



LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY

*LIGO Laboratory / LIGO Scientific Collaboration*

LIGO- E1200225-v8

Advanced LIGO

5/21/2018

---

Coding Standard for TwinCAT Slow Controls Software

---

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](mailto: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](mailto: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

<b>1</b>	<b><i>Introduction</i></b> .....	<b>4</b>
1.1	<b>Programming Languages</b> .....	<b>4</b>
1.2	<b>Project Directories</b> .....	<b>4</b>
1.2.1	Target Area.....	5
1.3	<b>Project Archive</b> .....	<b>5</b>
1.3.1	Organization.....	5
1.3.3	Version Numbers.....	7
1.4	<b>Cycle Time</b> .....	<b>7</b>
1.5	<b>Data Tags (Channels)</b> .....	<b>7</b>
1.5.1	Input/Output Convention.....	7
1.5.2	Interface Variables.....	7
1.6	<b>OPC Interface</b> .....	<b>7</b>
<b>2</b>	<b><i>Program Organization</i></b> .....	<b>8</b>
2.1	<b>Library</b> .....	<b>8</b>
2.1.1	Hardware Input Structure.....	8
2.1.3	Hardware Output Structure.....	9
2.1.4	Interface Structure.....	9
2.1.6	Error Handling.....	10
2.1.7	Function Block.....	12
2.1.8	Initialization.....	12
2.1.9	Visual Screen Templates.....	13
2.2	<b>Global Variables</b> .....	<b>14</b>
2.3	<b>Program</b> .....	<b>15</b>
2.5	<b>Project Files</b> .....	<b>16</b>
2.5.1	Imports.....	16
2.5.2	Site and Location Customization.....	17
2.5.3	Replacement Rules.....	18
2.6	<b>Installation and Configuration Scripts</b> .....	<b>18</b>
<b>3</b>	<b><i>TwinCAT Project Setup</i></b> .....	<b>19</b>
3.1	<b>Setting the Options in the Project</b> .....	<b>19</b>
3.1.1	Project Information.....	19
3.1.2	Fonts and Path Options.....	19
3.1.3	Build Options.....	21
3.1.4	TwinCAT Options.....	21
3.2	<b>PLC Common Infrastructure</b> .....	<b>22</b>
3.2.1	Common libraries.....	22
3.2.2	Error Handling.....	22
3.2.3	Software version.....	25

3.2.5	Run-time information.....	26
<b>4</b>	<b><i>Naming Scheme</i></b> .....	<b>27</b>
<b>4.1</b>	<b>Names</b> .....	<b>27</b>
4.1.1	Variable Names .....	27
4.1.2	Type Names .....	27
4.1.3	Function and Method Names.....	27
4.1.4	Function Block Names.....	27
4.1.5	Names of Visuals.....	27
4.1.6	Suffix Summary .....	28
<b>4.2</b>	<b>Hardware Channels</b> .....	<b>28</b>
<b>4.3</b>	<b>Library Objects</b> .....	<b>29</b>
4.3.1	Name Space .....	29
4.3.2	Folder Names .....	29
<b>4.4</b>	<b>External Tags</b> .....	<b>30</b>
<b>5</b>	<b><i>OPC Access and Properties</i></b> .....	<b>31</b>
<b>5.1</b>	<b>OPC Access</b> .....	<b>31</b>
<b>5.2</b>	<b>OPC Properties</b> .....	<b>31</b>
<b>5.3</b>	<b>Automatic Type Support</b> .....	<b>33</b>
<b>5.4</b>	<b>Enumerated Types</b> .....	<b>33</b>
<b>5.5</b>	<b>Array Variables</b> .....	<b>34</b>
<b>5.6</b>	<b>ICS Database Generation</b> .....	<b>35</b>
<b>6</b>	<b><i>Documentation</i></b> .....	<b>36</b>
<b>6.1</b>	<b>Project Information</b> .....	<b>36</b>
<b>6.2</b>	<b>Type Information</b> .....	<b>36</b>
<b>6.3</b>	<b>Global Variables</b> .....	<b>37</b>
<b>6.4</b>	<b>Interfaces</b> .....	<b>37</b>
<b>6.5</b>	<b>Functions</b> .....	<b>37</b>
<b>6.6</b>	<b>Function Blocks</b> .....	<b>38</b>
<b>6.7</b>	<b>Visuals</b> .....	<b>38</b>

## 1 Introduction

The purpose of this document is to facilitate a single coding standard among the slow controls software written for the TwinCAT system. TwinCAT contains an embedded IEC 61131-3 software PLC which is the main focus here. The document gives guidance how to build a reusable programming structure, how to name objects like variable, structures and function blocks, and how to document a library module.

### 1.1 Programming Languages

The IEC 61131-3 programming standard supports 5 different languages: structured text (ST), function block diagram (FBD), ladder diagram (LD), instruction list (IL) and sequential function chart (SFC). TwinCAT 3 also supports C/C++ and Matlab/Simulink. For the advanced LIGO slow control systems only structured text shall be used with TwinCAT 2.11. For TwinCAT 3 advanced LIGO also supports C/C++ for integrating already written modules.

Programming language	Description	TwinCAT version
Structured Text	One of the IEC 61131-3 programming languages, Pascal like	2.11 and 3
C/C++	For integrating previously written modules	3

**Table 1: Supported languages.**

### 1.2 Project Directories

The project directories on a front-end or development machine are organized in a development area under version control and a target area where the run-times reside.

Items	Path	Owner
TwinCAT	C:\TwinCAT	Beckhoff
Code	C:\SlowControls	Subversion
Target	C:\ SlowControls\Target	Run-time

### 1.2.1 Target Area

The target area contains the files associated with a specific run-time. The directory structure is organized by target and PLC. The run-time files associated with a specific run-time are copied to the target directory using an installation script. This requires that the code are is a committed revision within subversion.

Items	Path	
Target Area	C:\SlowControls\Target	
Example target system area	C:\SlowControls\Target\H1ECATC1	
Example target PLC 1 code	C:\SlowControls\Target\H1ECATC1\PLC1	
Example target PLC 2 code	C:\SlowControls\Target\H1ECATC1\PLC2	
Example target PLC 3 code	C:\SlowControls\Target\H1ECATC1\PLC3	
Example target PLC 4 code	C:\SlowControls\Target\H1ECATC1\PLC4	
TwinCAT boot files	C:\TwinCAT\Boot	

## 1.3 Project Archive

All project files are stored in a subversion (SVN) archive on [redoubt.ligo-wa.caltech.edu](http://redoubt.ligo-wa.caltech.edu).

Item	Link	Type
Server	<a href="http://redoubt.ligo-wa.caltech.edu">redoubt.ligo-wa.caltech.edu</a>	web
Archive	<a href="http://redoubt.ligo-wa.caltech.edu/svn/slowcontrols">/slowcontrols</a>	web
Full path	<a href="https://redoubt.ligo-wa.caltech.edu/svn/slowcontrols/trunk">https://redoubt.ligo-wa.caltech.edu/svn/slowcontrols/trunk</a>	checkout

**Table 2: Subversion archive.**

### 1.3.1 Organization

The slow controls archive contains the folder TwinCAT for storing all files related to TwinCAT. There are currently two sub folders TwinCAT\Library for storing libraries and TwinCAT\Source for the storing project files. Scripts are stored in the TwinCAT\Script\Common folder. The configuration scripts and the system configuration associated with a real-time computer are stored in a target folder within TwinCAT\Script\Configuration. There are up to 4 PLCs allowed in TwinCAT 2.11.

Items	Path	
System documents	SlowControls\Documents	
Network documents	SlowControls\Documents\Network	
TwinCAT files	SlowControls\TwinCAT	
TwinCAT documents	SlowControls\TwinCAT\Documents	
TwinCAT coding standard	SlowControls\TwinCAT\Documents\CodingStandard	
TwinCAT library files	SlowControls\TwinCAT\Library	
Individual TwinCAT library	SlowControls\TwinCAT\Library\CommonModeServo	
...	...	
TwinCAT program source files	SlowControls\TwinCAT\Source	
Current TwinCAT source files	SlowControls\TwinCAT\Source\Current	
Source files for interferometer	SlowControls\TwinCAT\Source\Current\Interferometer	
Corner source files	SlowControls\TwinCAT\Source\Current\Interferometer\Corner	
End station source files	SlowControls\TwinCAT\Source\Current\Interferometer\End	
Import files	SlowControls\TwinCAT\Source\Current\Import	
...	...	
Script files	SlowControls\Scripts	
Common script files	SlowControls\Scripts\Common	
Configuration files	SlowControls\Scripts\Configuration	
Target configuration files	SlowControls\Scripts\Configuration\H1ECATC1	
System configuration files	SlowControls\Scripts\Configuration\H1ECATC1\SYS	
PLC configuration files	SlowControls\Scripts\Configuration\H1ECATC1\PLC	
...	...	
EPICS related files	SlowControls\EPICS	
EPICS utilities	SlowControls\EPICS\Utilities	
...	...	
Modbus related files	SlowControls\Modbus	
Modbus target files	SlowControls\Modbus\Target	
Individual Modbus target	SlowControls\Modbus\Target\H1ModbusC1	
...	...	

**Table 3: Organization of the archive.**

### 1.3.3 Version Numbers

The production code is managed by subversion release numbers. The subversion number is part of the run-time code and is archived. When significant changes to a library are made that require supporting both the old and new versions, a new library project has to be created. If the original library was called `TimingMasterFanout` then new version would be called `TimingMasterFanoutV2`.

## 1.4 Cycle Time

An IEC 61131-3 system consists of system task and at least one programmable logic controller (PLC). The system task is responsible for interfacing the hardware and starting the PLC tasks. The field bus of choice in advance LIGO is EtherCAT. The system task transfers data between a shared memory region and hardware at a fixed cycle time. TwinCAT 2.11 supports up to four different update rates. For advanced LIGO the standard update rate is 10 ms. For a limited number of channels a faster update rate of 1 ms is supported.

Task	Description	Rate
Standard	All non time critical software and supervisory tasks	10 ms
Fast	Time critical functions such as RS422 support at 115kbaud	1 ms

**Table 4: Supported update rates.**

The tasks with the fast update rate are running at a higher priority (lower number).

## 1.5 Data Tags (Channels)

### 1.5.1 Input/Output Convention

From the perspective of the TwinCAT program and configuration input channels refer to inputs from the EtherCAT terminals, e.g., analog-to-digital converters and binary inputs, whereas output channels refer to outputs to the EtherCAT terminals, e.g., digital-to-analog converters and binary outputs. The same is true for user inputs which are inputs into TwinCAT and readbacks which are outputs from TwinCAT.

### 1.5.2 Interface Variables

All external tags (channels) have an initialization record which is periodically updated and is declared PERSISTENT. Upon power failure and loading a new code its value as retained as much as possible. Any initialization that is required, when the PLC is started or when a new version is loaded, needs to be dealt with in software. See the `SaveRestore` library.

## 1.6 OPC Interface

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. Variable names in TwinCAT are translated one-to-one into OPC tag names, which in

turn are translated into EPICS channels using a conversion rule. OPC properties are used to describe additional information such as limits, precision and state names. These OPC properties are translated into corresponding EPICS database fields.

## 2 Program Organization

The development blocks for the advanced LIGO slow controls software are individual libraries. Each of the basic libraries is tailored to control a single electronics chassis or controller.

A typically library consists of

- one or more type describing the hardware inputs,
- one or more type describing the hardware outputs,
- a type describing the user interface channels or tags (input and output),
- one or more function blocks containing the run-time code, and
- a set of visual templates that can be used for diagnostics.

The main program then consists of a global variable list and a series of function block calls.

### 2.1 Library

This section gives an example of the structures and the function block defined for the LowNoiseVco library.

#### 2.1.1 Hardware Input Structure

```

TYPE LowNoiseVcoInStruct :
STRUCT
    PowerOk:           BOOL;  (* Voltage monitor readback *)
    TuneMon:           INT;   (* Monitor for the frequency offset *)
    ReferenceMon:      INT;   (* RF power at the reference input *)
    DividerMon:        INT;   (* RF power at the divider input *)
    OutputMon:         INT;   (* RF power at the output amp *)
    ReferenceTemp:     INT;   (* Temperature of the reference RF detector *)
    DividerTemp:       INT;   (* Temperature of the divider RF detector *)
    OutputTemp:        INT;   (* Temperature of the output RF detector *)
    Excitation:        BOOL;  (* Monitors the excitation input enable *)
    Frequency:         LREAL; (* Measured frequency *)
    FrequencyLive:     BOOL;  (* Keep alive for frequency measurement *)
END_STRUCT
END_TYPE;

```



### 2.1.3 Hardware Output Structure

```

TYPE LowNoiseVcoOutStruct :
STRUCT
    TuneOfs:          INT;  (* Setpoint for the frequency offset  *)
    ExcitationEn:     BOOL; (* Enables the excitation input *)
END_STRUCT
END_TYPE;

```

### 2.1.4 Interface Structure

All elements of an interface structure are getting exported with read and write permission. To prevent output tags from showing an invalid value each output parameter has to be overwritten at each cycle. Output parameters in the interface structure should never be read.

```

TYPE LowNoiseVcoStruct :
STRUCT
    (* error handling *)
    Error:          ErrorStruct; (* Error struct *)
    (* output tags *)
    PowerOk:        BOOL; (* Voltage monitor readback *)
    TuneMon:        LREAL; (* Monitor for the frequency offset in V *)
    ReferenceMon:   LREAL; (* RF power at the reference input in dBm *)
    DividerMon:     LREAL; (* RF power at the divider input in dBm *)
    OutputMon:      LREAL; (* RF power after the output amplifier dBm *)
    ReferenceTemp:  LREAL; (* Temperature of the reference RF detector *)
    DividerTemp:    LREAL; (* Temperature of the divider RF detector *)
    OutputTemp:     LREAL; (* Temperature of the output RF detector in C *)
    ExcitationSwitch: BOOL; (* Monitor the excitation input enable *)
    Frequency:      LREAL; (* Frequency of the VCO output *)
    FrequencyServoFault: BOOL; (* Indicates a fault in the frequency servo *)
    (* input tags *)
    TuneOfs:        LREAL; (* Setpoint for the frequency offset in V *)
    ExcitationEn:   BOOL; (* Enables the excitation input *)
    FrequencySet:   LREAL; (* Setpoint for the VCO frequency output *)
    FrequencyServoEn: BOOL; (* Enables the frequency PID *)
END_STRUCT
END_TYPE;

```

## 2.1.6 Error Handling

Each main function block needs to provide error handling using a single struct defined as the first element of the interface structure: Error of type ErrorStruct. This struct contains a Flag, Code and Msg. If the error flag is set true, it indicates an error condition. The error code is a bit encoded value listing the error conditions with zero indicating no error. The error code number can be used to flag multiple errors by setting corresponding bits. Error conditions are described in the documentation associated with the library.

The error message is a human readable string describing the error condition. It can contain up to 80 characters. If multiple errors are flagged, the error message needs to reflect this. All error messages need to be defined in a global constant of type ErrorMessagesArray.

```

VAR_GLOBAL CONSTANT
    ThermistorStruct_Errors: ErrorMessagesArray := [
        (* 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 TwinCAT 2.11 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 medm screen generator is able to assemble an error list for each structure. For TwinCAT 3.1 no special export option is required.

The function block that is processing the user interface structure needs to call the error handler to flag the error messages. Typically, this works like this:

```

VAR
    ErrorHandler:      ErrorHandlerFB;
END_VAR

// Initialization (before any errors are flagged)
ErrorHandler (ErrorMethod := ErrorMethodEnum.Init, Error := Thermistor.Error,
              List := ADR(ThermistorStruct_Errors));

// Flagging an error
IF Thermistor.Resistance > 1000000 THEN
    ErrorHandler (ErrorMethod := ErrorMethodEnum.Report,
                 Error := Thermistor.Error, Num := 1);
END_IF;

// Making sure it is getting reported
ErrorHandler (ErrorMethod := ErrorMethodEnum.Commit, Error := Thermistor.Error);

```

A simple library without error conditions just needs to call the error handler with Init and Commit.

If a struct contains other structs that have error handlers themselves, the error handler of the containing struct must check the error flag of the sub structs and report them. For example:

```

TYPE TemperatureStruct :
STRUCT
    (* error handling *)
    Error:           ErrorStruct; (* Error struct *)
    (* output tags *)
    Thermistor_A:   ThermistorStruct;
    Thermistor_B:   ThermistorStruct;
    Thermistor_C:   ThermistorStruct;
END_STRUCT
END_TYPE;

VAR_GLOBAL CONSTANT
    TemperatureStruct _Errors: ErrorMessageArray := [
        (* 1 *) 'Thermistor_A',
        (* 2 *) 'Thermistor_B',
        (* 3 *) 'Thermistor_C'];
END_VAR

VAR
    ErrorHandler:   ErrorHandlerFB;
END_VAR

// Initialization (before any errors are flagged
ErrorHandler (ErrorMethod := ErrorMethodEnum.Init, Error := Temperature.Error,
              List := ADR(TemperatureStruct _Errors));
// Flagging the errors of sub structures
IF Temperature.Thermistor_A.Flag THEN
    ErrorHandler (ErrorMethod := ErrorMethodEnum.Report,
                Error := Temperature.Error, Num := 1);
END_IF;
IF Temperature.Thermistor_B.Flag THEN
    ErrorHandler (ErrorMethod := ErrorMethodEnum.Report,
                Error := Temperature.Error, Num := 2);
END_IF;
...
// Making sure it is getting reported
ErrorHandler (ErrorMethod := ErrorMethodEnum.Commit, Error := Temperature.Error);

```

It is important that the error messages for a sub structure have the exact same name as the element it represents. If not the auto generation of the screens will not pick them up.

### 2.1.7 Function Block

A function block has to declare input and output variables. In the simplest case the input parameter is the hardware input structure, the hardware output structure is the output parameter and the interface structure is the in/out parameter.

```

FUNCTION_BLOCK LowNoiseVcoFB
VAR_INPUT
    LowNoiseVcoIn:      LowNoiseVcoInStruct;      (* Input structure *)
END_VAR
VAR_OUTPUT
    LowNoiseVcoOut:     LowNoiseVcoOutStruct;     (* Output structure *)
END_VAR
VAR_IN_OUT
    LowNoiseVco:        LowNoiseVcoStruct;        (* Interface structure *)
END_VAR

```

### 2.1.8 Initialization

All function blocks controlling hardware have to support initialization and have to be able to store the current state. This is done by passing a SaveRestoreEnum parameter as well as an additional interface structure that holds the previously stored values.

```

FUNCTION_BLOCK LowNoiseVcoFB
...
VAR_INPUT
    Request:            SaveRestoreEnum;          (* init/save request *)
END_VAR
VAR_IN_OUT
    LowNoiseVcoInit:    LowNoiseVcoStruct;        (* saved interface struct *)
END_VAR
...

```

The additional interface structure should be used to pass the previously saved parameters to the initialization routine. It is also used to store these parameters. The Init parameter will either request no operation, an initialization operation, a save operation, or a transition to a safe operation mode. Typically, only state machines will have to implement the transition to a safe operation mode. Within the function block the initialization code would look like:

```

(* Code *)
CASE Request OF
  (* initialization *)
  Restore:
    LowNoiseVco := LowNoiseVcoInit;
    (* additional initialization steps can be added here *)
  SafeMode:
    (* only for state machines *)
  Save:
    LowNoiseVcoInit := LowNoiseVco;
  Noop:
    (* always ignore *)
END_CASE;
...

```

The LowNoiseVcoInit variable will be stored in a global persistent block. This means its values are preserved between reboots and recompilations as long as the LowNoiseVcoStruct stays the same. Changing the LowNoiseVcoStruct or one of its elements will invalidate its persistent memory and reinitialize with all zeros. However, a change in an unrelated structure should not affect the low noise VCO.

The restore request is basically an initialization request and will be issued once after a reboot or a reload of the program. The save request will be issued at regular intervals, but at a low rate, maybe once a minute. The safe mode request might be issued upon a fatal error or a user request. It would typically affect the entire PLC program and not just one library.

Restoring to the previously saved values is probably the best option in most circumstances, but is unlikely to be appropriate in all cases. For example, a state machine probably needs to start in a well-defined initialization state and not in whatever state it was left in. In these cases additional code needs to be added to the restore request.

### 2.1.9 Visual Screen Templates

Either one or a set of visual screen templates are associated with a library. The top-level screen template should be a representation of the hardware controlled by the library. It should interface the interface structure, and display all its input and output parameters. Input parameters should be modifiable by the user. Since the library only knows abstract data types, the visual screen template shall deploy placeholder variables to represent actual data. For example, the VCO template screen might reference “\$vco\$.OutputMon” in the numeric field describing the output RF power. \$vco\$ is the placeholder parameter that will be replaced with the actual data of type LowNoiseVcoStruct, when the visual template is embedded into a master screen. In most cases the visual template screens should leave their background transparent, so that it can be set by the master screen.

## 2.2 Global Variables

Global variables are used to store hardware input structure, the hardware output structure, the associated function blocks and a hierarchical structure representing all interface structure as part of an interferometer structure. The later is outlined in section 3.4 and is used to represent the opc naming tree which in turn is translated into an EPICS name.

Persistent global variables are used to store the initialization structures. Their values are retained between reboots and restarts of the program. They will be reinitialized with zeros, when the structure itself is modified, so.

```

VAR_GLOBAL
  I1:          IfoStruct;    (*~    (OPC : 1 : visible for OPC-Server) *)

  AlsVcoIn    AT %I*:       LowNoiseVcoInStruct;    (* Input *)
  AlsVcoOut   AT %Q*:       LowNoiseVcoOutStruct;    (* Output *)
  AlsVcoFB:   LowNoiseVcoFB;    (* Function block *)
  ...
END_VAR
VAR_GLOBAL PERSISTENT
  AlsVcoInit: LowNoiseVcoStruct;    (* Save/restore *)
  ...
END_VAR

```

The variable names for the input, output, interface and initialization structures follow the naming of the structure elements within I1 that lead to the VCO. The interferometer tag isn't included in the name to support copy/paste between different interferometers. The actual addresses are wildcards and are configured through the system manager.

## 2.3 Program

The main program can be a simple series of function block calls. There can be multiple programs to separate subsystems. These programs need to be attached to the standard task, which updates at the 10 ms rate. If a function block requires 1 ms update rate, it needs to be located in separate program that is attached to the fast task.

```
PROGRAM ALS
VAR
    SaveRestore: SaveRestoreFB; (* function block for save/restore *)
    GotoSafe:    BOOL;          (* goto safe mode when transitioning high *)
    Request:     SaveRestoreEnum; (* save/restore request *)
END_VAR

SaveRestore ( SaveInterval := T#1m,
              GotoSafe := GotoSafe,
              Request => Request );

AlsVcoFB (   LowNoiseVcoIn := AlsVcoIn,
            LowNoiseVcoOut => AlsVcoOut,
            LowNoiseVco := I1.Als.VCO,
            Request := Request,
            LowNoiseVcoInit := AlsVcoInit );

...
END_PROGRAM;
```

## 2.5 Project Files

Project files must not be written site specific. To facilitate this the source files for the interferometer are separated into two subdirectories: corner and end. Each project file must contain at least two imports to determine the subversion number and to specify the target interferometer as well as the appropriate end station. The project name indicates the PLC it is intended for and has the form “PLC1.pro” or similar.

### 2.5.1 Imports

A normal project files defines two global variable resources which are linked to a file and are imported before compilation: Global\_Variables\_Version and Global\_Variables\_IFOVAR.

Global\_Variables\_Version looks like this:

```
VAR_GLOBAL CONSTANT
    SvnRevision:      DINT := 0;
END_VAR
```

Whereas Global\_Variables\_IFOVAR looks like this

```
VAR_GLOBAL CONSTANT
    IfoId: IfoIdEnum := IfoT1; (* IfoH1, IfoL1 or IfoH2 *)
    LocId: LocationIdEnum := EndX; (* Corner, EndX or EndY, MidX or MidY *)
END_VAR
```

In the source directory the interferometer is set to the test system. The important point is that during installation these two global variable resources will be overwritten with the correct information appropriate for the target. In particular, the installation script guarantees that the current source code revision has no local modifications and sets the SvnRevision number accordingly. Similarly, the interferometer parameters are set to the appropriate configuration for the specific target. These variables are available in the code and can be used in the rare instances where sites or end stations need to be distinguished.



```

0001 VAR_GLOBAL CONSTANT
0002   IfoId:                      IfoIdEnum := IfoT1; (* IfoH1, IfoL1 or IfoH2 *)
0003   LocId:                       LocationIdEnum := EndX; (* Corner, EndX or EndY, MidX or MidY *)
0004 END_VAR
0005 VAR_GLOBAL
0006   (* Must reflect intrferometer and X/Y end station *)
0007   T1 AT %MBO:                   IfoXStruct;
0008   (*~
0009   (OPC                           :1   : visible for OPC-Server)
0010   (OPC_PROP[8610]                 :Plc2: OPC-Server name)
0011   *)
0012 END_VAR
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
0030
0031

```

## 2.5.2 Site and Location Customization

The main program defines

```

VAR_GLOBAL
  Ifo:                      IfoStruct;      (*~
    (OPC                     :1   : visible for OPC-Server)
    (OPC_Filter               :Export: Filter string for OPC server)
    (OPC_PROP[8610]           :Plc1: OPC-Server name)
    (OPC_PROP[8620]           :.${IFO}: alias name)
  *)
END_VAR

```

which is getting aliased to an H1 or L1 variable by the TwinCAT-EPICS-gateway. A program, which runs in either the X-end or the Y-end, usually defines a structure element “End”. This is then aliased to “X” or “Y” by the TwinCAT-EPICS-gateway. For example,

```

TYPE PemStruct :
STRUCT
  End:                      PemBuildingStruct;  (*~
    (OPC                     :1   : visible for OPC-Server)
    (OPC_PROP[8620]           :${END}: alias name)
  *)
END_STRUCT
END_TYPE

```

Dynamically, the variable IfoId and LocId (both enums) are available to distinguish site and location.

### 2.5.3 Replacement Rules

For the TwinCAT-EPICS-gateway to pick up the correct interferometer name and end station location, it needs to define a set of replacement rule. This is done in the second argument of the tcAlias command, e.g., “IFO=L1, END=Y”.

As shown above the IFO and END variables in the PLC programs need to define an alias name that contains the replacement variable name enclosed by “\${“ and “}”. It is possible to define more complicated aliases, such as “\${END}1PLC2”. The resulting name after replacement then becomes “Y1PLC2”.

One place to be careful is a replacement rule for a global variable. TwinCAT2 will prepend a dot, whereas TwinCAT3 will prepend a name and a dot. The TwinCAT-EPICS-gateway will typically remove and letters up and including the first dot. Therefore, if a global variable uses an alias name, it probably needs to start with an explicit dot.

## 2.6 Installation and Configuration Scripts

The installation scripts are located in the folder SlowControls\Scripts\Common. A separate documentation is available with [T1300175](#).

Typically, a user would run the “install\_tc\_target” shortcut which starts a GUI that allows the selection of a target with associated PLCs as well as the desired action. Targets are defined in the Scripts\Configuration folder. Each target requires its own folder in there, and each PLC requires its own folder within the target folder. Target specific configuration scripts are specified in the target folder, whereas PLC specific configurations are specified in the PLC folder. Typically, the target configuration script should be sufficient.

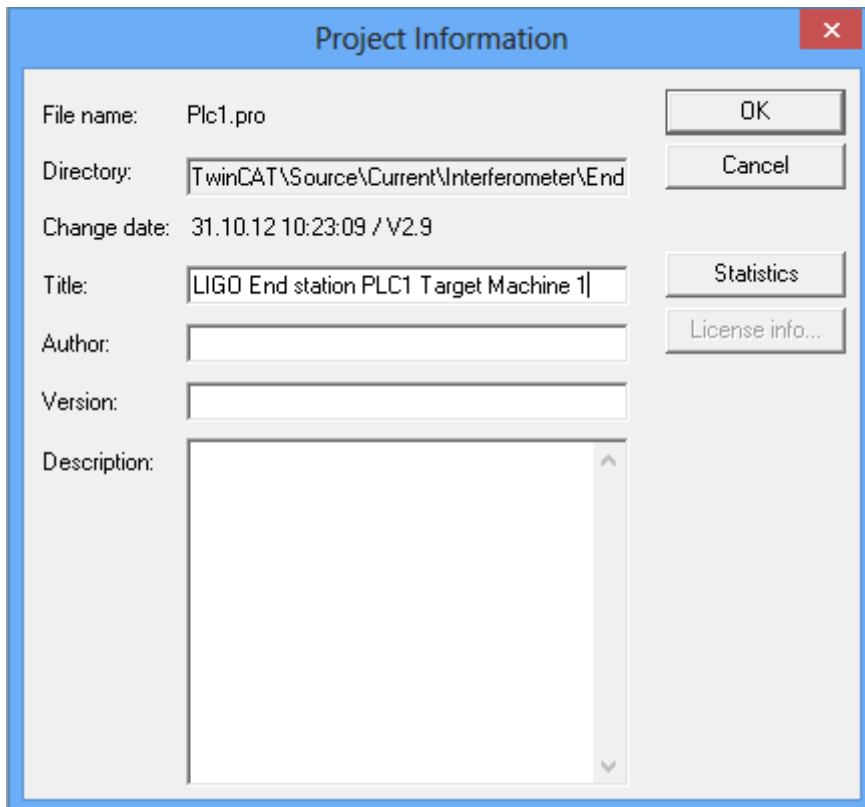
Each target folder also requires a SYS folder which contains the TwinCAT system configuration file (\*.tsm). The TwinCAT system configuration in principle is identical between sites and end stations, but there may be slight difference between locations that cannot be easily parameterized. Since software updates are much more frequent than hardware changes, this should not pose a problem.

## 3 TwinCAT Project Setup

### 3.1 Setting the Options in the Project

#### 3.1.1 Project Information

Set the title information to the appropriate PLC and station. Since there are typically only source files for corner and end station, there should be no indication of interferometer or which of the two end stations.

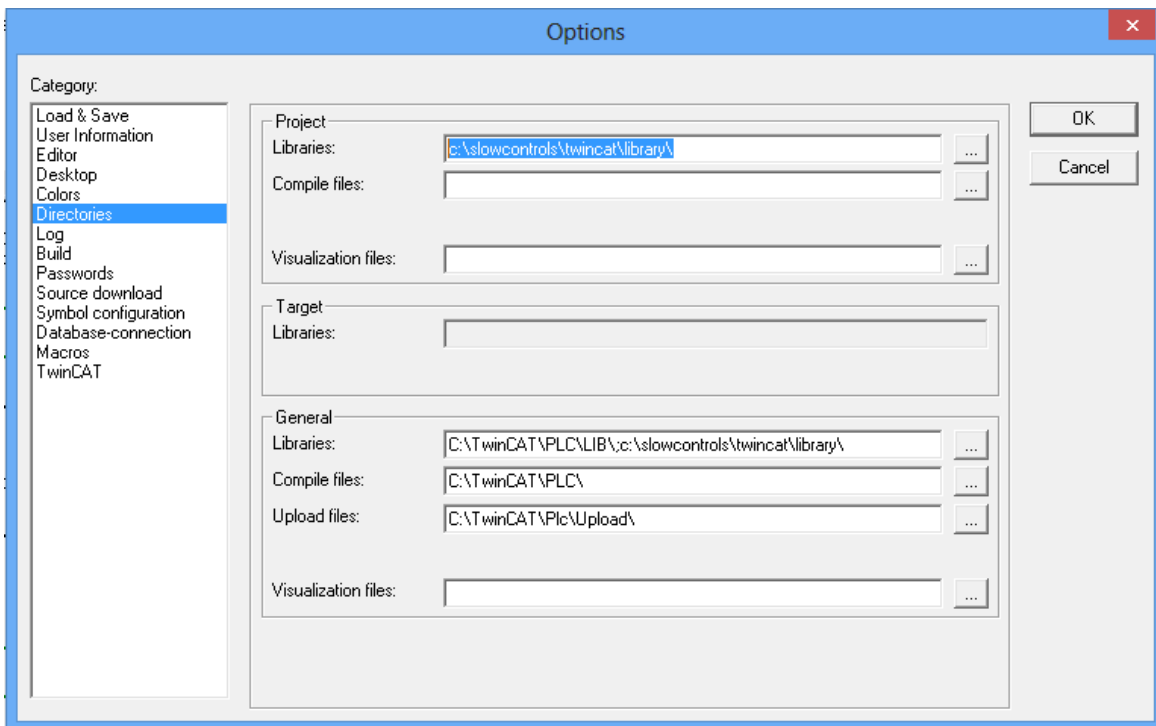
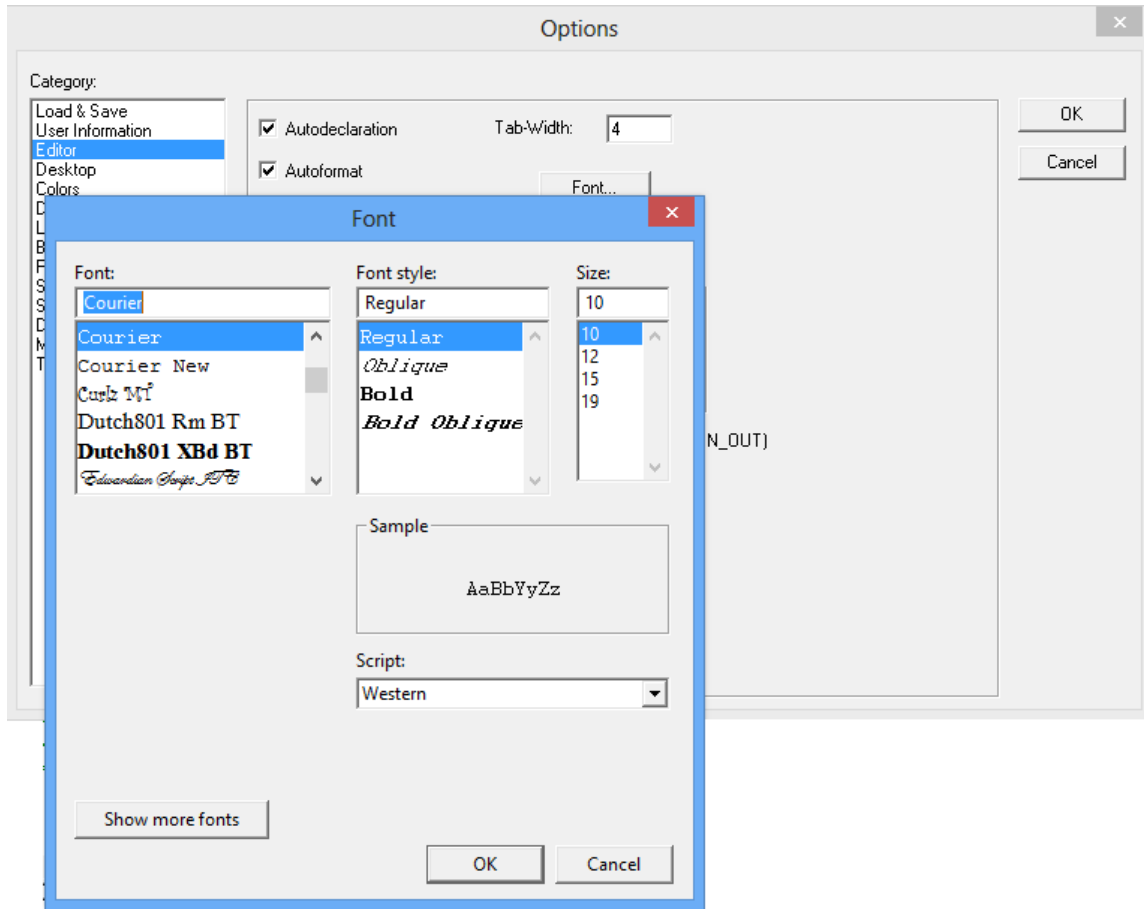


The screenshot shows a 'Project Information' dialog box with the following fields and buttons:

- File name: Plc1.pro
- Directory: TwinCAT\Source\Current\Interferometer\End
- Change date: 31.10.12 10:23:09 / V2.9
- Title: LIGO End station PLC1 Target Machine 1
- Author: (empty)
- Version: (empty)
- Description: (empty text area)
- Buttons: OK, Cancel, Statistics, License info...

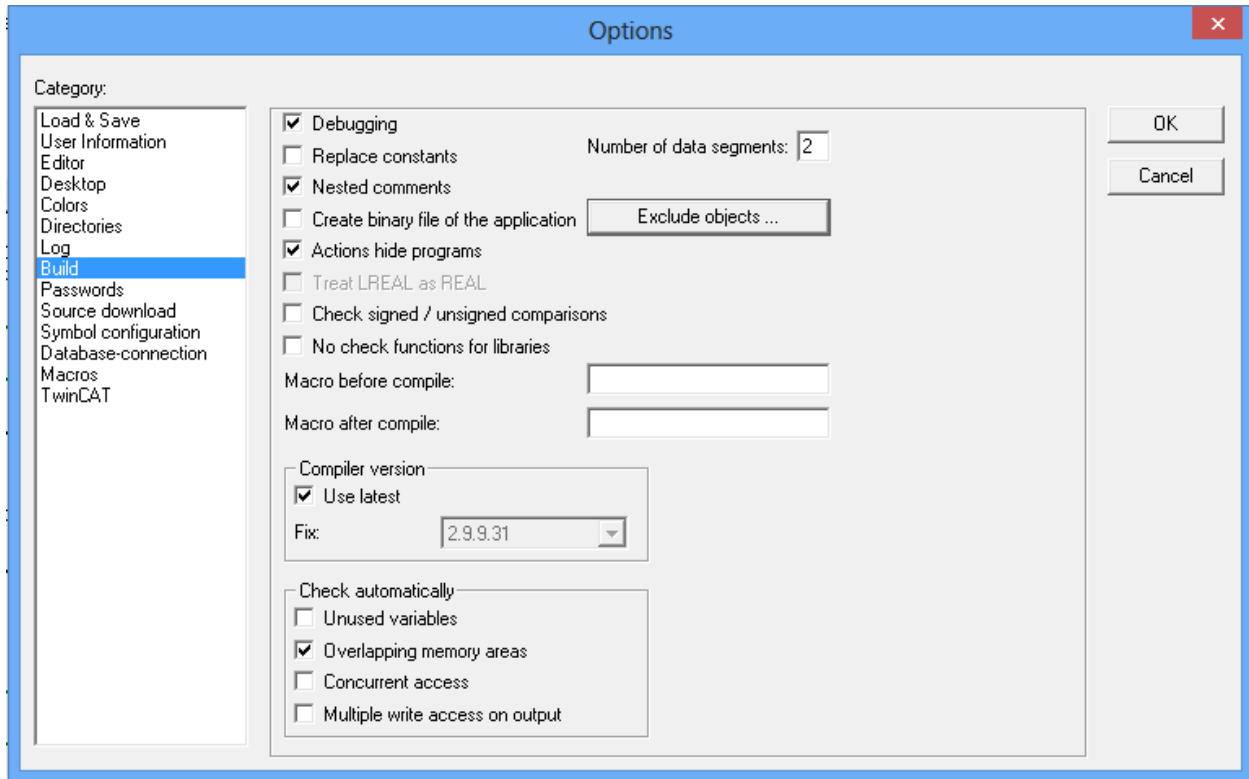
#### 3.1.2 Fonts and Path Options

Choose a fixed point font such as “Courier New”. The project library path should include the path to the slow controls libraries.



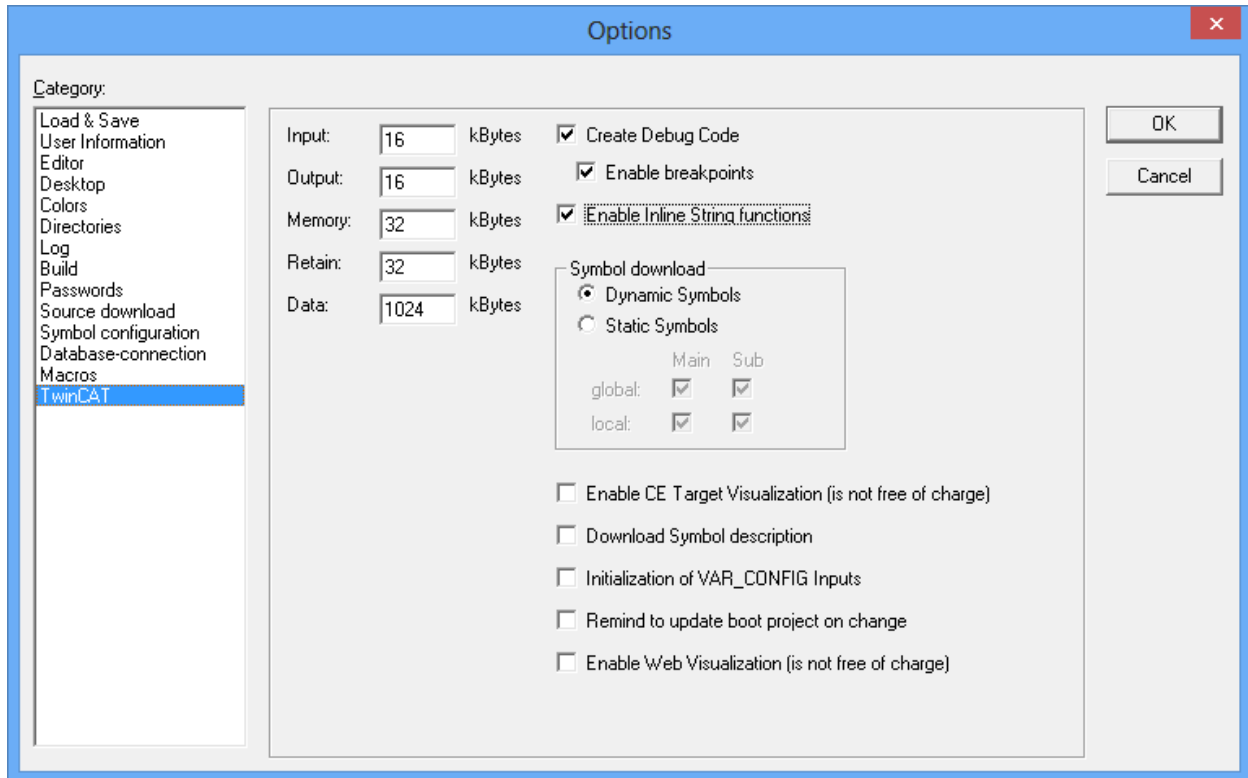
### 3.1.3 Build Options

Make sure the “check overlapping memory areas” option is selected.



### 3.1.4 TwinCAT Options

Make sure the “Enable inline string functions” option is selected. This will allow us to use the string functions in more than one task. By default they are not multi-thread safe. This option page also allows to change the allocation for the different memory regions. For large project these limits may have to be increased.



## 3.2 PLC Common Infrastructure

Each PLC project is required to implement the PlcInfo library. This library provides support for reporting errors at the PLC level, it exports the software subversion number and it provides a set of run-time information specific to the PLC.

### 3.2.1 Common libraries

The following libraries need to be imported for every project.

SaveRestore	Support for saving user interface settings to a persistent memory block. It also supports restoring these settings upon restart.
Error	Support for error handling and reporting.
PlcInfo	Support for reporting software version, run-time information and top level errors

### 3.2.2 Error Handling

Each user interface structure is required to contain an error reporting structure of type ErrorStruct. Error handling is supported with “Error” library. However, this abstraction cannot be propagated to the highest level because a PLC only covers a subset of the available channels. For instance, there are “H1:ALS” channels in the corner and both end stations. This means that each of the three

associated PLCs will have a user structure H1.Als defined. If it would contain an error structure, the names would collide. There is also no error at the H1 level for the same reason. Instead, the error hierarchical error structures stop at the level one below. For instance, corner station PLC would contain an error structure in the H1.Als.C structure, whereas the end stations would contain error structures in the H1.Als.X and H1.Als.Y structures, respectively. Within each PLC code each subsystem has a corresponding top level error reporting structure. These top level local error structures are combined in a PLC error structure which is reported in the PlcInfo structure. The PlcInfo structures are defined in the H1.Sys.EtherCAT structure. Each PLC has exactly one of these structures and the name of the structure contains the target designator. For instance, the user can access H1.Sys.EtherCat.C1Plc2, H1.Sys.EtherCat.X1Plc2 and H1.Sys.EtherCat.Y1Plc2 and the Error element therein to learn about errors occurring with PLC2 of corner, end X and end Y, respectively.

```

SysStruct
0001 (* This structure needs to be updated in synchronization with SysXStruct and SysYStruct *)
0002 TYPE SysStruct :
0003 STRUCT
0004     EtherCat:           EtherCatStatusStruct;
0005 END_STRUCT
0006 END_TYPE
0007
0008

SysXStruct
0001 (* This structure needs to be updated in synchronization with SysStruct and SysYStruct *)
0002 TYPE SysXStruct :
0003 STRUCT
0004     EtherCat:           EtherCatStatusXStruct;
0005 END_STRUCT
0006 END_TYPE
0007

EtherCATStatusStruct
0001 (* This structure needs to be updated in synchronization with EtherCATStatusXStruct and EtherCAT
0002 TYPE EtherCATStatusStruct :
0003 STRUCT
0004     End1Plc2:           PlcInfoStruct;
0005 END_STRUCT
0006 END_TYPE
0007

EtherCATStatusXStruct
0001 (* This structure needs to be updated in synchronization with EtherCATStatusStruct and EtherCATS
0002 TYPE EtherCATStatusXStruct :
0003 STRUCT
0004     X1Plc2:             PlcInfoStruct;
0005 END_STRUCT
0006 END_TYPE
0007
  
```

Since there is only one end station program, a little bit of magic with the overlapping H1 and Ifo variables is required to make it all work.

```

Global_Variables_Version C:\SlowControls\TwinCAT\Source\Current\Interferometer\Version.exp
0001 VAR_GLOBAL CONSTANT
0002     SvnRevision:      DINT := 0;
0003 END_VAR
0004
0005
Global_Variables_Ifo C:\SlowControls\TwinCAT\Source\Current\Interferometer\End\IfoVar.exp
0001 VAR_GLOBAL CONSTANT
0002     IfoIdEnum := IfoT1; (* IfoH1, IfoL1 or IfoH2 *)
0003     LocationIdEnum := EndX; (* Corner, EndX or EndY, MidX or MidY *)
0004 END_VAR
0005 VAR_GLOBAL
0006     (* Must reflect intrferometer and X/Y end station *)
0007     T1 AT %MBO:      IfoXStruct;
0008     (*~
0009     (OPC              :1      : visible for OPC-Server)
0010     (OPC_PROP[8610]  :Plc2: OPC-Server name)
0011     *)
0012 END_VAR
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
0100
0101
0102
0103
0104
0105
0106
0107
0108
0109
0110
0111
0112
0113
0114
0115
0116
0117
0118
0119
0120
0121
0122
0123
0124
0125
0126
0127
0128
0129
0130
0131
0132
0133
0134
0135
0136
0137
0138
0139
0140
0141
0142
0143
0144
0145
0146
0147
0148
0149
0150
0151
0152
0153
0154
0155
0156
0157
0158
0159
0160
0161
0162
0163
0164
0165
0166
0167
0168
0169
0170
0171
0172
0173
0174
0175
0176
0177
0178
0179
0180
0181
0182
0183
0184
0185
0186
0187
0188
0189
0190
0191
0192
0193
0194
0195
0196
0197
0198
0199
0200
0201
0202
0203
0204
0205
0206
0207
0208
0209
0210
0211
0212
0213
0214
0215
0216
0217
0218
0219
0220
0221
0222
0223
0224
0225
0226
0227
0228
0229
0230
0231
0232
0233
0234
0235
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247
0248
0249
0250
0251
0252
0253
0254
0255
0256
0257
0258
0259
0260
0261
0262
0263
0264
0265
0266
0267
0268
0269
0270
0271
0272
0273
0274
0275
0276
0277
0278
0279
0280
0281
0282
0283
0284
0285
0286
0287
0288
0289
0290
0291
0292
0293
0294
0295
0296
0297
0298
0299
0300
0301
0302
0303
0304
0305
0306
0307
0308
0309
0310
0311
0312
0313
0314
0315
0316
0317
0318
0319
0320
0321
0322
0323
0324
0325
0326
0327
0328
0329
0330
0331
0332
0333
0334
0335
0336
0337
0338
0339
0340
0341
0342
0343
0344
0345
0346
0347
0348
0349
0350
0351
0352
0353
0354
0355
0356
0357
0358
0359
0360
0361
0362
0363
0364
0365
0366
0367
0368
0369
0370
0371
0372
0373
0374
0375
0376
0377
0378
0379
0380
0381
0382
0383
0384
0385
0386
0387
0388
0389
0390
0391
0392
0393
0394
0395
0396
0397
0398
0399
0400
0401
0402
0403
0404
0405
0406
0407
0408
0409
0410
0411
0412
0413
0414
0415
0416
0417
0418
0419
0420
0421
0422
0423
0424
0425
0426
0427
0428
0429
0430
0431
0432
0433
0434
0435
0436
0437
0438
0439
0440
0441
0442
0443
0444
0445
0446
0447
0448
0449
0450
0451
0452
0453
0454
0455
0456
0457
0458
0459
0460
0461
0462
0463
0464
0465
0466
0467
0468
0469
0470
0471
0472
0473
0474
0475
0476
0477
0478
0479
0480
0481
0482
0483
0484
0485
0486
0487
0488
0489
0490
0491
0492
0493
0494
0495
0496
0497
0498
0499
0500
0501
0502
0503
0504
0505
0506
0507
0508
0509
0510
0511
0512
0513
0514
0515
0516
0517
0518
0519
0520
0521
0522
0523
0524
0525
0526
0527
0528
0529
0530
0531
0532
0533
0534
0535
0536
0537
0538
0539
0540
0541
0542
0543
0544
0545
0546
0547
0548
0549
0550
0551
0552
0553
0554
0555
0556
0557
0558
0559
0560
0561
0562
0563
0564
0565
0566
0567
0568
0569
0570
0571
0572
0573
0574
0575
0576
0577
0578
0579
0580
0581
0582
0583
0584
0585
0586
0587
0588
0589
0590
0591
0592
0593
0594
0595
0596
0597
0598
0599
0600
0601
0602
0603
0604
0605
0606
0607
0608
0609
0610
0611
0612
0613
0614
0615
0616
0617
0618
0619
0620
0621
0622
0623
0624
0625
0626
0627
0628
0629
0630
0631
0632
0633
0634
0635
0636
0637
0638
0639
0640
0641
0642
0643
0644
0645
0646
0647
0648
0649
0650
0651
0652
0653
0654
0655
0656
0657
0658
0659
0660
0661
0662
0663
0664
0665
0666
0667
0668
0669
0670
0671
0672
0673
0674
0675
0676
0677
0678
0679
0680
0681
0682
0683
0684
0685
0686
0687
0688
0689
0690
0691
0692
0693
0694
0695
0696
0697
0698
0699
0700
0701
0702
0703
0704
0705
0706
0707
0708
0709
0710
0711
0712
0713
0714
0715
0716
0717
0718
0719
0720
0721
0722
0723
0724
0725
0726
0727
0728
0729
0730
0731
0732
0733
0734
0735
0736
0737
0738
0739
0740
0741
0742
0743
0744
0745
0746
0747
0748
0749
0750
0751
0752
0753
0754
0755
0756
0757
0758
0759
0760
0761
0762
0763
0764
0765
0766
0767
0768
0769
0770
0771
0772
0773
0774
0775
0776
0777
0778
0779
0780
0781
0782
0783
0784
0785
0786
0787
0788
0789
0790
0791
0792
0793
0794
0795
0796
0797
0798
0799
0800
0801
0802
0803
0804
0805
0806
0807
0808
0809
0810
0811
0812
0813
0814
0815
0816
0817
0818
0819
0820
0821
0822
0823
0824
0825
0826
0827
0828
0829
0830
0831
0832
0833
0834
0835
0836
0837
0838
0839
0840
0841
0842
0843
0844
0845
0846
0847
0848
0849
0850
0851
0852
0853
0854
0855
0856
0857
0858
0859
0860
0861
0862
0863
0864
0865
0866
0867
0868
0869
0870
0871
0872
0873
0874
0875
0876
0877
0878
0879
0880
0881
0882
0883
0884
0885
0886
0887
0888
0889
0890
0891
0892
0893
0894
0895
0896
0897
0898
0899
0900
0901
0902
0903
0904
0905
0906
0907
0908
0909
0910
0911
0912
0913
0914
0915
0916
0917
0918
0919
0920
0921
0922
0923
0924
0925
0926
0927
0928
0929
0930
0931
0932
0933
0934
0935
0936
0937
0938
0939
0940
0941
0942
0943
0944
0945
0946
0947
0948
0949
0950
0951
0952
0953
0954
0955
0956
0957
0958
0959
0960
0961
0962
0963
0964
0965
0966
0967
0968
0969
0970
0971
0972
0973
0974
0975
0976
0977
0978
0979
0980
0981
0982
0983
0984
0985
0986
0987
0988
0989
0990
0991
0992
0993
0994
0995
0996
0997
0998
0999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400

```



```

0001 PROGRAM MAIN
0002 VAR
0003     SaveRestore:           SaveRestoreFB;
0004     GotoSafe:              BOOL;
0005     Request:               SaveRestoreEnum;
0006
0007     IfoErrorHandler:       IfoErrorHandlerFB;
0008     IfoError:               ErrorStruct;
0009     PlcInfo:                PlcInfoFB;
0010 END_VAR
0011
0001 SaveRestore( SaveInterval := T#1m,
0002               GotoSafe := GotoSafe,
0003               Request => Request );
0004
0005 ALS();
0006 ASC();
0007 ISC();
0008
0009 IfoErrorHandler(Ifo := Ifo, Error := IfoError);
0010
0011 PlcInfo (Request := Request, Ifo := IfoId, Loc := LocId,
0012         Svn := SvnRevision, Status := IfoError,
0013         Plc := Ifo.Sys.EtherCat.End1Plc2);
0014
0015
0016
0017
0001 ErrorHandler(ErrorMethod := Init, Error := Error);
0002
0003 IF Ifo.ALS.End.Error.Flag THEN
0004     ErrorHandler(ErrorMethod := Report, Error := Error, Code := 1, Msg := 'Als');
0005 END_IF
0006
0007 IF Ifo.ASC.End.Error.Flag THEN
0008     ErrorHandler(ErrorMethod := Report, Error := Error, Code := 2, Msg := 'Asc');
0009 END_IF
0010
0011 IF Ifo.ISC.Rf.End.Error.Flag THEN
0012     ErrorHandler(ErrorMethod := Report, Error := Error, Code := 4, Msg := 'Rf');
0013 END_IF
0014
0015 ErrorHandler(ErrorMethod := Commit, Error := Error);
0016
0017

```

### 3.2.3 Software version

The subversion revision is a number describing a snapshot of the software in the archive. The installation scripts for a PLC will make sure the archive is up-to-date before compiling the project. The active subversion number is imported into the code and available to the user as part of the PlcInfo structure.

### 3.2.5 Run-time information

Let's take a look at the most important elements of the PlcInfo structure.

IfoId	Interferometer identification, e.g. H1 or L1
LocationId	Location of the computer, i.e., Corner, EndX or EndY
Status	Top level error structure of the PLC
SvnRevision	Subversion revision number
StartTime	Time the PLC was started
CurrentTime	Current system time
Hostname	Computer name of the machine running the PLC
CpuUsage	Percentage of CPU usage
SysLatencyActual	Actual system latency
SysLatencyMax	Maximum system latency

## 4 Naming Scheme

### 4.1 Names

Generally, verbose and descriptive names are preferred to short and abbreviated ones. This will make the code more readable and help in maintenance and support. For example, `Index` is preferred over `I` and `TimingMasterFanout` is preferred over `Tmfo`.

#### 4.1.1 Variable Names

The naming of variables preferably should be unique in all libraries, following the camel case notation: For each variable a meaningful, preferably short, English name should be used, the base name. Always the first letter of a word of the base name is to be written uppercase, the remaining letters lowercase; example: `FastGain` or `InputOffset`. Abbreviations are written starting with an uppercase and then all lower case; example: `VcoGain` or `TimingMasterFanout`. Pointer variables shall use the suffix **Ptr**, whereas constant variables may use the suffix **Const**.

#### 4.1.2 Type Names

Type names follow the same rule as variable names. A complex type shall incorporate a suffix to denote its derivation: **Enum** for `ENUM`, **Struct** for `STRUCT` and **Array** for `ARRAY`.

Structure members follow the rules of variables.

#### 4.1.3 Function and Method Names

Function and method names follow the same rules as variables but with the suffix **Fun**. Internal helper functions such as conversion routines can also use a lowercase name, so that they look more in line with mathematical notation.

#### 4.1.4 Function Block Names

The names of function blocks follow the same rules as variables but with the suffix **FB**. Interfaces in TwinCAT 3 use the suffix **I**.

#### 4.1.5 Names of Visuals

Visual interfaces have the suffix **Vis**.

### 4.1.6 Suffix Summary

Element	Description	suffix
Constant	Constant value (optional, may be clear from context)	Const
Pointer	Pointer to a variable	Ptr
ENUM	Enumerated type	Enum
STRUCT	Record type	Struct
ARRAY	Array type	Array
Function	Function or Method declaration	Fun
Function block	Function block declaration	FB
Interface	Abstract function block or interface	I
Visual	Screen interface for diagnostics	Vis

**Table 5: Required suffix notation.**

## 4.2 Hardware Channels

Variables that are connected to hardware channels are separated into input variables and output variables. They must be located in the input and output shared memory regions, respectively. A variable describing a list of input channels must have the suffix **In**. The corresponding structure must have the suffix **InStruct**. An output channel list uses the suffix **Out**, whereas the output structure uses **OutStruct**. Channels with different cycle time must be placed into different structures. The above names are for the standard cycle time of 10 ms. Channels that need to be updated at the fast rate need to prepend **Fast** to the above suffixes.

Element	Description	suffix
Input variable	Input variable with standard update rate	In
Output variable	Output variable with standard update rate	Out
Input variable	Input variable with fast update rate	FastIn
Output variable	Output variable with fast update rate	FastOut
Input STRUCT	Input channel structure with standard update rate	InStruct
Output STRUCT	Output channel structure with standard update rate	OutStruct
Input STRUCT	Input channel structure with fast update rate	FastInStruct
Output STRUCT	Output channel structure with fast update rate	FastOutStruct

**Table 6: Input and output channel notation.**

A code fragment declaring input and output channels in the global variable space:

```
PicoMotorFastIn  AT %IB0100: PicoMotorFastInStruct;  
PicoMotorFastOut AT %QB0200: PicoMotorFastOutStruct;  
PicoMotorIn      AT %IB0102: PicoMotorInStruct;  
PicoMotorOut     AT %QB0204: PicoMotorOutStruct;
```

## 4.3 Library Objects

### 4.3.1 Name Space

Libraries can optionally choose a name space following the variable name notation. This name space is then used to prefix all exported objects. For example: the library `TimingMasterFanout` has the name space prefix `Timing`. Within this library `TimingSlaveDuoToneStructure`, `TimingReadSlaveFun` and `TimingMasterFanoutFB` are a valid structure, function and function block, respectively.

Simple libraries that consist of an input structure, an output structure, an interface structure and a function block are not required to choose an explicit name space, but are expected to use the library name as the base for all four objects. Hence, they are defining an implicit name space with the same name as the library name. For example: the library `CommonMode` may contain the structures `CommonModeInStruct`, `CommonModeOutStruct` and `CommonModeStruct` as well as the function block `CommonModeFB`.

### 4.3.2 Folder Names

Program object units (POUs) and data types are organized in folders. These folders are purely organizational and are intended to help grouping items together for easier maintenance. In a library all exported types, functions and function blocks are typically located at the top level. If there are many objects, it may make sense to group them into folders. In any case, internal objects should always be moved into a folder named `Internal`.

## 4.4 External Tags

External tags (channels) are organized in a hierarchical structure. Each system defines its own structure. This continues with structures for subsystems that are contained in the system structures.

```

TYPE AlsEndStruct :
STRUCT
    Laser:      ALSLaserStruct;
    VCO:        LowNoiseVcoStruct;
    PZT1:       PZTMirrorStruct;
    PZT2:       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;
    Asc:        AscStruct;
    Lsc:        LscStruct;
    Tcs:        TcsStruct;
END_STRUCT
END_TYPE;

VAR_GLOBAL PERSISTENT
    IFO:        IfoStruct;      (*~
    (OPC          :1      : visible for alias)
    (OPC_Filter   :Export: Filter string for OPC server)
    (OPC_PROP[8610] :Plc1: OPC server name)
    (OPC_PROP[8620] :.${IFO}: alias name)
    *)
END_VAR;

```

This allows for exporting the entire interferometer interface structure at once and it allows for generating tag names automatically while preserving the hierarchical organization.

## 5 OPC Access and Properties

### 5.1 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 for alias)
...   *)
```

will make the entire IFO variable with all its sub elements visible through the OPC interface. In turn, it can be interfaced to EPICS. Individual tags such as the FastGain of the LaserServo will be available from the OPC server as “H1.Isc.Als.LaserServo.FastGain”. The default EPICS channel name constructed from this tag will then become “H1:Isc-Als\_LaserServo\_FastGain”. Be aware that IEC 61131-3 names are not case sensitive. The same is true for the corresponding TwinCAT OPC names, whereas EPICS channel names are case sensitive.

### 5.2 OPC Properties

OPC properties are used to further describe the external tags. These properties are also used to fill in the EPICS database fields. The properties have to be attached to the elements at the end of the hierarchical structure. These are variables with a basic type like INT or LREAL. Due to the program organization most of these variables are defined in libraries through structures. Therefore, the OPC properties are written after the structure elements using the OPC comment structure. For example:

```
TYPE LowNoiseVcoStruct :
STRUCT
  (* output tags *)
  PowerOk:      BOOL; (*~
                    (OPC_PROP[0005] :1: read-only)
                    (OPC_PROP[0101] :Voltage monitor readback: DESC)
                    (OPC_PROP[0106] :OK: ONAM)
                    (OPC_PROP[0107] :OOR: ZNAM) *)
  TuneMon:      LREAL; (*~
                    (OPC_PROP[0005] :1: read-only)
                    (OPC_PROP[0101] :Frequency offset monitor: DESC)
                    (OPC_PROP[0100] :V: EGU)
                    (OPC_PROP[0103] :-10: LOPR)
                    (OPC_PROP[0102] :+10: HOPR)
                    (OPC_PROP[8500] :3: PREC) *)
  ...
END_STRUCT
END_TYPE;
```

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	EPICS 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 7: Supported OPC properties.**



Only a small 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 above table.

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 are currently not supported.

The alias property supports replacement rules.

### 5.3 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.

Type	Description	
longin/longout	SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT, BYTE, WORD, DWORD, LWORD	
bi/bo	BOOL	
mbbi/mbbo	Enumerated data type with 16 or fewer labels	
stringin/stringout	STRING	
ai/ao	REAL, LREAL, any other	

**Table 8: Automatic type support.**

### 5.4 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 TwinCAT, 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 specifically. 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.

## 5.5 Array Variables

Array variables are supported by IEC 61131-3 and can be exported through OPC as well. They will also be accessible through EPICS, but require an extension to the LIGO channel naming convention. For example, if the structure “L1.Io.Wfs1” contains the members:

```

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

Gain:    ARRAY [1..4] OF LREAL;
Rotation: ARRAY [1..4,1..4] OF LREAL;
Signal:  ARRAY [1..4] OF DemodComplex;

```

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

Type	OPC name	EPICS name
LREAL	L1.Io.Wfs1.Gain[ $m$ ]	L1:Io-Wfs1_Gain[ $m$ ]
LREAL	L1.Io.Wfs1.Rotation[ $m$ ][ $n$ ]	L1:Io-Wfs1_Rotation[ $m$ ][ $n$ ]
LREAL	L1.Io.Wfs1.Signal[ $m$ ].I	L1:Io-Wfs1_Signal[ $m$ ].I
LREAL	L1.Io.Wfs1.Signal[ $m$ ].Q	L1:Io-Wfs1_Signal[ $m$ ].Q

**Table 9: Array variables with OPC and EPICS.**

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

## 5.6 ICS Database Generation

All EPICS fields must be defined in the PLC through OPC properties. TwinCAT will automatically generate an XML file with the extension “.tpy” which can be parsed to generate an EPICS database file. The program EpicsDbGen is available to do this. It is called from the command line as follows:

```
Usage: EpicsDbGen ['options'] -i 'input' -o 'output'
Generates an EPICS database from a TwinCAT tpy file.
-ea exports all variables regardless of their opc setting
-l[l][a|e|b] generate an [extended] [atomic|epics|burt] channel listing
-r[n|d] [no|dot] conversion rule for EPICS names
-c[u|l] force upper/lower case for EPICS names
-nd eliminates leading dot
-ni replaces array brackets with underscore
-ns ignores channels of type string
-sio splits database into input only and input/output records
-sn 'num' splits database into files with no more than num records
-i 'input' input file name (stdin when omitted)
-o 'output' output database file (stdout when omitted)
```

The input file is the TwinCAT file with the extension “.tpy”. The output file can either be a list of channels or an EPICS database file depending on the arguments. If no input file is specified, input from the standard input is taken. If no output file is specified, the output is written to the standard output.

The argument “-a” specifies that all global variables will be exported. The “-l” argument generates a channel name list rather than a database file. With the “-ll” extension a long list is generated. Normally, the case of the TwinCAT variable is preserved. However, the option “-cu” forces all upper case names, whereas the option “-cl” forces all lower case names. If the option “-ni” is specified, array indices of the form “[n]” are replaced by “\_n” when translating to the EPICS channel name.

The option “-sio” can be used to split the database records between input and in/out into separate files. The option “-sn N” can be used to split files, so that they contain no more than N records. Both options can be used individually or combined. If either option is used, an output file has to be specified. For example, if the file name is “PLC.db”, the “-sio” option will generate two files “PLC.in.db” and “PLC.io.db”. With the option “-sn 1000”, we will get “PLC.001.db”, “PLC.002.db”, etc. Each file but the last will contain exactly 1000 records. Both options together will generate files of the form “PLC.in.001.db” and “PLC.io.001.db”.

## 6 Documentation

A template for documenting a TwinCAT library exists in the DCC, [F1200003](#). It contains the project information, a description of the function blocks as well as detailed listing of the input and output types. Some specialized libraries may require additional information for functions, interfaces or global variables. An example can be found in [E1200226](#).

### 6.1 Project Information

The following project information is required: title, version, name space, author and a short description.

Field	Description	Mandatory
Title	Name of the library, usually in camel case, e.g., LowNoiseVco	Yes
Version	Library version number, usually 1, 2, etc.	Yes
TwinCAT	Version of TwinCAT for which the library was developed	Yes
Name space	Name space of the library	Yes, if exists
Author	Name of the programmer	Yes
Description	Short description of the purpose of the library	Yes
Error code	Lists the available error codes	Yes

**Table 10: Project Information.**

### 6.2 Type Information

Each external type of a library require the following information: name, definition and short description. For a complex type each element should contain a short description as well.

Field	Description	Mandatory
Type name	Name of the type, e.g., LowNoiseVcoStruct	Yes
Definition	Type definition used by the library	Yes
Description	Short description of the purpose of the type	Yes
Elements	For complex types a list of elements	Yes, if exist

**Table 11: Type Information.**

### 6.3 Global Variables

Generally, there should be no need for global variables in a library. If they exist, the following information is required: name, type, a possible initialization value and a short description.

Field	Description	Mandatory
Variable name	Name of the global variable	Yes
Type	Type of the global variable	Yes
Initialization	Initialization value of the variable	Yes, if exist
Description	Short description of the purpose of the variable	Yes

**Table 12: Global variables.**

### 6.4 Interfaces

In TwinCAT 3 abstract classes are called interfaces. They contain a list of abstract methods. Each interface definition requires name, list of methods and a short description.

Field	Description	Mandatory
Interface name	Name of the type, e.g., LowNoiseVcoStruct	Yes
Methods	List of methods used by the interface	Yes
Arguments	Each method can have a list of arguments	Yes, if exist
Description	Short description of the purpose of the interface	Yes

**Table 13: Interfaces.**

### 6.5 Functions

Each function requires the following information: name, return type, list of input parameters, list of output parameters, list of in/out parameters and a short description.

Field	Description	Mandatory
Name	Name of the, e.g., TimingSlaveDuoToneReadFunc	Yes
Return	Return type	Yes
Inputs	List of input parameters	Yes, if exist
Outputs	List of output parameters	Yes, if exist
In/Outs	List of in/out parameters	Yes, if exist
Description	Short description of the purpose of the function or function block	Yes

**Table 14: Functions.**

## 6.6 Function Blocks

Each function and function block requires the following information: name, list of input parameters, list of output parameters, list of in/out parameters and a short description. In TwinCAT 3 function block are treated as classes and can extend a base class, inherit from an interface definition and contain methods. If used, the information of all class elements are required.

Field	Description	Mandatory
Name	Name of the function or function block, e.g., LowNoiseVcoFB	Yes
Parent	For classes that extend a parent function block	Yes, if exist
Interfaces	For classes that implement an interface	Yes, if exist
Inputs	List of input parameters	Yes, if exist
Outputs	List of output parameters	Yes, if exist
In/Outs	List of in/out parameters	Yes, if exist
Methods	List of methods used by the function block	Yes, if exist
Description	Short description of the purpose of the function or function block	Yes

**Table 15: Function blocks.**

## 6.7 Visuals

Each visual screen element requires the following information: screen snapshot, name, a short description and a list of placeholders. Placeholders are parameters denoted by \$paramter\_name\$ in the visuals that are required to be defined when the visual is embedded. Since the visual of a library usually represents an interface structure, there should be at least one placeholder parameter denoting a variable of this type.

Field	Description	Mandatory
Name	Name of the function or function block, e.g., lscWhiteningVis	Yes
Description	Short description of the purpose of the function or function block	Yes
Placeholder	Parameters used for variable substitution	Yes, if exist

**Table 16: Visuals.**