



LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY

LIGO Laboratory / LIGO Scientific Collaboration

LIGO-E1900185-v3

Advanced LIGO

10/25/2021

TwinCAT EPICS IOC (tcIoc) User Guide

Daniel Sigg

Distribution of this document:
LIGO Scientific Collaboration

This is an internal working note
of the LIGO Laboratory.

California Institute of Technology
LIGO Project – MS 18-34
1200 E. California Blvd.
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project – NW22-295
185 Albany St
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
P.O. Box 159
Richland WA 99352
Phone 509-372-8106
Fax 509-372-8137

LIGO Livingston Observatory
P.O. Box 940
Livingston, LA 70754
Phone 225-686-3100
Fax 225-686-7189

<http://www.ligo.caltech.edu/>

Table of Contents

1	<i>Introduction</i>	3
2	<i>IOC Commands</i>	3
2.1	EPICS Initialization	3
2.2	Scan Rate	4
2.3	Loading Records	4
2.4	Alias and Replacement Rules	5
2.5	Generating Listings	5
2.6	Information Records	5
2.7	Generating Macros	6
3	<i>TwinCAT EPICS Options</i>	8
4	<i>OPC Specification</i>	10
4.1	OPC Access	12
4.2	OPC Properties	12
4.3	Automatic Type Support	14
4.4	Enumerated Types	14
4.5	Array Variables	15

1 Introduction

The TwinCAT EPICS IOC, tcIoc, is an interface between the [TwinCAT automation software](#) and the [EPICS channel access](#). It makes TwinCAT variables available to the EPICS system using an annotation mechanism similar to the TwinCAT OPC server. The exported variables of one or multiple TwinCAT PLCs are scanned at a fixed rate using the TwinCAT ADS communication library. A memory pool is setup in the IOC which mirrors the variable value in the TwinCAT PLCs. On the EPICS side a device driver has been written for a “tcat” device that interfaces this memory pool and makes all variables available using channel access. Special care was taken to map EPICS output channels to TwinCAT variables that are both readable and writeable. EPICS input channels are mapped to TwinCAT read only variables. Write only variables are not supported.

In the current TwinCAT EPICS IOC implementation, the exported variables have to be declared as global symbols. They can be of complex types such as structures and arrays—even structures of structures. Since EPICS doesn’t support complex types, complex types are expanded down to the simple types, which are then exported each as an EPICS database record. Floating point types are mapped to ai/ao records, integer types to longin/longout records, character strings to stringin/stringout records, and Boolean types to bi/bo records. Enumerated types are mapped to mbbi/mbbo records, if the numerical representation lies within 0 to 15. Otherwise, they map to longin/longout records as well. EPICS only supports short strings, so TwinCAT string values that are larger than 40 characters are truncated.

EPICS also has limitation on the length of channels names. The current limitation is 56 characters maximum. Since the EPICS channel names are typically derived from the full TwinCAT variable name, one has to be careful to stay within this limit. Deep structures of structures can exceed this limit more easily, since each element name is appended to the base variable name.

If a TwinCAT PLC is halted, the data exchange will stop and all EPICS variables are set to invalid. If the TwinCAT PLC is restarted, the data exchange will be reinitialized and the EPICS variables become valid again. The tcIoc is periodically checking the TwinCAT configuration. If it has changed, e.g., after a recompiling and reloading, the data exchange will also stop. Since the tcIoc cannot alter the EPICS database, it has to be terminated and restarted to load the new configuration.

2 IOC Commands

The TwinCAT EPICS IOC looks like a normal IOC, but supports an extra command set to configure the TwinCAT interface. A typical tcIoc configuration file is listed in Table 1. The last command in a tcIoc configuration file should be the “iocInit()” command to start the EPICS server.

2.1 EPICS Initialization

The first three commands in Table 1 initialize the EPICS system. The first line loads the “tCat.dbd” database file which defines the TwinCAT database records. The second line registers the TwinCAT specific device driver commands, and the third line increases the callback queue size, which may be needed for large databases.

```

dbLoadDatabase("C:\SlowControls\tcIoc\tCat.dbd",0,0)
tCat_registerRecordDeviceDriver(pdbbase)
callbackSetQueueSize(50000)

tcSetScanRate(10, 5)
tcSetAlias("C1PLC1", "IFO=H1,END=X")
tcGenerateList("C:\SlowControls\Target\PLC1.opc.txt","-l -rn -yi -cp")
tcGenerateList("C:\SlowControls\Target\PLC1.chn.txt","-l")
tcGenerateList("C:\SlowControls\Target\PLC1.req","-lb")
tcGenerateMacros("C:\SlowControls\Target\ADL")
tcInfoPrefix(".\${IFO}.Sys.\${ALIAS}.info")
tcLoadRecords ("C:\SlowControls\Target\PLC1.tpy")

iocInit()

```

Table 1: Typical configuration file for the TwinCAT EPICS IOC.

2.2 Scan Rate

The command “tcScanRate” sets the scan rate for the TwinCAT PLC. The first argument is the scan rate in milli seconds for the read or write scanners reading and writing TwinCAT variables. The second number is a multiplier which describes the slow down for updating the EPICS read-only channels. The update rate for read/write channels is the same as the TwinCAT scan rate. In Table 1, “tcSetScanRate(10,5)” yields a 10ms TwinCAT update rate, and a 50ms EPICS update rate for read-only channels. EPICS writes are always propagated to TwinCAT at the next write update cycle.

2.3 Loading Records

TwinCAT2 will generate a “.tpy” file, whenever a PLC is compiled. This tpy file is an XML documents describing the PLC configuration. It lists all used types and the variables. This file is also available in TwinCAT3, but is not generated automatically and has to be enabled in the TwinCAT project. After loading a tpy file using the “tcLoadRecords” command, the tcIoc will go through all exported TwinCAT variables, build an internal memory pool and generated an EPICS database file with the same name, but the extension “.db”. Next, it will load the newly generated EPICS database using the standard EPICS load database command. When EPICS loads a database record, it will call the “tcat” device driver, which in turn will link the record back to the memory pool.

The tcIoc watches the modification time of the tpy file to determine, if the TwinCAT configuration has changed. TwinCAT read and write cycles will stop, if the tpy has changed.

The second argument to the “tcLoadRecords” command is an optional set of arguments that are described in Section 3. This command must come after setting the alias and replacement rules, after setting the info prefix, and after setting up listings and macro processing.

2.4 Alias and Replacement Rules

The command “tcAlias” is used to set the alias name for a PLC as well as define replacement rules that are used in the channel name generation. The first argument is simply the alias name, whereas the second argument is a string assigning values to substitution variables, separated by commas. The alias defines an implicit replacement rule of the form “ALIAS=alias name”.

When specifying the OPC properties associated with a TwinCAT variable or structure element, see Section 4, it is possible to define an alternate name which can use substitution variable. For example, specifying “\${Ifo}End\${End}” uses the substitution variables, “Ifo” and “End”, which will lead to “H1EndX” after the replacement rules in Table 1 are applied.

One place to be careful is a replacement rule for a global variable. TwinCAT2 will silently prepend a dot to all global variables, whereas TwinCAT3 will prepend a namespace designator and a dot. Typically, this leading dot and namespace designator are unwanted, and the TwinCAT EPICS IOC will by default remove any letters up to and including the first dot. Since replacement rules are applied globally after any alias substitution, a top level alias name will also have its leading dot and namespace designator removed. If it doesn’t have a leading dot, it may be misinterpreted as a namespace designator. Therefore, if a global variable uses an alias name, it probably needs to start with an explicit dot, and possibly a namespace.

2.5 Generating Listings

Channel listing can be generated using the “tcGenerateList” command. The first argument is the listing file name, while the second argument a set of options that are described in Section 3.

2.6 Information Records

The TwinCAT EPICS IOC supports a set of database records that give information about the PLC configuration within the IOC and are not linked to TwinCAT variables. Supported info records are listed in Table 2.

Information records are enabled by specifying a prefix using the “tcInfoPrefix”. The prefix is prepended to the information records with a dot in between. The same conversion and replacement rules apply as with TwinCAT variables. So, the prefix probably needs to start with an explicit dot.

The “cb.queue” information records report the size of the callback ring buffer size, as well as the used and free entries. It requires a patch of the EPICS libraries to work. Consult the documentation of “EpicsInterface::get_callback_queue_size” function.

Information	Description	Type
name	Name of PLC	STRING
alias	Alias name	STRING
active	Running state of PLC (ONLINE/OFFLINE)	BOOL
state	AMS state of PLC	ENUM
statestr	AMS state of PLC	STRING
timestamp.str	PLC time stamp (format 2019-1-31 09:13:56)	STRING
timestamp.x	x is year, month, day, hour, minute or second	INT
rate.read	Period of read scanner in ms	INT
rate.write	Period of write scanner in ms	INT
rate.update	Period of update scanner in ms	INT
records.num	Number of records	INT
tpy.filename	Name of tpy file	STRING
tpy.valid	Validity of tpy file (VALID/INVALID)	BOOL
tpy.time.str	Modification time of tpy file	STRING
tpy.time.x	x is year, month, day, hour, minute or second	INT
ads.version	ADS library version	INT
ads.revision	ADS library revision	INT
ads.build	ADS library build	INT
ads.port	ADS/AMS port of PLC	INT
ads.netid.str	ADS/AMS address of PLC	STRING
ads.netid.b0..5	ADS/AMS address b0, b1, b2, b3, b4 and b5	INT
svn.local	SVN local modifications (MODIFIED/COMMITTED)	BOOL
svn.revision	SVN revision if fully committed (zero otherwise)	INT
svn.time.str	SVN compile time	STRING
cb.queue.size	Size of EPICS callback queue	INT
cb.queue.used	Used entries in EPICS callback queue	INT
cb.queue.free	Free entries in EPICS callback queue	INT

Table 2: Supported information records.

2.7 Generating Macros

Macros can be used to automatically generate user screens. The command “tcGenerateMacros” with an argument pointing to a directory is used to enable macro generation.

Generating macros will create one file per structure which describes the structure elements. If a structure contains an ErrorStruct of the form:

```

TYPE ErrorStruct :
STRUCT
    Flag:          BOOL;          (*~
                                (OPC          :1:   Make variable visible for OPC-Server)
                                (OPC_PROP[005] :1:   OPC_PROP_RIGHTS)
                                (OPC_PROP[0101] :Error flag: Description)
                                (OPC_PROP[0106] :Error: ONAM)
                                (OPC_PROP[0107] :OK: ZNAM)
                                *)
    Code:          DWORD;         (*~
                                (OPC          :1:   Make variable visible for OPC-Server)
                                (OPC_PROP[005] :1:   OPC_PROP_RIGHTS)
                                (OPC_PROP[0101] :Bit encoded error condition: Description)
                                *)
    Msg:           STRING;        (*~
                                (OPC          :1:   Make variable visible for OPC-Server)
                                (OPC_PROP[005] :1:   OPC_PROP_RIGHTS)
                                (OPC_PROP[0101] :Human readable error message: Description)
                                *)
END_STRUCT
END_TYPE;

```

It will also generate macros for error screens. For this to work properly, a global list needs to be defined that contains all error messages. These error messages need to be defined in a global constant of type ErrorMessageArray. Example:

```

TYPE ErrorMessageArray : ARRAY [1..32] OF STRING;
END_TYPE;

VAR_GLOBAL CONSTANT
    ThermistorStruct_Errors: ErrorMessageArray :=
        (* 1 *) 'Thermistor resistance is too high',
        (* 2 *) 'Thermistor resistance is too low',
        (* 3 *) 'Thermistor data invalid',
        (* 4 *) 'Thermistor measurement error';
END_VAR

```

The name of the constant string array has to reflect the name of the structure that contains the error structure with the extension “_Errors” added. In TwinCAT2 this constant has to be linked to a file with the name “ThermistorStruct_Errors.exp” with the option “Export before compile” selected. This will guarantee that the automatic screen generator is able to assemble an error list for each structure.

3 TwinCAT EPICS Options

Variable names in TwinCAT are translated into EPICS channel names using conversion rules. When generating a database file or a listing, a set of options specifying the conversion rules is available. Options can be specified in either Windows or Unix style. Meaning, both /ea and -ea will produce the same result.

Option	Description
Channel Processing:	
/eo (default)	Only export variables which are marked by an OPC export directive in the tpy file
/ea	Export all global variables regardless of the OPC settings in the tpy file
/ys (default)	String variables are processed
/ns	No string variables are processed
/pa (default)	Process all types
/ps	Process only simple types types, e.g., INT, BOOL, DWORD, etc.
/pc	Process only complex types, e.g., STRUCT, ARRAY
Channel Name Conversion:	
/rl (default)	LIGO standard conversion rule
/rv	LIGO rules for initial vacuum channel names (version 1.1)
/rd	Replace dots with underscores in channel names
/rn	Do not apply any special conversion rules
/cp	Preserve case in EPICS channel names
/cu (default)	Force upper case in EPICS channel names (default)
/cl	Force lower case in EPICS channel names
/nd (default)	Eliminate leading dot in channel name as well as all prior characters
/yd	Leave leading dot in channel name as well as all prior characters
/yi	Leave array indices in channel names
/ni	Replace array brackets with a single leading underscore
/p 'name'	Include a prefix of 'name' for every channel (defaults to no prefix)
Split File Support:	
/nsio (default)	Do not split database or listing by record type
/ysio	Split database or listing into input only and input/ouput recrods
/sn 'num'	Split database or listing into files with no more than 'num' records
/sn 0 (default)	Does not split database or listing into multiple files

Database Generation:	
/devopc (def.)	Use OPC name in INPUT/OUTPUT field
/devtc	Use TwinCAT name in INPUT/OUTPUT fields instead of OPC
/ss	Use standard strings (stringin/stringout)
/sl	Use long strings (sli/slo)
/sd (default)	Use long or short strings depending on size
/is	Use standard 32-bit integers (longin/longout)
/il	Use 64-bit integers (int64in/int64out)
/id (default)	Use 32 or 64-bit integers depending on size
List Generation:	
/l (default)	Generate a standard listing, name only
/ll	Generate a long listing, name and opc parameters
/lb	Generate an autoburt save/restore file
/li	Generate a LIGO DAQ ini file
Macro Generation:	
/ma (default)	Generate a macro file for each structure describing fields and errors
/me	Generate a macro file for each structure describing the error messages
/mf	Generate a macro file for each structure describing all fields

Table 3: TwinCAT EPICS Options

The applicable options are as follows:

	Available options	Enforced options
Utility programs		
tpyinfo	Channel processing	
EpicsDbGen	All (program depreciated)	
tcloc commands:		
tcLoadRecords	Channel processing, channel name conversion For EPICS versions below 7	-ps -nsio -sn 0 -devtc -is
tcGenerateList	Channel processing, channel name conversion, list generation For EPICS versions below 7	-ps -nsio -sn 0 -is
tcGenerateMacros	macro generation	

Table 4: Applicable Options

4 OPC Specification

We are using the TwinCAT OPC comments denoted by (*~ ... *) to make global variables accessible to the OPC server. The opening bracket annotation needs to be on the same line as the variable. OPC properties are used to describe additional database fields such as limits (HOPR/LOPR), precision (PREC) and many more. These OPC properties are translated into corresponding EPICS database fields. An extended example can be found in Table 5.

Individual tags such as the LaserType of the ALSLaserStruct will have the TwinCAT name “H1.Als.X.Laser.LaserType”. The default EPICS channel name constructed from this tag will then become “H1:ALS-X_LASER_LASERTYPE”, if the LIGO standard conversion option is selected. Be aware that TwinCAT names are not case sensitive, whereas EPICS channel names are case sensitive.

```

TYPE ALSLaserEnum : (
    NPRO, (*NPRO*)
    Diode, (*DIODE*)
    Argon (*ARGON*)
);
END_TYPE; (*~
    (OPC_PROP[8510]      :NPRO: ZRST)
    (OPC_PROP[8511]      :DIODE: ONST)
    (OPC_PROP[8512]      :ARGON: TWST)
*)

TYPE ALSLaserStruct:
STRUCT
    Error:                ErrorStruct;
    LaserType:            ALSLaserEnum;
    LaserDiodePowerMonitor: LREAL; (*~
        (OPC
        (OPC_PROP[005]      :1:  Make variable visible for OPC-Server)
        (OPC_PROP[0100]     :1:  OPC_PROP_RIGHTS)
        (OPC_PROP[0101]     :A:  Unit/EGU)
        (OPC_PROP[0101]     :Laser diode 1 power monitor: Description/DESC)
        (OPC_PROP[8500]     :3:  Display precision/PREC)
    *)
    LaserDiodePowerNominal: LREAL; (*~
        (OPC
        (OPC_PROP[005]      :1:  Make variable visible for OPC-Server)
        (OPC_PROP[005]      :3:  OPC_PROP_RIGHTS)
        (OPC_PROP[0100]     :A:  Unit)
        (OPC_PROP[0101]     :Laser diode power nominal: Description)
        (OPC_PROP[8500]     :3:  Display precision)
    *)
    NoiseEaterRelay:      BOOL; (*~
        (OPC
        (OPC_PROP[005]      :1:  Make variable visible for OPC-Server)
        (OPC_PROP[005]      :3:  OPC_PROP_RIGHTS)
        (OPC_PROP[0101]     :Noise Eater Relay: Description)

```

```

                (OPC_PROP[0106]      :On: ONAM)
                (OPC_PROP[0107]      :Off: ZNAM)
        *)
        CrystalTemperature:          LREAL; (*~
                (OPC                  :1:   Make variable visible for OPC-Server)
                (OPC_PROP[005]         :3:   OPC_PROP_RIGHTS)
                (OPC_PROP[0100]        :V:   Unit)
                (OPC_PROP[0101]        :Crystal temperature: Description)
                (OPC_PROP[0102]        : 10: HOPR)
                (OPC_PROP[0103]        : -10: LOPR)
                (OPC_PROP[8500]        :7:   Display precision)
        *)
END_STRUCT
END_TYPE;

TYPE AlsEndStruct :
STRUCT
        Laser:          ALSLaserStruct;
        VCO:             LowNoiseVcoStruct;
        PZT:             ARRAY [1..2] OF PZTMirrorStruct;
        ...
END_STRUCT
END_TYPE;

TYPE AlsStruct :
STRUCT
        End:            AlsEndStruct;      (*~
                (OPC                  :1: visible)
                (OPC_PROP[8620] :${END}: alias name)
        *)
END_STRUCT
END_TYPE
...
TYPE IfoStruct:
STRUCT
        Als:            AlsStruct;
        ...
END_STRUCT
END_TYPE;

VAR_GLOBAL
        IFO:            IfoStruct;        (*~
                (OPC                  :1: visible)
                (OPC_PROP[8620] :.${IFO}: alias name)
        *)
END_VAR;

```

Table 5: Example TwinCAT global variable with OPC comments

4.1 OPC Access

The global variable describing the interface structure of the interferometer is made accessible to the OPC server by using the OPC comments. Meaning,

```

    IFO:          IfoStruct; (*~
                (OPC          :1: visible)
...            *)

```

will make the entire IFO variable with all its sub elements visible through the OPC interface. In turn, it can be interfaced by EPICS. The default behavior of a structure element is to be visible, when the corresponding global variable is visible. However, if an OPC comment is added to a structure element, the visibility has to be set explicitly again.

4.2 OPC Properties

OPC properties are used to further describe the external EPICS records. These properties have to be attached to the last element in a hierarchical levels of structures. These are the elements with a basic type like BOOL, INT, STRING or LREAL.

Only a subset of EPICS database fields are supported. In general, fields associated with conversion and calculations are not supported, since all processing should be done within the PLC program. The supported general properties are listed in the Table 6.

If a property is specified for a structure, it is used as the default value for all its elements. It can be overwritten by each element below. NO_ALARM, MINOR and MAJOR are the allowed alarm severity values. HIHI and LOLO alarms are assigned major severity, if they are defined; whereas LOW and HIGH alarms are assigned minor severity, if they are defined. Custom fields for alarm are currently not supported.

The alias property can be used, if a desired EPICS name clashes with a built-in TwinCAT name. Employing substitution variables, it can also be used to configure names during tcIoc startup based on the replacement rules.

Property ID	Description	Record
5	Access control: 1 – read-only, 3- read/write	all
100	EGU: Engineering units	numeric
101	DESC: Description	all
102	HOPR: High operations value	numeric
103	LOPR: Low operation value	numeric
104	DRVH: Maximum instrument range	numeric
105	DRVL: Minimum instrument range	numeric
106	ONAM: Label for closed (one) state	binary
107	ZNAM: Label for open (zero) state	binary
306	HYST: alarm deadband	numeric
307	HIHI: hihi alarm level	numeric
308	HIGH: high alarm level	numeric
309	LOW: low alarm level	numeric
310	LOLO: lolo alarm level	numeric
8500	PREC: Display precision	numeric
8510 to 8525	ZRST, ONST, ... FFST: Zero string, one string, ... fifteen string	mb binary
8600	EPCS data type (bi, bo, ai, ao, longin, longout, stringin, stringout, mbbi, mbbo, mbbiDirect, and mbboDirect)	all
8601	Input or output: overwrites the default behavior	all
8602	TSE: Time stamp; default is -2	all
8603	PINI: default 1 for input and 0 for output records	all
8604	DTYP: default is opc; can be overwritten with opcRaw	all
8610	Default OPC server name; default is opc	top level
8611	TwinCAT runtime name including ads routing info and port	top level
8620	Alias for structure item or top level symbol	top & items
8700	OSV: one alarm severity	binary
8701	ZSV: zero alarm severity	binary
8702	COSV: change of state alarm severity	(mb) binary
8703	UNSV: unknown state alarm severity	mb binary
8710 to 8725	ZRSV, ONSV, ... FFSV: zero, one, ... fifteen state alarm severity	mb binary
8727 to 8730	HHSV, HSV, LSV and LLSV	analog
8800 to 8999	FIELD: Any database field can be specified in the comment string; does not perform any checks; use only when truly desperate	don't use

Table 6: Supported OPC properties.

4.3 Automatic Type Support

By default all variables that are read-only will be represented by EPICS input records, whereas all variables that have read/write access will be represented by EPICS output records. This behavior can be overwritten, but there should never be a reason to.

The table below shows the default EPICS type selected for the database depending on the TwinCAT datatype.

EPICS	TwinCAT
longin/longout	SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT, BYTE, WORD, DWORD, LWORD
Int64in/int64out	LINT, ULINT, LWORD; with options, see section 3.
bi/bo	BOOL
mbbi/mbbo	Enumerated data type with 16 or fewer labels
stringin/stringout	STRING
lsi/lso	STRING(nnn); long strings supported with options, see section 3.
ai/ao	REAL, LREAL, any other

Table 7: Automatic type support.

4.4 Enumerated Types

An enumerated type will be converted into a multi-bit binary record, if there are 16 or fewer labels and if all numeric representations are between 0 and 15. There is no conversion possible. The numeric value of the enum type has to be the same as its EPICS representation, i.e., the zero value will be set to 0, etc. The string values of the multi-bit binary record are automatically set to the labels of the enumerated type.

Since enum labels need to be unique in TwinCAT2, one usually has to add a prefix to guarantee that there are no name conflicts. This leads to somewhat cumbersome names in EPICS. It is therefore possible to specify the EPICS enum labels differently. Example:

```

TYPE IfoIdEnum : (IfoH1, IfoL1, IfoH2, IfoT1, IfoI1);
END_TYPE (*~
    (OPC_PROP[8510] :H1: ZRST)
    (OPC_PROP[8511] :L1: ONST)
    (OPC_PROP[8512] :H2: TWST)
    (OPC_PROP[8513] :T1: THST)
    (OPC_PROP[8514] :I1: FRST)
*)

```

This leads to EPICS labels of the form H1, L1, etc. rather than the default IfoH1, IfoL1, etc.

TwinCAT3 no longer requires enum labels to be unique, so they can be defined exactly as they will be seen in EPICS. Unfortunately, the OPC support in TwinCAT3 is broken for enumerated types, and one has to revert to normal comments after each tag; see Table 5 for an example.

4.5 Array Variables

Array variables are supported by TwinCAT and can be exported through OPC as well. They will be accessible through EPICS, but require an extension to the LIGO channel naming convention. For example, if the structure “L1.Io.Wfs1” is of type WfsStruct and contains the members:

```

TYPE DemodComplex:
STRUCT
    I:    LREAL;
    Q:    LREAL;
END_STUCT
END_TYPE;

TYPE WfsStruct:
STRUCT
    Gain:    ARRAY [1..4] OF LREAL;
    Rotation: ARRAY [1..4,1..4] OF LREAL;
    Signal:  ARRAY [1..4] OF DemodComplex;
END_STUCT
END_TYPE;

```

The corresponding OPC and EPICS variables are (with m and n ranging from 1 to 4):

Type	TwinCAT name	EPICS name (LIGO standard)
LREAL	L1.Io.Wfs1.Gain[1]	L1:IO-WFS1_GAIN_1
LREAL	L1.Io.Wfs1.Rotation[1][2]	L1:IO-WFS1_ROTATION_1_2
LREAL	L1.Io.Wfs1.Signal[3].I	L1:IO-WFS1_SIGNAL_3_I
LREAL	L1.Io.Wfs1.Signal[3].Q	L1:IO-WFS1_SIGNAL_3_Q

Table 8: Array variables in TwinCAT and EPICS.

Each individual array index will be exported as separate EPICS channel.