



---

# EPICS Interface

## M-Class Oscilloscopes

LXI Models    ZT4211, ZT4212, ZT4421, ZT4422,  
                  ZT4431, ZT4432, ZT4441, ZT4442  
                  ZT4611, ZT4612, ZT4628, ZT4629,  
                  ZT4421-DP, ZT4422-DP, ZT4431-DP,  
                  ZT4432-DP, ZT4441-DP, ZT4442-DP  
                  ZT4421-HV, ZT4422-HV, ZT4431-HV,  
                  ZT4432-HV, ZT4441-HV, ZT4442-HV

User's Manual: 0004-000067  
Revision 3

September 16, 2010

---

# Contact

ZTEC Instruments 7715 Tiburon Street NE Albuquerque, NM 87109	Telephone: (505) 342-0132 Fax: (505) 342-0222 Web Site: <a href="http://www.ztecinstruments.com">www.ztecinstruments.com</a>
---	--

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.

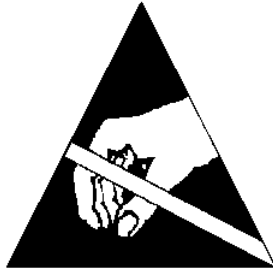
**All rights reserved under copyright laws of the United States and other countries.**

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.



## 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.

# 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	<p>All Sections revised, new sections added.            Added new PVs: operation complete enable, operation complete query, segment view, segment mode, segment count, accessory id, memory clear, undo, reference position,            Created Status Register Section            Renamed and Moved some previous 'Util' PVs to 'Status' PVs:            UtilStatus → Status (fixed instr error also)            UtilStandard → StatusStandard (fixed instr error also)            UtilFreqFault → StatusFreq            UtilTestResult → StatusTest            UtilOperation → StatusOper            UtilQuesReg → StatusQues            UtilCalResult → StatusCal            Fixed record type on above Status PVs, now mbb*Direct            Added new PVs: StatusTestDigN            Added new Status PVs: register enables and conditions            Moved set/getEnvView and set/getAverView to mbbo/i (Changed default NELM size to 1000)            (Added SCAN field to all PVs)            (Fixed ranges to account for multiple instrument series)            Added missing entry for CalcNGateMethod</p>
2a	12-7-09	Process Variables, Capturing	Added Auto Load PV
3	9-16-10	Process Variables	<p>Added Trigger Noise Reject PV, Input Filter LFR PV, Initiate State PV, corrected StatusOper name, added Measurement Reference PVs, Reference Format PVs, Interpolation Format PV, LXI Mode PV, Trigger Time PV, and Force Trigger A PV.            Implemented repeated capture and added wave count PVs</p>

# Table of Contents

<b>Introduction</b> .....	<b>6</b>
Instrument Discovery.....	6
Versions .....	6
Instrument Information .....	6
EPICS Information.....	6
Installed Files .....	7
<b>Configuration</b> .....	<b>8</b>
Instrument Configuration .....	8
Database Configuration.....	9
PV Database File.....	9
Database Template Files.....	10
Database Definition File .....	10
Changing PVs.....	10
PV Naming Conventions .....	11
Set/Get PVs.....	11
FLNKs & Fanout PVs .....	11
Viewing the Current PV Database File .....	12
Uploading a PV Database File .....	12
Channel Access Interface .....	13
EDM & MEDM Interfaces .....	14
<b>Functionality</b> .....	<b>15</b>
Updating .....	15
Interfaces.....	15
Capturing.....	16
Continuous .....	16
Single.....	16
Repeated .....	16
Decimation.....	17
Uploading Waveforms .....	17
Instrument Use Example .....	19
<b>Process Variables</b> .....	<b>20</b>
Process Variable Table .....	20
Input Process Variables.....	20
Horizontal Process Variables .....	23
Acquisition Process Variables .....	24
Trigger Process Variables .....	25
Advanced Trigger Process Variables .....	27
Arm Process Variables .....	29
Output Process Variables .....	30
Calculate Process Variables.....	32
Advanced Calculate Process Variables.....	34
Reference Process Variables .....	40
Measure Process Variables.....	42
Operate Process Variables.....	46
Waveform Process Variables .....	47
Utility Process Variables .....	50
Status Register Variables .....	53

# Introduction



This manual provides detailed information on the ZTEC<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> *webLXI* interface enables the network addresses to be altered, made static, and/or given an alias.

## Versions

The ZTEC<sup>®</sup> 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.

# Installed Files

Both Windows and Linux versions of the ZTEC<sup>®</sup> M-Class ZScope<sup>®</sup> 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

# Configuration



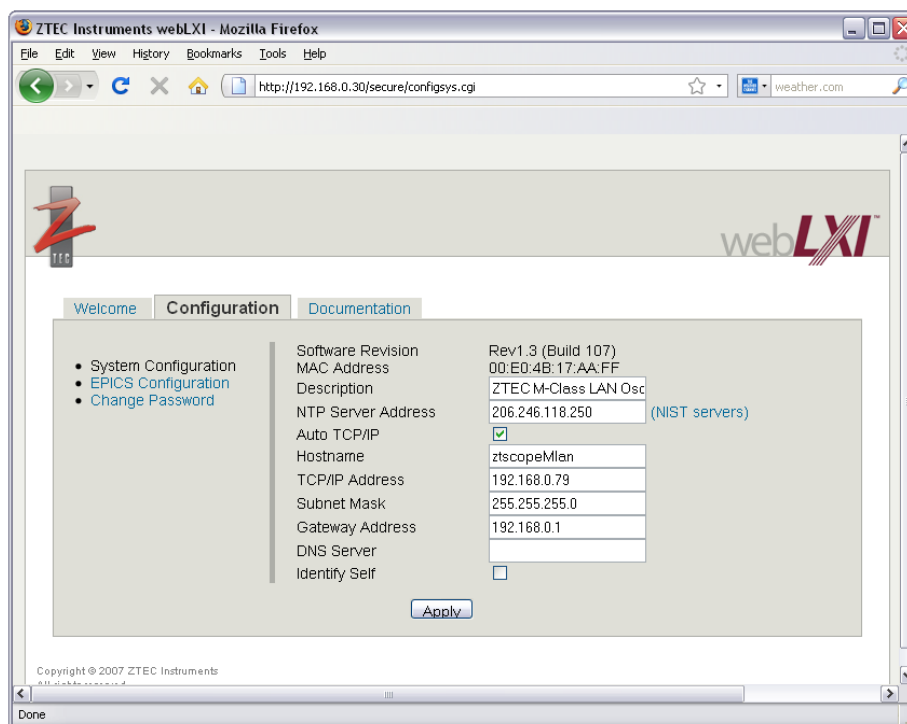
## Instrument Configuration

The instrument's Network and Channel Access settings may be changed using the ZTEC<sup>®</sup> *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<sup>®</sup> and various hyperlinks.

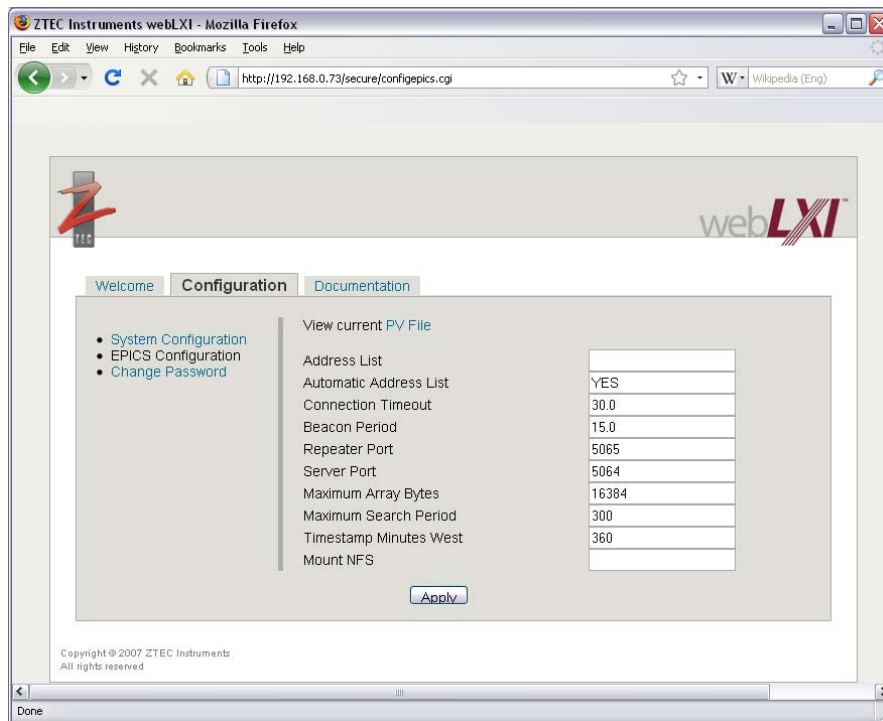
The actions following actions should facilitate communication, but more detail on the *webLXI* interface can be found in the *ZFind User's Guide*.

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

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\data\`  
`[ztscopeM-ch2 | ztscopeM-ch4]\ztscopeM.pv`  
Linux: `/usr/local/share/ZscopeM/EPICS/data/`  
`[ztscopeM-ch2 | ztscopeM-ch4]/ztscopeM.pv`

There is a two-channel version, which includes ZTEC<sup>®</sup>'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 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<sup>®</sup>'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<sup>®</sup> 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<sup>®</sup> EPICS oscilloscopes, changing the PV prefixes to separate identifiers will clear up any ambiguity about which instrument is 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.

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.

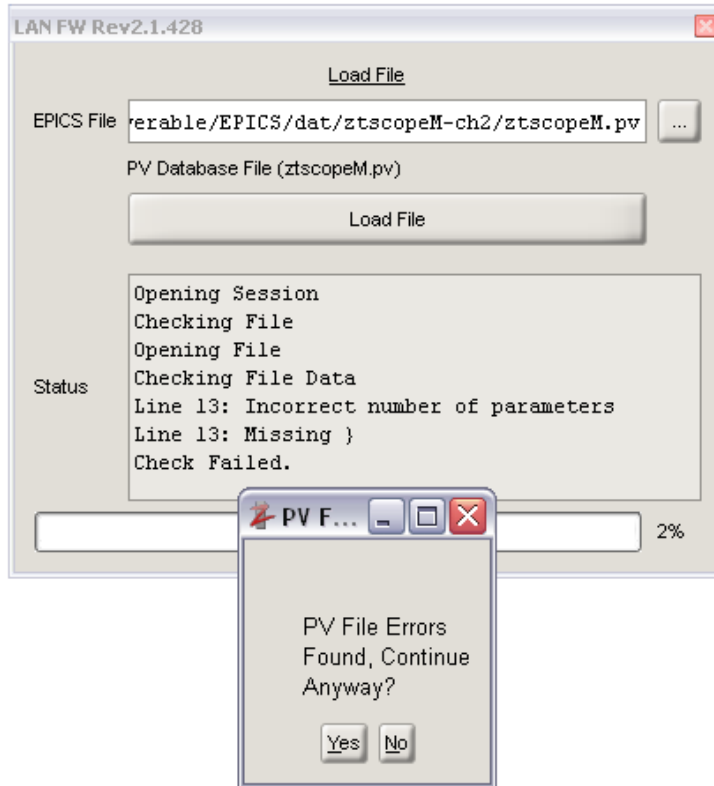
## Uploading a PV Database File

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. Detected errors are reported with their line number in the PV file so they may be tracked down and corrected. The automated checking will only report gross formatting errors and may not catch small errors or typos.



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.

## 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<sup>®</sup> 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 may need to be edited to adjust for desired functionality. In the EDM folder, modify the \*.edl files, or in the MEDM folder, modify the \*.adl files to edit the panels.

# Functionality



## Updating

ZTEC<sup>®</sup> 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<sup>®</sup> LXI instruments are accessible through several interfaces: drivers, GUIs and EPICS. Only one non-Channel Access (CA) interface, such as the drivers or ZScope<sup>®</sup>, 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

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).

There are three capture operation modes that can be started using OpInitiate. OpAbort will halt all capture modes. These are described in detail in the following sections.

During capture, waveforms may be set to update automatically using WaveAutoUpdate. If this is set to None, no waveform PVs will update automatically, and they will only update when the PV is processed (see *Updating*). If this is set to Scaled or Wave, all waveform PVs of the selected type will update if their channel is enabled.

### Continuous

During continuous capture, the instrument captures new data as fast as possible given the acquisition setup. However, this mode has some limitations on waveform downloads. Normal waveform updating is not allowed; waveforms may only be updated using the WaveAutoUpdate capabilities. Since captures may occur very quickly, it is not guaranteed that every capture will be downloaded during automatic update. Also, the waveforms are only available decimated to  $\leq 1000$  points. Larger captures are allowed and all data points are used for measurements and calculations, the data is decimated only for reading waveforms.

Continuous capture is ideal for times when it is not necessary to monitor all waveform data manually. It is good for tests that do on-board processing such as limit and mask testing. This mode is also best for capturing many events that happen closely.

### Single

Single capture mode will initiate a single instrument capture event, wait for capture completion, then perform any enabled automatic waveform updates. Waveforms are downloaded with all available data (see *Decimation*).

### Repeated

Repeated capture mode will do repeated single captures. Once capture and waveform update are complete, the instrument will automatically begin a new capture.

Repeated capture mode should be used in place of continuous mode if it is necessary to have access to every capture or to more than 1000 points per channel. Due to the continuous initiation, repeated capture will make it difficult for non-EPICS interfaces to download waveform data.

## Decimation

When reading waveform data, data may be automatically decimated. The data on the instrument is always captured with `HorzPoints` data points and all on-board calculations and measurements are done to the entire waveform. However, limitations may exist on the data transfer that will force data decimation.

### Continuous Capture

During continuous capture waveforms are always decimated to  $\leq 1000$  points.

### Number of Elements

Waveform PVs are limited by their Number of Elements (NELM) field. If more points are requested, they will automatically decimate to fit in the PV buffer.

### Max Array Bytes

The `EPICS_CA_MAX_ARRAY_BYTES` environment variable on both the host and client must be set to larger than the NELM or channel access errors will occur.

## Uploading Waveforms

Waveforms can be uploaded as an array of type `DBR_DOUBLE` via Channel Access using the `CalcNUpload`, `CalcNScaledUpload`, `RefNUpload` and `RefNScaledUpload` 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/>).

# 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

# Process Variables



## Process Variable Table

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

### 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. The ZT4xx1 has two channels and the ZT4xx2 has four channels.

Process Variable	Type	Values	Description
setInp1Atten setInp2Atten setInp3Atten setInp4Atten	ao	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
getInp1Atten getInp2Atten getInp3Atten getInp4Atten	ai		
setInp1Couple setInp2Couple setInp3Couple setInp4Couple	mbbo	Discrete	Input coupling for selected channel: "AC" = AC Coupling "DC" = DC Coupling
getInp1Couple getInp2Couple getInp3Couple getInp4Couple	mbbi		
setInp1Delay setInp2Delay setInp3Delay setInp4Delay	ao	0.0 to 10.0 $\mu$ s	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.
getInp1Delay getInp2Delay getInp3Delay getInp4Delay	ai		

Process Variable	Type	Values	Description
setInp1Enable setInp2Enable setInp3Enable setInp4Enable  getInp1Enable getInp2Enable getInp3Enable getInp4Enable	bo    bi	0 or 1	Input channel enable for selected channel: 0 = disable 1 = enable
setInp1Filter setInp2Filter setInp3Filter setInp4Filter  getInp1Filter getInp2Filter getInp3Filter getInp4Filter	bo bi	0 or 1	Input lowpass filter enable for selected channel: 0 = bypass 1 = filter enable
setInp1FilterLFR setInp2FilterLFR setInp3FilterLFR setInp4FilterLFR  getInp1FilterLFR getInp2FilterLFR getInp3FilterLFR getInp4FilterLFR	bo    bi	0 or 1	Input low frequency reject filter enable (ZT4620 only) 0 = bypass 1 = filter enable
setInp1Imped setInp2Imped setInp3Imped setInp4Imped  getInp1Imped getInp2Imped getInp3Imped getInp4Imped	ao    ai	50 or 1e6	Input impedance for selected channel: 50 = 50 Ohm (low) 1e6 = 1 Mohm (high)
setInp1Offset setInp2Offset setInp3Offset setInp4Offset  getInp1Offset getInp2Offset getInp3Offset getInp4Offset	ao    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).

Process Variable	Type	Values	Description
setInp1Pos setInp2Pos setInp3Pos setInp4Pos  getInp1Pos getInp2Pos getInp3Pos getInp4Pos	ao     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.
setInp1ProtState setInp2ProtState setInp3ProtState setInp4ProtState  getInp1ProtState getInp2ProtState getInp3ProtState getInp4ProtState	bo     bi	0 or 1	Sets or queries the specified input channel voltage protection state. 0 = Input channel voltage protection OFF 1 = Input channel voltage protection ON
setInp1Range setInp2Range setInp3Range setInp4Range  getInp1Range getInp2Range getInp3Range getInp4Range	ao     ai	Varies by model, See Instrument Manual	Input range in Vpp for selected channel, Input range is limited to 10Vpp for 50 Ohm impedance

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

## 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
setAcqCount getAcqCount	longout longin	2 to 65536 in powers of 2	Waveform count for multiple-capture acquisition types of average, envelope, equivalent-time, and fast acquisition
setAcqType getAcqType	mbbo mbbi	Discrete	Waveform acquisition type: “NORM” = normal “AVER” = average “ENV” = envelope “PDET” = peak detect “HRES” = high resolution “FAST” = fast acquisition “ETIM” = equivalent-time
setAverView getAverView	mbbo mbbi	Discrete	Average view select: “AVER” = average “SEGM” = segments
setEnvView getEnvView	mbbo mbbi	Discrete	Envelope view select: “MIN” = minimum “MAX” = maximum
SegCount	longin	variable, >=0	Queries the number of historical segments captured. This is only valid for History mode.
setSegMode getSegMode	mbbo mbbi	Discrete	Segment mode select: “HIST” = history mode “ZOOM” = zoom mode
setSegView getSegView	longin 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.

## 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. The ZT4xx1 has two channels and the ZT4xx2 has four channels.

Process Variable	Type	Values	Description	
setTrigCount	longout	1 to 65535	Trigger event count to qualify trigger	
getTrigCount	longin			
setTrigHoldoff	ao	0 to 100	Holdoff time in seconds to wait before detecting trigger	
getTrigHoldoff	ai			
setTrigImpedExt	ao	50 or 1e6	Input impedance for external input trigger: 50 = 50 Ohm 1e6 = 1 Mohm	
getTrigImpedExt	ai			
setTrigLevExt	ao	-2.0 to 2.0	Trigger level in Volts for the external input trigger	
getTrigLevExt	ai			
setTrigLevInp1 setTrigLevInp2 setTrigLevInp3 setTrigLevInp4	ao	(offset -range/2) to (offset + range/2)	Trigger level in Volts for an input channel. The trigger level must be within the vertical range and offset settings for the corresponding input channel	
getTrigLevInp1 getTrigLevInp2 getTrigLevInp3 getTrigLevInp4	ai			
setTrigMode	mbbo	Discrete		Trigger mode: "AUTO" = auto-trigger "NORM" = normal trigger
getTrigMode	mbbi			
setTrigSlope	mbbo	Discrete		Trigger slope or polarity: "NEG" = negative "POS" = positive
getTrigSlope	mbbi			

Process Variable	Type	Values	Description
setTrigSource getTrigSource	mbbo mbbi	Discrete	Trigger source: "TTLT0" = TTLTRG0 (external header) "TTLT1" = TTLTRG1 (external header) "TTLT2" = TTLTRG2 (external header) "TTLT3" = TTLTRG3 (external header) "TTLT4" = TTLTRG4 (external header) "TTLT5" = TTLTRG5 (external header) "TTLT6" = TTLTRG6 (external header) "TTLT7" = TTLTRG7 (external header) "INP1" = input channel 1 "INP2" = input channel 2 "INP3" = input channel 3 (ZT4xx2) "INP4" = input channel 4 (ZT4xx2) "EXT" = external input "MAN" = software only "PATT" = pattern
setTrigType getTrigType	mbbo mbbi	Discrete	Trigger type: "EDGE" = edge "IN" = pulse width in "OUT" = pulse width out "LTH" = pulse width less than "GTH" = pulse width greater than "EQU" = pulse width equal "NEQ" = pulse width not equal "GLIT" = glitch "VID" = video

## 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
setTrigBCount getTrigBCount	longout longin	1 to 65535	Trigger B event count to qualify trigger B
setTrigBEnable getTrigBEnable	bo bi	0 or 1	Trigger B enable: 0 = disable 1 = enable
setTrigGlitchLimit getTrigGlitchLimit	ao ai	10e-9 to 500e-3	Limit for glitch triggering time in seconds: 10 ns to 500 ms, 5 ns resolution
setTrigBHoldoff getTrigBHoldoff	ao ai	0 to 100	Holdoff time in seconds to wait before detecting trigger B
setTrigNoiseRejExt getTrigNoiseRejExt	bo bi	0 or 1	Settings for the noise rejection state of the external trigger. (ZT4620 only) 0 = disabled 1 = enabled
setTrigNoiseRejInp1 setTrigNoiseRejInp2 setTrigNoiseRejInp3 setTrigNoiseRejInp4  getTrigNoiseRejInp1 getTrigNoiseRejInp2 getTrigNoiseRejInp3 getTrigNoiseRejInp4	bo  bi	0 or 1	Settings for the noise rejection state of an input channel trigger. (ZT4620 only) 0 = disabled 1 = enabled
setTrigPattMask getTrigPattMask	longout longin	0000 <sub>16</sub> to 1FFF <sub>16</sub>	Pattern trigger mask for defining pattern match criteria (0 = ignore, 1 = match required), used in conjunction with set/getTrigPattTruth: Bit 0 = input channel 1 Bit 1 = input channel 2 Bit 2 = input channel 3 (ZT4xx2) Bit 3 = input channel 4 (ZT4xx2) Bit 4 = external input Bit 5 = TTLTRG0 (external header) Bit 6 = TTLTRG1 (external header) Bit 7 = TTLTRG2 (external header) Bit 8 = TTLTRG3 (external header) Bit 9 = TTLTRG4 (external header) Bit 10 = TTLTRG5 (external header) Bit 11 = TTLTRG6 (external header) Bit 12 = TTLTRG7 (external header)

Process Variable	Type	Values	Description
setTrigPattTruth getTrigPattTruth	longout longin	0000 <sub>16</sub> to 1FFF <sub>16</sub>	Pattern trigger truth for defining pattern match criteria (0 = low, 1 = high), used in conjunction with set/getTrigPattMask: Bit 0 = input channel 1 Bit 1 = input channel 2 Bit 2 = input channel 3 (ZT4xx2) Bit 3 = input channel 4 (ZT4xx2) Bit 4 = external input Bit 5 = TTLTRG0 (external header) Bit 6 = TTLTRG1 (external header) Bit 7 = TTLTRG2 (external header) Bit 8 = TTLTRG3 (external header) Bit 9 = TTLTRG4 (external header) Bit 10 = TTLTRG5 (external header) Bit 11 = TTLTRG6 (external header) Bit 12 = TTLTRG7 (external header)
setTrigPWLLower getTrigPWLLower	ao ai	10e-9 to 500e-3	Lower limit for pulse width triggering time in seconds: 10 ns to 500 ms, 5 ns resolution
setTrigPWUpper getTrigPWUpper	ao ai	10e-9 to 500e-3	Upper limit for pulse width triggering time in seconds: 10 ns to 500 ms, 5 ns resolution
setTrigBSlope getTrigBSlope	mbbo mbbi	Discrete	Trigger slope or polarity: "NEG" = negative "POS" = positive
setTrigBSource getTrigBSource	mbbo mbbi	Discrete	Trigger B source: "TTLT0" = TTLTRG0 (external header) "TTLT1" = TTLTRG1 (external header) "TTLT2" = TTLTRG2 (external header) "TTLT3" = TTLTRG3 (external header) "TTLT4" = TTLTRG4 (external header) "TTLT5" = TTLTRG5 (external header) "TTLT6" = TTLTRG6 (external header) "TTLT7" = TTLTRG7 (external header) "INP1" = input channel 1 "INP2" = input channel 2 "INP3" = input channel 3 (ZT4xx2) "INP4" = input channel 4 (ZT4xx2) "EXT" = external input "MAN" = software only "PATT" = pattern
setTrigVidField getTrigVidField	longout longin	1 or 2	Video trigger field: 1 = field 1 2 = field 2

Process Variable	Type	Values	Description
setTrigVidLine getTrigVidLine	longout longin	1 to 625	Video trigger line, depends upon TrigVidStan and TrigVidField: For NTSC: field 1, line = 1 to 263, field 2, line = 1 to 262 For PAL or SECAM; field 1, line = 1 to 313 field 2, line = 314 to 625
setTrigVidStan getTrigVidStan	mbbo mbbi	Discrete	Video trigger standard: "PAL" = PAL "NTSC" = NTSC "SEC" = SECAM

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

## 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 TTLTRGn outputs on the external trigger header.

Process Variable	Type	Values	Description
setOutLXI0Mode setOutLXI1Mode setOutLXI2Mode setOutLXI3Mode setOutLXI4Mode setOutLXI5Mode setOutLXI6Mode setOutLXI7Mode	mbbo	Discrete	Sets or queries the unit LXI output mode for the given line, <i>N</i> . “WIR” = Wired Or “DRIV” = Driven “BIAS” = Bias
getOutLXI0Mode getOutLXI1Mode getOutLXI2Mode getOutLXI3Mode getOutLXI4Mode getOutLXI5Mode getOutLXI6Mode getOutLXI7Mode	mbbi		
setOutputEventTime getOutputEventTime	ao ai	50e-9 to 163e-3	External output programmable event driven pulse width in seconds: 50 ns to 163 ms
setOutExtEnable getOutExtEnable	bo bi	0 or 1	External output enable: 0 = disable 1 = enable
setOutExtPolarity getOutExtPolarity	mbbo mbbi	Discrete	Output level or pulse polarity: “NEG” = negative “POS” = positive
setOutExtPulsePer getOutExtPulsePer	ao ai	26.67e-9 to 100	External output programmable pulse or clock period in seconds: 26.667 ns to 100 seconds
setOutExtSource getOutExtSource	mbbo mbbi	Discrete	External output source: “ARM” = arm event “TRIG” = trigger complete event “ATR” = trigger A event “BTR” = trigger B event “CAPT” = capture complete event “OPC” = operation complete event “CONS” = constant state “PULS” = programmable pulse “REF” = 10 MHz reference “CLOC” = programmable clock “LIM” = passed limit test “MSS” = master summary status event “PROB” = probe compensation output

Process Variable	Type	Values	Description
setOutTrig0Enable setOutTrig1Enable setOutTrig2Enable setOutTrig3Enable setOutTrig4Enable setOutTrig5Enable setOutTrig6Enable setOutTrig7Enable  getOutTrig0Enable getOutTrig1Enable getOutTrig2Enable getOutTrig3Enable getOutTrig4Enable getOutTrig5Enable getOutTrig6Enable getOutTrig7Enable	bo          bi	0 or 1	Output enable for selected trigger: 0 = disable 1 = enable
setOutTrig0Polarity setOutTrig1Polarity setOutTrig2Polarity setOutTrig3Polarity setOutTrig4Polarity setOutTrig5Polarity setOutTrig6Polarity setOutTrig7Polarity  getOutTrig0Polarity getOutTrig1Polarity getOutTrig2Polarity getOutTrig3Polarity getOutTrig4Polarity getOutTrig5Polarity getOutTrig6Polarity getOutTrig7Polarity	mbbo          mbbi	Discrete	Output level or pulse polarity: “NEG” = negative “POS” = positive
setOutTrig0Source setOutTrig1Source setOutTrig2Source setOutTrig3Source setOutTrig4Source setOutTrig5Source setOutTrig6Source setOutTrig7Source  getOutTrig0Source getOutTrig1Source getOutTrig2Source getOutTrig3Source getOutTrig4Source getOutTrig5Source getOutTrig6Source getOutTrig7Source	mbbo          mbbi	Discrete	Output source for selected trigger: “ARM” = arm event “TRIG” = trigger complete event “ATR” = trigger A event “BTR” = trigger B event “CAPT” = capture complete event “OPC” = operation complete event “CONS” = constant state “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. Range and offset variables need to be updated after a capture.

Process Variable	Type	Values	Description
setCalc1Enable setCalc2Enable setCalc3Enable setCalc4Enable	bo	0 or 1	Channel enable for selected calculate channel: 0 = disable 1 = enable
getCalc1Enable getCalc2Enable getCalc3Enable getCalc4Enable	bi		
Calc1Imm Calc2Imm Calc3Imm Calc4Imm	bo	0 or 1	Causes a calculation channel to immediately update: 0 = disable 1 = enable
setCalc1Offset setCalc2Offset setCalc3Offset setCalc4Offset	ao	-inf to +inf	Offset in Volts for selected calculate channel, Calculate offset is set automatically when the calculate operation is first selected
getCalc1Offset getCalc2Offset getCalc3Offset getCalc4Offset	ai		
setCalc1Op setCalc2Op setCalc3Op setCalc4Op	mbbo	Discrete	Operation for selected calculate channel: "ADD" = add "AVAL" = absolute value "COPY" = copy "DER" = derivative "INT" = integrate "INV" = invert "MULT" = multiply "SUBT" = subtract "LIM" = limit test "FTR" = FFT frequency transform "TTR" = time transform(smooth) "MEAS" = measurement statistics "HIST" = histogram "NOOP" = no operation; static channel
getCalc1Op getCalc2Op getCalc3Op getCalc4Op	mbbi		
setCalc1Pos setCalc2Pos setCalc3Pos setCalc4Pos	ao	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%).
getCalc1Pos getCalc2Pos getCalc3Pos getCalc4Pos	ai		Changing a Calculate Channel's position resets the calculation. Processing Calc/Imm will recalculate using the new offset.

Process Variable	Type	Values	Description
setCalc1Range setCalc2Range setCalc3Range setCalc4Range  getCalc1Range getCalc2Range getCalc3Range getCalc4Range	ao     ai	-inf to +inf	Range in Vpp for selected calculate channel, Calculate range is set automatically when the calculate operation is first selected
setCalc1Source1 setCalc2Source1 setCalc3Source1 setCalc4Source1  getCalc1Source1 getCalc2Source1 getCalc3Source1 getCalc4Source1	mbbo     mbbi	Discrete	Source 1 for selected calculate channel: "INP1" = input channel 1 "INP2" = input channel 2 "INP3" = input channel 3 (4 channel only) "INP4" = input channel 4 (4 channel only) "CALC1" = calculate channel 1 "CALC2" = calculate channel 2 "CALC3" = calculate channel 3 "CALC4" = calculate channel 4 "REF1" = reference channel 1 "REF2" = reference channel 2 "REF3" = reference channel 3 "REF4" = reference channel 4
setCalc1Source2 setCalc2Source2 setCalc3Source2 setCalc4Source2  getCalc1Source2 getCalc2Source2 getCalc3Source2 getCalc4Source2	mbbo mbbi	Discrete	Source 2 for selected calculate channel, Only required for add, subtract, and multiply calculate operations: "INP1" = input channel 1 "INP2" = input channel 2 "INP3" = input channel 3 (ZT4xx2) "INP4" = input channel 4 (ZT4xx2) "CALC1" = calculate channel 1 "CALC2" = calculate channel 2 "CALC3" = calculate channel 3 "CALC4" = calculate channel 4 "REF1" = reference channel 1 "REF2" = reference channel 2 "REF3" = reference channel 3 "REF4" = reference channel 4
Calc1Upload Calc2Upload Calc3Upload Calc4Upload	aao	Array	Upload a signed short integer waveform data array in codes into the selected calculation channel.
Calc1ScaledUpload Calc2ScaledUpload Calc3ScaledUpload Calc4ScaledUpload	aao	Array	Upload a signed short integer waveform data array in Volts into the selected calculation channel.

## 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.

Process Variable	Type	Values	Description
setCalc1FFTFormat setCalc2FFTFormat setCalc3FFTFormat setCalc4FFTFormat  getCalc1FFTFormat getCalc2FFTFormat getCalc3FFTFormat getCalc4FFTFormat	mbbo     mbbi	Discrete	FFT format for selected calculate channel: “MLIN” = magnitude linear “MLOG” = magnitude logarithmic “PHAS” = phase “REAL” = real “IMAG” = imaginary
setCalc1FFTWin setCalc2FFTWin setCalc3FFTWin setCalc4FFTWin  getCalc1FFTWin getCalc2FFTWin getCalc3FFTWin getCalc4FFTWin	mbbo     mbbi	Discrete	FFT window for selected calculate channel: “RECT” = rectangular “HAMM” = hamming “HANN” = hann “BLAC” = Blackman “FLAT” = flattop
setCalc1FiltPoints setCalc2FiltPoints setCalc3FiltPoints setCalc4FiltPoints  getCalc1FiltPoints getCalc2FiltPoints getCalc3FiltPoints getCalc4FiltPoints	longout     longin	2 to 40	Digital lowpass IIR filter smoothing points for selected calculate channel
Calc1GateCursor Calc2GateCursor Calc3GateCursor Calc4GateCursor	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.
setCalc1GateFreqStart setCalc2GateFreqStart setCalc3GateFreqStart setCalc4GateFreqStart  getCalc1GateFreqStart getCalc2GateFreqStart getCalc3GateFreqStart getCalc4GateFreqStart	ao     ai	0 to Maximum frequency	Sets or gets the start frequency for the mask test gate, if setting the gates by frequency.

Process Variable	Type	Values	Description
setCalc1GateFreqStop setCalc2GateFreqStop setCalc3GateFreqStop setCalc4GateFreqStop  getCalc1GateFreqStop getCalc2GateFreqStop getCalc3GateFreqStop getCalc4GateFreqStop	ao     ai	0 to Maximum frequency	Sets or gets the stop frequency for the mask test gate, if setting the gates by frequency.
setCalc1GatePointStart setCalc2GatePointStart setCalc3GatePointStart setCalc4GatePointStart  getCalc1GatePointStart getCalc2GatePointStart getCalc3GatePointStart getCalc4GatePointStart	ao     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.
setCalc1GatePointStop setCalc2GatePointStop setCalc3GatePointStop setCalc4GatePointStop  getCalc1GatePointStop getCalc2GatePointStop getCalc3GatePointStop getCalc4GatePointStop	ao     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.
setCalc1GateTimeStart setCalc2GateTimeStart setCalc3GateTimeStart setCalc4GateTimeStart  getCalc1GateTimeStart getCalc2GateTimeStart getCalc3GateTimeStart getCalc4GateTimeStart	ao     ai	0 to Maximum waveform length	Sets or gets the start time for the mask test gate, if setting the gates by time.
setCalc1GateTimeStop setCalc2GateTimeStop setCalc3GateTimeStop setCalc4GateTimeStop  getCalc1GateTimeStop getCalc2GateTimeStop getCalc3GateTimeStop getCalc4GateTimeStop	ao     ai	0 to Maximum waveform length	Sets or gets the stop time for the mask test gate, if setting the gates by time.
Calc1LimitClear Calc2LimitClear Calc3LimitClear Calc4LimitClear	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

Process Variable	Type	Values	Description
Calc1LimitFail Calc2LimitFail Calc3LimitFail Calc4LimitFail	bi	0 or 1	Calculation limit test results: 0 = no failures 1 = a failure has occurred
setCalc1LimitMeas setCalc2LimitMeas setCalc3LimitMeas setCalc4LimitMeas  getCalc1LimitMeas getCalc2LimitMeas getCalc3LimitMeas getCalc4LimitMeas	stringout     stringin	string	The measurement to check during a limit test. “MASK” = Mask Test “AC” = AC RMS “AMPL” = Amplitude “AVER” = Average “CAV” = Cycle Average “CFR” = Cycle Frequency “CPER” = Cycle Period “CRMS” = Cycle RMS “DC” = DC RMS “ENOB” = Effective number of bits (FFT) “FEDG” = Number of Falling Edges “FOV” = Falling edge overshoot “FPR” = Falling edge preshoot “FREQ” = Frequency “FTCR” = Falling edge crossing time “FTIM” = Fall time “HIGH” = High “LOW” = Low “MAX” = Maximum “MIN” = Minimum “MID” = Mid “NDUT” = Negative duty cycle “NWID” = Negative pulse width “PDUT” = Positive duty cycle “PER” = Period “PHAS” = Phase “PTP” = Peak-to-peak “PWID” = Positive width “REDG” = Number of Rising Edges “ROV” = Rising edge overshoot “RPR” = Rising edge preshoot “RTCR” = Rising edge crossing time “RTIM” = Rise time “SDEV” = Standard deviation “SFDR” = Spurious-free dynamic range (FFT) “SNDR” = Signal-to-noise+distortion (FFT) “SNR” = Signal-to-noise (FFT) “THD” = Total harmonic distortion (FFT) “TMAX” = Time of maximum “TMIN” = Time of minimum

Process Variable	Type	Values	Description
setCalc1LimitLower setCalc2LimitLower setCalc3LimitLower setCalc4LimitLower  getCalc1LimitLower getCalc2LimitLower getCalc3LimitLower getCalc4LimitLower	ao     ai	-inf to +inf	The lower boundary for limit tests
setCalcLimitNumber  getCalcLimitNumber	longout  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 <sup>th</sup> failure.
setCalc1LimitUpper setCalc2LimitUpper setCalc3LimitUpper setCalc4LimitUpper  getCalc1LimitUpper getCalc2LimitUpper getCalc3LimitUpper getCalc4LimitUpper	ao     ai	-inf to +inf	The upper boundary for limit tests
Calc1LimRepAvg Calc2LimRepAvg Calc3LimRepAvg Calc4LimRepAvg	ai	-inf to +inf	Gets the average value of the waveform reported from the calculation limit test.
Calc1LimRepCount Calc2LimRepCount Calc3LimRepCount Calc4LimRepCount	longin	0 to 2147483647	Gets the number of waveforms captured in the calculation limit test.
Calc1LimRepFail Calc2LimRepFail Calc3LimRepFail Calc4LimRepFail	longin	0 to 2147483647	Gets the number of waveform capture failures in the calculation limit test.
Calc1LimRepLast Calc2LimRepLast Calc3LimRepLast Calc4LimRepLast	ai	-inf to +inf	Gets the last value of the waveform reported from the calculation limit test.
Calc1LimRepMin Calc2LimRepMin Calc3LimRepMin Calc4LimRepMin	ai	-inf to +inf	Gets the minimum value of the waveform reported from the calculation limit test.
Calc1LimRepMax Calc2LimRepMax Calc3LimRepMax Calc4LimRepMax	ai	-inf to +inf	Gets the maximum value of the waveform reported from the calculation limit test.

Process Variable	Type	Values	Description
Calc1LimRepSdev Calc2LimRepSdev Calc3LimRepSdev Calc4LimRepSdev	ai	-inf to +inf	Gets the standard deviation of the waveform reported from the calculation limit test.
Calc1MaskGen Calc2MaskGen Calc3MaskGen Calc4MaskGen	bo	0 or 1	Generates reference waveforms for mask testing based on calculation source.
setCalcMaskHorzOffset getCalcMaskHorzOffset	ao ai	0 to sweep time	Sets the horizontal offset in seconds used in generate reference masks
setCalc1MaskLower setCalc2MaskLower setCalc3MaskLower setCalc4MaskLower  getCalc1MaskLower getCalc2MaskLower getCalc3MaskLower getCalc4MaskLower	mbbo    mbbi	Discrete	The lower boundary for mask tests: "CALC1" = calculate channel 1 "CALC2" = calculate channel 2 "CALC3" = calculate channel 3 "CALC4" = calculate channel 4 "REF1" = reference channel 1 "REF2" = reference channel 2 "REF3" = reference channel 3 "REF4" = reference channel 4
setCalc1MaskUpper setCalc2MaskUpper setCalc3MaskUpper setCalc4MaskUpper  getCalc1MaskUpper getCalc2MaskUpper getCalc3MaskUpper getCalc4MaskUpper	mbbo    mbbi	Discrete	The upper boundary for mask tests: "CALC1" = calculate channel 1 "CALC2" = calculate channel 2 "CALC3" = calculate channel 3 "CALC4" = calculate channel 4 "REF1" = reference channel 1 "REF2" = reference channel 2 "REF3" = reference channel 3 "REF4" = reference channel 4
setCalcMaskVertOffset getCalcMaskVertOffset	ao ai	0 to full scale	Sets the vertical offset in volts used in generate reference masks
Calc1MeasClear Calc2MeasClear Calc3MeasClear Calc4MeasClear	bo	0 or 1	Clears measurement history. A transition from 0 to 1 causes the clear, after which the value returns to 0
setCalc1MeasType setCalc2MeasType setCalc3MeasType setCalc4MeasType  getCalc1MeasType getCalc2MeasType getCalc3MeasType getCalc4MeasType	stringout    stringin	string	The type of measurement to store in a calculation channel: "AC" = AC RMS "AMPL" = Amplitude "AVER" = Average "CAV" = Cycle average "CFR" = Cycle frequency "CPER" = Cycle period "CRMS" = Cycle RMS "DC" = DC RMS "ENOB" = Effective number of bits (FFT) "FEDG" = Number of Falling Edges "FOV" = Falling edge overshoot

Process Variable	Type	Values	Description
			<p> “FPR” = Falling edge preshoot  “FREQ” = Frequency  “FTCR” = Falling edge crossing time  “FTIM” = Fall time  “HIGH” = High  “LOW” = Low  “MAX” = Maximum  “MIN” = Minimum  “MID” = Mid  “NDUT” = Negative duty cycle  “NWID” = Negative pulse width  “PDUT” = Positive duty cycle  “PER” = Period  “PHAS” = Phase  “PTP” = Peak-to-peak  “PWID” = Positive width  “REDG” = Number of Rising Edges  “ROV” = Rising edge overshoot  “RPR” = Rising edge preshoot  “RTCR” = Rising edge crossing time  “RTIM” = Rise time  “SDEV” = Standard deviation  “SFDR” = Spurious-free dynamic range (FFT)  “SNDR” = Signal-to-noise+distortion (FFT)  “SNR” = Signal-to-noise (FFT only)  “THD” = Total harmonic distortion (FFT)  “TMAX” = Time of maximum  “TMIN” = Time of minimum </p>

## 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.

Process Variable	Type	Values	Description
setRef1Enable setRef2Enable setRef3Enable setRef4Enable  getRef1Enable getRef2Enable getRef3Enable getRef4Enable	bo     bi	0 or 1	Channel enable for selected reference channel: 0 = disable 1 = enable
setRef1FFTFormat setRef2FFTFormat setRef3FFTFormat setRef4FFTFormat  getRef1FFTFormat getRef2FFTFormat getRef3FFTFormat getRef4FFTFormat	mbbo mbbi	Discrete	FFT format for selected reference channel: “MLIN” = magnitude linear “MLOG” = magnitude logarithmic “PHAS” = phase “REAL” = real “IMAG” = imaginary
setRef1Offset setRef2Offset setRef3Offset setRef4Offset  getRef1Offset getRef2Offset getRef3Offset getRef4Offset	ao     ai	-inf to +inf	Offset in Volts for selected reference channel, Reference offset is set automatically when the reference waveform is stored
setRef1Pos setRef2Pos setRef3Pos setRef4Pos  getRef1Pos getRef2Pos getRef3Pos getRef4Pos	ao     ai	0.0 to 1.0	Offset as a fraction of full scale for selected reference channel, Reference offset is set automatically when the reference waveform is stored
setRef1Range setRef2Range setRef3Range setRef4Range  getRef1Range getRef2Range getRef3Range getRef4Range	ao     ai	-inf to +inf	Range in Vpp for selected reference channel, Reference range is set automatically when the reference waveform is stored
Ref1ScaledUpload	aaao	Array	Upload a floating point scaled waveform data array

Process Variable	Type	Values	Description
Ref2ScaledUpload Ref3ScaledUpload Ref4ScaledUpload			in Volts into the selected scaled reference channel.
Ref1Store Ref2Store Ref3Store Ref4Store	mbbo	Discrete	Store a waveform from the selected channel to the reference channel. Source channels: “INP1” = input channel 1 “INP2” = input channel 2 “INP3” = input channel 3 (ZT4xx2) “INP4” = input channel 4 (ZT4xx2) “CALC1” = calculate channel 1 “CALC2” = calculate channel 2 “CALC3” = calculate channel 3 “CALC4” = calculate channel 4 “REF1” = reference channel 1 “REF2” = reference channel 2 “REF3” = reference channel 3 “REF4” = reference channel 4
Ref1Upload Ref2Upload Ref3Upload Ref4Upload	aa0	Array	Upload a signed short integer waveform data array in codes into the selected reference channel.

## 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. The corresponding measurements can also be viewed through ZScope™ or the zscopeM drivers as four lists of eight measurements.

Process Variable	Type	Values	Description
Meas1Clear Meas2Clear Meas3Clear Meas4Clear	bo	0 or 1	Set to clear one measurement ( <i>N</i> is 1-32). Idle value is 0. After setting to 1, value returns to idle.
setMeasCurs1X setMeasCurs2X  getMeasCurs1X getMeasCurs2X	ao  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.
setMeasCurs1Y setMeasCurs2Y  getMeasCurs1Y getMeasCurs2Y	ao  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.
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  getMeasErrState	bo  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. 0 = Off 1 = On
setMeasGateTimeStart  getMeasGateTimeStart	ao  ai	0 to Maximum waveform length	Sets or gets the start measurement gate in seconds, if setting the gates by time.
setMeasGateTimeStop  getMeasGateTimeStop	ao  ai	0 to Maximum waveform length	Sets or gets the start measurement gate in seconds, if setting the gates by time.
setMeasGateFreqStart  getMeasGateFreqStart	ao  ai	0 to ½ of the current Sample Frequency	Sets or gets the start measurement gate in Hertz, if setting the gates by frequency.

Process Variable	Type	Values	Description
setMeasGateFreqStop getMeasGateFreqStop	ao 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 getMeasGatePointStart	longout 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 getMeasGatePointStop	longout longin	0 to Maximum waveform points	Sets or gets the stop measurement gate in points, if setting the gates by the number of samples.
setMeasList1Enable setMeasList2Enable setMeasList3Enable setMeasList4Enable  getMeasList1Enable getMeasList2Enable getMeasList3Enable getMeasList4Enable	bo    bi	0 or 1	Sets or queries the state of a list of measurements. A disabled measurement list retains its settings.
MeasList1Clear MeasList2Clear MeasList3Clear MeasList4Clear	bo	0 or 1	Set to clear all eight measurements from the measurement list ( <i>N</i> is 1 -4). Idle value is 0. After setting to 1, value returns to idle.
setMeasNMax getMeasNMax	longout 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.
setMeasRefHigh getMeasRefHigh	ao ai	-Inf to +Inf	Set the high voltage reference for edge based measurements. Units depend on the measurement reference method.
setMeasRefLow getMeasRefLow	ao ai	-Inf to +Inf	Set the low voltage reference for edge based measurements. Units depend on the measurement reference method.
setMeasRefMeth getMeasRefMeth	mbbo mbbi	Discrete	Set the method for reading and settings measurement reference levels “ABS” = Absolute voltage levels “REAL” = Fractions of voltage maximum
setMeasRefMid getMeasRefMid	ao ai	-Inf to +Inf	Set the middle voltage reference for edge based measurements. Units depend on the measurement reference method.
MeasNResult	ai	-Inf to +Inf	Measurement result in appropriate units ( <i>N</i> is 1 – 32).

Process Variable	Type	Values	Description
setMeasNSource getMeasNSource	mbbo mbbi	Discrete	Source for selected measurement number( <i>N</i> is 1 – 32): “INP1” = input channel 1 “INP2” = input channel 2 “INP3” = input channel 3 (ZT4xx2) “INP4” = input channel 4 (ZT4xx2) “CALC1” = calculate channel 1 “CALC2” = calculate channel 2 “CALC3” = calculate channel 3 “CALC4” = calculate channel 4 “REF1” = reference channel 1 “REF2” = reference channel 2 “REF3” = reference channel 3 “REF4” = reference channel 4
MeasNStatus	longin	0000 <sub>16</sub> to 0F03 <sub>16</sub>	Status of selected measurement number ( <i>N</i> is 1 – 32): 0000 <sub>16</sub> = measurement valid Bit 0 = invalid waveform Bit 1 = edge not found Bit 8 = over-voltage maximum Bit 9 = over-voltage minimum Bit 10 = low signal level Bit 11 = measurement command invalid
setMeasNType getMeasNType	stringout stringin	string	Type for selected measurement number ( <i>N</i> is 1 – 32): “NONE” = none selected “AC” = AC RMS “AMPL” = Amplitude “AVER” = Average “CAV” = Cycle average “CFR” = Cycle frequency “CPER” = Cycle period “CRMS” = Cycle RMS “DC” = DC RMS “ENOB” = Effective number of bits (FFT) “FEDG” = Number of Falling Edges “FOV” = Falling edge overshoot “FPR” = Falling edge preshoot “FREQ” = Frequency “FTCR” = Falling edge crossing time “FTIM” = Fall time “HIGH” = High “LOW” = Low “MAX” = Maximum “MIN” = Minimum “MID” = Mid “NDUT” = Negative duty cycle “NWID” = Negative pulse width “PDUT” = Positive duty cycle “PER” = Period “PHAS” = Phase “PTP” = Peak-to-peak “PWID” = Positive width

Process Variable	Type	Values	Description
			<p> “REDG” = Number of Rising Edges  “ROV” = Rising edge overshoot  “RPR” = Rising edge preshoot  “RTCR” = Rising edge crossing time  “RTIM” = Rise time  “SDEV” = Standard deviation  “SFDR” = Spurious-free dynamic range (FFT)  “SNDR” = Signal-to-noise+distortion (FFT)  “SNR” = Signal-to-noise (FFT)  “THD” = Total harmonic distortion (FFT)  “TMAX” = Time of maximum  “TMIN” = Time of minimum </p>

## Operate Process Variables

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

Process Variable	Type	Values	Description
OpAutoLoad	bo	0 or 1	Download new waveforms during continuous acquisition. 0 = no operation 1 = download 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: 0 = disarm 1 = arm 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: 0 = no operation 1 = force auto setup A transition from 0 to 1 causes the auto setup, after which the value returns to 0
OpAbort	bo	0 or 1	Aborts waveform acquisition.
getOpComplete	bi	0 or 1	Check to see if all pending operations have completed. 0 = all pending operations have <u>not</u> completed 1 = all pending operations have completed.
OpCompleteEnable	bo	0 or 1	Enable the operation complete flag to be set when all pending operations have completed. 0 = operation complete register is disabled 1 = operation complete register is enabled until pending operations complete, then is disabled again. This enables the operation complete bit, which can be read in Status as well as OPC driven outputs.
OpForceCap	bo	0 or 1	Force waveform capture (all triggers): 0 = no operation 1 = force capture A transition from 0 to 1 causes the waveform acquisition to complete, after which the value returns to 0
OpForceTrigA	bo	0 or 1	Force trigger A event, this does not effect trigger B functionality. 0 = no operation 1 = force trigger A A transition from 0 to 1 causes the trigger event to complete, after which the value returns to 0
OpInitiate	mbbo	0 to 2	Initiate waveform acquisition. "SING" = single capture "CONT" = continuous capture "REP" = repeated capture

Process Variable	Type	Values	Description
OpInitiateState	bi	0 or 1	Returns the instrument's initiation state 0 = Idle 1 = Initiated
trigTime	ai	0 to 1	Returns the trigger timestamp of the most recent trigger event in fractional seconds with a 1 second wrap.

## Waveform Process Variables

The following table lists the PVs for the acquired waveforms. 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:  $(NELM \times 4) \leq EPICS\_CA\_MAX\_ARRAY\_BYTES$

If FTVL is DOUBLE then:  $(NELM \times 8) \leq EPICS\_CA\_MAX\_ARRAY\_BYTES$

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
setFormInterp	mbbo	Discrete	Selects the format for interpolated waveforms "ZERO" = No interpolation (zero hold) "FIRS" = First Order (linear) "SINC" = $\sin(x)/x$
getFormInterp	mbbi		
InpScaledTime	waveform	0 to 100	Floating point scaled time (X-axis) waveform data in seconds for all input channel
Calc1ScaledTime Calc2ScaledTime Calc3ScaledTime Calc4ScaledTime	waveform	-inf to +inf	Floating point scaled time (X-axis) waveform data for selected calculate channel

Process Variable	Type	Values	Description
Calc1ScaledWave Calc2ScaledWave Calc3ScaledWave Calc4ScaledWave	waveform	-inf to +inf	Floating point scaled waveform data in Volts for selected calculate channel
Calc1Wave Calc2Wave Calc3Wave Calc4Wave	waveform	-2147483648 to 2147483647	Signed short integer waveform data in codes for selected calculate channel
Calc1WaveCount Calc2WaveCount Calc3WaveCount Calc4WaveCount	longin	10 to 65535	The number of times that CalcNWave or CalcNScaledWave has been processed. Resets on instrument start or WaveCountReset.
Calc1WavePoints Calc2WavePoints Calc3WavePoints Calc4WavePoints	longin	10 to 65535	The number of actual data points stored in CalcMWave, CalcNScaledWave and CalcNScaledTime.
Inp1ScaledWave Inp2ScaledWave Inp3ScaledWave Inp4ScaledWave	waveform	-inf to +inf	Floating point scaled waveform data in Volts for selected input channel
Inp1Timestamp Inp2Timestamp Inp3Timestamp Inp4Timestamp	ai	0.0 to 0.9999999	Relative timestamp in seconds for acquired waveform, 1 second wrap period, 100 ns resolution
Inp1Wave Inp2Wave Inp3Wave Inp4Wave	waveform	-2147483648 to 2147483647	Signed short integer waveform data in codes for selected input channel
Inp1WaveCount Inp2WaveCount Inp3WaveCount Inp4WaveCount	longin	10 to 65535	The number of times that InpNWave or InpNScaledWave has been processed. Resets on instrument start or WaveCountReset.
Inp1WavePoints Inp2WavePoints Inp3WavePoints Inp4WavePoints	longin	10 to 65535	The number of actual data points stored in InpMWave, InpNScaledWave and InpScaledTime.
Ref1ScaledTime Ref2ScaledTime Ref3ScaledTime Ref4ScaledTime	waveform	-inf to +inf	Floating point scaled time (X-axis) waveform data for selected reference channel
Ref1ScaledWave Ref2ScaledWave Ref3ScaledWave Ref4ScaledWave	waveform	-inf to +inf	Floating point scaled waveform data in Volts for selected reference channel
Ref1Wave Ref2Wave Ref3Wave Ref4Wave	waveform	-2147483648 to 2147483647	Signed short integer waveform data in codes for selected reference channel

Process Variable	Type	Values	Description
Ref1WaveCount Ref2WaveCount Ref3WaveCount Ref4WaveCount	longin	10 to 65535	The number of times that RefNWave or RefNScaledWave has been processed. Resets on instrument start or WaveCountReset.
Ref1WavePoints Ref2WavePoints Ref3WavePoints Ref4WavePoints	longin	10 to 32768	The number of actual data points stored in RefNWave, RefNScaledWave and RefNScaledTime.
WaveAutoUpdate	mbbo	0 to 2	Set whether to automatically update waveform PVs on capture. “NONE” = Don’t update “SCAL” = Update Float Scaled Waves “CODE” = Update Integer Code Waves
WaveCountReset	bo	0 or 1	Reset all wave count values

## 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
setOutCoerce	bo	0 or 1	Selects whether set parameters update their values based on instrument coercion.
getOutCoerce	bi		
setRestoreState	longout	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 <i>N</i> PVs.
getRestoreState	longin		
UtilAccMD	stringin	ID string	Accessory Identification string with 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). 0 = no operation 1 = reset A transition from 0 to 1 causes the instrument reset, after which the value returns to 0.
UtilErr	stringin	error string	Error string explaining current error code
UtilErrCount	longin	0 to 32	Number of errors in error log, 0 = no errors
UtilErrNext	longin	0 to -32767	Current error code, processing clears current error and retrieves next error.
UtilErrReport	stringin	command string	Returns last command to cause an error. <b>Note:</b> This functionality is only valid for the ZT44xx
UtilID	stringin	ID string	Identification string of following format: "ZTEC,model_number,serial_number,version"
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.
UtilRecall	longout	1 to 30	Instrument state recall: Causes the instrument state to be recalled from one of thirty one locations in non-volatile memory.
UtilRefresh	bo	0 or 1	Manual trigger to update all PVs.

Process Variable	Type	Values	Description
UtilReset	bo	0 or 1	Instrument reset: 0 = no operation 1 = reset 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: Causes the instrument state to be saved to one of thirty one locations in non-volatile memory.
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).
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.  <b>Note:</b> The input channels <u>must</u> be disconnected or be driven with 0.0 VDC before starting the calibration.  The timeout value should be set to infinite before starting the calibration, and reset to the default value when completed.  <b>Note:</b> Do <u>not</u> interrupt the instrument during calibration or the calibration tables could be corrupted.  0 Pass 1 Fail (Did not converge) 2 Corrupt
setUtilSelfID getUtilSelfID	bo bi	0 or 1	Instrument HST LED state: 0 = LAN status 1 = blink Allows instrument identification.

Process Variable	Type	Values	Description
UtilSelfTest	mbbiDirect	0000 <sub>16</sub> to 0375 <sub>16</sub>	Initiates an instrument self test and returns the test results as a 16-bit code. The self test is initiated on instrument power up. Bit 0 = baseboard test failed Bit 1 = unused Bit 2 = ROM test failed Bit 3 = unused Bit 4 = reference oscillator test failed Bit 5 = SDRAM test failed Bit 6 = flash memory test failed Bit 7 = unused Bit 8 = digitizer submodule 1 test failed Bit 9 = digitizer submodule 2 test failed (ZT4xx2) Bit 10 -15 = unused
UtilTemp	ai	0.0 to 100.0	Internal temperature in degrees C
UtilTestCountBB	longin	integer	Returns the number of failure reports from last baseboard self test. 0 to 9 failures.  <b>Note:</b> 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.  <b>Note:</b> 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.  <b>Note:</b> This functionality is only valid for the ZT44xx
UtilTestReportBB	stringin	string	Returns last baseboard command that caused a failure.  <b>Note:</b> This functionality is only valid for the ZT44xx
UtilTestReportSM1	stringin	string	Returns last submodule1 command that caused a failure.  <b>Note:</b> This functionality is only valid for the ZT44xx
UtilTestReportSM2	stringin	string	ZT4xx2 Only. Returns last submodule2 command that caused a failure.  <b>Note:</b> This functionality is only valid for the ZT44xx
UtilUndo	bo	0 or 1	Reverses instrument reset, auto setup or state recall events.

## 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 <sub>16</sub> to FF <sub>16</sub>	Returns the latched instrument status: 1 = latched event Bit 0 = unused Bit 1 = unused Bit 2 = error log not empty Bit 3 = questionable event Bit 4 = message available Bit 5 = standard event Bit 6 = master summary status Bit 7 = operation event
setStatusEnable getStatusEnable	mbbiDirect mbbiDirect	00 <sub>16</sub> to FF <sub>16</sub>	Status register bit enable mask Bit 0 = unused Bit 1 = unused Bit 2 = error log not empty Bit 3 = questionable event Bit 4 = message available Bit 5 = standard event Bit 6 = master summary status Bit 7 = operation event
StatusStandard	mbbiDirect	00 <sub>16</sub> to FF <sub>16</sub>	Returns the latched standard register events: 1 = latched event Bit 0 = operation complete Bit 1 = request control Bit 2 = query error Bit 3 = device dependent error Bit 4 = execution error Bit 5 = command error Bit 6 = user request Bit 7 = power on
setStatusStandardEnable getStatusStandardEnable	mbbiDirect mbbiDirect	00 <sub>16</sub> to FF <sub>16</sub>	Standard register bit enable mask Bit 0 = operation complete Bit 1 = request control Bit 2 = query error Bit 3 = device dependent error Bit 4 = execution error Bit 5 = command error Bit 6 = user request Bit 7 = power on

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

Process Variable	Type	Values	Description
setStatusFreqEnable	mbbiDirect	0000 <sub>16</sub> to FFFF <sub>16</sub>	<p>Frequency register bit enable mask:</p> <p>ZT4610:</p> <ul style="list-style-type: none"> <li>Bit 0 = sample clock unlocked</li> <li>Bit 1 = unused</li> <li>Bit 2 = memory 1-2 clock unlocked</li> <li>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)</li> <li>Bits 4-6 = unused</li> <li>Bit 7 = baseboard clock unlocked</li> <li>Bits 8-15 = unused</li> </ul> <p>ZT4400:</p> <ul style="list-style-type: none"> <li>Bit 0 = sample clock unlocked</li> <li>Bit 1 = unused</li> <li>Bit 2 = memory 1-2 clock unlocked</li> <li>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)</li> <li>Bit 4 = reference 1-2 clock unlocked</li> <li>Bit 5 = reference 3-4 clock unlocked</li> <li>Bit 6 = reference 1-2 loss of signal</li> <li>Bit 7 = reference 3-4 loss of signal</li> <li>Bit 8 = ADC 1 clock unlocked</li> <li>Bit 9 = ADC 2 clock unlocked</li> <li>Bit 10 = ADC 3 clock unlocked</li> <li>Bit 11 = ADC 4 clock unlocked</li> <li>Bits 12-14 = unused</li> <li>Bit 15 = baseboard clock unlocked</li> <li>Bits 8-15 = unused</li> </ul> <p>ZT4210:</p> <ul style="list-style-type: none"> <li>Bit 0 = sample clock unlocked</li> <li>Bit 1 = unused</li> <li>Bit 2 = memory 1-2 clock unlocked</li> <li>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)</li> <li>Bit 4 = trigger 1-2 clock unlocked</li> <li>Bit 5 = trigger 3-4 clock unlocked</li> <li>Bits 6-14 = unused</li> <li>Bit 15 = baseboard clock unlocked</li> </ul>
getStatusFreqEnable	mbbiDirect		

Process Variable	Type	Values	Description
StatusFreqCond	mbbiDirect	0000 <sub>16</sub> to FFFF <sub>16</sub>	<p>Frequency fault status conditions:</p> <p>ZT4610:</p> <ul style="list-style-type: none"> <li>Bit 0 = sample clock unlocked</li> <li>Bit 1 = unused</li> <li>Bit 2 = memory 1-2 clock unlocked</li> <li>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)</li> <li>Bits 4-6 = unused</li> <li>Bit 7 = baseboard clock unlocked</li> <li>Bits 8-15 = unused</li> </ul> <p>ZT4400:</p> <ul style="list-style-type: none"> <li>Bit 0 = sample clock unlocked</li> <li>Bit 1 = unused</li> <li>Bit 2 = memory 1-2 clock unlocked</li> <li>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)</li> <li>Bit 4 = reference 1-2 clock unlocked</li> <li>Bit 5 = reference 3-4 clock unlocked</li> <li>Bit 6 = reference 1-2 loss of signal</li> <li>Bit 7 = reference 3-4 loss of signal</li> <li>Bit 8 = ADC 1 clock unlocked</li> <li>Bit 9 = ADC 2 clock unlocked</li> <li>Bit 10 = ADC 3 clock unlocked</li> <li>Bit 11 = ADC 4 clock unlocked</li> <li>Bits 12-14 = unused</li> <li>Bit 15 = baseboard clock unlocked</li> <li>Bits 8-15 = unused</li> </ul> <p>ZT4210:</p> <ul style="list-style-type: none"> <li>Bit 0 = sample clock unlocked</li> <li>Bit 1 = unused</li> <li>Bit 2 = memory 1-2 clock unlocked</li> <li>Bit 3 = memory 3-4 clock unlocked (ZT4xx2)</li> <li>Bit 4 = trigger 1-2 clock unlocked</li> <li>Bit 5 = trigger 3-4 clock unlocked</li> <li>Bits 6-14 = unused</li> <li>Bit 15 = baseboard clock unlocked</li> </ul>
StatusTest	mbbiDirect	0000 <sub>16</sub> to FFFF <sub>16</sub>	<p>Test status results:</p> <ul style="list-style-type: none"> <li>1 = latched event</li> <li>Bit 0 = baseboard register test failed</li> <li>Bit 1 = n/a</li> <li>Bit 2 = baseboard ROM test failed</li> <li>Bit 3 = n/a</li> <li>Bit 4 = reference oscillator test failed</li> <li>Bit 5 = DRAM test failed</li> <li>Bit 6 = flash memory test failed</li> <li>Bit 7 = unused</li> <li>Bit 8 = digitizer 1 test failed</li> <li>Bit 9 = digitizer 2 test failed (ZT4xx2)</li> <li>Bits 10-15 = unused</li> </ul>

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

Process Variable	Type	Values	Description
setStatusOperEnable getStatusOperEnable	mbbiDirect mbbiDirect	0000 <sub>16</sub> to FFFF <sub>16</sub>	Operation register bit enable mask Bit 0 = calibrating Bit 1 = settling Bit 2 = ranging Bit 3 = sweeping Bit 4 = measuring Bit 5 = waiting for trigger Bit 6 = waiting for arm Bit 7 = unused Bit 8 = trigger event Bit 9 = data capture event Bit 10 = limit test event Bit 11 = auto download event Bits 12-15 = unused
StatusOperCond	mbbiDirect	0000 <sub>16</sub> to FFFF <sub>16</sub>	Operation condition register: Bit 0 = calibrating Bit 1 = settling Bit 2 = ranging Bit 3 = sweeping Bit 4 = measuring Bit 5 = waiting for trigger Bit 6 = waiting for arm Bit 7 = unused Bit 8 = trigger event Bit 9 = data capture event Bit 10 = limit test event Bit 11 = auto download event 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 <sub>16</sub> (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 <sub>16</sub> . The Status Byte and Standard Event Status Registers as defined by IEEE 488.2 are not affected

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

Process Variable	Type	Values	Description
StatusVolt	mbbiDirect	000 <sub>16</sub> to FFF <sub>16</sub>	<p>Voltage status faults: 1 = latched event</p> <p>ZT4210:</p> <ul style="list-style-type: none"> <li>Bit 0 = input 1 overload</li> <li>Bit 1 = input 2 overload</li> <li>Bit 2 = input 3 overload (ZT4xx2)</li> <li>Bit 3 = input 4 overload (ZT4xx2)</li> <li>Bit 4 = input 1-2 overvoltage</li> <li>Bit 5 = unused</li> <li>Bit 6 = input 3-4 overvoltage (ZT4xx2)</li> <li>Bit 7 = unused</li> <li>Bit 8 = accessory 1 fault</li> <li>Bit 9 = accessory 2 fault</li> <li>Bit 10 = accessory 3 fault (ZT4xx2)</li> <li>Bit 11 = accessory 4 fault (ZT4xx2)</li> </ul> <p>ZT4400 &amp; ZT4610</p> <ul style="list-style-type: none"> <li>Bit 0 = input 1 overload</li> <li>Bit 1 = input 2 overload</li> <li>Bit 2 = input 3 overload (ZT4xx2)</li> <li>Bit 3 = input 4 overload (ZT4xx2)</li> <li>Bit 4 = input 1 overvoltage</li> <li>Bit 5 = input 2 overvoltage</li> <li>Bit 6 = input 3 overvoltage (ZT4xx2)</li> <li>Bit 7 = input 4 overvoltage (ZT4xx2)</li> <li>Bit 8 = accessory 1 fault</li> <li>Bit 9 = accessory 2 fault</li> <li>Bit 10 = accessory 3 fault (ZT4xx2)</li> <li>Bit 11 = accessory 4 fault (ZT4xx2)</li> </ul>

Process Variable	Type	Values	Description
setStatusVoltEnable	mbbiDirect	000 <sub>16</sub> to FFF <sub>16</sub>	Voltage register bit enable mask
getStatusVoltEnable	mbbiDirect		<p>ZT4210:</p> <ul style="list-style-type: none"> <li>Bit 0 = input 1 overload</li> <li>Bit 1 = input 2 overload</li> <li>Bit 2 = input 3 overload (ZT4xx2)</li> <li>Bit 3 = input 4 overload (ZT4xx2)</li> <li>Bit 4 = input 1-2 overvoltage</li> <li>Bit 5 = unused</li> <li>Bit 6 = input 3-4 overvoltage (ZT4xx2)</li> <li>Bit 7 = unused</li> <li>Bit 8 = accessory 1 fault</li> <li>Bit 9 = accessory 2 fault</li> <li>Bit 10 = accessory 3 fault (ZT4xx2)</li> <li>Bit 11 = accessory 4 fault (ZT4xx2)</li> </ul> <p>ZT4400 &amp; ZT4610</p> <ul style="list-style-type: none"> <li>Bit 0 = input 1 overload</li> <li>Bit 1 = input 2 overload</li> <li>Bit 2 = input 3 overload (ZT4xx2)</li> <li>Bit 3 = input 4 overload (ZT4xx2)</li> <li>Bit 4 = input 1 overvoltage</li> <li>Bit 5 = input 2 overvoltage</li> <li>Bit 6 = input 3 overvoltage (ZT4xx2)</li> <li>Bit 7 = input 4 overvoltage (ZT4xx2)</li> <li>Bit 8 = accessory 1 fault</li> <li>Bit 9 = accessory 2 fault</li> <li>Bit 10 = accessory 3 fault (ZT4xx2)</li> <li>Bit 11 = accessory 4 fault (ZT4xx2)</li> </ul>

Process Variable	Type	Values	Description
StatusVoltCond	mbbiDirect	000 <sub>16</sub> to FFF <sub>16</sub>	<p>Voltage condition register:</p> <p>ZT4210:</p> <ul style="list-style-type: none"> <li>Bit 0 = input 1 overload</li> <li>Bit 1 = input 2 overload</li> <li>Bit 2 = input 3 overload (ZT4xx2)</li> <li>Bit 3 = input 4 overload (ZT4xx2)</li> <li>Bit 4 = input 1-2 overvoltage</li> <li>Bit 5 = unused</li> <li>Bit 6 = input 3-4 overvoltage (ZT4xx2)</li> <li>Bit 7 = unused</li> <li>Bit 8 = accessory 1 fault</li> <li>Bit 9 = accessory 2 fault</li> <li>Bit 10 = accessory 3 fault (ZT4xx2)</li> <li>Bit 11 = accessory 4 fault (ZT4xx2)</li> </ul> <p>ZT4400 &amp; ZT4610</p> <ul style="list-style-type: none"> <li>Bit 0 = input 1 overload</li> <li>Bit 1 = input 2 overload</li> <li>Bit 2 = input 3 overload (ZT4xx2)</li> <li>Bit 3 = input 4 overload (ZT4xx2)</li> <li>Bit 4 = input 1 overvoltage</li> <li>Bit 5 = input 2 overvoltage</li> <li>Bit 6 = input 3 overvoltage (ZT4xx2)</li> <li>Bit 7 = input 4 overvoltage (ZT4xx2)</li> <li>Bit 8 = accessory 1 fault</li> <li>Bit 9 = accessory 2 fault</li> <li>Bit 10 = accessory 3 fault (ZT4xx2)</li> <li>Bit 11 = accessory 4 fault (ZT4xx2)</li> </ul>

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

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

Process Variable	Type	Values	Description
StatusCalCond	mbbiDirect	0000 <sub>16</sub> to FFFF <sub>16</sub>	<p>Calibration condition register:</p> <p>ZT4610:</p> <ul style="list-style-type: none"> <li>Bit 0 = calibration storage failed</li> <li>Bit 1 = offset zero failed</li> <li>Bit 2 = offset scale failed</li> <li>Bit 3 = null balance failed</li> <li>Bit 4 = gain balance failed</li> <li>Bit 5 = trigger level failed</li> <li>Bits 6-15 = unused</li> </ul> <p>ZT4400:</p> <ul style="list-style-type: none"> <li>Bit 0 = calibration storage failed</li> <li>Bit 1 = offset zero failed</li> <li>Bit 2 = offset scale failed</li> <li>Bit 3 = unused</li> <li>Bit 4 = gain balance failed</li> <li>Bit 5 = unused</li> <li>Bit 6 = trigger zero failed</li> <li>Bit 7 = trigger gain failed</li> <li>Bit 8 = external trigger zero failed</li> <li>Bits 9-15 = unused</li> </ul> <p>ZT4210:</p> <ul style="list-style-type: none"> <li>Bit 0 = calibration storage failed</li> <li>Bit 1 = offset zero failed</li> <li>Bit 2 = offset scale failed</li> <li>Bit 3 = null balance failed</li> <li>Bit 4 = gain balance failed</li> <li>Bit 5 = ADC failed</li> <li>Bit 6 = trigger zero failed</li> <li>Bit 7 = trigger gain failed</li> <li>Bit 8 = external trigger zero failed</li> <li>Bits 9-15 = unused</li> </ul>

Process Variable	Type	Values	Description
StatusTestDigN	mmbiDirect	00 <sub>16</sub> to FF <sub>16</sub>	<p>Digitizer status results:</p> <p>Digitizer 1:</p> <ul style="list-style-type: none"> <li>Bit 0 = register test failed</li> <li>Bit 1 = ROM test failed</li> <li>Bit 2 = sample clock test failed</li> <li>Bit 3 = memory clock test failed</li> <li>Bit 4 = ZT4410 &amp; ZT4210: unused ZT4610: ADC1</li> <li>Bit 5 = ZT4410 &amp; ZT4210: unused ZT4610: ADC2</li> <li>Bit 6 = RAM1 test failed</li> <li>Bit 7 = RAM2 test failed</li> </ul> <p>Digitizer 2: (ZT4xx2 only)</p> <ul style="list-style-type: none"> <li>Bit 0 = register test failed</li> <li>Bit 1 = ROM test failed</li> <li>Bit 2 = ZT4400: sample clock test failed ZT4610 &amp; ZT4210: unused</li> <li>Bit 3 = memory clock test failed</li> <li>Bit 4 = ZT4410 &amp; ZT4210: unused ZT4610: ADC3</li> <li>Bit 5 = ZT4410 &amp; ZT4210: unused ZT4610: ADC4</li> <li>Bit 6 = RAM3 test failed</li> <li>Bit 7 = RAM4 test failed</li> </ul>

Process Variable	Type	Values	Description
setStatusTestDigNEnable	mbbiDirect	00 <sub>16</sub> to FF <sub>16</sub>	Digitizer register bit enable mask
getStatusTestDigNEnable	mbbiDirect		Digitizer 1: Bit 0 = register test failed Bit 1 = ROM test failed Bit 2 = sample clock test failed Bit 3 = memory clock test failed Bit 4 = ZT4410 & ZT4210: unused ZT4610: ADC1 Bit 5 = ZT4410 & ZT4210: unused ZT4610: ADC2 Bit 6 = RAM1 test failed Bit 7 = RAM2 test failed  Digitizer 2: (ZT4xx2 only) Bit 0 = register test failed Bit 1 = ROM test failed Bit 2 = ZT4400: sample clock test failed ZT4610 & ZT4210: unused Bit 3 = memory clock test failed Bit 4 = ZT4410 & ZT4210: unused ZT4610: ADC3 Bit 5 = ZT4410 & ZT4210: unused ZT4610: ADC4 Bit 6 = RAM3 test failed Bit 7 = RAM4 test failed
StatusTestDigNCond	mbbiDirect	00 <sub>16</sub> to FF <sub>16</sub>	Digitizer condition register: Digitizer 1: Bit 0 = register test failed Bit 1 = ROM test failed Bit 2 = sample clock test failed Bit 3 = memory clock test failed Bit 4 = ZT4410 & ZT4210: unused ZT4610: ADC1 Bit 5 = ZT4410 & ZT4210: unused ZT4610: ADC2 Bit 6 = RAM1 test failed Bit 7 = RAM2 test failed  Digitizer 2: (ZT4xx2 only) Bit 0 = register test failed Bit 1 = ROM test failed Bit 2 = ZT4400: sample clock test failed ZT4610 & ZT4210: unused Bit 3 = memory clock test failed Bit 4 = ZT4410 & ZT4210: unused ZT4610: ADC3 Bit 5 = ZT4410 & ZT4210: unused ZT4610: ADC4 Bit 6 = RAM3 test failed Bit 7 = RAM4 test failed

