<br>Bit 4 = execution error<br>Bit 5 = command error<br>Bit 6 = user request<br>Bit 7 = power on |
| setStatusStandardEnable<br>getStatusStandardEnable | mbboDirect<br>mbbiDirect | $00_{16}$ to $FF_{16}$ | Standard register bit enable mask<br>Bit 0 = operation complete<br>Bit 1 = request control<br>Bit 2 = query error<br>Bit 3 = device dependent error<br>Bit 4 = execution error<br>Bit 5 = command error<br>Bit 6 = user request<br>Bit 7 = power on |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| StatusFreq | mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Frequency fault status results:<br>1 = latched event<br><br>ZT4610:<br>Bit 0 = sample clock unlocked<br>Bit 1 = unused<br>Bit 2 = memory 1-2 clock unlocked<br>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)<br>Bits 4-6 = unused<br>Bit 7 = baseboard clock unlocked<br>Bits 8-15 = unused<br><br>ZT4400:<br>Bit 0 = sample clock unlocked<br>Bit 1 = unused<br>Bit 2 = memory 1-2 clock unlocked<br>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)<br>Bit 4 = reference 1-2 clock unlocked<br>Bit 5 = reference 3-4 clock unlocked<br>Bit 6 = reference 1-2 loss of signal<br>Bit 7 = reference 3-4 loss of signal<br>Bit 8 = ADC 1 clock unlocked<br>Bit 9 = ADC 2 clock unlocked<br>Bit 10 = ADC 3 clock unlocked<br>Bit 11 = ADC 4 clock unlocked<br>Bits 12-14 = unused<br>Bit 15 = baseboard clock unlocked<br>Bits 8-15 = unused<br><br>ZT4210:<br>Bit 0 = sample clock unlocked<br>Bit 1 = unused<br>Bit 2 = memory 1-2 clock unlocked<br>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)<br>Bit 4 = trigger 1-2 clock unlocked<br>Bit 5 = trigger 3-4 clock unlocked<br>Bits 6-14 = unused<br>Bit 15 = baseboard clock unlocked |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setStatusFreqEnable<br>getStatusFreqEnable | mbboDirect<br>mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Frequency register bit enable mask:<br><br>ZT4610:<br>Bit 0 = sample clock unlocked<br>Bit 1 = unused<br>Bit 2 = memory 1-2 clock unlocked<br>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)<br>Bits 4-6 = unused<br>Bit 7 = baseboard clock unlocked<br>Bits 8-15 = unused<br><br>ZT4400:<br>Bit 0 = sample clock unlocked<br>Bit 1 = unused<br>Bit 2 = memory 1-2 clock unlocked<br>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)<br>Bit 4 = reference 1-2 clock unlocked<br>Bit 5 = reference 3-4 clock unlocked<br>Bit 6 = reference 1-2 loss of signal<br>Bit 7 = reference 3-4 loss of signal<br>Bit 8 = ADC 1 clock unlocked<br>Bit 9 = ADC 2 clock unlocked<br>Bit 10 = ADC 3 clock unlocked<br>Bit 11 = ADC 4 clock unlocked<br>Bits 12-14 = unused<br>Bit 15 = baseboard clock unlocked<br>Bits 8-15 = unused<br><br>ZT4210:<br>Bit 0 = sample clock unlocked<br>Bit 1 = unused<br>Bit 2 = memory 1-2 clock unlocked<br>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)<br>Bit 4 = trigger 1-2 clock unlocked<br>Bit 5 = trigger 3-4 clock unlocked<br>Bits 6-14 = unused<br>Bit 15 = baseboard clock unlocked |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| StatusFreqCond | mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Frequency fault status conditions:<br><br>ZT4610:<br>Bit 0 = sample clock unlocked<br>Bit 1 = unused<br>Bit 2 = memory 1-2 clock unlocked<br>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)<br>Bits 4-6 = unused<br>Bit 7 = baseboard clock unlocked<br>Bits 8-15 = unused<br><br>ZT4400:<br>Bit 0 = sample clock unlocked<br>Bit 1 = unused<br>Bit 2 = memory 1-2 clock unlocked<br>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)<br>Bit 4 = reference 1-2 clock unlocked<br>Bit 5 = reference 3-4 clock unlocked<br>Bit 6 = reference 1-2 loss of signal<br>Bit 7 = reference 3-4 loss of signal<br>Bit 8 = ADC 1 clock unlocked<br>Bit 9 = ADC 2 clock unlocked<br>Bit 10 = ADC 3 clock unlocked<br>Bit 11 = ADC 4 clock unlocked<br>Bits 12-14 = unused<br>Bit 15 = baseboard clock unlocked<br>Bits 8-15 = unused<br><br>ZT4210:<br>Bit 0 = sample clock unlocked<br>Bit 1 = unused<br>Bit 2 = memory 1-2 clock unlocked<br>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)<br>Bit 4 = trigger 1-2 clock unlocked<br>Bit 5 = trigger 3-4 clock unlocked<br>Bits 6-14 = unused<br>Bit 15 = baseboard clock unlocked |
| StatusTest | mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Test status results:<br>1 = latched event<br>Bit 0 = baseboard register test failed<br>Bit 1 = n/a<br>Bit 2 = baseboard ROM test failed<br>Bit 3 = n/a<br>Bit 4 = reference oscillator test failed<br>Bit 5 = DRAM test failed<br>Bit 6 = flash memory test failed<br>Bit 7 = unused<br>Bit 8 = digitizer 1 test failed<br>Bit 9 = digitizer 2 test failed (ZT4xx2)<br>Bits 10-15 = unused |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setStatusTestEnable<br>getStatusTestEnable | mbboDirect<br>mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Test register bit enable mask<br>Bit 0 = baseboard register test failed<br>Bit 1 = n/a<br>Bit 2 = baseboard ROM test failed<br>Bit 3 = n/a<br>Bit 4 = reference oscillator test failed<br>Bit 5 = DRAM test failed<br>Bit 6 = flash memory test failed<br>Bit 7 = unused<br>Bit 8 = digitizer 1 test failed<br>Bit 9 = digitizer 2 test failed (ZT4xx2)<br>Bits 10-15 = unused |
| StatusTestCond | mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Test condition<br>Bit 0 = baseboard register test failed<br>Bit 1 = n/a<br>Bit 2 = baseboard ROM test failed<br>Bit 3 = n/a<br>Bit 4 = reference oscillator test failed<br>Bit 5 = DRAM test failed<br>Bit 6 = flash memory test failed<br>Bit 7 = unused<br>Bit 8 = digitizer 1 test failed<br>Bit 9 = digitizer 2 test failed (ZT4xx2)<br>Bits 10-15 = unused |
| StatusOperation | mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Operation status results:<br>1 = latched event<br>Bit 0 = calibrating<br>Bit 1 = settling<br>Bit 2 = ranging<br>Bit 3 = sweeping<br>Bit 4 = measuring<br>Bit 5 = waiting for trigger<br>Bit 6 = waiting for arm<br>Bit 7 = unused<br>Bit 8 = trigger event<br>Bit 9 = data capture event<br>Bit 10 = limit test event<br>Bit 11 = auto download event<br>Bits 12-15 = unused |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setStatusOperEnable<br>getStatusOperEnable | mbboDirect<br>mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Operation register bit enable mask<br>Bit 0 = calibrating<br>Bit 1 = settling<br>Bit 2 = ranging<br>Bit 3 = sweeping<br>Bit 4 = measuring<br>Bit 5 = waiting for trigger<br>Bit 6 = waiting for arm<br>Bit 7 = unused<br>Bit 8 = trigger event<br>Bit 9 = data capture event<br>Bit 10 = limit test event<br>Bit 11 = auto download event<br>Bits 12-15 = unused |
| StatusOperCond | mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Operation condition register:<br>Bit 0 = calibrating<br>Bit 1 = settling<br>Bit 2 = ranging<br>Bit 3 = sweeping<br>Bit 4 = measuring<br>Bit 5 = waiting for trigger<br>Bit 6 = waiting for arm<br>Bit 7 = unused<br>Bit 8 = trigger event<br>Bit 9 = data capture event<br>Bit 10 = limit test event<br>Bit 11 = auto download event<br>Bits 12-15 = unused |
| StatusPreset | bo | 0 or 1 | Sets the status reporting event enable data structures to a known state. The condition and event register contents are not affected. All device-dependent status registers which cascade events into the Questionable Status and the Operation Status Registers are enabled by setting those device-dependent event enable registers to $7FFF_{16}$ (the 15 LSBs set). The IEEE-488.2 mandatory status data structures are disabled by setting the Questionable Status and Operation Status event enable registers to $0000_{16}$. The Status Byte and Standard Event Status Registers as defined by IEEE 488.2 are not affected |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| StatusQues | mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Questionable status:<br>1 = latched event<br>Bit 0 = voltage fault<br>Bits 1-4 = unused<br>Bit 5 = frequency fault<br>Bits 6-7 = unused<br>Bit 8 = calibration failure<br>Bit 9 = test failure<br>Bits 10-15 = unused |
| setStatusQuesEnable<br>getStatusQuesEnable | mbboDirect<br>mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Questionable register bit enable mask:<br>Bit 0 = voltage fault<br>Bits 1-4 = unused<br>Bit 5 = frequency fault<br>Bits 6-7 = unused<br>Bit 8 = calibration failure<br>Bit 9 = test failure<br>Bits 10-15 = unused |
| StatusQuesCond | mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Questionable condition register:<br>Bit 0 = voltage fault<br>Bits 1-4 = unused<br>Bit 5 = frequency fault<br>Bits 6-7 = unused<br>Bit 8 = calibration failure<br>Bit 9 = test failure<br>Bits 10-15 = unused |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| StatusVolt | mbbiDirect | $000_{16}$ to $FFF_{16}$ | Voltage status faults:<br>1 = latched event<br>ZT4210:<br>Bit 0 = input 1 overload<br>Bit 1 = input 2 overload<br>Bit 2 = input 3 overload (ZT4xx2)<br>Bit 3 = input 4 overload (ZT4xx2)<br>Bit 4 = input 1-2 overvoltage<br>Bit 5 = unused<br>Bit 6 = input 3-4 overvoltage (ZT4xx2)<br>Bit 7 = unused<br>Bit 8 = accessory 1 fault<br>Bit 9 = accessory 2 fault<br>Bit 10 = accessory 3 fault (ZT4xx2)<br>Bit 11 = accessory 4 fault (ZT4xx2)<br><br>ZT4400 & ZT4610<br>Bit 0 = input 1 overload<br>Bit 1 = input 2 overload<br>Bit 2 = input 3 overload (ZT4xx2)<br>Bit 3 = input 4 overload (ZT4xx2)<br>Bit 4 = input 1 overvoltage<br>Bit 5 = input 2 overvoltage<br>Bit 6 = input 3 overvoltage (ZT4xx2)<br>Bit 7 = input 4 overvoltage (ZT4xx2)<br>Bit 8 = accessory 1 fault<br>Bit 9 = accessory 2 fault<br>Bit 10 = accessory 3 fault (ZT4xx2)<br>Bit 11 = accessory 4 fault (ZT4xx2) |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setStatusVoltEnable<br>getStatusVoltEnable | mbboDirect<br>mbbiDirect | $000_{16}$ to $FFF_{16}$ | Voltage register bit enable mask<br><br>ZT4210:<br>Bit 0 = input 1 overload<br>Bit 1 = input 2 overload<br>Bit 2 = input 3 overload (ZT4xx2)<br>Bit 3 = input 4 overload (ZT4xx2)<br>Bit 4 = input 1-2 overvoltage<br>Bit 5 = unused<br>Bit 6 = input 3-4 overvoltage (ZT4xx2)<br>Bit 7 = unused<br>Bit 8 = accessory 1 fault<br>Bit 9 = accessory 2 fault<br>Bit 10 = accessory 3 fault (ZT4xx2)<br>Bit 11 = accessory 4 fault (ZT4xx2)<br><br>ZT4400 & ZT4610<br>Bit 0 = input 1 overload<br>Bit 1 = input 2 overload<br>Bit 2 = input 3 overload (ZT4xx2)<br>Bit 3 = input 4 overload (ZT4xx2)<br>Bit 4 = input 1 overvoltage<br>Bit 5 = input 2 overvoltage<br>Bit 6 = input 3 overvoltage (ZT4xx2)<br>Bit 7 = input 4 overvoltage (ZT4xx2)<br>Bit 8 = accessory 1 fault<br>Bit 9 = accessory 2 fault<br>Bit 10 = accessory 3 fault (ZT4xx2)<br>Bit 11 = accessory 4 fault (ZT4xx2) |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| StatusVoltCond | mbbiDirect | $000_{16}$ to $FFF_{16}$ | Voltage condition register:<br><br>ZT4210:<br>Bit 0 = input 1 overload<br>Bit 1 = input 2 overload<br>Bit 2 = input 3 overload (ZT4xx2)<br>Bit 3 = input 4 overload (ZT4xx2)<br>Bit 4 = input 1-2 overvoltage<br>Bit 5 = unused<br>Bit 6 = input 3-4 overvoltage (ZT4xx2)<br>Bit 7 = unused<br>Bit 8 = accessory 1 fault<br>Bit 9 = accessory 2 fault<br>Bit 10 = accessory 3 fault (ZT4xx2)<br>Bit 11 = accessory 4 fault (ZT4xx2)<br><br>ZT4400 & ZT4610<br>Bit 0 = input 1 overload<br>Bit 1 = input 2 overload<br>Bit 2 = input 3 overload (ZT4xx2)<br>Bit 3 = input 4 overload (ZT4xx2)<br>Bit 4 = input 1 overvoltage<br>Bit 5 = input 2 overvoltage<br>Bit 6 = input 3 overvoltage (ZT4xx2)<br>Bit 7 = input 4 overvoltage (ZT4xx2)<br>Bit 8 = accessory 1 fault<br>Bit 9 = accessory 2 fault<br>Bit 10 = accessory 3 fault (ZT4xx2)<br>Bit 11 = accessory 4 fault (ZT4xx2) |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| StatusCal | mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Calibration status results:<br>1 = latched event<br><br>ZT4610:<br>Bit 0 = calibration storage failed<br>Bit 1 = offset zero failed<br>Bit 2 = offset scale failed<br>Bit 3 = null balance failed<br>Bit 4 = gain balance failed<br>Bit 5 = trigger level failed<br>Bits 6-15 = unused<br><br>ZT4400:<br>Bit 0 = calibration storage failed<br>Bit 1 = offset zero failed<br>Bit 2 = offset scale failed<br>Bit 3 = unused<br>Bit 4 = gain balance failed<br>Bit 5 = unused<br>Bit 6 = trigger zero failed<br>Bit 7 = trigger gain failed<br>Bit 8 = external trigger zero failed<br>Bits 9-15 = unused<br><br>ZT4210:<br>Bit 0 = calibration storage failed<br>Bit 1 = offset zero failed<br>Bit 2 = offset scale failed<br>Bit 3 = null balance failed<br>Bit 4 = gain balance failed<br>Bit 5 = ADC failed<br>Bit 6 = trigger zero failed<br>Bit 7 = trigger gain failed<br>Bit 8 = external trigger zero failed<br>Bits 9-15 = unused |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setStatusCalEnable<br>getStatusCalEnable | mbboDirect<br>mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Calibration register bit enable mask:<br><br>ZT4610:<br>Bit 0 = calibration storage failed<br>Bit 1 = offset zero failed<br>Bit 2 = offset scale failed<br>Bit 3 = null balance failed<br>Bit 4 = gain balance failed<br>Bit 5 = trigger level failed<br>Bits 6-15 = unused<br><br>ZT4400:<br>Bit 0 = calibration storage failed<br>Bit 1 = offset zero failed<br>Bit 2 = offset scale failed<br>Bit 3 = unused<br>Bit 4 = gain balance failed<br>Bit 5 = unused<br>Bit 6 = trigger zero failed<br>Bit 7 = trigger gain failed<br>Bit 8 = external trigger zero failed<br>Bits 9-15 = unused<br><br>ZT4210:<br>Bit 0 = calibration storage failed<br>Bit 1 = offset zero failed<br>Bit 2 = offset scale failed<br>Bit 3 = null balance failed<br>Bit 4 = gain balance failed<br>Bit 5 = ADC failed<br>Bit 6 = trigger zero failed<br>Bit 7 = trigger gain failed<br>Bit 8 = external trigger zero failed<br>Bits 9-15 = unused |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| StatusCalCond | mbbiDirect | $0000_{16}$ to $FFFF_{16}$ | Calibration condition register:<br><br>ZT4610:<br>Bit 0 = calibration storage failed<br>Bit 1 = offset zero failed<br>Bit 2 = offset scale failed<br>Bit 3 = null balance failed<br>Bit 4 = gain balance failed<br>Bit 5 = trigger level failed<br>Bits 6-15 = unused<br><br>ZT4400:<br>Bit 0 = calibration storage failed<br>Bit 1 = offset zero failed<br>Bit 2 = offset scale failed<br>Bit 3 = unused<br>Bit 4 = gain balance failed<br>Bit 5 = unused<br>Bit 6 = trigger zero failed<br>Bit 7 = trigger gain failed<br>Bit 8 = external trigger zero failed<br>Bits 9-15 = unused<br><br>ZT4210:<br>Bit 0 = calibration storage failed<br>Bit 1 = offset zero failed<br>Bit 2 = offset scale failed<br>Bit 3 = null balance failed<br>Bit 4 = gain balance failed<br>Bit 5 = ADC failed<br>Bit 6 = trigger zero failed<br>Bit 7 = trigger gain failed<br>Bit 8 = external trigger zero failed<br>Bits 9-15 = unused |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| StatusTestDig*N* | mbbiDirect | $00_{16}$ to $FF_{16}$ | Digitizer status results:<br><br>Digitizer 1:<br>Bit 0 = register test failed<br>Bit 1 = ROM test failed<br>Bit 2 = sample clock test failed<br>Bit 3 = memory clock test failed<br>Bit 4 = *ZT4410 & ZT4210*: unused<br>*ZT4610*: ADC1<br>Bit 5 = *ZT4410 & ZT4210*: unused<br>*ZT4610*: ADC2<br>Bit 6 = RAM1 test failed<br>Bit 7 = RAM2 test failed<br><br>Digitizer 2: (ZT4xx2 only)<br>Bit 0 = register test failed<br>Bit 1 = ROM test failed<br>Bit 2 = *ZT4400*: sample clock test failed<br>*ZT4610 & ZT4210*: unused<br>Bit 3 = memory clock test failed<br>Bit 4 = *ZT4410 & ZT4210*: unused<br>*ZT4610*: ADC3<br>Bit 5 = *ZT4410 & ZT4210*: unused<br>*ZT4610*: ADC4<br>Bit 6 = RAM3 test failed<br>Bit 7 = RAM4 test failed |



Downloaded from www.Manualslib.com manuals search engine

| Process Variable | Type | Values | Description |
|---|---|---|---|
| setStatusTestDig*N*Enable<br>getStatusTestDig*N*Enable | mbboDirect<br>mbbiDirect | $00_{16}$ to $FF_{16}$ | Digitizer register bit enable mask<br><br>Digitizer 1:<br>Bit 0 = register test failed<br>Bit 1 = ROM test failed<br>Bit 2 = sample clock test failed<br>Bit 3 = memory clock test failed<br>Bit 4 = *ZT4410 & ZT4210*: unused<br>*ZT4610*: ADC1<br>Bit 5 = *ZT4410 & ZT4210*: unused<br>*ZT4610*: ADC2<br>Bit 6 = RAM1 test failed<br>Bit 7 = RAM2 test failed<br><br>Digitizer 2: (ZT4xx2 only)<br>Bit 0 = register test failed<br>Bit 1 = ROM test failed<br>Bit 2 = *ZT4400*: sample clock test failed<br>*ZT4610 & ZT4210*: unused<br>Bit 3 = memory clock test failed<br>Bit 4 = *ZT4410 & ZT4210*: unused<br>*ZT4610*: ADC3<br>Bit 5 = *ZT4410 & ZT4210*: unused<br>*ZT4610*: ADC4<br>Bit 6 = RAM3 test failed<br>Bit 7 = RAM4 test failed |
| StatusTestDig*N*Cond | mbbiDirect | $00_{16}$ to $FF_{16}$ | Digitizer condition register:<br>Digitizer 1:<br>Bit 0 = register test failed<br>Bit 1 = ROM test failed<br>Bit 2 = sample clock test failed<br>Bit 3 = memory clock test failed<br>Bit 4 = *ZT4410 & ZT4210*: unused<br>*ZT4610*: ADC1<br>Bit 5 = *ZT4410 & ZT4210*: unused<br>*ZT4610*: ADC2<br>Bit 6 = RAM1 test failed<br>Bit 7 = RAM2 test failed<br><br>Digitizer 2: (ZT4xx2 only)<br>Bit 0 = register test failed<br>Bit 1 = ROM test failed<br>Bit 2 = *ZT4400*: sample clock test failed<br>*ZT4610 & ZT4210*: unused<br>Bit 3 = memory clock test failed<br>Bit 4 = *ZT4410 & ZT4210*: unused<br>*ZT4610*: ADC3<br>Bit 5 = *ZT4410 & ZT4210*: unused<br>*ZT4610*: ADC4<br>Bit 6 = RAM3 test failed<br>Bit 7 = RAM4 test failed |



Downloaded from www.Manualslib.com manuals search engine




**ZTEC® Instruments**



Downloaded from www.Manualslib.com manuals search engine

**Artisan Technology Group** is an independent supplier of quality pre-owned equipment

## Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

## We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

## Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

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

ARTISAN® TECHNOLOGY GROUP