IDM UID
3QPL4H
VERSION CREATED ON / VERSION / STATUS
30 Jan 2013 / 1.4/ Approved
EXTERNAL REFERENCE

IT Technical Specifications

PLC Software Engineering Handbook

This document lists the rules and guidelines applicable to the development of software for PLCs deployed on the ITER project.

Approval Process			
	Name	*Action*	*Affiliation*
Author	**Evrard B.**	**30-Jan-2013:signed**	**IO/DG/DIP/CHD/CSD/PCI**
CoAuthor	**Prasad S.**	**12-Feb-2013:signed**	**IO/DG/DIP/CHD/CSD/PCI**
Reviewers	**Wallander A.**	**12-Feb-2013:recommended**	**IO/DG/DIP/CHD/CSD**
Approver	**Thomas P.**	**24-Mar-2013:approved**	**IO/DG/DIP/CHD**
Document Security: level 1 (IO unclassified) *RO: Evrard Bruno*			
Read Access	**LG: QA DOC Editors, AD: ITER, AD: External Collaborators, AD: Division - Control System Division - EXT, AD: Section - CODAC - EXT, AD: Section - CODAC, AD: Section - Plant Control and Instrumentation, project administrator, RO, LG: PLC group, LG: CODAC team**		

Change Log				
Title (Uid)	***Version***	***Latest Status***	***Issue Date***	***Description of Change***
PLC Software Engineering Handbook (3QPL4H_v1_4)	v1.4	Approved	30 Jan 2013	V7.0 Update
PLC Software Engineering Handbook (3QPL4H_v1_3)	v1.3	Approved	09 Feb 2011	Version after external review of PCDH V6.
PLC Software Engineering Handbook (3QPL4H_v1_2)	v1.2	Signed	10 Jan 2011	Integration of John Poole Comments. Ready for External Review.
PLC Software Engineering Handbook (3QPL4H_v1_1)	v1.1	Signed	04 Jan 2011	Version after CODAC internal review. Version ready for external review.
PLC Software Engineering Handbook (3QPL4H_v1_0)	v1.0	In Work	06 Dec 2010	

PLC Software Engineering

Technical note

Abstract

This document is listing the applicable rules and guidelines to be applied for the development of Software for PLCs deployed on the ITER project.

	External Number: ITER_D_3QPL4H v 1.0	**Date:** 13 September 2010
	Name	**Affiliation**
Author	Evrard Bruno	IO/DG/DIP/CHD/CODAC
CoAuthor	Prasad Sawantdesai	
Reviewers	ITER I&C IPT	IO/DG/DIP/CHD/CODAC
Approver	D Bora	IO/DG/DIP/CHD

Document Revision History

Version	Status	Date	Summary of Changes
1.0	Draft	13/12/2010	Draft
1.1	Issued	04/01/2011	First Version

Table of Contents

Table of Figures

1 Introduction

1.1 PCDH Context

The Plant Control Design Handbook (PCDH) [RD 10] defines methodology, standards, specifications and interfaces applicable to ITER plant systems Instrumentation & Control (I&C) system life cycle. I&C standards are essential for ITER to:

- Integrate all plant systems into one integrated control system.
- Maintain all plant systems after delivery acceptance.
- Contain cost by economy of scale.

PCDH comprises a core document which presents the plant system I&C life cycle and recaps the main rules to be applied to the plant system I&Cs for conventional controls, interlocks and safety controls. Some I&C topics will be explained in greater detail in dedicated documents associated with PCDH as presented in Figure 1.1. This document is one of them.

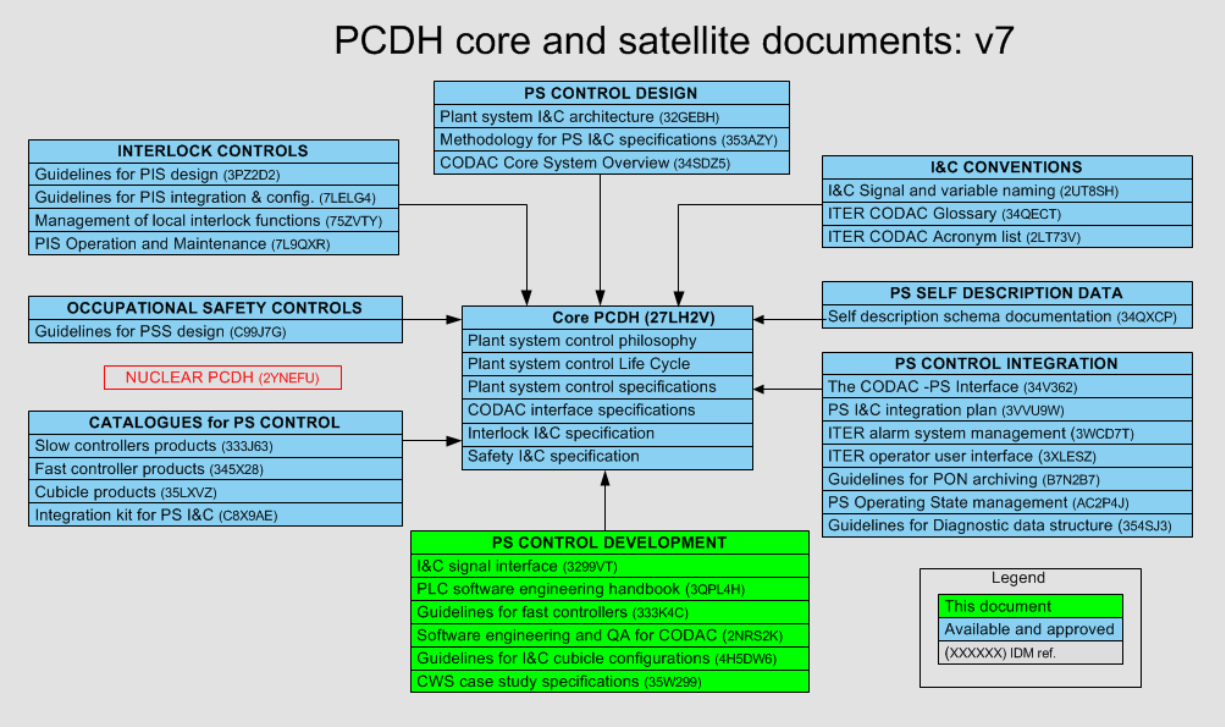

Figure 1 : Schema of PCDH documents

1.2 Purpose of document

This document intends to

- Define a standard software architecture for PLC applications developed in the ITER Project .

- Provide rules to have a standard approach for the development of the Control Functions.

1.3 Scope

This document covers the Development of Software for PLC Conventional Controllers. It is not covering SIL-3 PLCs, Simatic F/FH Series, Interlock (PIS) or Safety (PSS) Controllers.

1.4 Organization of document.

A preliminary Chapter will present the generic requirements that every PLC should fulfil. The rest of the document will give details on how to meet these requirements
The succession of the following chapters will follow as far as possible the PLC application development process followed by a Plant System I&C Programmer, trying to give all information in the order the programmer needs them:

- Standard Functional Architecture of the PLC Application
- Naming and Numbering Conventions required all along the development of the application
- Hardware Configuration of the PLC
- Standard PLC Software Structure.
- Interfaces
- Core Application Development
- Software Configuration Management
- Examples and Templates

In the document the following markers will precede some paragraphs:

[NR<w>] for naming rules,

[CR<x>] for coding rules,

[RD<y>] for reference documents,

[D<z>] for reference to PCDH ([RD 10]) Deliverables.

These markers will be referenced in the document.

1.5 Acronyms

SSPS	Standard Software PLC Structure
PLC	Programmable Logic Controller
FBS	Functional Breakdown Structure
PBS	Process Breakdown Structure
CBS	Component Breakdown Structure
FC	Function Chart
FB	Function Block

DB	Data Block
UDT	User Data Type
SFC	System Function Chart
SFB	System Function Block
FBD	Functional Block Diagram
LAD	Ladder Diagram
CFC	Continuous Flow Chart
SDD	Self Description Data
PSH	Plant System Host
COS	Common Operating State
PCDH	Plant Control Design Handbook
I/Q	Inputs/Outputs

1.6 Definitions

PLC Application	All Software developed in a PLC
PLC Core Application	All Software or Control Blocks implementing the Control Functions. All what is not implemented in the Peripheral Blocks
Shared DB variable	Generic term used for any variable in a PLC
Process Variable	Generic term used for a Variable in the EPICS environment.
Plant System I&C Programmer	Person Responsible of Programming CODAC, PLC or Fast Controller applications.
Configuration Database	
Peripheral Blocks	PLC software Blocks implementing the interfaces and the Health Monitoring.
Configuration	Set of all configuration variables for a PLC.
Configuration Variable	An EPICS PV transmitted to the PLC through a Shared DB Variable.
States	Set of all PLC Shared DB variables transmitted to the CODAC.
State Variable	A PLC Shared DB variable transmitted to the CODAC through an EPICS PV.
Standard Control Block Interface	In and Out parameters of a FC or a FB for a Control Block deployed in the PLC Core Application
Control Function	Function achieved by a Controller in the Context of a Functional Analysis.
Control Block	FC of FB in the Context of a Siemens Step 7 application.

1.7 Reference Documents

	IDM Number	Title
[RD 1]	2UT8SH	"I&C Signal and Process Variable Naming Convention"
[RD 2]	28QDBS	"ITER numbering system for parts/components"
[RD 3]	34V362	"The CODAC – Plant System Interface"
[RD 4]	353AZY	"Methodology for PS I&C design"

[RD 5]	2FJMPY	"ITER Function Category and Type for ITER Numbering System"
[RD 6]	32GEBH	"Plant System I&C Architecture"
[RD 7]	32Z4W2	"Self-description data editor - User manual"
[RD 8]	35W299	"Cooling Water System Prototype Specification"
[RD 9]	333J63	Siemens S7 PLC Catalogue
[RD 10]	27LH2V	Plant Control Design Handbook

2 Context and Constraints

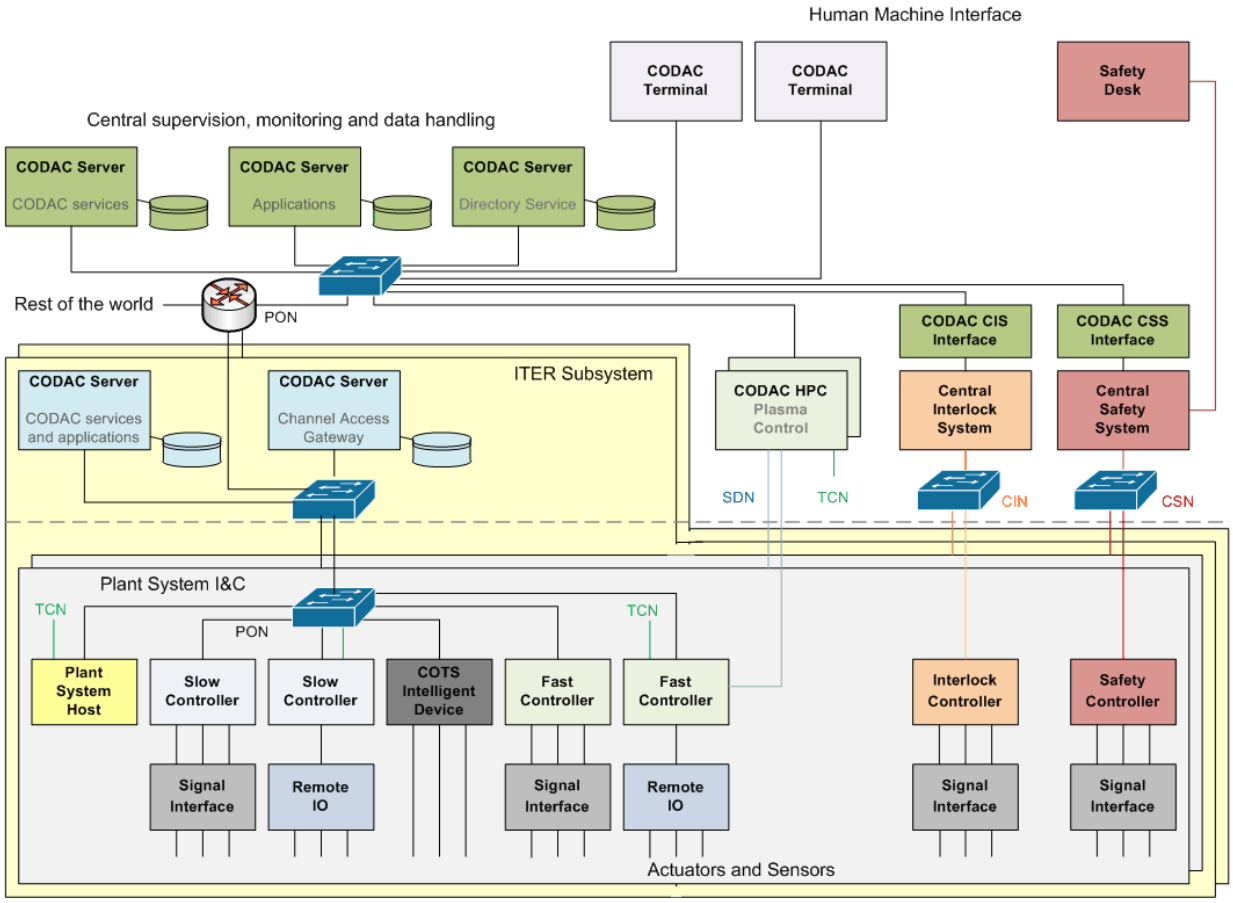

Figure 2: CODAC Architecture

The architecture of Plant System I&C is defined in [RD 6]. The PLCs will communicate with the CODAC through the PSH. The PSH is a standard computer running EPICS. Its configuration will be generated for each Plant System I&C. The communication with Step7 PLCs will be done through TCP/IP Socket communication. The general structure of the frames has already been settled.

The PSH will implement the COS that has to be synchronized with the State of the PLC.

PLCs inside a Plant System may have functional interfaces with other PLCs, Fast Controllers and COTS Intelligent Devices. These interfaces will be supported by the PON.

3 Generic Requirements of PLC Applications on ITER

- Flexibility.
    - During integration and Commissioning, all interfaces may be not available. The application should give possibility to force some signals, or to simulate partially the missing interface.

- Maintainability
    - Enough system information of the system should be provided.
    - The PLC Application should be built in a way that modifications has only located impact.

- Ability to be tested.
    - Unit testing of PLC Functions should be made easier
    - Control Systems Software should be tested independently from the system. The idea is to test the Control System disconnected from the System and connected to a Simulator. The plant System has to define beforehand what Controllers has to be tested together.
- Readability
    - Every information transformation should be easy to track.

4 Software Architecture of a PLC application.

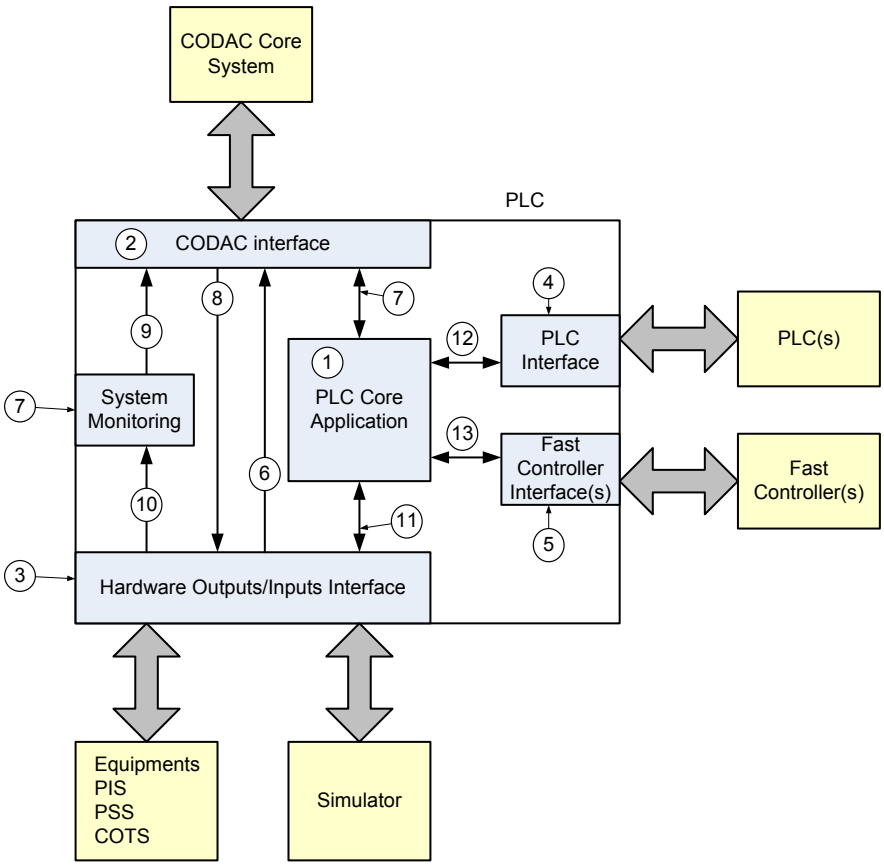

Figure 3: PLC Conceptual Architecture

The idea is to have a common architecture of the application inside all the PLCs deployed on the Project. Depending of PLC Application all the blocks might not be present. For example:

- A "Master Controller", in an I&C architecture (see [RD 6]) will not have any Hardware Inputs/ Outputs Interface and will have a lot of Interfaces with other PLCs of the Plant System.
- Fast Controllers Interfaces will probably be very rare and may use the CODAC interface, as Fast Controllers are running EPICS and are consequently connected to Channel Access. TBD

Except for the PLC Core Application, the inside structure of all other blocks will be standard for all PLCs deployed on the Project. Only the volume and structure of the datas computed in these blocks will be different. As far as possible, these blocks will be generated automatically, using the "Configuration Database" as input. The Codac Interface ("2"on *Figure 3*) is already fully generated by the SDD package. For the other blocks, the static inside structure will be developed in this document. Further generation activities will be based on these structures.

4.1 PLC Core Application

The PLC Core Application ("1"on *Figure 3*) is the place where the Control Logics, Grafcets, State Charts, Regulation Loops of the process will be implemented. In this place we should find only the process programming. The PLC Core Application will implement the "Control Functions." Its operation will be affected by all the interfaces represented on *Figure 3*. All programming or treatment not directly involving the process are performed in the other "Peripheral blocks" (Interfaces, System monitoring).

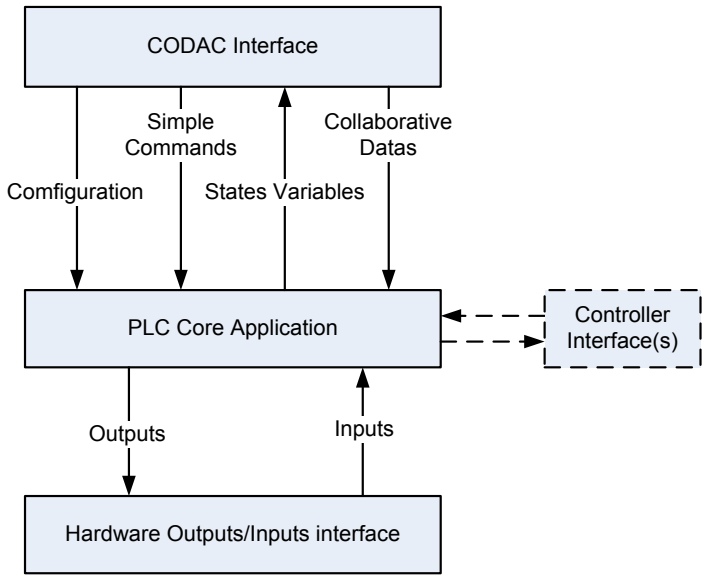

Figure 4: PLC Core Application Environment

The PLC Core application will use the configuration variables (See *Figure 4*) transmitted by the CODAC interface as main inputs from operation.
Some Configuration variables examples:

- ON/OFF requests
- OPEN/CLOSE requests,
- HIGH VACUUM/ROUGHING/VENTING request,
- Current Setpoint,
- Temperature Setpoint
- …

Hardware Inputs and Outputs (See *Figure 4*) are in their engineering format. The PLC Core Application, make a complete abstraction of the fact that these values are coming from real equipments or simulated or forced.
PLC Core Application will compute the CODAC Configuration variables and the Hardware Inputs and generate the outputs in order to reach the configuration requested. The States variables report the effective State of the process. The main principle is that on the CODAC, it is always possible to have an easy comparison between the state (configuration) that was requested to the Process, and the effective state of the Process. A simple example is given in *Figure 5* of what will be a CODAC HMI for a simple device.

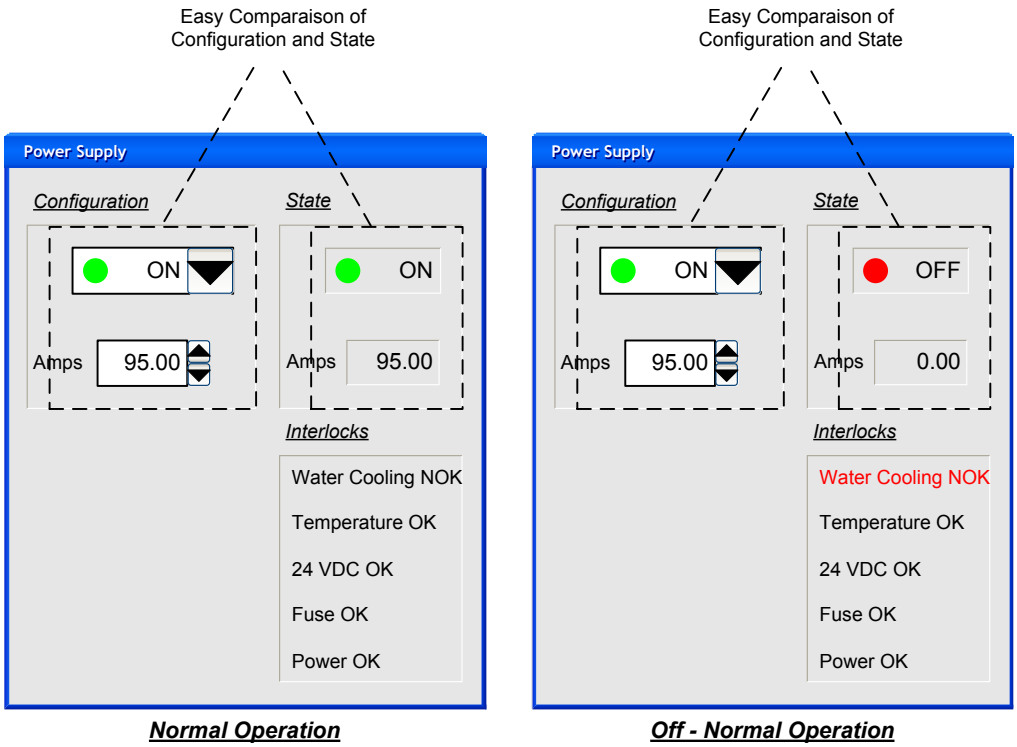

Figure 5: Simple Example of CODAC HMI

Collaborative Datas (See *Figure 4*) are State Variables produced by other Plant Systems and Transmitted by CODAC Core System. In PCDH transversal wired links between Plant System is strictly forbidden. Transmission of information between Plant System will use the Collaborative Data link.
The interfaces with other Controllers within the same Plant System I&C impacts also the processing, it will be developed in § 4.4 and § 4.5.

4.2 CODAC Interface

The main function of the CODAC Interface ("2"in *Figure 3*) is to manage the PLC side of the communication with the CODAC developed in an EPICS environment. The CODAC side of the communication is managed in the PSH running a specific driver.
This communication is broken down in 4 categories, as represented in *Figure 4*:

- Configuration Variables
- State Variables
- Simple Commands
- Collaborative Data

The main use of Configuration variables is developed in § 4.1. In *Figure 3* the link "8" represents another use of these variables: it will give configuration to the Hardware Outputs/Inputs Interface. Mainly, it will provide Physical to engineering conversion parameters, forcing values and inhibitions, it will also affect the simulation mode. It is developed in § 4.3

The States Variables are transmitting the state of the Process:

- Directly from the Hardware Inputs/Outputs Interface ("6"in ***Figure 3***). This direct link is necessary as the CODAC Core Applications will use these variables without computing required in the PLC Core Application.
  It is important to note here that these variables here are in their engineering values, they can also can be forced or simulated.
- From the computed variables issued by the PLC Core Application ("7"in ***Figure 3***).
- From the System Monitoring ("9"in ***Figure 3***).

Simple Commands are variables set to "TRUE" during one Cycle in the PLC. These simple Commands are used in the cases where it is not required to memorize the action related to this command, like with configuration variables.. Typical examples are "Reset" of some devices. Reset is not a stable configuration, it is a transient command.

Collaborative Data are state variables transmitted between Plant System I&Cs. A Strong requirement of the PCDH is a that no transversal wired link is allowed between Plant System I&Cs. This link will be a "Software link" between 2 PLCs from 2 different Plant System I&Cs. This collaborative Datas will be States Variables with a specific Status of "Collaborative Data".
Note, that if several Controllers have to share the same information (A temperature, a Pressure, …) it is important that this information has exactly the same origin.

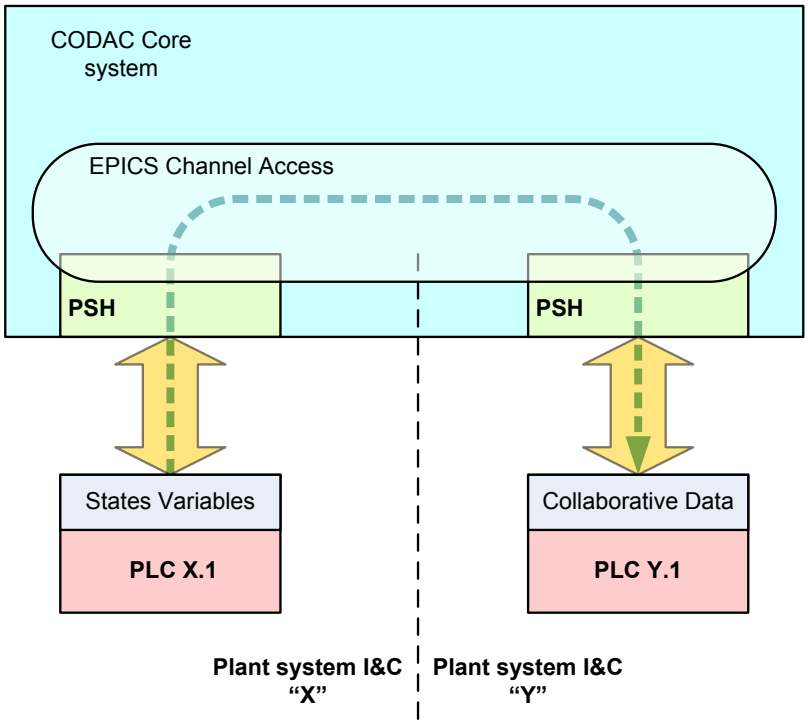

Figure 6: Collaborative Data

4.3 Hardware inputs/outputs interface

4.3.1 General Description

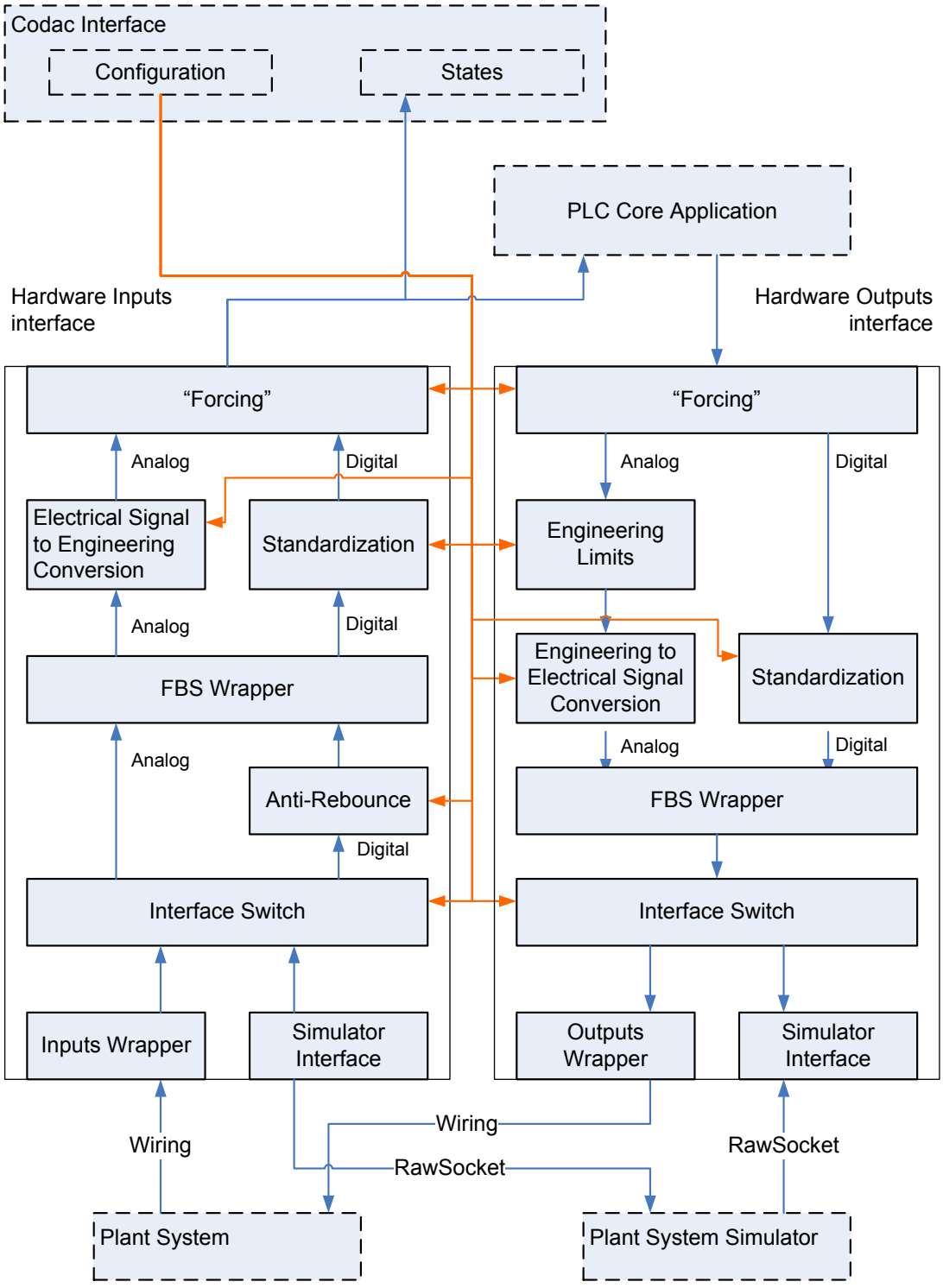

Figure 7: Hardware Inputs/Outputs Interface Block Diagram

The Hardware Interface is divided in two parts: inputs interface and outputs interface. Almost same functions are present in both parts but are processed in reverse order. In order to explain the working of this interface, here is the process flow of a wired input coming from a Plant System:

- The signal is wired between the Plant System and the Input Board of the PLC .

- In a first Software Function called the “Inputs Wrapper” (2), the signal is copied from an I/Q addressing area to a DB addressing area. Example Input “I0.0” is copied in “DB1.DBX0.0”. Note: PLC absolute addressing is used here for better understanding, but Symbols should be used. The signal is now becoming a Shared DB variable.
- If the signal is Boolean (coming originally from a digital signal), the variable is going through an “Anti-Rebounce” layer.
- The Shared DB variable is transmitted to an “Interface Switch” Block where it is chosen to use the wired signal or a signal coming from a Simulator. Another Shared DB variable is issued.
- The issued Shared DB variable is transmitted to a “FBS Wrapper”, where the variable is copied from a Component Naming Convention (“PPPPPP-TTT-NNNN:AAAASSSS”) to a FBS Level 3 convention (FBS-L3.variable). See [RD 1]. Another Shared DB variable is issued.
- From the “FBS Wrapper”, the Shared DB variable can have a different processing, depending if it is a numerical (coming originally form a analog signal) or a boolean variable (coming originally from a digital signal).
    - Numerical Variable: it is being transformed in an engineering value according to a linear regression, or a Look Up Table, etc… “Signal to Engineering Conversion” (5). Another Shared DB variable is issued.
    - Boolean Variable: here the Boolean value can be negated or not, depending on the logic the Developer wants to use in the Core Application. Example: in order to have a fail-safe logic, the status of a device could be notified by a “0V” signal, what it is more convenient to program a “TRUE” in the code. “Standardization Layer” (6). Another Shared DB variable is issued.
- The variable is going through a “Forcing” Layer (7), where its value can be forced by the user, for commissioning or maintenance purposes. The variable issued is the one issued by the Hardware Input Interface.
- The variable is systematically transmitted to the “States Variables” (8) transmission mechanism of the CODAC Interface. And can be used by the PLC Core Application (9).

The following paragraphs give a more detail description of t every layer.

4.3.2 Inputs/Outputs Wrapper

The purpose of this layer is to directly use at the lowest level aSiemens Data Blocks addressing area. (Shared DB variables) There are 2 advantages:

- Information can be organized in hierarchy in systems and subsystems with different depth.
- The whole volume of variables can be handled with only one simple block.

Complex interfaces like FM453 positioning modules, CP441 serial communications modules will also be implemented in this layer.
There is link between this layer and the Health Monitoring function. All the the variables issued by the Wrapper will be transmitted to the Health Monitoring System. The Health

Monitoring System transmits these variables to the CODAC interface. The purpose is to have the raw values of the CODAC available on a system screen. For debugging purposes.

4.3.3 *Interface Switch*

Connecting a process simulator to the controller will give the following possibilities:

- Validate the software without being connected to the process
- During integration and commissioning, modify the software and test these modifications on a different platform, before loading on the real control unit.

The Interface Switch is just Switching the origin of the signal variables to the real process or to a simulator. Whatever the Simulator is, we can consider that the interface will be a Data Block
The control of the Interface Switch will be a CODAC configuration variable (*Figure 3* – "10").
This command has to be secured in the sense that it cannot be operated during real operation.

4.3.4 *Anti-Rebounce*

TBD

4.3.5 *Engineering Limits*

For numerical outputs, it is necessary set limits expressed in engineering format, reflecting the limit of the actuator or of the physical process. If these limits are exceeded, the PLC output may be erroneous.
The limits will be set by configuration variables.

4.3.6 *FBS Wrapper*

This blocks simply copies the signal variables presented in a PBS naming convention (PPPPPP-TTT-NNN:AAAASSSS) to a FBS naming convention (FBS-L3.variable).

A segregation is made between digital and numerical signal variable because the above layers are different.

4.3.7 *Electrical Signal to Engineering/Engineering to Electrical Signal Conversion*

For most of the numerical signal variables, a conversion will be required. This conversion can be linear, quadratic, of superior orders. It can be also a look-up table.
All the conversion parameters will be provided by CODAC configuration variables.

4.3.8 *Standardization*

The idea here is to standardize the code as much as possible in the PLC Core Application. For a same type of devices, we should always control it with the same PLC function. The fact is that sometimes same type of devices will be wired with a different logic. If we take the example of a valve. The Limit Switches of some valves will be wired in a positive logic (24VDC – position reached) and some in a negative logic (0VDC – position reached). While the Control of the valve is the identical...

The function of this standardization block would be to process all the negation required to all discrete signals, in order to present a standard signal interface of the different types of devices to the PLC Core Application
All the negation parameters will be provided by CODAC configuration variables.

4.3.9 Forcing

During integration, commissioning and sometimes during maintenance, engineers will inevitably want to force some signal variables to a value, because the related signal is not connected, is missing, is not operational or is failing. This is the reality of highly integrated systems during non-operational phases. It is better to take this fact into account into the software design, so that this "unregular" (and can-be-dangerous) behaviour will be kept under control. The idea here is to avoid dangerous "temporary-permanent" practices like forcing some signal with PLC hardcoded modifications, hardwired modifications, screwdrivers sticked in the relays.
This forcing layer has to be developped. We can consider ie:

- some signals that cannot be forced at any time because impact can be destructive.
- The control unit (or the all I&C) cannot reach an operational state as long as signal variables are forced.
- Inhibiting this forcing feature.

All permanent and runtime parameters will be provided by CODAC configuration variables.

4.4 PLC Interface

This interface addresses communications between PLCs of a same Plant System I&C, in case a functional interface is required. The Siemens Protocol used will be defined later in the document.
From conceptual point of view, we can consider 3 different cases:

- It can be master/slave link where the master PLC is sending commands (Boolean or numerical) to a slave PLC
- A point-to-point link where 2 PLCs are exchanging states between each other. This state transmission can be Inputs/outputs of another PLC.
- A Multipoint Communication where a PLC is Publishing states to a group of PLCs.

In a Master/Slave architecture, the Master Coordinator will send orders to the Slaves. A communication paradigm has to be defined for the communication of these orders. TBD

Each case will be implemented with the most appropriate Siemens Technology.

4.5 Fast Controller Interface

This interface addresses communications between a PLC and a Fast Controller. We consider here 3 cases.

- It can be master/slave link where the PLC is sending orders (Boolean or numerical) to a Fast Controller.

- It can be master/slave link where the Fast Controller is sending orders (Boolean or numerical) to a PLC.
- A point-to-point link where a Fast Controler and a PLC are exchanging states between each other.

In a Master/Slave architecture, the Master will send orders to the Slaves. A communication paradigm has to be defined for the communication of these orders. TBD

The Technology used will be defined in a later Paragraph.

4.6 System Monitoring

A Task will be dedicated to PLC System Monitoring: the following Parameters will be monitored:

- Operating Mode: RUN/STOP
- Memory:
  - Load Memory Assigned: 0..100%
  - Work Memory Assigned: 0..100%
  - Retentive: 0..100%
- Scan cycles:
  - Shortest
  - Longest
  - Average
  - Standard Deviation
- CPU Time: Date and Hour
- Communication
  - Configured
  - Max numbers of connexion available
  - Number of connection used
- I/Os:
  - Board Statuses
  - Raw value of each signal
- Alive Counter.

5 Numbering and naming conventions.

5.1 Block numbering convention

A Siemens PLC Program is composed of several Blocks. There are different Block Types : OB, FC, DB, etc.. A number is attributed to each of these blocks. The numbering areas will be divided in 3 Categories:

1. "System" Blocks
2. "CODAC Reserved" Blocks including:
   - Control Blocks produced by the CODAC in the scope of Standardization. Some of these Blocks will be used the Peripheral Blocks, some in the Core Application.
   - DBs used in Peripheral Blocks with content specific to the application but unique.
3. "Application Specific" Blocks including:
   - DBs used in Peripheral Blocks with content specific to the application and with a number of Blocks specific to the application.
   - All Blocks in the Core Application produced by the Plant System I&C Developer.

			Numbering
OB			Siemens Default
UDT			
	System		1..99
	CODAC Reserved		100..299
	Application Specific		300..65535
DB			
	CODAC Reserved		
		Shared	1..49
		Instance	50..99
	Application Specific		
		Shared	100..299
		Instance	300..999
FC			
	System		Siemens Default:1..99
	CODAC Reserved		100..199
	Application Specific		200..999
FB			
	System		Siemens Default:1..99
	CODAC Reserved		100..199

	Application Specific	200..999
SFC		Siemens Default
SFB		Siemens Default

5.2 Block Naming Convention

As we want to enforce symbolic Programming, a name will be attributed to each Block. Naming is an important topic in large projects. ITER has already issued documents that have to be applied. See [RD 1], [RD 2] and [RD 5]. Naming of components inside the PLCs is not simple as many factors has to be considered:

- FBS and PBS naming Conventions
- System Blocks have predefined names
- Some Blocks are related to the Core Application, some to the Peripheral Blocks
- Few Metacharacters are allowed in Siemens

Naming rules will be spread all over the document. However some generic rules can already be mentioned here.

[NR 1] UDTs names will always begin with ”_” (underscore character)
.

[NR 2] Instance DBs will always begin with “i”.

[NR 3] When a block name or part of name is related to FBS, the FBS identifiers will always be in capital letters, and separated by “_” (underscore character).

Examples: "_WFC_CIStates", “WFC”

5.2.1 Core Application Blocks naming convention

The rules will be illustrated with example taken from the FBS of the Cooling Water System Prototype in its actual state. See Annex §14.3 for summary or [RD 8] We will take the case of the Water Flow Control Function.

There will be one FC in the PLC for the Control of the WFC. A FBD representation of a Siemens Control Block of the WFC is represented in Figure 8. The figure represents an example with a FC and an example with a FB.

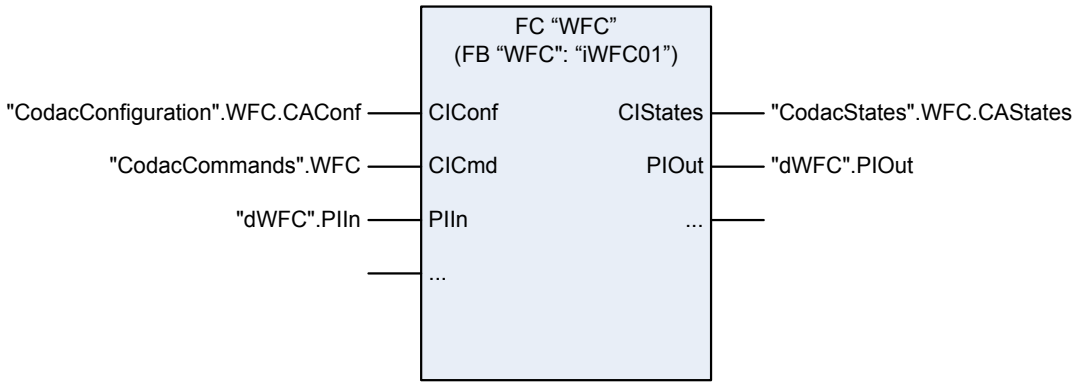

Figure 8 : Control Block of a FBS Level 4 Function.

The FC is named according to the FBS Level 4 name of the Function it is implementing: "WFC".
But in some cases, the PLC could be assimilated at level 2 and implementing FBS Level 3 functions.

[NR 4] In Core Application Blocks, FCs will be named according to the lowest FBS level control function they are implementing. The upper levels are not required in the name.

.

In many cases, the same Control Function will be instantiated many times. In this case, the use of a unique FB with several instances is more adapted. If we consider the hypothetic case where there would be several WFCs, we would define a FB called "WFC" and the instance DBs would be named "iWFC01", "iWFC02", if we agree on the fact that the Control Function names would be "WFC01", "WFC02", etc… In Figure 8 it is represented by the Block name between brackets.

[NR 5] In Core Application Blocks, FBs will be named according to the FBS level function type they are implementing. The instance DB will be named according to the lowest FBS level function instance.

.

The following will developed in §11 but every Core Application Control Block will have the same Interface broken down in 5 connexions. . Let's call this interface the "Standard Control Block Interface".

- "CIConf", the Configuration variables sent by the CODAC.
- "CICmd", the Simple Commands sent by the CODAC.
- "CIStates", the State Variables sent to the CODAC
- "PIIn", the Process Interface Inputs. All input signals of the controlled device
- "PIOut", the Process Interface Outputs. All outputs signals of the controlled device.

[NR 6] In Control Blocks, the 5 Connections of the "Standard Control Interface will be named : "CIConf", "CICmd", "CIStates", "PIIn", "PIOut**".**

.

Each connexion is defined by a UDT. These 5 UDTs are specific to the Function or Function Type. Each UDT will be composed of the variable of the interface it is defining. The name of the UDT is submitted to rules. In the example of the CWS, we would have:

For “CIConf”:

UDT	***Variable***	
"_WFC_CIConf"	Name	Type
	CWFC	BOOL
	HSRQ	BOOL
	PT2SP	REAL
	LFSP	REAL
	HFSP	REAL

For “CICmd”:

UDT	***Variable***	
"_WFC_CICmd"	Name	Type
	dummy	BYTE

For “CIStates”:

UDT	***Variable***	
"_WFC_CIStates"	Name	Type
	STOPWFC	BOOL
	LFST	BOOL
	HFST	REAL

For “PIIn”:

UDT	***Variable***	
"_WFC_CIStates"	Name	Type
	PL1_CY	BOOL
	PL1_YT	BOOL
	VC8_FVY	BOOL
	MP2_PT	BOOL
	MF1_FT	BOOL
	PL1_SY	REAL

For “PIOut”:

UDT	***Variable***	
"_WFC_CIStates"	Name	Type
	PL1_CZ	BOOL
	VC8_FVZ	BOOL
	PL1_CS	REAL

The examples above are applying the following rule:

[NR 7] UDTs related to the Standard Control Block Interface will be named according to the following Pattern:
“_”+<Control Block Name>+”_”+<Connection Type>.
<Connection Type> can be the names defined in [NR 6].

Inside these UDTs, the variables names are the one defined in [RD 8]. These names are following naming rules defined in [RD 1] for the signals.

[NR 8] In UDTs defining "PIIn", "PIOut", interfaces, the variables names has to follow the rule of FBS signal names, defined in [RD 1].

For UDTs defining "CIConf", "CICmd", "CIStates", no strict convention is applied so far except that they have to be in capital letters.

5.2.2 Peripheral Blocks and Generic Functions naming convention

The Peripheral Blocks will not be directly related to Plant System Control Functions. They will address internal organization of the PLC, communication functions, system functions, etc…. Iter will also provide a generic library with tools to perform engineering conversion, standard UDTs, etc…

[NR 9] Peripheral Blocks and generic Blocks will be named with undefined number of fields, each field beginning with a capital letter. The rest of the field will be in minor letter.

Examples:"CodacInterface", "InputsProcessing", "DigInProcess".

Many UDTs will be created in order structure the information related to a unique Control Function. Within this framework, a part of the name will be related to FBS, and the other part will be related to the scope of the UDT. Both parts will be separated by a "_".

Examples: "_WFC_HwiConf", "_WFC_CIStates".

5.3 Variables naming convention

5.3.1 Imputs and Outputs variables.

Even if inputs and outputs will almost not be used in their raw state, they have to be named.

[NR 10] Inputs and outputs will be named according tot their full PBS name. As the PBS name begin with a number, and it is not accepted by STEP 7, the name will begin with a "p". Dash and colons will be replaced by underscores ("_").

Examples:
p26PHDL_PL_1_CY_CCC;
p26PHDL_PL_1_YT_CCC;
p26PHDL_VC_8_FVY_CCC;
p26PHDL_MP_2_PT_CCC;
p26PHDL_MF_1_FT_CCC;
p26PHDL_PL_1_SY_CCC;
p26PHDL_PL_1_CS_CCC;
p26PHDL_PL_1_CZ_CCC;
p26PHDL_VC_8_FVZ_CCC;

5.3.2 DB variables.

Naming rules of variables inside DBs will be detailed in §10, along with the details of Peripheral Blocks development.

6 Programming Environment Standard Configuration.

6.1 Step 7 Config

Some applications will be edited on several workstations. It is important that the Development environment is identically configured when editing the project.
All the default settings of STEP 7 will be used except for the following:

1. Language

In Simatic Manager menu, choose "Options/Customize/Language". Choose english as National Language and check English for Mnemonics.

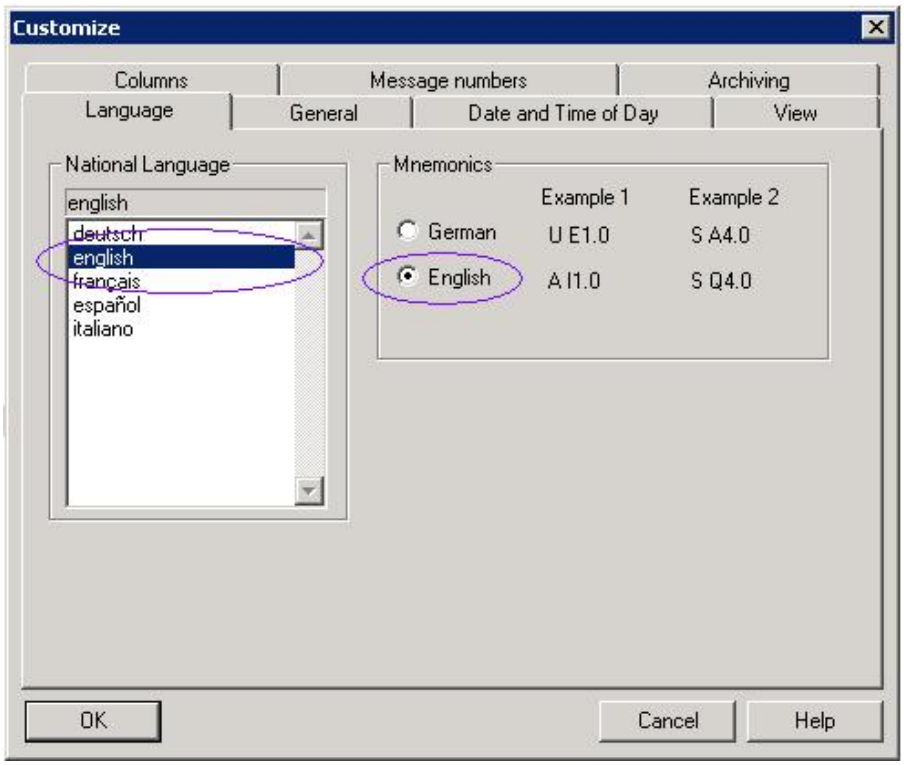

Figure 9 : Step 7 Language Setting.

2. Date and Time

In Simatic Manager menu, choose “Options/Customize/Date and Time of Day”. Check “ISO 8601” as Format for Date and Time of Day..

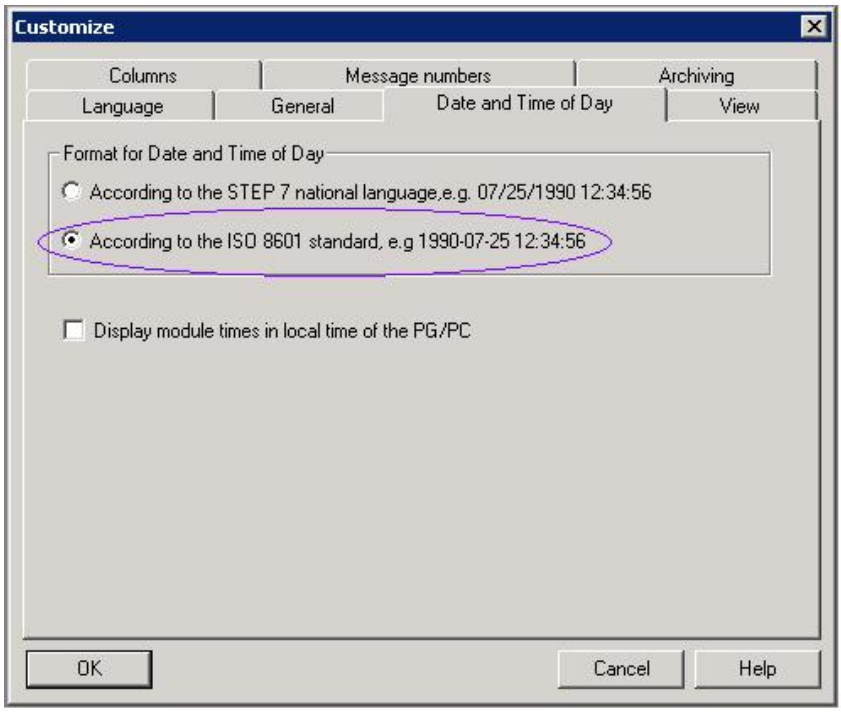

Figure 10 : Step 7 Date and Time of Day Format.

3. LAD/FDB Layout

In LAD/FBD/STL editor menu, choose "Options/Customize/LAD/FBD". Choose DIN A4 Landscape as Layout.

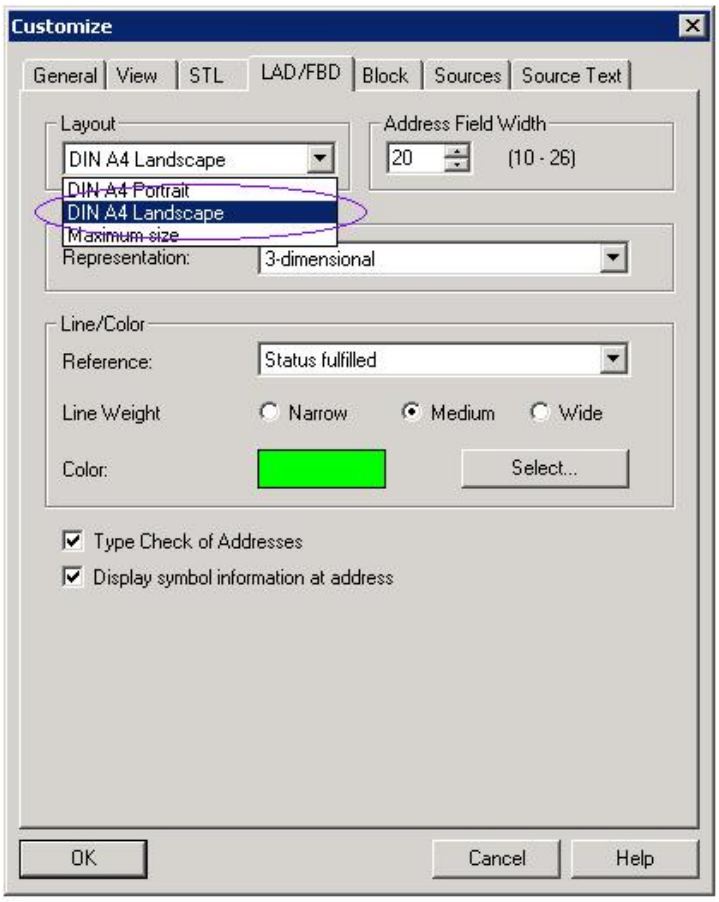

Figure 11 : LAD/FBD Layout.

4. LAD/FDB/STL Sources.

In LAD/FBD/STL editor menu, choose “Options/Customize/Sources”. Check “Generate “Sources automatically”, “Symbolic Identifier of the Block.”, “Symbolic Adresses”.

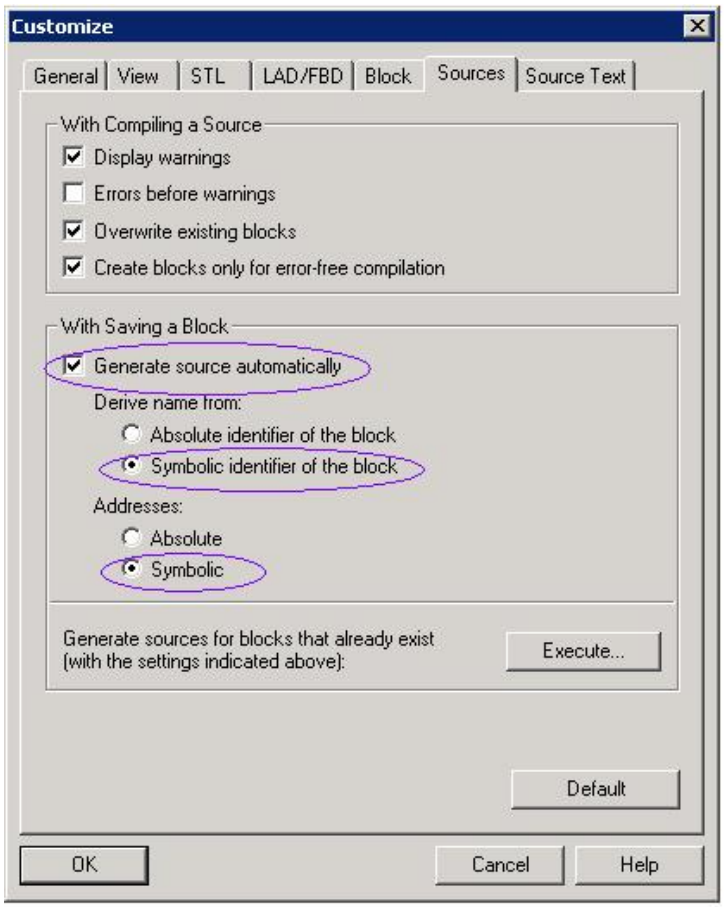

Figure 12 : Block Sources.

5. Symbol Editor Import.

In Symbl editor menu, choose “Options/Customize/Import”. Check “Overwrite Mode” and “Symbol Name”.

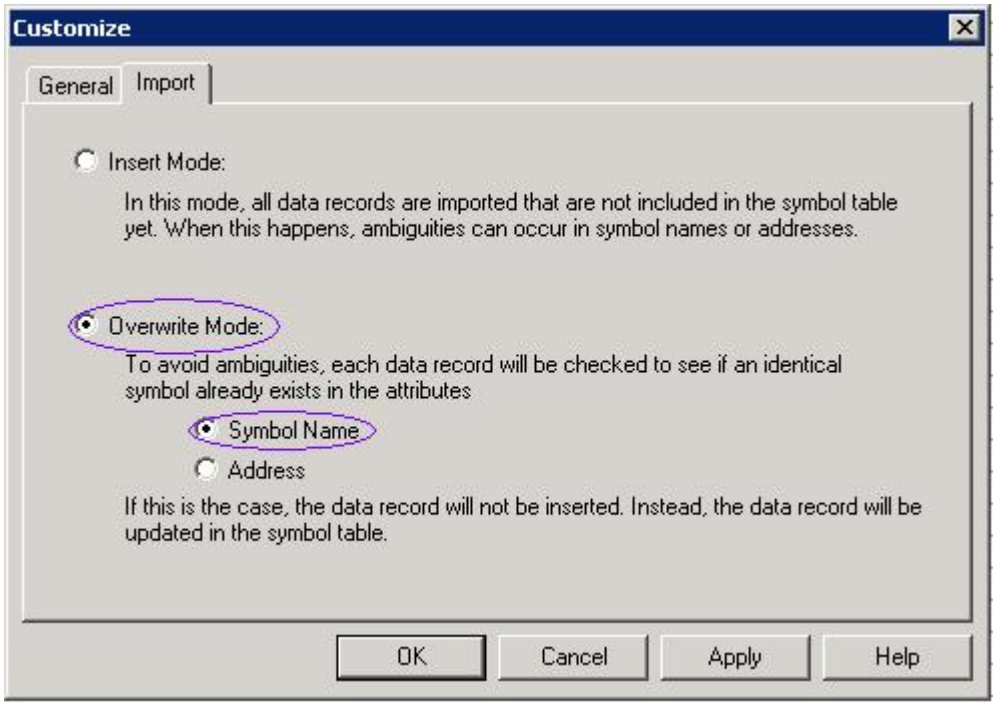

Figure 13 : Symbol Editor Import.

6.2 Project Config

Project settings has also to be standardized in order to make it as "portable" as possible.
All the default settings of Project will be used except for the following:

1. Address Priority:

Click right on the "Block Folder" of the Program and check "For all accesses". This setting intends to enforce Symbolic Programming.

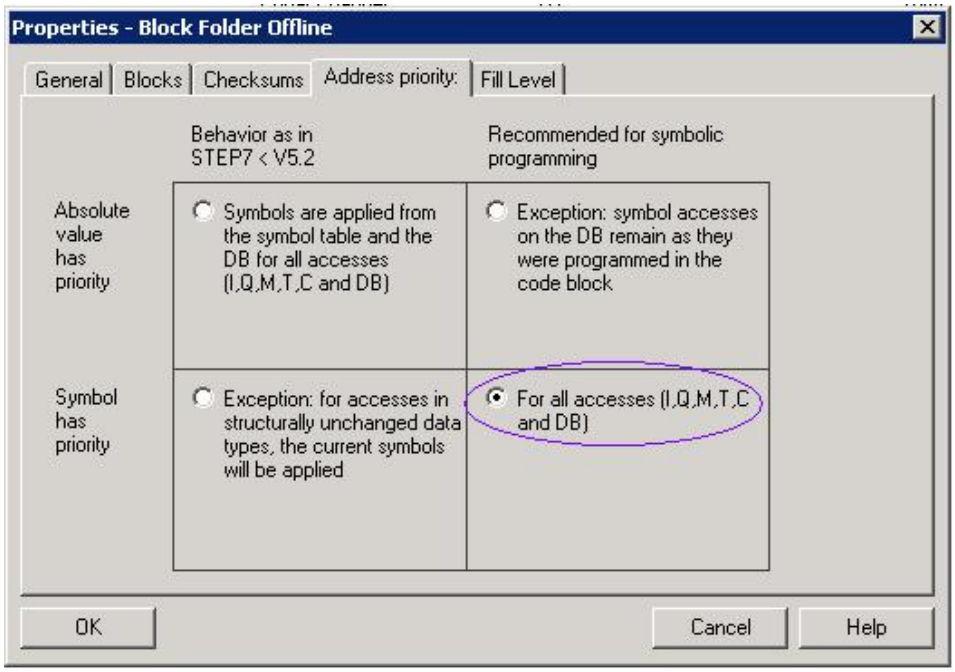

Figure 14 : Address .Priority.

7 Hardware Config

When creating a PLC STEP7 Application, the first step is to create the S7 Project and configure the Hardware. This chapter is giving the rules to be applied when choosing parameters. Most of the parameters are the default one. The below rules are addressing the exceptions.
All hardware components must be chosen in the PCDH Catalog. See [RD 9].

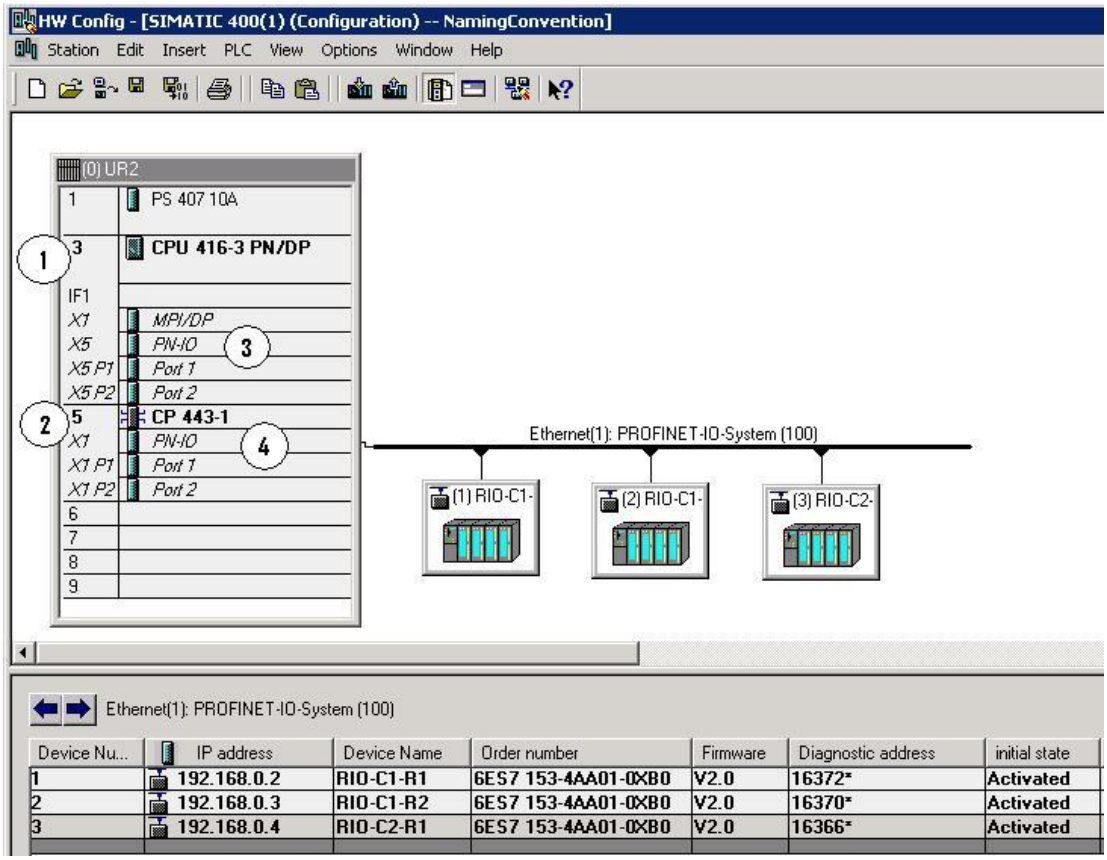

Figure 15 : STEP 7 HW Config Screen for a CPU Stations with 3 Remote IO Racks.

1. *CPU Configuration*
   - Immediately when a CPU is inserted in a Rack, (*Figure 15* –1) the first parameters requested is the IP Address on the network. This interface is connected to the PON. CODAC is managing the IP Address plan of the PON and will provide the information. The rest is the default parameters. There is no need to configure a network. It will be required if you have PLCs interconnected in the Plant System, this is developed in §10.3.
   - Double-Click on the CPU in your hardware config (*Figure 15* –1) and choose the "General" tab. In the "Comment" text field, introduce the PBS number of the PLC, the Cubicle PBS and Location.
   - Choose the "Cycle/Clock Memory" tab. Check "Clock Memory" and introduce "100" in the "Memory Byte" field. (*Figure 16*)

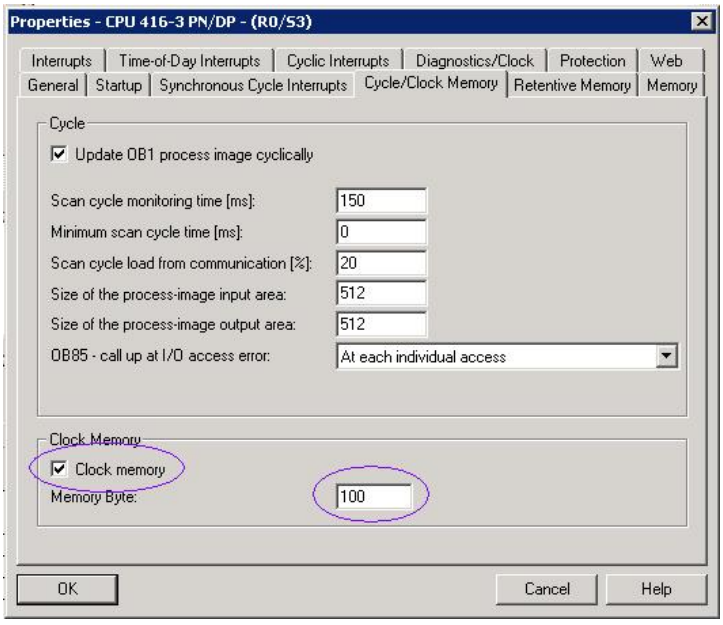

Figure 16 : CPU Config Clock Memory Setup

- Choose the Startup tab. Check "Cold Restart" for Startup after Power On. (*Figure 17*)

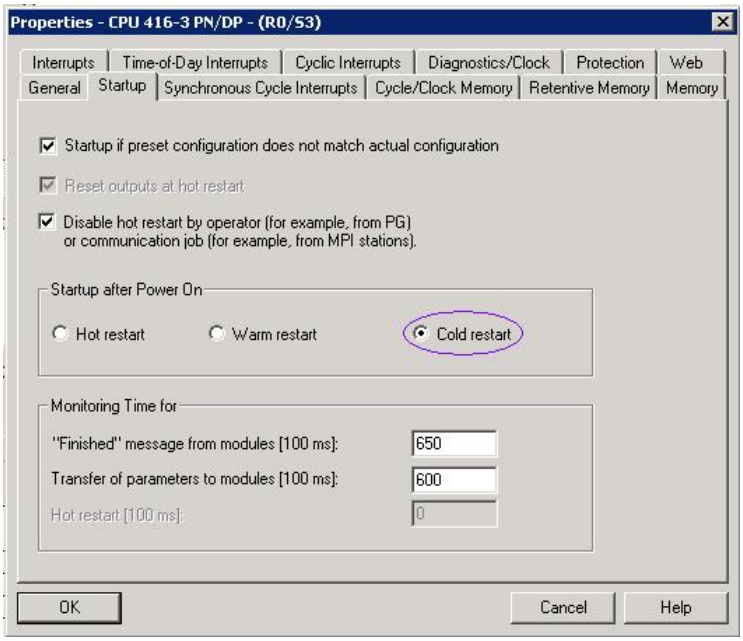

Figure 17 : CPU Config Startup

- Double-Click on X5-PN:IO filed of the CPU ((*Figure 15* –3).
  - Choose "time-of-Day Synchronization tab. Check "Enable Time of Day synchro in NTP Mode". (*Figure 18*)
  - Introduce 2 NTP Servers Address. This information is managed by CODAC. (*Figure 18*)
  - Write 60 in "Update Interval". (*Figure 18*)

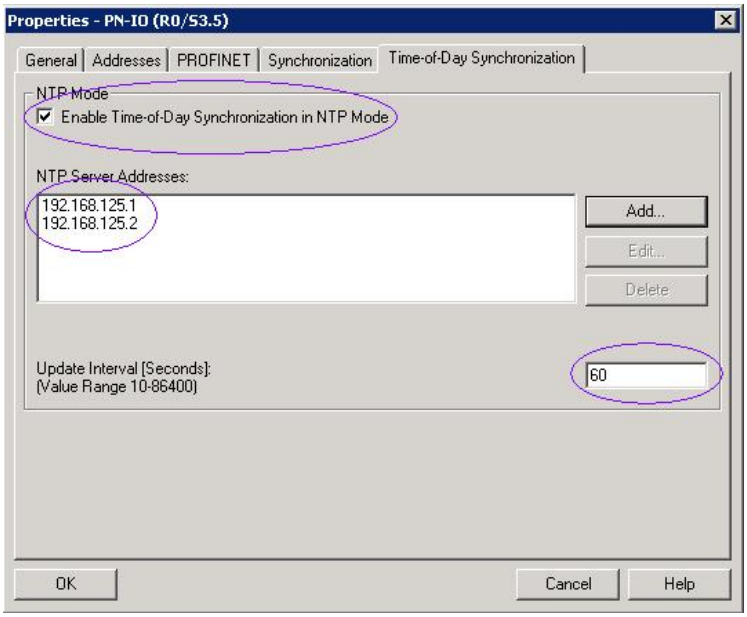

Figure 18 : CPU Time-of-Day Synchronization.

2. *CP Configuration*
   - Immediately when a CP is inserted in the Rack (*Figure 15* –2), the first parameter requested is the IP Address on the network. This Network is the Profinet Network, physically separated from the PON. The default address can be left as it is. There is no need to have a specific IP Adress Plan.
   - In Subnet, Create a New Network, with default parameters..
   - Click-right on the "X1 PN-IO" (*Figure 15* –4), field of the CP and select "insert PROFINET IO System".

3. *Remote IO Rack Configuration*
   - When installing a Remote IO Rack in the Profinet Network the only parameter to impose is the name. This name is important because it is used by the Profinet network communications. The name will follow the following Pattern:
     "RIO-C<x>-R<y>"
     Where <x> is the cubicle number and <y> a rack number in the cubicle. These numbers are own to the PLC and to the rack.
     In the "Comment" text field of the Rack, the PBS number and location of the Cubicle should be mentioned.
   - When inserting the IO boards in the Racks, the default addresses proposed by STEP 7 will be applied. It is important to follow this rule because depending of the type of Boards (digital/analog, input/output) Step 7 is choosing specific Inputs/Outputs areas.
     - The boards in the Remote IO Racks has to be arranged in the following order (*Figure 19*):
       1. Digital Input Boards
       2. Digital Output Boards
       3. Analog Input Boards:

3.1. 0..10V, 4-20mA, etc..
3.2. RTD Input boards
3.3. Thermocouples Input Boards
4. Analog Output Boards

- In a group of board of the same type, the signal Addresses has to be kept in an ascending order. See arrows on *Figure 19* It is the default behaviour of STEP7, but not if should reshuffle boards manually afterwards.

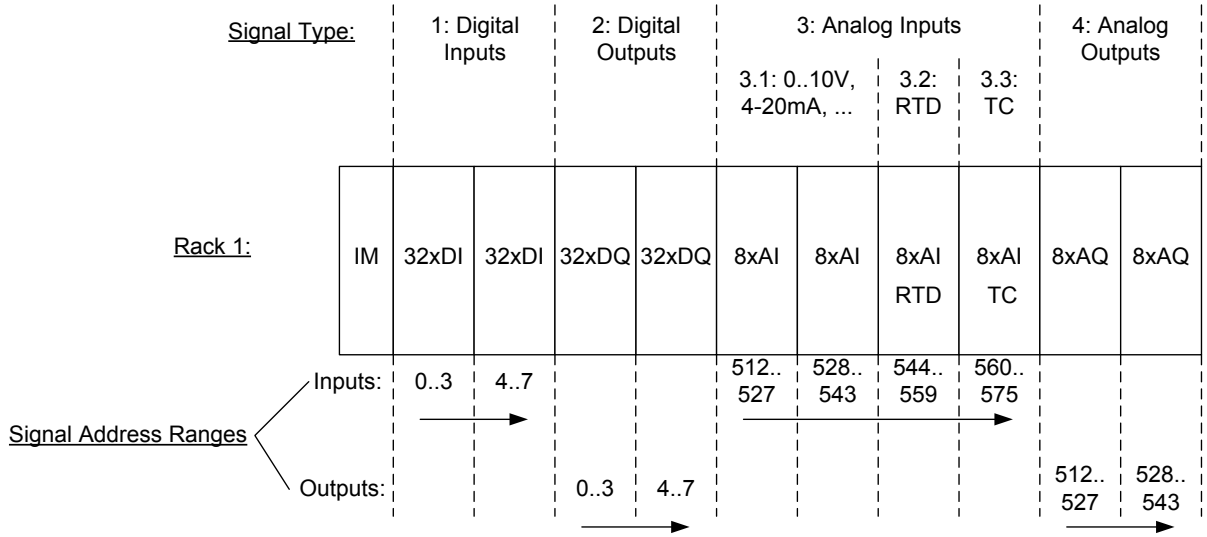

Figure 19 : Remote IO Rack Board organization.

8 Symbol Table

Most of the rules for the edition activity in the Symbol table is covered by the Naming and Numbering Rules. However, an important additional rule is to declare all numerical Inputs and Outputs as INT or DINT. The Symbol Table is declaring them by default as WORD. It makes sense only for a few status information. .. If these signals are declared as WORD, it requires an additional conversion WORD-> INT before processing.
This concerns : PIW, IW,PQW and QW address types.

9 Standard PLC Software Structure (SPSS)

9.1 SPSS Description

The root structure of the Control Blocks in the PLC will be the same in every PLCs deployed on ITER. The diagrams below describe this standard structure.

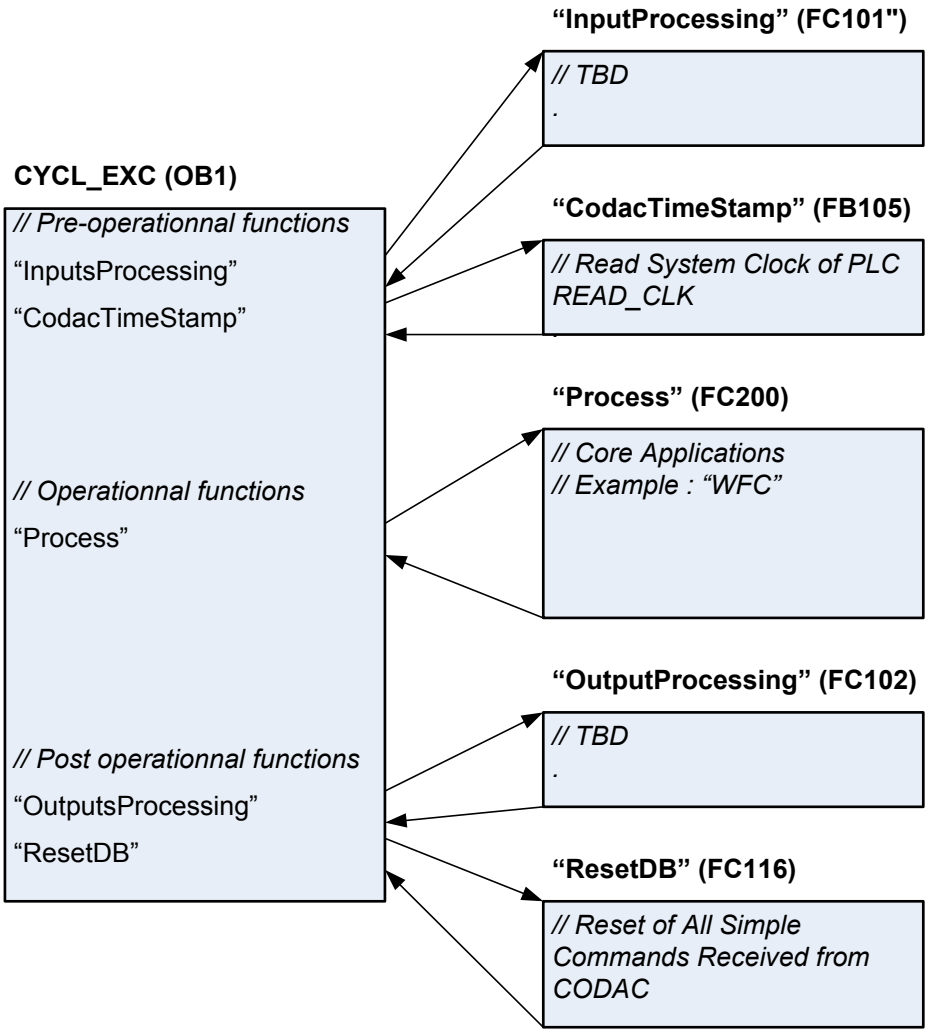

Figure 20 : Main Cycle Loop Standard Structure

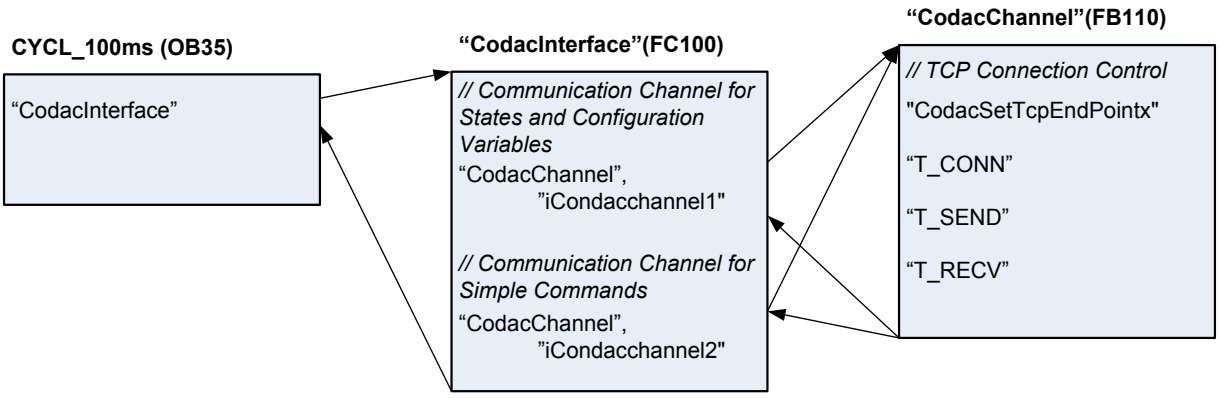

Figure 21 : 100ms Cycle Loop Standard Structure

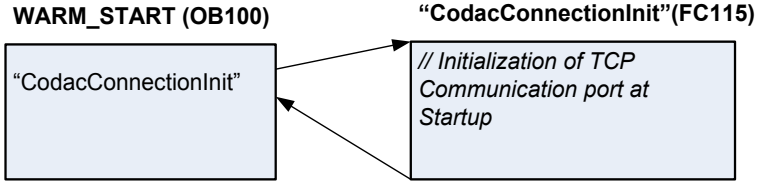

Figure 22 : Warm Restart Block Standard Structure

This Standard Structure is developed and maintained by ITER. It has to be imported in any application before developing the Peripheral Blocks and the Core Application.
Peripheral Blocks will be generated automatically or coded manually using templates and coding rules. It is developed in §10.

This standard Structure is currently supporting the backbone for the Codac Interface. In §10.1 it is explained how to define a Codac Interface and how to generate the code automatically.

The Core Application Blocks will be called in the "Process" Control Block represented in *Figure 20*. Ie, in the example described in §5.2, the call to FC "WFC" would be integrated in the "Process" Block.

9.2 SPSS Creation Procedure

A first Step is to create a suitable Hardware Configuration.
Second Step is to Import the SPSS. There are 2 Options: Import source file or integrate directly the binaries in the Project. The files mentioned here-under can be found on the Mini-CODAC at the following location:

```
/opt/codac-<CCS-version>/step7/STEP7
/opt/codac-<CCS-version>/step7/STL
```

Where `<CCS-version>` is release-dependent.

9.2.1 Hardware Configuration

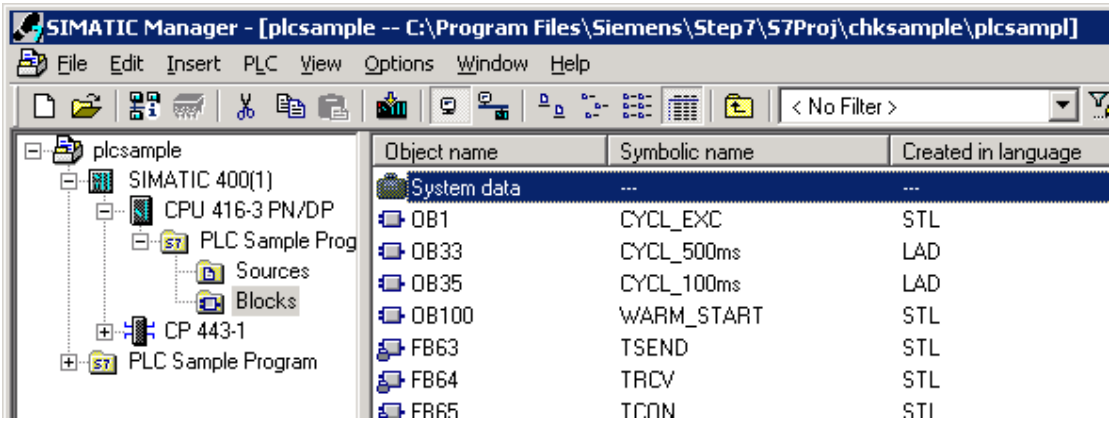

Figure 23: Hardware configuration compiled to generate 'System data'.

After a STEP7 project is created the PLC hardware can be specified with following steps.

1. Add a “Simatic 400 Station” for S7-400 PLC, a “Simatic 300 Station” for S7-300 PLC.

2. Edit the hardware of this station using “HwConfig” which is opened by double-clicking the “Hardware”. Add a rack and populate the rack with appropriate Power Supply and CPU. Refer to PLC Catalog for appropriate reference [RD 9]. Other CPUs can be used as long as they have an Ethernet Interface and they support “Open IE Communication”. Nethertheless, CPUs not included in the catalog will not be supported by CODAC.

3. Using “NetPro” specify the IP addresses of the CPU and CP modules in the rack. See §7 The IP addresses must be same as previously configured in the CPU.

4. The hardware configuration should be saved and compiled either in “HwConfig” or “NetPro”. After the hardware configuration is compiled it gets reflected as “System data” in the “CPU | S7 Program | Blocks” folder under the Simatic Station. See *Figure 23.*

9.2.2 *Import the “Standard PLC Software Structure” from external source files.*

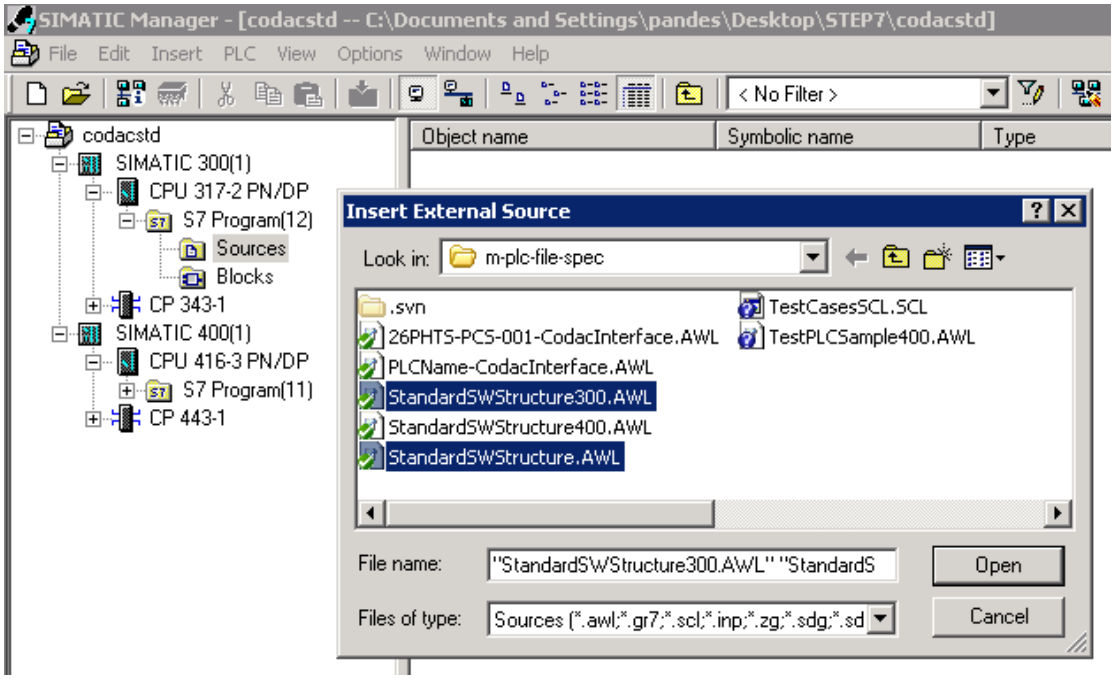

Figure 24: Add standard software structure as external source.

1. Configure the PLC hardware as described in §9.2.1

2. In the Simatic Manager open "Libraries | Standard Libraries | Communication Blocks", and drag-and-drop FB63 (TSEND), FB64 (TRECV), FB65 (TCON), FB66 (TDISCON) and UDT65 (TCON_PAR) in the “CPU | S7 Program | Blocks” folder.

3. Open the symbol table and import "StandardSoftwareStructure.sdf" and save. It is necessary to save the Symbol Table at this stage to be able to compile the STL source in following step.

4. Insert external source from the "StandardSWStructure.AWL" file in the “CPU | S7 Program | Sources” folder and compile. The compilation must not give any error if step 2 and 3 are performed correctly.

5. Only for S7-400 CPU, insert the "StandardSWStructure400.AWL" file in the “CPU | S7 Program | Sources” folder and compile to perform the S7-400 specific initialization.

6. Only for S7-300 CPU, insert the "StandardSWStructure300.AWL" file in the “CPU | S7 Program | Sources” folder and compile to perform the S7-300 specific initialization.

9.2.3 Import the “Standard PLC Software Structure” from STEP7 Archive

1. In Simatic Manager open the ‘codacstd.zip’ file (“File | Retrieve…”) to create a STEP7 project which includes the “S7 Program” folder containing SPSS.
2. Configure the PLC hardware as described in §9.2.1.
3. Drag and drop the “S7 Program” folder to the PLC-CPU.
4. Compile the hardware through “HwConfig” or “NetPro”.
5. Only for S7-400 CPU, compile the "StandardSWStructure400" in the “CPU | S7 Program | Sources” folder to perform the S7-400 specific initialization.
6. Only for S7-300 CPU, compile the "StandardSWStructure300" in the “CPU | S7 Program | Sources” folder to perform the S7-300 specific initialization.

10 Peripheral Blocks Development

10.1 CODAC Interface

10.1.1 Description

From protocol point of view, the CODAC interface is based on raw socket communication between the PLC and the PSH. In the PLC the "Open Communications IE" Blocks are used to implement this communication. This Block family is available only on CPU embedding an Ethernet Interface. This choice is based on an assessment of communications possibilities with Siemens STEP 7 PLCs.
As described in §4.2, this interface will support 4 types of information:

- States Variables
- Configuration Variables
- Simple Commands
- Collaborative Data

Each type of Data will be transmitted in one DB of Maximum 8 kBytes. States and Configuration will use the same TCP connexion, Simple Commands and Collaborative data will have their respective TCP Connection.
Each DB will be build with UDTs. Each UDT will represent the Interface Type (States, Configuration, Commands) of one Control Function identified in the FBS of its Plant System. The diagram below represents the Block organisation and dependencies from the Control Function "CWS-DHLT-WFC" in the Cooling Water System. The boxes in pale Blue are generated. The other ones are part of the SPSS.

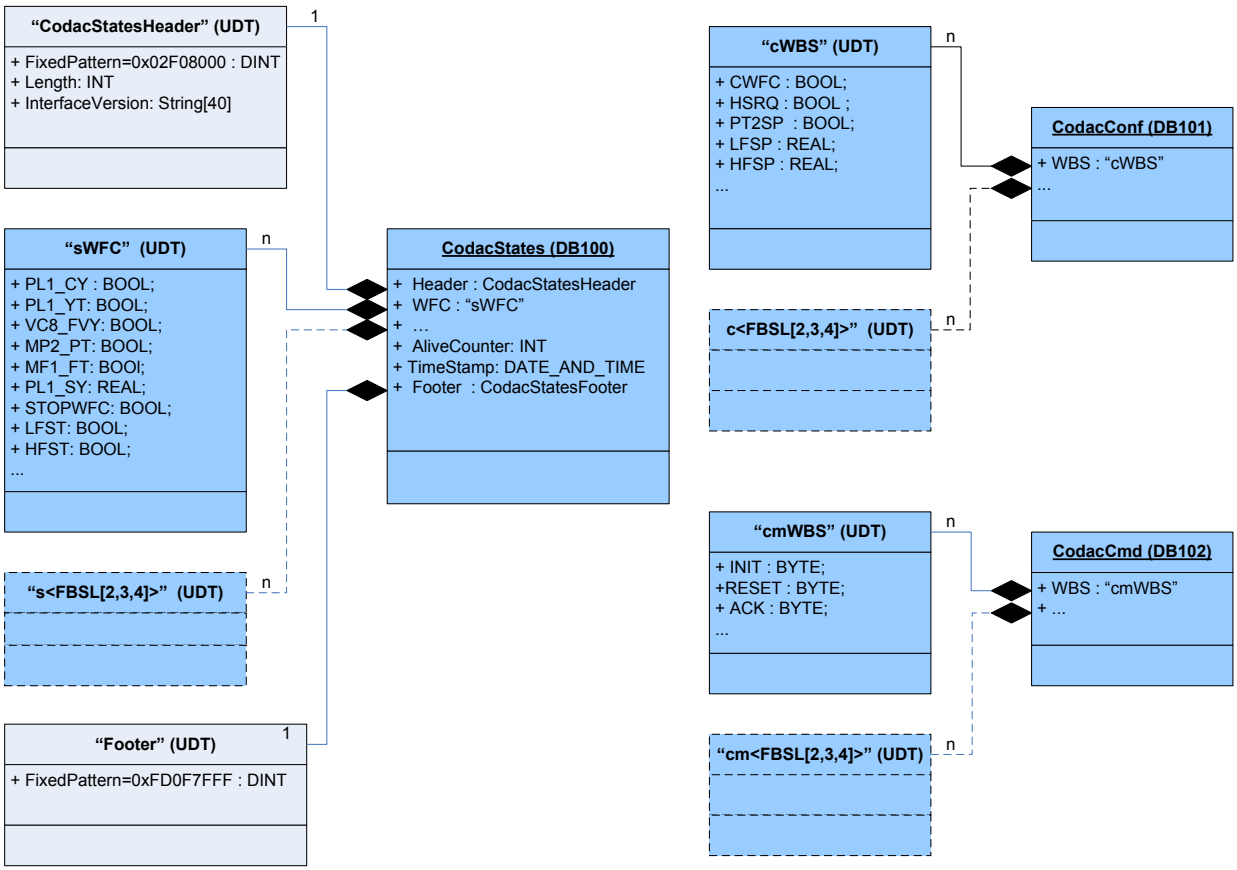

Figure 25:UDTs and DBs organisation and dependencies for Control Function "CWS-DHLT-WFC".

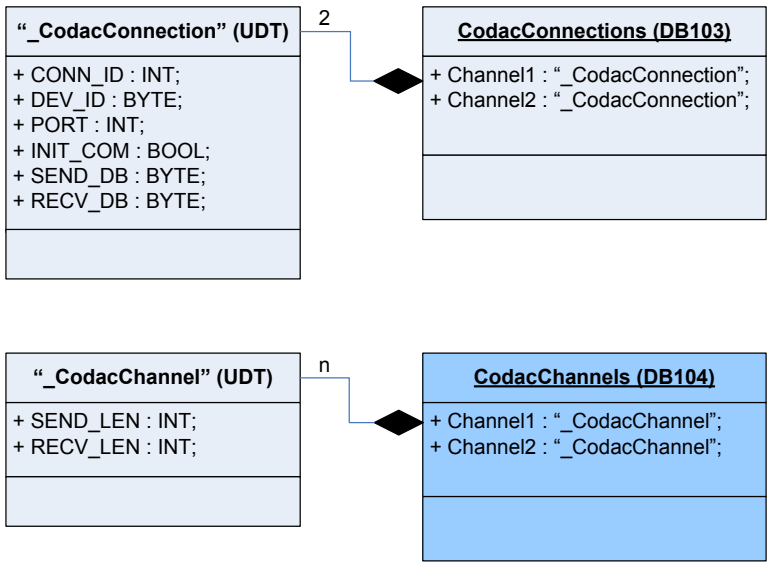

Figure 26:UDTs and DBs organisation and dependencies for TCP connexion parameters.

10.1.2 Generation procedure

The different STEP 7 components described in previous Chapter will be integrated in Source files automatically generated by the SDD. SDD is a CODAC toolkit used to describe the Interface with a PLC and generated automatically the STEP 7 and PSH files required to build this interface. The SDD Toolkit generate a set of plant system specific Symbol Table (*.sdf) file and an STL (*.awl) file. These files implement the "CodacStates", "CodacConfiguration" and "CodacCommands" data blocks and initialize the "CodacChannels" data block. The following screenshot shows the example of SDD Editor generated files for Cooling Water System. The use of SDD is described in [RD 7]

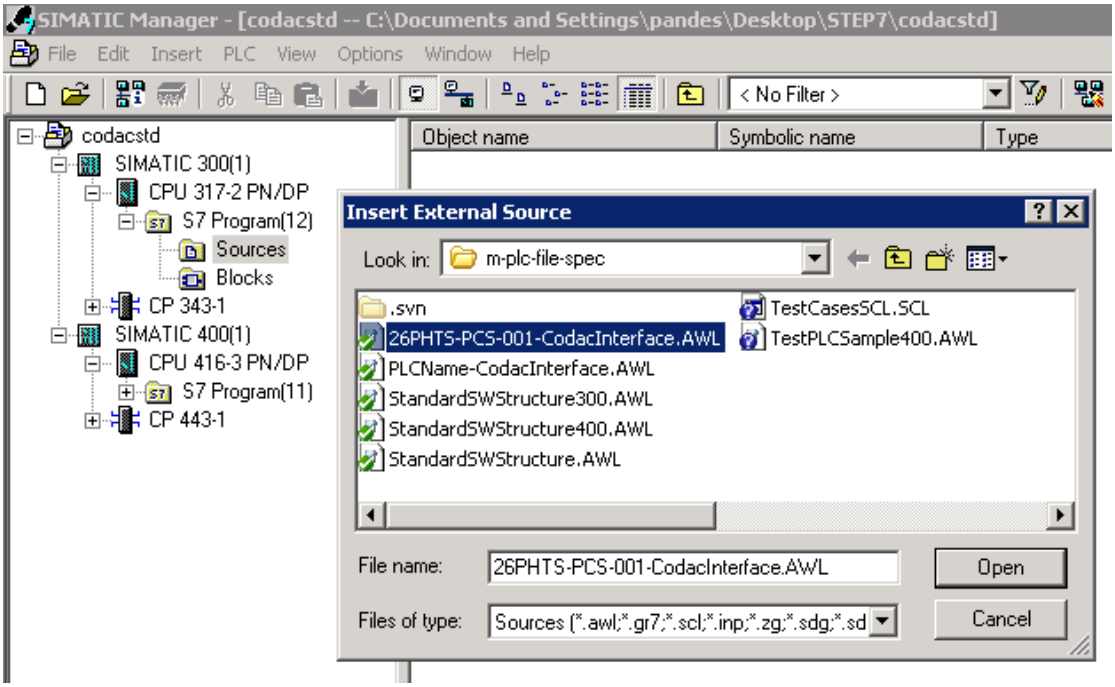

Figure 3: Add plant system specific STL file generated by SDD Editor.

After the SPSS is built the Symbol Table (*.sdf) file and the STL (*.awl) file should be imported in the STEP 7 Project as given below.

1. Open the symbol table and import the plant system specific Symbol Table (*.sdf) file and save.
2. Insert external source from the plant system specific SDD generated '*.AWL' file in the "CPU | S7 Program | Sources" folder and compile.

10.2 Hardware Inputs/Ouputs interface

TBD

10.3 PLC inside plant System interface

TBD

10.4 Fast Controllers interface

TBD

10.5 Simulator interface

TBD

10.6 System Health Monitoring

The Implementation of Health Monitoring is still TBD.
In order to avoid unrehearsed crashes of the PLC, the following OBs will be loaded in the PLC:

- OB82 : Diagnostic interrupt.
- OB86 : IO-Device failure interrupt.

- OB121 : Programming error interrupt.
- OB122 : I/O error access interrupt.

The content of these OBs is TBD.

11 PLC Core Application Development

11.1 Development Cycle and Deliverables

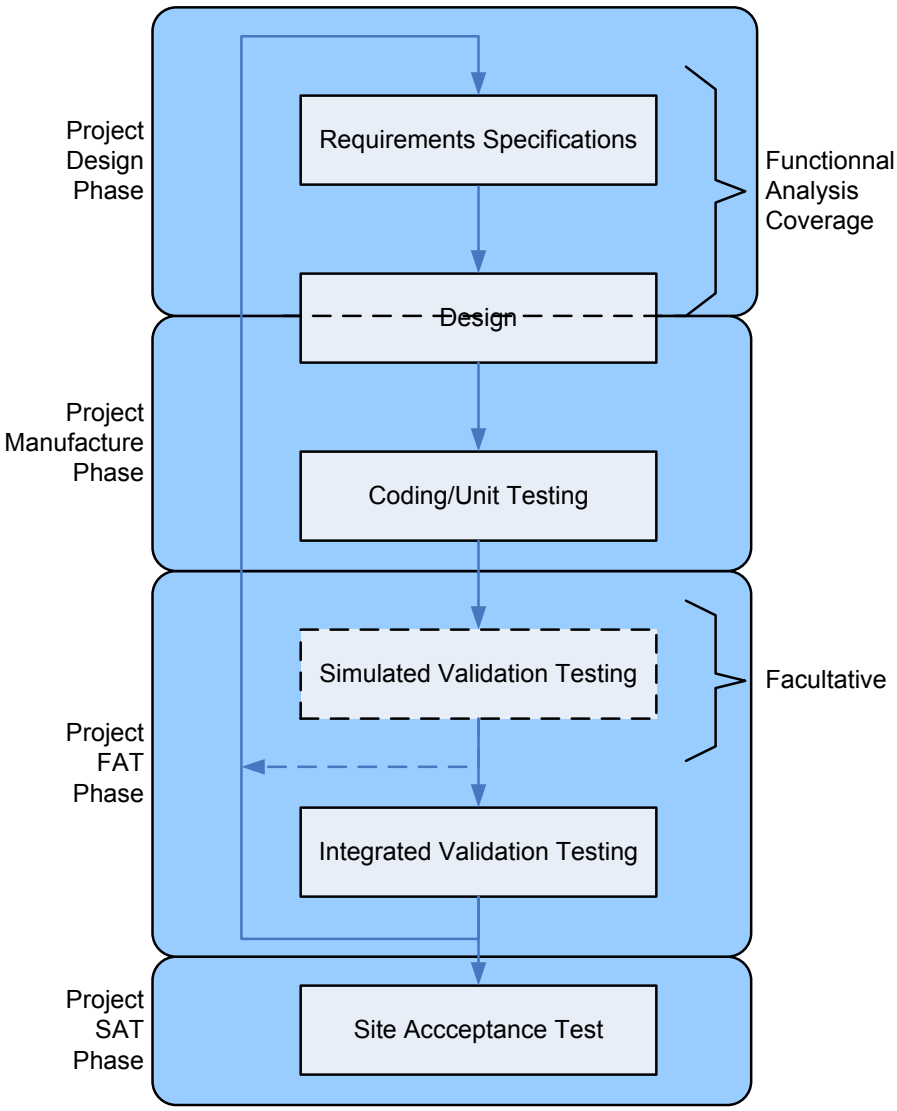

Figure 27: Core Application Development Life Cycle

A first constraint in the development of the application is that requirements activities, design activities and coding activities will be geographically distant. So in order to track the development and verify that the transitions between phases are possible, a minimum of formal documentation will be required.
A second constraint is that one Plant System I&C may control several components coming from different procurements. The strategy applied to address this problem has to be explained in document PCDH [D30].

The lifecycle proposed is addressing the development of the Core Application Software of every PLC of a complete Plant system I&C. Meanwhile, some Plant System I&C will be very big and will implement several high level functions. In this case, the development of one plant

system I&C can be broken down in several life cycles. This has to be specified in PCDH [D30].
Usually, a PLC developer is in charge of one (or several) high-level function, distributed on one or several PLCs (*). It is suggested that one life-cycle is covering the activity of one high-level function, so the activity of one PLC developer.
The life cycle is nothing more than the different steps followed in a normal Software Engineering life cycle. The only originality here is in the addition of an "integrated validation testing".
As the detailed design of the Plant System I&C will be completed at the time the software development will begin, a full set of documents will be available, broken down in 9 deliverables defined in the PCDH. These deliverables will include an I&C hardware Architecture, a functional analysis, a list of Inputs/Outputs, etc…

(*)Note that it is also recommended not to have more than one PLC developer on one PLC. PLC Development Environments are not well designed for collaborative development.

11.1.1 Requirements Specification

Software Requirements are fully covered during the I&C Design Phase. They will be mainly covered by the Functional Analysis. But other inputs State Machines, Control Philosophy will be used as a input. See PCDH.

11.1.2 Design Specification

We will consider 2 phases in the Software Design: an Architectural Design and a Detailed Design. The architectural design will be also covered by the Functional Analysis, as it will define the main treatment blocks inside the Core Application. It is not required to re-define the Peripheral Blocks, as they will be all developed according to templates or generated.
The detailed design consists in defining how all the functions will be implemented. The following information should be provided for each Controller for this Step:

- The naming and numbering of each Programming Blocks: OB,FC,FB,DB
- The Full Program Structure of each OB, for the Core Application Section.
- Any Specific Hardware, Network, Project, configuration used.
- Any deviation on the rules defined in this handbook.

All this information will be gathered in PCDH [D31]

11.1.3 Coding/Unit Testing

Coding and unit testing will be performed simultaneously. It is proven that unit testing is improving drastically the reliability and the robustness of the code produced.
It means that every FB and FC has to be tested independently. The standard architecture is making this easier, as the Core Application has no direct external interface. So any interface of block programmed in the Core Application can be replaced by a variable in order to simulate the behaviour of the interface.
Unit testing doesn't require a formal document. Meanwhile if too many mistakes are noticed during validation phases, a formal unit test document may be requested by IO

11.1.4 Simulated Validation Testing

The idea here is to have a Simulation Test performed before the test connected to the System. There are several purposes:

- The real system doesn't even have to be connected/ready to go through testing activities. Their life cycles can be desynchronised until connection.
- Tracking as many functional problems as possible before connecting to the real system. The software will be more mature at the time of the connection so less time will be lost in Software troubleshooting during System testing.
- When the I&C is composed of several Controllers, it will be possible to test the I&C in its integrality even in the case the systems comes from different procurements.
- If Simulation is available, corrective and adaptative maintenance will be possible without having the system connected.

This phase is optional, as it may not have a lot of sense for small Plant Systems.
For large Plant Systems with many controllers, simulation will cover only low level functionalities. The development of complex algorithms with multiple couplings is too time consuming and doesn't produce any value.
Meanwhile, it is strictly recommended. The strategy retained according simulation has to be described in PCDH [D30].

The development of the Simulator and the Development of the Control units will be made by different people. In that sense the different understandings on how the Control System should operate will be confronted in an early phase of the project.
The simulator will be connected to the Controllers through the field network. It will read/write Shared DB Variables defined in the Simulator Interface of the Controller. These variables will replace the real I/Os connected to the Controllers. There is no requirement so far on the technology to be used for the development of the Simulator. But the Simulator will be delivered to ITER/IO after FAT. PCDH [D TBD].

For this phase, a validation book has to be provided before the beginning of the validation. PCDH [D TBD].

11.1.5 Integrated Validation Testing

This test will be done during FAT and will be made with real system connected. As some components may not be available during this phase, the strategy applied will be defined in PCDH [D13].

11.1.6 Site Acceptance Test

TBD

11.2 Languages

The languages allowed in the application will be basically the one defined in IEC 61131-3:

IEC61131-3 Language Name	Siemens Equivalent
• Ladder Diagram (LD)	Ladder Logic (LAD)
• Function Block Diagram (FBD)	Function Block Diagram (FBD)
• Structured Text (ST)	Structured Control Language (SCL)
• Instruction list (IL)	Statement List (STL)
• Sequential function Chart (SFC)	Sequential Control System (SCS)

Siemens CFC (Flow Charts) will not be allowed in conventional controllers. First it is not defined in IEC 61131-3, second it has the major drawback that you can not mix CFC with other languages. TBC. But CFC will be used for redundant architectures, deployed in Interlock and Safety. This topic is not covered in this document.
Siemens HiGraph (Petri Nets) will not be allowed either because it is not defined in IEC 61131-3.

You can almost implement anything in any language. Meanwhile, every language has its own characteristics and has been created in order to solve some type of problems. The following rules has to be applied in order to keep the PLC Program as clean and readable as possible:

- LAD and FBD will be used to implement boolean logic and interlocking. Typically all the logic required to start/stop a device will be implemented in LAD. No complex numerical computation allowed in LAD and FBD. The choice of LAD or FBD is left to the programmer. Usually electrical engineers use LAD and electronic engineers use FBD. It doesn't make any difference as we can switch the representation from LAD to FBD in STEP7. See later in the paragraph.
- SCS will be used to implement sequences (GRAFCET). But outputs will not be written directly in Grafcets. See §11.4.
- SCL will be used to implement complex numerical algorithms, loop algorithms, complex state machines ore Petri nets where a sequence (SCS) is not sufficient to express it. As SCL is a structured language quite close to Pascal, it makes it much more readable than STL.
- STL will be avoided as far as we can, as assembler is not really easy to read and to maintain for the people that didn't write the code. It will be used only in cases where for example a specific instruction is not available in SCL or optimization of performances is required.

The Mix of languages is allowed in one block as far as it respects the rules defined in §11.4. and in § 11.8.

Organization of languages in STEP7.

The base language in STEP 7 is STL. All other graphical languages and meta-languages are built as an abstraction of the STL language. If you create a program in LAD, FBD, SCL or SFC, they will end up in LIST blocks after eventual compilation.

The same editor is used to program in LAD, FBD and STL. You can switch very easily from graphical languages LAD and FBD to LIST without recompilation. The other way is not so obvious, STL can be shown in LAD or FBD only if some rules are respected.

SCS is using a specific graphical editor : S7 Graph. After edition, the code is compiled in uncommented STL Blocks. You can watch this Blocks in STL, but any modification will corrupt the graphical representation.

SCL is using a specific text editor. After edition, the code is compiled in uncommented LIST Blocks. You can watch this Blocks in STL, but any modification done in STL will corrupt the text representation. The case of SCL is a bit specific as the code is first saved in the "Sources" folder of the project and stored in the "Blocks" folder after compilation.

11.3 CODAC Interface good practice.

The CODAC interface is built upon a concept described in [RD 3] and summarized in §4.2. It is very important that configuration variables are not overwritten in the Core Application. There is no readback of these variables, so if one of these variables is modified in the PLC, the associated EPICS PV will be misaligned.

A configuration variable mustn't be considered as a simple setting for a PLC output. In the Conceptual Design of the PLC in §4, it is clearly described that PLC Outputs are managed by the Core Application. There is not direct writing allowed from the CODAC to the outputs.

An example of a Control Loop is represented in *Figure 27*. The Command of the device is controlled by the Control Loop. So the command value issued by the Control Loop has to be transmitted to the CODAC as a State Variable, not as a Configuration Variable. If an "open loop" mode has been implemented, than the user input has to be implemented as separated configuration Variable.

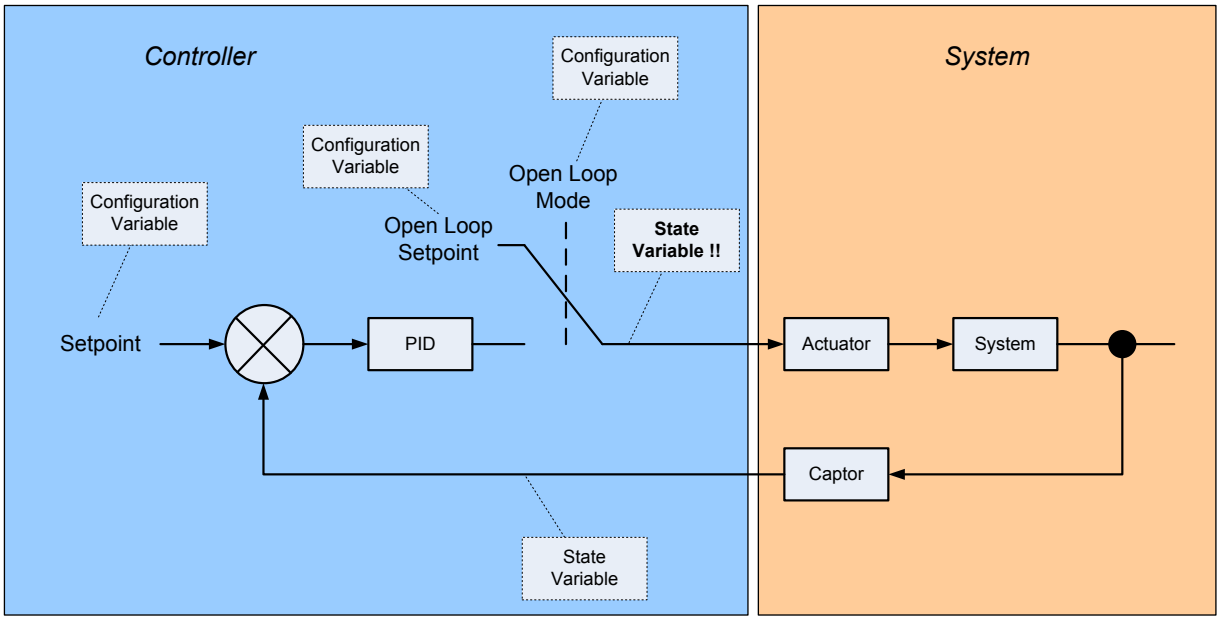

Figure 28: Closed Loop Control Example

11.4 Standard Structure of a Process Function

In *Figure 29*, the conceptual organization of a Control Function is represented: it is mainly composed of three parts: the State Management, the Interlocks and Control Logic and the Control loop.

- The State Management part is represented by a Grafcet or a Petri Net. The State Management will be influenced by exchanges with the CODAC (operations) and the Process Inputs. The State Management never activates directly the Process Outputs.
- The Interlocks and Control Logic Part is managing mainly digital Process Outputs of the Control Function. It is influenced by the State Management and the Process Inputs. The idea is that the command of and actuator is written only at one place in the code, in order to make the code easier to read and to maintain. A Logic diagram is gathering
    - All Process conditions ensuring the actuator is technically ready.
    - State conditions.
- The Control Loop is managing the continuous (analog) Process Outputs. It is influenced by the continous Process Inputs and the State Management. It is very common that depending the operating state, you may want to change the configuration of the Control Loop: change the Setpoint, open the loop, inhibit the integral action, etc..

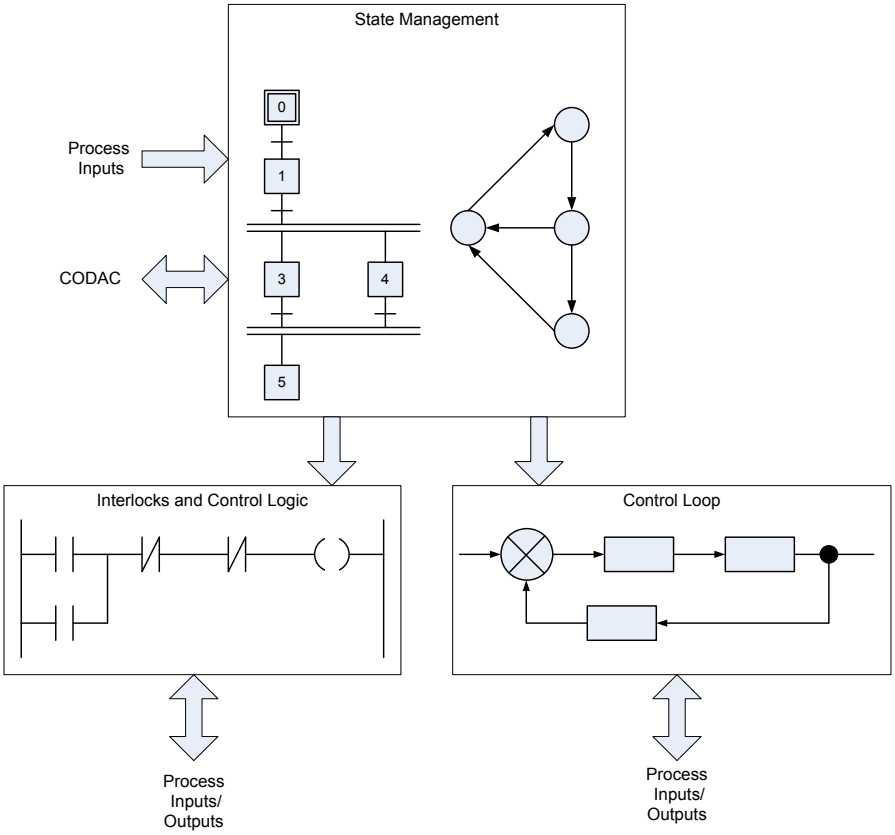

Figure 29: Conceptual Design of a Control Function in the Core Application

The implementation of a Control Function is also standardized. Every Control function will be implemented in one Control Block composed of the three parts described above:

- State Management
- Interlocks and Control Logic
- Control Loop

The Control Block can be an FB or a FC, depending of the fact it is a unique instance or multiple instances. The three parts can be implemented in the same Control Block or in dedicated one. It depends of the complexity of the function.

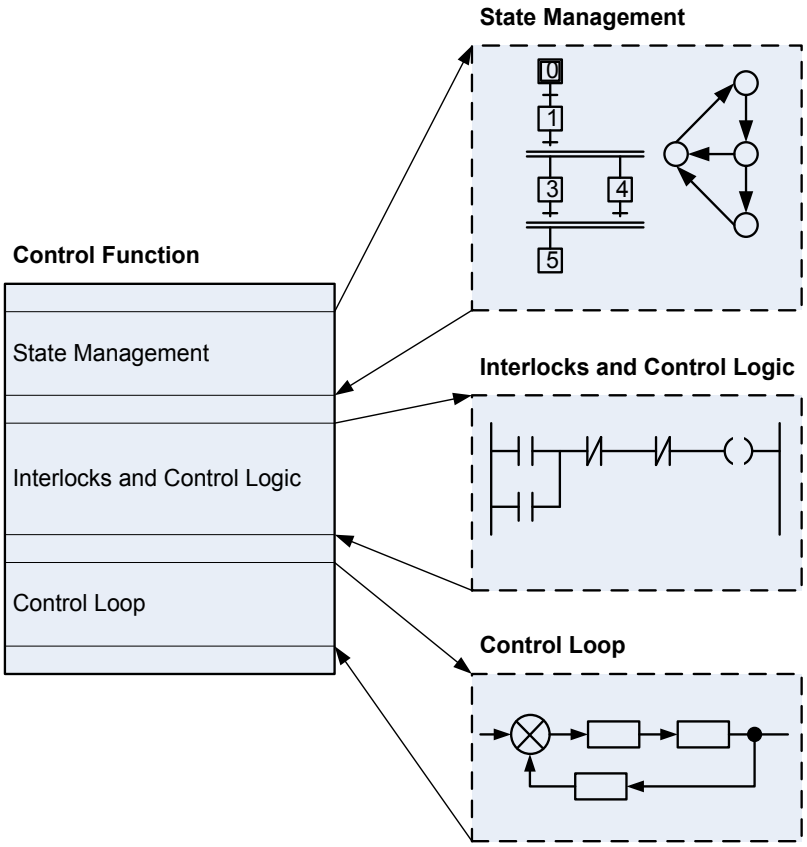

Figure 30: Standard Implementation of a Control Function in the Core Application

Every Control Blocks managing one Control Function will have a standard interface, composed of (See *Figure 8*):

- "IN"
  - "CIConf"
  - "CICmd"
  - "PIIn"
- "OUT"
  - "PIOut"
  - "CIStates"

"CIConf", "CICmd" and "CIStates" will be UDTs generated by SDD Toolkit. See §10.1.2 The advantage if a new interface is generated, the Control Block interface will be updated automatically, avoiding a lot of potential errors.
"PIIn" and "PIOut" will follow the same approach.

11.5Siemens Libraries

Siemens STEP 7 is providing a System Library. Some blocks are allowed, some other ones are prohibited.
TBD

11.6ITER library

ITER will provide a library for the most common standard functions deployed on the Project.
TBD

11.7 Alarms Management

Alarms are managed within the CODAC. No Specific programming is required regarding alarms. If a combination of information is required in order to generate an alarm in the CODAC, it will be including in the regular control Blocks and this information will be transmitted to the CODAC with a State Variable.

11.8 Coding Rules

[CR 1] A generic rule is to implement everything possible by coding more than by configuration. Some features can be implemented by simply configuring the CPU and almost no coding. It may increase development time, but coding as some advantages:
- It is easier to trace code modifications than configuration modifications. Maintenance will be easier especially on our organization where several persons may follow one to another on the program.
- Project will face a least one upgrade to the next generation of Siemens PLCs. Code will be portable in a major part, while we cannot make any assumption regarding CPU configuration. Pushing everything in the code is a good way to reduce risk of a migration.

[CR 2] Programming in FBD or in LAD has to be done in the same way as if it was an electronic or an electric diagram. The writing of coils or latches has to be unique. Set and reset of variables spread everywhere in the code is prohibited.

[CR 3] Use loops in SCL (“FOR”, “WHILE”, …) instead of backwards jump in STL. Backwards jumps are dangerous and difficult to troubleshoot, as any “goto” instruction.

[CR 4] The passing or arguments from one block to another will be done through the interface of the FBs or FCs.: Input Variable, Output Variable. Use of Shared DB Global variable directly in the Control Block is prohibited. It is not always technically possible so exceptions will be clearly stated in PCDH [D31]
This rule will:
- make the code portable
- shorten the length of variables inside blocks, as local variables names don’t have to include name of the block.
- make easier the Unit Testing of blocks.

[CR 5] Use only Shared DB variables, instead of Mementos. The advantage of using DBs is that variables can be grouped functionally. This is helping structuring the code. Use of Memento is consequently prohibited.

[CR 6] Use Clock Memories as far as possible. Clock Memories can be used to generate up to 50 ms delay. This is delay is Small enough to manage most of the industrial control problems.

This is making the code more portable and easier to unit test. This is an enforcement of [CR 1].

[CR 7] Organization Blocks will include only calls Control Blocks. The implementation of Logic in the OBs is prohibited.

[CR 8] Every variable, block has to be commented.

[CR 9] No Logic Programming in LST. Use LAD/FBD.

[CR 10] Each Network: Maximum 1 A4 landscape page. If not possible, use intermediate TEMP Variables.

[CR 11] The use of the "Enable" inputs of LAD and FBD block is prohibited.

[CR 12] No use of "OPN DB" Instruction. Use of complete absolute Shared DB variable: " "WFC". PL1-CY", …

[CR 13] No explicit use of Address Register instructions in STL.

12 Simulator Development

TBD

13 Version Control

TBD

14 Annexes

14.1 Already Reserved Blocks for CODAC

Block Symbol	Block Number	Bloc Type	Description
_CodacStatesHeader	UDT 100	UDT 100	CodacStatesHeader
_CodacStatesFooter	UDT 101	UDT 101	CodacStatesFooter
_CodacConnection	UDT 110	UDT 110	Codac Connection
_CodacChannel	UDT 111	UDT 111	
CodacStates	DB 100	DB 100	Codac Interface States Communications
CodacConfiguration	DB 101	DB 101	
CodacCommands	DB 102	DB 102	Codac Interface Simple Command Communications
CodacConnections	DB 103	DB 103	Communication Parameters of Codac Interface
PlcHwiWiredInputs	DB 1	DB 1	Plc Hardware Interface Wired Inputs
PlcHwiWireOutputs	DB 2	DB 2	Plc Hardware Interface Wired Outputs
PlcHwiSimInputs	DB 3	DB 3	Plc Hardware Interface Simulated Inputs
PlcHwiSimOutputs	DB 4	DB 4	Plc Hardware Interface Simulated Outputs
PlcHwiInputs	DB 5	DB 5	Plc Hardware Interface Inputs
PlcHwiOutputs	DB 6	DB 6	Plc Hardware Interface Outputs
iCodacChannel1	DB 50	FB 110	Codac Interface States and Configuration Channel (1)
iCodacChannel2	DB 51	FB 110	Codac Interface Simple Commands Channel (2)
CodacTimestamp	FB 105	FB 105	
CodacChannel	FB 110	FB 110	Codac Interface Open Communication Control
CodacInterface	FC 100	FC 100	Codac Interface Communications
InputsProcessing	FC 101	FC 101	
OutputsProcessing	FC 102	FC 102	Hardware interface Outputs Processing Block
Process	FC 103	FC 103	Process Function Blocks
CodacSetTcpEndPointx	FC 111	FC 111	Codac Interface TCP Endpoint Setting
CodacConnectionInit	FC 115	FC 115	
ResetDB	FC 116	FC 116	

14.2 Already Reserved Global Variables for CODAC

Variable Name	Address	Type	Description
SystemClockMemory	MB 100	BYTE	System Clock Memory
SysClock100ms	M 100.0	BOOL	System Clock Memory 100ms Period
SysClock200ms	M 100.1	BOOL	System Clock Memory 200ms Period
SysClock400ms	M 100.2	BOOL	System Clock Memory 400ms Period
SysClock500ms	M 100.3	BOOL	System Clock Memory 500ms Period
SysClock800ms	M 100.4	BOOL	System Clock Memory 800ms Period
SysClock1s	M 100.5	BOOL	System Clock Memory 1s Period
SysClock1600ms	M 100.6	BOOL	System Clock Memory 1600ms Period
SysClock2s	M 100.7	BOOL	System Clock Memory 2s Period

14.3 Cooling Water System Example

FBS L1	FBS L2	FBS L3	FBS L4
CWS	PHTS	DHLT	WFC

"CWS" identify the Cooling Water Supply Function.
"PHTS" identify the Primary Heat Transfer System.
"DHLT" identify the Divertor Loop Heat Transfer..
"WFC" identify the Water Flow Control.

The Water Flow Control has the following Hardware Interface:

Signal	Type	I&C Name
Pump state	Digital Input	PL1-CY
Pump electric failure	Digital Input	PL1-YT
Valve state	Digital Input	VC8-FVY
Pressure sensor signal	Digital Input	MP2-PT
Flow sensor signal	Digital Input	MF1-FT
Speed measurement	Analog Input	PL1-SY
Power contactor command	Digit Output	PL1-CZ
Valve command	Digit Output	VC8-FVZ
Speed command	Analog Output	PL1-CS

Note that all the signals of the Hardware Interface are tramsmitted to the CODAC as States.
The Water Flow Control Function will have the following interface with the CODAC:

CODAC Classification	Signal	Type	I&C Name

Configuration Variables	WFC start/stop	BOOL	CWFC
	High speed request	BOOL	HSRQ
	Delta P pump set point, range [2- 6 bars]	REAL	PT2SP
	Low flow set point, range [100- 400 m3/h]	REAL	LFSP
	High flow set point, range [400- 800 m3/h]	REAL	HFSP
States Variables	Stop state achieved	BOOL	STOPWFC
	Low flow state achieved	BOOL	LFST
	High flow state achieved	BOOL	HFST
Simple Commands (*)	dummy	BYTE	dummy

(*) WFC doesn't require any Simple Command. A dummy one has been added to make the example complete.

A unique PLC will assume the DHLT Control. So the PLC will be a part of FBS level 3.