**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.

<table>
<thead>
<tr>
<th>Approval Process</th>
<th>Name</th>
<th>Action</th>
<th>Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Simelio A.</td>
<td>18 Jan 2019:signed</td>
<td>IO/DG/COO/SCOD/CSD/PCI</td>
</tr>
<tr>
<td>Co-Authors</td>
<td>De Frutos Bolzoni L.</td>
<td>21 Jan 2019:signed</td>
<td>IO/DG/COO/SCOD/CSD/PCI</td>
</tr>
<tr>
<td></td>
<td>Kusurkar A.</td>
<td>21 Jan 2019:signed</td>
<td>IO/DG/COO/SCOD/CSD/PCI</td>
</tr>
<tr>
<td>Reviewers</td>
<td>Evrard B.</td>
<td>25 Jan 2019:recommended</td>
<td>IO/DG/COO/SCOD/CSD/PCI</td>
</tr>
<tr>
<td></td>
<td>Kim C.</td>
<td>26 Feb 2019:recommended</td>
<td>IO/DG/COO/SCOD/CSD/PCI</td>
</tr>
<tr>
<td></td>
<td>Petitpas P.</td>
<td>18 Feb 2019:recommended</td>
<td>IO/DG/COO/SCOD/CSD/PCI</td>
</tr>
<tr>
<td></td>
<td>Pimentel L.</td>
<td>31 Jan 2019:recommended</td>
<td>IO/DG/COO/SCOD/CSD/PCI</td>
</tr>
<tr>
<td></td>
<td>Soni J.</td>
<td>05 Feb 2019:reviewed</td>
<td>IO/DG/COO/SCOD/CSD/PCI</td>
</tr>
<tr>
<td>Approver</td>
<td>Wallander A.</td>
<td>27 Feb 2019:approved</td>
<td>IO/DG/COO/SCOD/CSD</td>
</tr>
</tbody>
</table>

**Read Access**

LG: PLC group, LG: CODAC team, AD: ITER, AD: External Collaborators, AD: IO, Director-General, AD: EMAB, AD: OBS - Plant Control and Instrumentation Section (PCI), AD: Auditors, AD: ITER Management Assessor, project administrator, RO, AD: OBS - Control System Division (CSD) - EXT, AD: OBS - CODAC Sec...
<table>
<thead>
<tr>
<th>Version</th>
<th>Latest Status</th>
<th>Issue Date</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.0</td>
<td>In Work</td>
<td>06 Dec 2010</td>
<td></td>
</tr>
<tr>
<td>v1.2</td>
<td>Signed</td>
<td>10 Jan 2011</td>
<td>Integration of John Poole Comments. Ready for External Review.</td>
</tr>
<tr>
<td>v1.4</td>
<td>Approved</td>
<td>30 Jan 2013</td>
<td>V7.0 Update</td>
</tr>
<tr>
<td>v2.0</td>
<td>Approved</td>
<td>18 Jan 2019</td>
<td>This is a Major Update. Adapted to the current approach of the Control System Division.</td>
</tr>
</tbody>
</table>

**[Software]**:

- Software Architecture of the PLC Application is totally revamped.
- PLC Objects Library concept is added.
- Hardware Access Layer is completely reshaped.
- Health Monitoring System information is increased.
- OPC-UA Transmission Protocol is added.
- New SPSS for 1500 PLC is added.

Numbering and naming convention rebuild.

**SIEMENS Software**:

- TIA Portal added as a Tool.

Development of the PLC Application revamped.
- Software Generation Tools Added.

Examples and templates adapted to the current philosophy.

**[Hardware]**:

- 1500 Family is added.

Hardware Naming Convention is added.

**[Supporting documents]**:

The supporting documentation has been updated and improved.
# Table of Contents

Table of Contents .................................................................................................................. 1

Table of Figures .................................................................................................................... 4

1 Introduction .......................................................................................................................... 6
  1.1 PCDH Context .................................................................................................................. 6
  1.2 Purpose of document ....................................................................................................... 7
  1.3 Scope ................................................................................................................................ 7
  1.4 Organization of document ............................................................................................... 7
  1.5 Acronyms .......................................................................................................................... 8
  1.6 Definitions ........................................................................................................................ 9
  1.7 Reference Documents .................................................................................................... 10

2 Context and Constraints ..................................................................................................... 11

3 General Requirements of PLC-Applications of the ITER Plant Systems ...................... 12

4 Software Architecture of the PLC application ................................................................... 13
  4.1 Introduction: ..................................................................................................................... 13
  4.2 PLC Core Application .................................................................................................... 17
    4.2.1 PLC Core Application Surroundings: ....................................................................... 17
    4.2.2 PLC Core Application Body: ................................................................................... 20
      4.2.2.1 Generic Object Description: .............................................................................. 21
        4.2.2.1.1 Object Inputs: .......................................................................................... 21
        4.2.2.1.2 Object Outputs: ...................................................................................... 21
        4.2.2.1.3 Operation Modes: .................................................................................. 22
      4.2.2.2 The objects library: ............................................................................................ 23
        4.2.2.2.1 The Objects: .......................................................................................... 23
        4.2.2.2.2 IO Objects: ............................................................................................ 24
        4.2.2.2.3 Interface Objects: .................................................................................. 26
        4.2.2.2.4 Field Device Objects: .......................................................................... 26
        4.2.2.2.5 Control Objects: .................................................................................... 27
      4.2.2.3 Software Architecture of the PLC Core Application: .................................... 28
        4.2.2.4 Standard structure of a Process Function (PCO) ........................................... 29
  4.3 CODAC Interface ............................................................................................................. 31
    4.3.1 Introduction to the CODAC Interface .................................................................. 31
      4.3.1.1 SPSS ........................................................................................................... 31
      4.3.1.2 OPC-UA (currently only for 1500 PLC) ......................................................... 31
    4.3.2 Exchanged data in the CODAC Interface .............................................................. 33
    4.3.3 Configuration of the CODAC Interface ................................................................. 35
      4.3.3.1 OPC-UA Configuration .............................................................................. 35
        4.3.3.1.1 OPC UA Server Configuration ............................................................... 35
        4.3.3.2 SPSS Configuration .................................................................................. 38
        4.3.3.2.1 SPSS Creation Procedure .................................................................. 39
  4.4 Hardware inputs/outputs interface (Hardware Access Layer) ....................................... 42
    4.4.1 General Description ............................................................................................... 42

---

Page 1 of 91
7.8.1 Introduction .......................................................................................................................... 77
7.8.2 Construction .......................................................................................................................... 77
  7.8.2.1 Development Cycle and deliverables ............................................................................... 77
    7.8.2.1.1 Requirements Specification ......................................................................................... 78
    7.8.2.1.2 Design Specification .................................................................................................. 78
    7.8.2.1.3 Coding/Unit Testing .................................................................................................. 79
    7.8.2.1.4 Simulated Validation Testing (FAT First phase) .......................................................... 79
    7.8.2.1.5 Integrated Validation Testing (FAT second phase) ..................................................... 80
    7.8.2.1.6 Site Acceptance Test .................................................................................................. 80
  7.8.2.2 Languages ......................................................................................................................... 80
  7.8.3 Generation ............................................................................................................................ 82

7.9 Summary of the ITER Development Tools/Libraries ............................................................... 83

7.10 Alarms Management .............................................................................................................. 85

7.11 Coding Rules............................................................................................................................ 85

8 Software Configuration Management .......................................................................................... 87

9 Annexes ........................................................................................................................................ 88
  9.1 Already Reserved Blocks for CODAC ...................................................................................... 88
  9.2 Already Reserved Global Variables for CODAC ....................................................................... 89
  9.3 Brief Example about Cooling Water System ............................................................................ 89
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Schema of PCDH satellite documents</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Schema of ‘PLC Software Engineering Handbook’ satellite documents</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>CODAC Architecture</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>PLC Conceptual Software Architecture</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>PLC Core Application Environment</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Simple Example of CODAC HMI</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>Generic Object Description</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>Software Architecture of the PLC Core Application</td>
<td>28</td>
</tr>
<tr>
<td>9</td>
<td>Conceptual Design of a Control Function in the Core Application</td>
<td>29</td>
</tr>
<tr>
<td>10</td>
<td>Collaborative Data</td>
<td>33</td>
</tr>
<tr>
<td>11</td>
<td>Activate OPC UA Server</td>
<td>35</td>
</tr>
<tr>
<td>12</td>
<td>OPC UA License</td>
<td>36</td>
</tr>
<tr>
<td>13</td>
<td>Server Settings</td>
<td>36</td>
</tr>
<tr>
<td>14</td>
<td>Add New Subnet</td>
<td>37</td>
</tr>
<tr>
<td>15</td>
<td>Main Cycle Loop Standard Structure</td>
<td>38</td>
</tr>
<tr>
<td>16</td>
<td>Main Cycle Loop Standard Structure</td>
<td>38</td>
</tr>
<tr>
<td>17</td>
<td>Warm Restart Block Standard Structure</td>
<td>39</td>
</tr>
<tr>
<td>18</td>
<td>Hardware configuration compiled to generate ‘System data’.</td>
<td>39</td>
</tr>
<tr>
<td>19</td>
<td>Add standard software structure as external source.</td>
<td>40</td>
</tr>
<tr>
<td>20</td>
<td>Hardware I/O Interface (Hardware Access Layer) SW Architecture</td>
<td>42</td>
</tr>
<tr>
<td>21</td>
<td>Flow of the Health Data</td>
<td>47</td>
</tr>
<tr>
<td>22</td>
<td>Control Block of a CBS Level 4 Function.</td>
<td>52</td>
</tr>
<tr>
<td>23</td>
<td>General Network Naming</td>
<td>56</td>
</tr>
<tr>
<td>24</td>
<td>SIMATIC Manager Language Setting.</td>
<td>57</td>
</tr>
<tr>
<td>25</td>
<td>SIMATIC Manager Date and Time of Day Format.</td>
<td>58</td>
</tr>
<tr>
<td>26</td>
<td>LAD/FBD Layout</td>
<td>58</td>
</tr>
<tr>
<td>27</td>
<td>Symbol Editor Import</td>
<td>59</td>
</tr>
<tr>
<td>28</td>
<td>Address Priority</td>
<td>59</td>
</tr>
<tr>
<td>29</td>
<td>HW Config Screen for a CPU Stations with 3 Remote IO Racks.</td>
<td>60</td>
</tr>
<tr>
<td>30</td>
<td>CPU Config Clock Memory Setup</td>
<td>61</td>
</tr>
<tr>
<td>31</td>
<td>CPU Config Startup</td>
<td>61</td>
</tr>
<tr>
<td>32</td>
<td>CP Time-of-Day Synchronization</td>
<td>62</td>
</tr>
<tr>
<td>33</td>
<td>Remote IO Rack Board organization.</td>
<td>63</td>
</tr>
<tr>
<td>34</td>
<td>Ethernet Port Config End of Topology.</td>
<td>64</td>
</tr>
<tr>
<td>35</td>
<td>TIA Language Setting</td>
<td>65</td>
</tr>
<tr>
<td>36</td>
<td>LAD/FBD Layout</td>
<td>65</td>
</tr>
<tr>
<td>37</td>
<td>TIA Network View for CPU Stations with 3 Remote IO Racks.</td>
<td>66</td>
</tr>
<tr>
<td>38</td>
<td>CPU Config System and Clock Memory Setup.</td>
<td>66</td>
</tr>
<tr>
<td>39</td>
<td>CP Time Synchronization</td>
<td>67</td>
</tr>
<tr>
<td>40</td>
<td>PROFINET Port Config End of Topology.</td>
<td>68</td>
</tr>
<tr>
<td>41</td>
<td>General PLC Software Architecture</td>
<td>69</td>
</tr>
<tr>
<td>42</td>
<td>UDTs and DBs organisation and dependencies for “CWS-DHLT-WFC”</td>
<td>71</td>
</tr>
<tr>
<td>43</td>
<td>UDTs and DBs organisation and dependencies for TCP connexion</td>
<td>72</td>
</tr>
<tr>
<td>44</td>
<td>Closed Loop Control Example</td>
<td>72</td>
</tr>
<tr>
<td>45</td>
<td>Add plant system specific STL file generated by SDD Editor.</td>
<td>73</td>
</tr>
<tr>
<td>46</td>
<td>Software Architecture of the Hardware Access Layer</td>
<td>74</td>
</tr>
<tr>
<td>47</td>
<td>Core Application Development Life Cycle</td>
<td>77</td>
</tr>
<tr>
<td>48</td>
<td>Scope of the ‘Code Generator Tools’</td>
<td>84</td>
</tr>
</tbody>
</table>
Figure 49: CWS Case of Study Example
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 satellite 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 and industrial approach for the development of the Control Functions of the controllers.

1.3 Scope

This document covers the Development of Software for PLC Conventional Controllers. It is not covering, nor Safety (Occupational or Nuclear) neither Interlock (so SIL-3 PLC, Simatic F/FH Series).

1.4 Organization of document.

The ‘PLC Software Engineering Handbook’ is a satellite of the ‘core PCDH’ (as we can see in Figure 1) and, in turn, is orbited by a constellation of satellite documents intended to support it.

Figure 2: Schema of ‘PLC Software Engineering Handbook’ satellite documents

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 depict, 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 it:

a) Software Architecture of the PLC Application
   - PLC Core Application
   - CODAC Interface
   - Hardware Access Layer
   - PLC and Fast Controller Interfaces
   - Health System Monitoring

b) Numbering and naming convention (required all along the development of the PLC Application)

c) Programming Environment Configuration (including the Hardware Configuration)

d) Development of the PLC Application
   - Introduction, Construction and Generation of all the parts with the ITER provided tools.

e) Software Control Management

f) Examples and Templates (Annex)

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

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSPS</td>
<td>Standard Software PLC Structure</td>
</tr>
<tr>
<td>PLC</td>
<td>Programmable Logic Controller</td>
</tr>
<tr>
<td>PBS</td>
<td>Process Breakdown Structure</td>
</tr>
<tr>
<td>CBS</td>
<td>Control Breakdown Structure</td>
</tr>
<tr>
<td>FC</td>
<td>Function Chart</td>
</tr>
<tr>
<td>FB</td>
<td>Function Block</td>
</tr>
<tr>
<td>DB</td>
<td>Data Block</td>
</tr>
<tr>
<td>UDT</td>
<td>User Data Type</td>
</tr>
</tbody>
</table>
1.6 Definitions

<table>
<thead>
<tr>
<th>term</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC Application</td>
<td>All Software developed in a PLC</td>
</tr>
<tr>
<td>PLC Core Application</td>
<td>All Software or Control Blocks implementing the Control Functions. All what is not implemented in the Peripheral Blocks</td>
</tr>
<tr>
<td>Shared DB variable</td>
<td>Generic term used for any global access variable in a PLC</td>
</tr>
<tr>
<td>Process Variable</td>
<td>Generic term used for a Variable in the EPICS environment.</td>
</tr>
<tr>
<td>Plant System I&amp;C</td>
<td>Person Responsible of Programming CODAC, PLC or Fast Controller applications.</td>
</tr>
<tr>
<td>Configuration Database</td>
<td>Database with the ‘SW’ or ‘HW’ configuration of the project.</td>
</tr>
<tr>
<td>Peripheral Blocks</td>
<td>PLC software Blocks implementing the interfaces, the Health Monitoring and the Hardware Access Layer.</td>
</tr>
<tr>
<td>Configuration</td>
<td>Set of all configuration variables transmitted to the PLC.</td>
</tr>
<tr>
<td>Configuration Variable</td>
<td>An EPICS PV transmitted to the PLC through a Shared DB Variable.</td>
</tr>
<tr>
<td>States</td>
<td>Set of all PLC variables transmitted to the CODAC.</td>
</tr>
<tr>
<td>State Variable</td>
<td>A PLC Shared DB variable transmitted to the CODAC through an EPICS PV.</td>
</tr>
<tr>
<td>Standard Control Block</td>
<td>In and Out parameters of a FC or a FB for a Control Block deployed in the PLC Core Application</td>
</tr>
<tr>
<td>Control Function</td>
<td>Function achieved by a Controller in the Context of a Functional Analysis.</td>
</tr>
<tr>
<td>Control Block</td>
<td>FC of FB in the Context of a Siemens Step 7 application.</td>
</tr>
</tbody>
</table>
## 1.7 Reference Documents

<table>
<thead>
<tr>
<th>IDM Number</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>[RD 1] 2UT8SH</td>
<td>Signal and plant system I&amp;C Variable Naming Convention</td>
</tr>
<tr>
<td>[RD 2] 28QDBS</td>
<td>ITER Numbering System for Components and Parts</td>
</tr>
<tr>
<td>[RD 3] 34V362</td>
<td>The CODAC – Plant System Interface</td>
</tr>
<tr>
<td>[RD 4] 353AZY</td>
<td>Methodology for Plant System I&amp;C specifications</td>
</tr>
<tr>
<td>[RD 5] 2FJMPY</td>
<td>ITER Function Category and Type for ITER Numbering System</td>
</tr>
<tr>
<td>[RD 6] 32GEBH</td>
<td>Plant System I&amp;C Architecture</td>
</tr>
<tr>
<td>[RD 8] 35W299</td>
<td>CWS case study specifications</td>
</tr>
<tr>
<td>[RD 9] 333J63</td>
<td>ITER catalogue for I&amp;C products - Slow controllers PLC</td>
</tr>
<tr>
<td>[RD 11] N3Y8M7</td>
<td>PLC control logic standardisation for PBS 34</td>
</tr>
<tr>
<td>[RD 14] 34QXCP</td>
<td>Self-description Data Documentation and Guidelines</td>
</tr>
<tr>
<td>[RD 17] Q6L4ZV</td>
<td>S7 Hardware Naming Conventions</td>
</tr>
<tr>
<td>[RD 18] UVYE98</td>
<td>Benchmark information about the communication performance between CODAC and the PLCs</td>
</tr>
<tr>
<td>[RD 19] 4CK4MT</td>
<td>ITER System Design Process</td>
</tr>
<tr>
<td>[RD 20] R32A77</td>
<td>Version Management Using Subversion</td>
</tr>
</tbody>
</table>
2 Context and Constraints

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. The PLC or the PSH will implement the PSOS.

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; the standard Plant Operational Network.

Figure 3: CODAC Architecture
3 General Requirements of PLC-Applications of the ITER Plant Systems

- **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**:
  - Following the deliverables advised [RD 10] the system must be provided with enough information to be understandable and maintained for ITER IO.
  - The PLC Application should be built in a way that modifications have 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 objective is to test the Control System disconnected from the real process and, eventually, 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 the PLC application.

4.1 Introduction:

The objective is to have a common software architecture of the application inside all the PLCs deployed on the Project. Depending on the ‘Plant System Hardware Architecture’ all the blocks of the ‘PLC Conceptual Software Architecture’ (depicted above) might not be present.

The ITER software architecture is based on the following standards:

- ISA-88 : Standard addressing batch process control
- IEC-61512 : Standard addressing batch process control
- IEC-61499 : Standard addressing the topic of function blocks for industrial process measurement and control systems
Briefly; number by number:

1. **PLC Core Application**: Core Structure of the PLC Code. Implements the main functions of the PLC Application.
2. **CODAC Interface**: Implements the bidirectional communication with CODAC.
3. **Hardware Outputs/Inputs Interface**: Implements the layer of access to the I/O Modules (Hardware). Also called Hardware Access Layer.
4. **PLC Interface**: Implements the interface to other/s PLC on the network; normally using a plant local network.
5. **Fast Controllers Interface**: Implements the interface to Fast controller/s; using a dedicated network.
6. **Link ‘Hardware Outputs/Inputs Interface’ to ‘CODAC Interface’**: Unidirectional transfer of ‘Raw data’ from the ‘Hardware Access Layer’ to CODAC.
7. **System Monitoring**: Implements the ‘Health Status’ of the Hardware. Also called ‘Health monitoring’.
8. **Link ‘CODAC Interface’ to ‘Hardware O/I Interface’**: Unidirectional transfer of ‘Configuration data’ from CODAC to the ‘Hardware I/O Interface’ to parametrize this layer.
9. **Link ‘System Monitoring’ to ‘CODAC Interface’**: Unidirectional transfer of ‘Health monitoring information’ to CODAC.
10. **Link ‘HW O/I Interface’ to ‘System Monitoring’**: Unidirectional transfer of data from ‘Hardware access layer’ to ‘System Monitoring’ in order to be processed and to extract ‘Health Status’ information.
11. **Link ‘HW O/I Interface’ to ‘PLC Core Application’**. Bidirectional transfer of information:
   - ‘Hardware Access layer’ data is send to ‘PLC Core Application’ in order to be processed and to extract the status of the variables of the process system.
   - ‘PLC Core Application’ data is sent to the ‘Hardware Access Layer’ to configure the process system.
12. **Link ‘PLC Interface’ to/from ‘PLC Core Application’**: Bidirectional exchange of information between the ‘Core Application’ and the ‘Interfaced PLC/s’.
13. **Link ‘Fast Controllers Interface’ to/from ‘PLC Core Application’**: Bidirectional exchange of information between the ‘Core Application’ and the ‘Interfaced Fast Controllers/s’.
14. **Link ‘System Monitoring to ‘PLC Core Application’**: Unidirectional transfer of ‘System Monitoring’ data sent to ‘PLC Core Application’ to be processed.
15. **Link ‘PLC Core Application’ to/from CODAC Interface**: Bidirectional exchange of data to/from the ‘Core Application’.
   - States/Events performed in the ‘Core Application’ are reported to CODAC.
   - Configurations/Commands sent from CODAC to set up the parameters of the functions in the ‘PLC Core application’.
The Objective is the standardization:

The main blocks and the structure depicted in Figure 4 are looking for an objective; the standardization.

PLC Core Application (1 on Figure 4):
The ‘High Level Functions’ on the ‘PLC Core Application’ will be dedicated to every controller, so, different for every PLC.
But, except for these ‘High Level Functions’ of the PLC Core Application the inside structure of this block and all the other will be standard for all PLCs deployed on the Project. Only the volume and structure of the data computed in these blocks will be different. As far as possible, these blocks will be generated automatically, using a “Configuration Database” as input.
The ITER guidelines, tools and methods for the standardization and automatic generation of this block will be discussed in dedicated subchapters for the ‘PLC Core Application’: [4.2] and [7.8].

CODAC Interface (2 on Figure 4):
The ‘CODAC Interface’ is already fully transparent to the user.

a) The ‘CODAC Interface’ will be implemented by:
  - **SPSS** for S7-300/400 and S7-1500. Refer to [RD 12].
  Or
  - **OPC-UA** for S7-1500. Refer to [RD 13].

  As you can see, in case the developer is working with S7-1500, the ‘CODAC Interface’ could be ‘SPSS’ or ‘OPC-UA; user must take a decision between these two.
  For a comparison between these two methods refer to [RD 13] (Chapter 3.4.3).

b) The User DataBlocks containing the exchanged variables will be generated through an ‘AWL File’ by the SDD application; then imported to the SIEMENS software.

The ITER guidelines, tools and methods for the standardization and automatic generation of this block will be discussed in the dedicated subchapters for the ‘CODAC Interface’ [4.2.2.4] and [7.2].

Hardware Access Layer and System Monitoring (3 and 7 in the Figure 4)
The structures of these two blocks must be the same for every PLC in the ITER Control System.
The ITER guidelines, tools and methods for the standardization and automatic generation of this block will be discussed in ‘Hardware Access Layer’ [4.4] and [7.3] and ‘System Monitoring’[4.7] and [7.7].
To highlight:

- A “Master Controller”, so a ‘High Level PLC’ managing other ‘process PLCs’, in an I&C architecture (see [RD 6], Figure 3.2.) will not have any ‘Hardware Inputs/ Outputs Interface’ and it will have a lot of interfaces with other PLCs of the Plant System, eventually using ‘Plant Local Networks’. Still, ITER methodology strongly encourages to connect every single PLC to PON (Plant Operative Network).

- Fast Controllers Interfaces will probably be very rare and may use the CODAC interface, not necessarily the ‘Fast Controller Interface’, as Fast Controllers are running EPICS and are consequently connected to Channel Access. Another option is to use a HW Link, e.g. via a DIO connection.
4.2 PLC Core Application

4.2.1 PLC Core Application Surroundings:

The PLC Core Application will implement the “Control Functions.” Its operation will be affected by all the interfaces represented on Figure 4. All programming or treatment not directly involving the process is performed in the other “Peripheral blocks” (i.e.: Interfaces, System monitoring).

The PLC Core Application (“1” on Figure 4) is the place where the Control Logics, Grafcets, State Charts, Regulation Loops of the process will be implemented. Is the place to find only the process programming. The developer should focus the efforts in it.

In some Plant Systems the ‘PLC Application’ will be highly complex; however in others, it will be merely a gateway between the Field and Central Control System, a simple mapping from the signals to the variables.

The following figure represents a simplified view of the ‘PLC Core Application Environment’:

![Diagram of PLC Core Application Environment]

*Figure 5: PLC Core Application Environment*
Upstream data of the PLC Core Application:
The ‘PLC Core Application’ delivers to CODAC two different types of data: State variables and events. So, they are processed by the ‘Core Application’ and then reported to the CODAC.

The ‘PLC Core Application’ receives from CODAC three different types of data: Configuration, Commands and Collaborative Data. Configuration variables (See Figure 5) transmitted by the ‘CODAC interface’ are the main inputs for the operation; they are set by for the operator (or for the ‘supervisory PLC control system’ in case, ‘PLC program’ reacts automatically).

Some States variables examples:
- Current of a Phase
- Voltage of bus bar
- Local/Remote position of a key to access the control system
- …

Some Events:
- Trip of a circuit breaker
- Instantaneous earth fault overcurrent
- …

Some Configuration variables examples:
- Current Setpoint,
- Temperature Setpoint
- …

Some ‘Simple Commands’ examples:
- ON/OFF requests
- OPEN/CLOSE requests,
- HIGH VACUUM/ROUGHING/VENTING request,
- …

Finally, collaborative data (Figure 5) is a special type of data; it is said about a variable or a set of variables coming from another ‘Plant System’ with the aim to influence the process of the Plant System owning the programmed controller. So, they are State Variables produced by other Plant Systems and Transmitted by CODAC Core System in the form of Configurations. In PCDH transversal wired links between Plant System are strictly forbidden. Transmission of information between Plant System will use the Collaborative Data link.

Downstream data of the PLC Core Application:
Hardware Inputs and Outputs Interface (Figure above) deals with the field hardware, it gathers the inputs and delivers the outputs; this data is raw, not processed or treated in any way. The PLC Core Application, make a complete abstraction of the fact that these values are coming from real equipment or simulated.
Sides of the PLC Core Application:
The ‘System Monitoring’ and the ‘PLC or Fast Controllers’ interfaces within the same Plant System I&C also impacts the processing inside the ‘Core Application’, this point will be developed in further chapters.

In conclusion:
The PLC Core Application:
   a) It will compute the CODAC Configuration variables, the commands and the ‘HW Field Inputs’.

   b) It will execute the process logic.

   c) It will generate the ‘HW outputs’ in order for the ‘Plant System’ to reach the configuration requested.
      It will generate the States and the events reporting to CODAC the effective Status of the process.

Assets of this software architecture:
The main principle is that on the CODAC it should be always possible to have an easy comparison between a ‘Configuration’ requested from the HMI to the Process, and the effective ‘State’ of this Process.

A simple example is given in Figure 6 of what will be a CODAC HMI for a simple device.

![Figure 6: Simple Example of CODAC HMI](image)
4.2.2 **PLC Core Application Body:**

The body of the PLC Core Application is the brain of the system. The body of the software structure of the ITER PLCs will be programmed object-oriented. Object-oriented programming (OOP) is a programming language model organized around “objects” rather than “actions” and “data” rather than “logic.” The ‘PLC Core Application body’ will be a logical procedure that takes input data, processes it, and produces output data.

**Assets:**

- Modularity: optimize design, tests and maintainability.
- Abstraction: During the programming, the developer will work with objects, only the interface of the object is visible when the object is placed in the code, it reduces complexity.
- Generic: Any module can be plug-in without major impact.
- Re-use: Minimize engineering effort.

Object-oriented programming is especially useful in the phase of commissioning in order to identify physical objects of the field with ‘software objects’ inside the programming code.

A ‘Standard PLC Object Library’ will be used in the PLCs in order to develop the ‘core application’.

This document is going to explain in the following chapters the generalities of this methodology; in three steps, from the basic object to the general software architecture:

1. Generic Object Description
2. The objects in the library
3. General Software Architecture of the Core Application

*To notice:

a) ITER Cryogenic System Section targeting specifically the Cryoplant (PBS 34) developed a ‘PLC Object Library’, currently available. To know more information about it refer to [RD 11].

b) ITER Control System Division targeting all ITER Plant Systems is in phase of development of a ‘PLC Standard Object Library’. (* document to come)
4.2.2.1 Generic Object Description:

The Generic description of any object in the library is presented in the following figure.

Following, the description of every box in the figure.

4.2.2.1.1 Object Inputs:

In the input side, the following specialized roles overload the mere interface definition – when applicable and/or necessary:

- **Environment Inputs**, which collects signals both the field, by means of specialized hardware, and other objects hierarchically linked.
- **Manual Requests**, refers to commands and parameters (configurations), which are defined by the operator of the system through the HMI at the supervisory level by means the operator workstation.
- **Interlock Requests**, refers to process signals that convey executive orders to force the component into a controlled state (nominally “STOP”) or to not-exit the controlled state (to not START).
- **Auto Requests**, refers to commands and parameters produced by other components or by the process logic inside the software system.

4.2.2.1.2 Object Outputs:

In the output side, variables can belong to more than one group, but following roles are considered:

- **Environment Outputs**, which conveys signals both the field, by means of specialized hardware, and other objects hierarchically linked.
• **Control Logic Outputs**, refers to statuses produced by the component that may be used both by other components and process control logic inside the software system.

• **HMI Outputs** is an interface, which echoes internal variables of the component in order to be accessed by the supervisory level through the operator workstation.

**Hardcoded parameters** refer to a hard-coded configuration, which may imply a forced constraint placed to control the diversity of a meta-component into a specific component instance and, therefore, only available from the engineer workstation.

4.2.2.1.3 *Operation Modes:*

The object operation mode identifies whether the object is driven by the operator, the control logic or the process field. Only one mode can be on at a given time. The setting of one of the operation modes automatically resets the other modes. These modes are:

**Auto Mode:** The object is driven by the control logic of a higher object of the hierarchy.

**Manual Mode:** The operator drives the object via the terminal. The operator will be able to set the output of the object, selected for the operator, no matter of the input.

**Forced Mode:** The operator drives the object via the terminal. The operator will be able to force the output of the object, selected for the operator, no matter of the input. The automatic return to the auto mode is impossible by the control logic.

Not to confuse the ‘Forced Mode’ with the complete ‘Simulation Mode’ that could be selected using the ‘Hardware Access Layer’.

**Local Mode:**

Two types of local mode exist: Hardware and Software.

• **Hardware Local Mode:**
  o The object is driven locally by the process field, mostly for maintenance purposes.

• **Software Local Mode:**
  o Requested by operators via the Local panels.
  o The Local software is writing directly in the manual requests of the objects.

It is convenient to remember that in the requirement [R11] of [RD 10] the use of ‘Local Control Rooms’ is strictly forbidden.
4.2.2.2 The objects library:

**Introduction:**
The objective of the library 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 using 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); still the Control of the valve is identical. The function of these standard blocks would be to implement the internal logic required to all the discrete signals, in order to present a ‘standard signal interface’ for the different types of devices inside the PLC Core Application. All the ‘internal logic’ parameters will be provided by ‘CODAC configuration variables’. The only exception will be the ‘Hardcoded Parameters’, directly provided for the ‘Engineering Workstation’.

4.2.2.2.1 The Objects:

The components of the library will be divided in the following groups:

- IO Objects
- Interface Objects
- Field Objects
- Control Objects
4.2.2.2 IO Objects:

General functionality:
Base components; interface with the field, with fieldbuses or even with internal memory.

In generic terms, the IO Object:

a) Input Object:
   • Receives Field Inputs from the ‘HW Access Layer’.
   • Delivers data to HMI and to the above object in the ‘Field Objects Layer’.
   • Receives data from the ‘Health System Monitoring’.

b) Output Object:
   • Delivers Field Outputs to the ‘HW Access Layer’
   • Receives data from the HMI and from the above object in the ‘Field Objects Layer’.
   • Receives data from the ‘Health System Monitoring’.

The objects integrating this layer perform conditioning, scaling, filtering, etc. as it will be explained in the following paragraphs.

Objects list:


Examples:
The Analog Input could represent, for instance, a temperature transmitter.
The Analog Output could represent, for instance, a proportional valve position order.
The Digital Input could represent, for instance, an ‘end contact’.
The Digital Output could represent, for instance, a run command for a pump.

General skills of the object:
In the following paragraph, we summarize set of basic characteristics of these objects:

CBS Wrapper:
A deep and important task of the ‘IO Objects’ is to transform a Signal to a Variable, so, to transform a PLC Variable, from a PBS format to a CBS format (and viceversa in the case of the outputs).
Focusing in an input object, it will receive a signal, following a ‘PBS’ structure as ‘PPPPPP- TTT-NNNN:AAAA[RRRR]-SSS’ and it will convert to a variable, following a ‘CBS’ structure as ‘CBS1-CBS2-CBS3-CBS4-CBS5:TTTNNNN-AAAA’, the depth of the CBS level depends on the application. The output will work the other way around. See [RD 1].

Example:

- Signal PBS 43 : 43AGT0-BR-1000:CRST01-CCC
- Variable CBS UTIL : UTIL-HV-T15:BR1000-CRST.
Filtering:
Targeting digital objects, the debounce filter feature permits to fix a bouncing input to a stable value by configuration.
Targeting Analog objects, the first order filter feature permits to fix a trembling input to a stable value by configuration.
(* The implementation of the filtering feature for the ‘Digital Input Object’ in the UCL library is under discussion.)

Engineering Conversion (scaling):
Targeting Analog objects, this feature permits to transform raw data to an engineering value or vice versa.
For the 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’ are part of the ‘Hardware Parameters’ and they will be provided by the ‘Engineering Console’.

Engineering Limits (Out of range detection):
Targeting Analog objects.
For Analog inputs, whenever the raw value is less or more than what it is configured the out of range warning will be generated.
For Analog 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.

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 consider this fact in the software design; implementing a standard functionality to force a signal and a traceable way to check which signals are currently forced. The idea here is to avoid non-regular and even dangerous “temporary-permanent” practices like forcing some signal with PLC hardcoded modifications, hardwired modifications or screwdrivers stuck in the relays.

Important points to take in account about this feature:
• Some signals are particularly perilous to be forced at any time because of its risk to endanger equipment or personal.
• The ‘control system’ must not reach a standard operational state as long as signal variables are forced. The developer will program this behaviour.
• The object will incorporate the option to inhibit this forcing feature.

All permanent and runtime parameters will be provided by CODAC configuration variables.
4.2.2.3  Interface Objects:

General functionality:
These objects have a more simple architecture than IO Objects. Parametrization and status; can be connected to the periphery.

Objects list:


General skills of the object:
The object parameter is used basically to send data from the HMI to the PLC. The object status is used basically to send data from the PLC to the HMI.

4.2.2.4  Field Device Objects:

General Functionality:
They represent in the ‘PLC Software’ the acting devices in the field. They model the real field equipment, like pumps, valves, etc. As a general rule, the field objects are connected to the I/O Objects. NO direct connection to the PLC periphery.

Objects list:

| OnOff, Analog, AnaDO, Local |

General skills of the object:
OnOff is a binary object, for instance an on/off valve, motor or a pump. Analog is an Analog object, for instance a control valve or a heater. AnaDO behaves like an OnOff object plus an Analog object, for instance motors driven by inverters, thyristors or heaters can be represented with it. Local is a non-remotely controlled object like a manual valve.

Point Inversion: Targeting the OnOff Object, this feature permits to invert the output data of the object by configuration. This parameter allows the user to program logic functions using positive logic. The final value will be automatically inverted by the object. In some particular cases, the Plant System will need complex dedicated Field Device Objects. For these objects (like positioning modules) the solution will consist:

- Create a bundle of the presented field objects, combining functionalities and giving birth to a super object.
- OR
- Creating a new complex object from scratch, and follow the same programming principles as much as possible.
4.2.2.2.5 Control Objects:

Functionality:
Main objects holding the control logic, the alarms, and the feedback of the controllers. Control objects are acting on the Field Objects.

Objects list:

| Process Control Object, Controller, Alarm |

General skills of the object:
**Controller:** It contains a feedback regulation mechanism. For instance, acts as a PID; calculates PLC output actions based on set-point and PLC input measurements.
**Alarm:** It models alarms; triggers an alarm depending of a PLC input.
**Process Control Object** (PCO): Finally the PCO implements the high level control logic. It is the highest object in hierarchy. It gives orders to their affected fields or to the dependant PCO Objects.

In addition, the Process Control Objects is subdivided in categories, as per the scope of the control object:

a) **PCO A: Unit (Highest hierarchy level).**
   A collection of associated control modules and/or equipment modules and other process equipment in which one or more higher level processing activities are conducted.

b) **PCO B: Equipment Module : Functional Group**
   It is a functional group of equipment that can carry out a finite number of specific minor processing activities.
   An equipment module is typically centered around a piece of process equipment (a weigh tank, a process heater, a scrubber, etc.).
   In a complex process we can have more than one level of ‘equipment modules’, depending of the complexity of the process.

b) **PCO C: Control Module : Device**
   It is the lowest level grouping of equipment in the physical model that can carry out basic control.
   This term applies to both the physical equipment and the equipment entity.
4.2.2.3 Software Architecture of the PLC Core Application:

The objects will be structured in a series of juxtaposed layers:

- From the ‘Basic IO objects’ layer, at the bottom of the architecture; in direct contact with the hardware; so, feed by the ‘Hardware Access Layer’ and the ‘System Monitoring’.
- To the ‘Control objects’ layer, at the top of the architecture; implementing the ‘High Level functions’ that define the general functionality of the system.

The objects in the code will ‘mirror’ each element of the real process to control and they will be interrelated.

The Software architecture is depicted in the figure below:

![Diagram of Software Architecture of the PLC Core Application](image)

Figure 8: Software Architecture of the PLC Core Application

To know about the currently suitable ITER tools for the automatic generation of the ‘PLC Core Application’ refer to the chapter: [Summary of the ITER Development Tools/Libraries].
4.2.2.4 Standard structure of a Process Function (PCO)

In figure below, the conceptual organization of a Control Function is represented: it is mainly composed of four parts: the Interlocks Logic, Configuration Logic, Standard Object Logic, and Specific Process Object Logic.

![Diagram of Control Function organization](image)

*Figure 9: Conceptual Design of a Control Function in the Core Application*

- **Interlocks logic:** 3 interlocks are possible:
  1. Start Interlock (prevents the object from starting).
  2. Full Stop Interlock (stops the object). Restarting requires the elimination of the interlock and the operator attending to it).
  3. Temporary Stop Interlock (stops of the object). Restarting does not require the intervention of the operator except when in manual or forced mode.

- **Configuration logic:**
  This logic determines whether the object will be on or off pending on the status of dependent objects. One can also originate a controlled stop. In the case of an orderly pause at the request or an interlock (off), all orders associated with the objects dependent on the process are stopped. In the case of a controlled stop, before
interrupting the orders, the object is put in a configuration defined by the specific logic of the process.

- **Standard Object Logic:**
  There are several optional modes such as the start-up phase, the warm up phase. There are eight possible modes which can be set either by the command logic or by the operator. If the process is stopped, one can go from one to another without restrictions. On the other hand, if the process is active, the passage from one mode to another must be authorized.

- **Specific Process Object Logic:**
  The operation point of the process depends on the logic of the interlocks, logic of configuration and standard logic of the object. A command is then sent to the specific logic of the process.

- Run Order: the process must be activated except if there are stop interlocks or if there is a start interlock and that the process is stopped.

It is necessary also to describe the rest of the boxes:

- **Inputs:**
  Process Inputs, Manual Requests, Auto Requests or Internal Parameters

- **Status:**
  Information to other objects or to the HMI

- **Actions:**
  Actions taken affecting other objects or the Field itself.
4.3 CODAC Interface

The main function of the CODAC Interface (“2” in Figure 4) is to manage the ‘PLC side’ of the communication with the CODAC. The CODAC side of the communication is managed in the PSH running a specific driver. The ‘CODAC Interface’ will use the ‘Exchange-data DBs’ inside the PLC to communicate with CODAC.

4.3.1 Introduction to the CODAC Interface

Currently, ITER provides two methods to implement the ‘CODAC Interface’: SPSS and OPC-UA.

4.3.1.1 SPSS

SPSS is a set of standard blocks inserted in the ‘PLC Application’ that implements the communication tasks between the PLC and the CODAC.

- These blocks are clearly segregated from the user’s application blocks.
- SPSS supports standalone and redundant PLCs to communicate with EPICS.
- It allows communication on integrated or external interfaces (CP) for all ITER PLCs.
- It allows to work with events and commands.
- It accesses the variables in the ‘exchange-data DBs’ by ‘memory offset’.
- The transfer rate is related to the ‘PLC Cycle’ time and to the defined ‘communication load’ percentage.
- All the data of the ‘DB states’ or ‘DB events’ is sent at once.

To know how to use (and much more) about SPSS, refer to [RD 14].

4.3.1.2 OPC-UA (currently only for 1500 PLC)

OPC-UA is a machine-to-machine communication protocol for industrial automation developed by the OPC Foundation. 
OPC-UA does not need inserted blocks in the PLC Application.
The same PLC includes an OPC-UA Server, totally transparent for the user.
The OPC-UA server implements the communication tasks between the PLC and the CODAC.

- OPC-UA supports only standalone PLC S7-1500 (still no redundant on the market).
- It allows communication only in the on-board interface of S7-1500.
- Currently, only states and configurations are working. Not events, neither commands.
- It accesses the variables in the ‘exchange-data DBs’ by ‘name’. So, no need to generate the DBs using AWL coming from SDD (still it can be used). It allows to decouple the PLC Implementation from SDD.
- ‘PLC cycle time’ is affected at the moment the data is subscribed. It is stable only after subscription. It is affected as well by the amount of data but more importantly by the number of nodes accessed.
- Only data changing in ‘DB states’ is sent to CODAC.
- EPICS Driver for OPC-UA is supported for a community (increases robustness).
- The transfer rate it is not related to the ‘PLC Cycle Time’. The OPC-UA server controls it and it is stable (not affected by the amount of data sent).

To know how to use (and more) about OPC-UA, refer to [RD 15].

To have a factual comparison between SPSS and OPC-UA, refer to the same [RD 15].
4.3.2 *Exchanged data in the CODAC Interface*

This communication is broken down in 4 categories, as represented in *Figure 5*:

- **States (DB 100 in the PLC)**
  - Collaborative Data: Data sent to another Plant System and needed to operate the remote one.
- **Events (DB 105 in the PLC)**
- **Configurations (DB 101 in the PLC)**
  - Collaborative Data: Data received from another Plant System and needed to operate the local one.
- **(Simple) Commands (DB 102 in the PLC)**

**States**

The States Variables are transmitting the status of the process:

- Directly from the ‘Hardware Inputs/Outputs Interface’ (Hardware Access Layer “6” in *Figure 4*). This direct link is necessary as the ‘CODAC Applications’ may use these raw variables without being computed by the ‘Core Application’ for debugging aims. The purpose is to have the raw data values of the system in the CODAC screen.
- From the computed variables issued by the PLC Core Application (“15” in *Figure 4*). It is important to note here that these variables here are in their engineering values, after the treatment in the ‘PLC Core Application’ where they can also be forced (just to recall that for the complete simulation we will use the ‘Hardware Access Layer’ set of skills).
- From the System Monitoring (“9” in *Figure 4*).

**Special case, inside States: Collaborative Data**

Collaborative Data are state variables transmitted between Plant System I&Cs. A Strong requirement of the PCDH is that no transversal wired link is allowed between Plant System I&Cs. This connection will be implemented with a “Software link” between 2 PLCs from 2 different Plant System I&Cs.

The ‘Collaborative Data’ will be States Variables with a specific Status of “Collaborative Data”. It is important in order to keep the consistency of the ‘Collaborative Data’ shared among Plant Systems to use always the same source of information (temperature, pressure…), so the same origin.

![Figure 10: Collaborative Data](image-url)
Events (currently only for SPSS, foreseen for OPC-UA)
The Event Variables are transmitting the status of the process.
The main difference with the states is that the ‘TimeStamp’ of the events is generated in the PLC, not in the PSH; their time stamp is as fast as the ‘PLC Cycle’ and it does not depend of the ‘transfer rate’ of the ‘CODAC Interface’. So, the ‘TimeStamp’ of the events recorded in the ‘DB 105’ will be different from each other.
The ‘events’ are ideal to capture a cascade of fast occurrences and be able to distinguish them temporarily.
The events are also transmitting the status of the Process (only binary data) and normally alarms:

- From the computed variables issued by the PLC Core Application (“15”in Figure 4).

**The PVs (Process variables) in EPICS are the direct representation of the states and the events in the PLC.**

**Configurations:**
The configuration variables are materializing the desires of the operator in how the system should work.
The configuration variables have two main objectives:

- The main use of this data is explained in (§4.2): Feed the PLC Core Application (“15” in Figure 4) with the data selected by the operator or the personal service:
  - Configuration of the ‘PLC Objects’ (i.e.: physical to engineering conversion, forcing values, inhibitions, filter configurations, PID Settings, Timer settings).
  - Assignment of set-points.
- Second use: Configure the Hardware Outputs/Inputs Interface (‘Hardware Access Layer’): Controlling the simulation mode (“8” in Figure 4).

**Simple Commands (* only for SPSS):**
As the configurations, the commands are materializing the desires of the operator in how the system should work.
The Command 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.

To know about the currently suitable ITER tools for the automatic generation of the ‘Exchanged data in the CODAC Interface’ refer to the chapter: [ ].
4.3.3 Configuration of the CODAC Interface

4.3.3.1 OPC-UA Configuration

OPC UA (Open Platform Communications Unified Architecture) is an M2M communication protocol adopted in 2009 that was specified by the OPC foundation. The OPC specification has been developed to create an interoperable, secure and reliable communication protocol.

4.3.3.1.1 OPC UA Server Configuration

S7-1500 PLC has an integrated ‘OPC-UA Server’, moreover from the firmware version 2.6 the functionality of ‘OPC-UA Client’ is also available (using TIA PORTAL v15.1). Regarding S7-400 PLC, with the introduction of the ‘CP 443-1 OPC UA’ the PLC is able to function as an ‘OPC UA server’ or as an ‘OPC UA client’.

Note: To know more information about it refer [RD 11]. This document shall be updated for CP443-1 OPC UA Server and CODAC OPC UA client configuration.

S7-1500:

The OPC UA server of the S7-1500 is disabled by default. Navigate to the “Properties” of the configured S7-1500 CPU in the TIA Portal.

Navigate to “OPC UA/Server/General” and select the “Activate OPC UA server”.

![Activate OPC UA Server](image.png)

*Figure 11 : Activate OPC UA Server.*
Navigate to “Runtime licenses/OPC UA” and select the required license in “Type of purchased license”.

![Image of OPC UA License](image1)

*Figure 12: OPC UA License.*

**S7-400:**

The OPC UA server of the S7-400 is enabled by default. Navigate to the “Properties” of the configured CP 443-1 in the TIA Portal.
Go to “Server settings”. Enter the smallest possible desired sampling and publishing interval of the OPC UA server in the “Minimum sampling interval” field and in the ”Minimum supported publishing interval” field.

![Image of Server Settings](image2)

*Figure 13: Server Settings.*
Add/Select new subnet for the CP as PON.

Figure 14: Add New Subnet.
4.3.3.2 SPSS Configuration

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.

![Diagram of ITER Controller Structure](image)

**Figure 15 : Main Cycle Loop Standard Structure**

![Diagram of Codac Interface](image)

**Figure 16 : Main Cycle Loop 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. The ‘CODAC Communication or Exchange Data’ Blocks will be generated automatically for the tool delivered for the CODAC Core System or coded manually using templates and coding rules. It is developed in [7.2].

This standard Structure is currently supporting the backbone for the Codac Interface. In [7.2] it is explained how to define a Codac Interface and how to generate the code automatically.

4.3.3.2.1 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.

a) Hardware Configuration

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 Catalogue for appropriate reference. The CPUs to be used are the ones included in this catalogue.

3. Using “NetPro” specify the IP addresses of the CPU and CP modules in the rack. See §0. 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 §.

b) Import the “Standard PLC Software Structure” from external source files.

1. Configure the PLC hardware as described in §0.

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.

c) 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 §0.
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.
4.4 **Hardware inputs/outputs interface (Hardware Access Layer)**

4.4.1 **General Description**

![Diagram of Hardware I/O Interface](image)

*Figure 20: Hardware I/O Interface (Hardware Access Layer) SW Architecture*
The Hardware I/O Interface has two basic objectives:

- **I/O HW Access:** Access the ‘field’ in both directions: gather inputs & deliver outputs.
- **Interface Switch:** Operate as a ‘switch’ between ‘real field data’ and ‘simulated data’.

The following chapters will give a deeper detail about every function.

In the exceptional case that the PLC needs to communicate with a ‘serial module’ using, for instance, a CP441, or with a ‘Modbus device’ implementing ‘Modbus protocol’; a function must be created to handle this kind of communication and it must be placed in the ‘Hardware Access Layer’. Refer to [Block numbering convention].

### 4.4.2 Inputs/Outputs HW Access

Also called ‘Inputs/Outputs Wrapper’, the purpose of this function is:

- Read the data from the ‘Input boards’ and store it in the ‘Shared DB for Wired Inputs’.
- Write the data from the ‘Shared DB for Wired Outputs’ and print it in the ‘Output Boards’.

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.

### 4.4.3 Interface Switch

The ‘Hardware Inputs/Outputs Interface’ (Hardware access Layer) allows the controller to make a complete simulation of the whole set of field values. This task is implemented through the Interface Switch (HW Access Switch).

Not to confuse with the forcing capabilities of every I/O Object presented in the ‘PLC Object Library in the chapter [PLC Core Application Body]:

The ‘Inputs HW Access switch’ allows you to choose the origin of the data to store in the ‘shared DB for inputs’: ['field signals’ or ‘simulator’].

The ‘Outputs HW Access switch’ allows you to choose the destination of the data coming from the ‘shared DB for outputs’: ['field signals’ or ‘simulator’].

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 4 – “8”). This command has to be secured in the sense that it cannot be operated during real operation. Another possibility, due to the importance of the switch, is to keep the control as an ‘internal PLC Variable’ with no connection to CODAC, being a bit to be reset every time PLC starts, and only control it from the ‘Engineering Console’.
4.4.4 Flow process of the Data in the Hardware Access Layer

The Hardware I/O Interface will allow the controller to access the field data. Observing the key Figure 20; the ‘Hardware Access Layer’ is divided, vertically, in two parts: HW Inputs interface and HW Outputs interface.

The same functions are present in both parts but they are processed in reverse order.

To explain how this interface works, here is the flow process of a wired input coming from a Plant System.

Process Flow of an input (from the field to the top of the ‘Hardware Access Layer’) following the structure shown in [Figure 20]:

- The signal is wired between the Plant System and the Input Board of the PLC.
- In a first Software function called the ‘Inputs HW Access’ (FC160), the signal is copied from a Hardware Input Address area to a DB (DB21). Example Input ‘I0.0’ is copied in ‘DB21.DBX0.0’.
  
  This signal now becomes a ‘Shared DB’ variable → ‘Shared DB for Wired Inputs’.
  
  *Note: PLC absolute addressing is used here for better understanding of this example, but Symbols must be used.

- The Shared DB variable is transmitted to an “Interface Switch” function (Inputs HW Access Switch - FC162). The job of the ‘Interface Switch’ is to choose between a wired signal (as it is the case of this example) or a signal coming from a Simulator.

  Then, the data is stored in another ‘Shared DB’ (DB25) → ‘Shared DB for Inputs’.

- Once the variable is in the ‘Shared DB for Inputs’ (DB 25), it is at the top of the ‘Hardware Access Layer’. From here, as it is shown in [Figure 4] it can be transmitted to the ‘PLC Core Application’ (Link 6) or directly to the ‘Codac Interface’ (Link 11).

The Process Flow of an output (from the top of the ‘Hardware Access Layer’ to the field) will go through the same functions but, logically, it will follow the opposite direction.

Finally, there is a sort of link between the ‘Hardware Access Layer’ and the ‘System Monitoring’ function; the data of the ‘Hardware Access Layer’ must be transmitted to the ‘System Monitoring’ in order to extract the ‘Health Status’ out of it. ‘10’ in [Figure 4].

To know more about the ‘Hardware Access Layer’, refer to [RD 16].

To know about the currently suitable ITER tools for the automatic generation of the ‘Hardware Access Layer’ refer to the chapter: [Summary of the ITER Development Tools/Libraries].
4.5 **PLC Interface**

This interface addresses the communication between PLCs of a same Plant System I&C (in case a functional interface is required). This ‘PLC Link’ is commonly known as ‘Plant Local Network’.

This type of connection could have place when the PLC to communicate with, for any reason, is not connected to PON (if the PLC is connected to PON there is no need to transfer data using a ‘Plant Local Network’ or ‘PLC Link’, better use CODAC).

It is important to highlight that this type of connection is advised against in the PCDH, and a strong reason must be exposed and approved for the ‘I&C Responsible’ of the system to go forward with it.

From conceptual point of view, we can consider 3 different cases:

- **Master/Slave link**
  Where the master PLC is sending commands (Boolean or numerical) to a slave PLC.  
  In a ‘Master/Slave’ architecture, the Master Coordinator will send orders to the Slaves.  
  A communication protocol must be defined for the communication of the devices.

- **Point to point link**
  Where 2 PLCs are exchanging states between each other. This state transmission can be Inputs/outputs of another PLC.

- **Multipoint communication:**
  Where a PLC is publishing states to a group of PLCs.

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

As a summary, in the exceptional situation of an independent ‘PLC Link’, ITER recommends the use of the ‘Point to point link’ using any of the following SIEMENS protocols to implement the communication: **S7 Communication** or **TCP Connection**. Refer to [RD 9] chapter [5.5.C] to know more about the hardware configurations of this type of link.
4.6 Fast Controller Interface

ITER recommends to use the network PON, via the PSH, as a standard communication between a PLC and a Fast Controller.

However, the direct communication between a PLC and a Fast Controller is allowed. We consider here three cases (similar at the mentioned above in the ‘PLC Interface’):

- A) Master/Slave Link, B) Slave/Master Link, C) Point to Point Link.

In the case this direct communication is necessary for the needs of the ‘Plant System’, ITER allows the use of the case C) ‘Point to point link’ using a SIEMENS protocol to implement the communication: **S7 Communication** or **TCP Connection**.

The proposal, regarding the hardware, is to use a dedicated Ethernet board in the Fast Controller, to build the private network. Regarding the software the suggestion is to put the ‘Transfer Function’ in the PLC Side, making it transparent for the ‘Fast Controller’, in order to better manage the function.

To highlight that the case of a ‘PLC’ Interfacing a ‘Fast Controller’ will be rare and particular. The design must be exposed and approved for the ‘I&C Responsibles’.
4.7 System Monitoring

System Monitoring of the PLC is an important task. The objective of the ‘System Monitoring’ is to report a set of data summarizing the ‘health status’ of the PLC, so, be able to assess the status of the controller and also the quality of the ‘Process Variable’ that will end up in the CODAC HMI.

The following figure shows the final goal of the system monitoring:

![Diagram](image_url)

*Figure 21: Flow of the Health Data*

There are two types of parameters to extract:

- **General Health Data**: General status of the PLC. These parameters must be presented in the CODAC HMI as the general status of the Plant System controller. So, they must be sent to the ‘CODAC Interface’ (‘9’ in [Figure 4]) in order to be part of the data sent to CODAC.

- **Board Health Data** (board = remote IO card): General status of the board and the status of every channel on it. These parameters are a necessary input to the ‘IO Objects’ of the ‘PLC Core Application’. Refer to [Object Inputs:]. So, they must be sent to the ‘PLC Core Application’ (‘14’ in [Figure 4]).
The following set of data is a summary of the most important information that could be extracted from a PLC to monitor its state; so it is in the interest of CODAC. The bold parameters are mandatory.

- **PLC Health Data**:
  - Operating Mode: RUN
  - Health Status of the CPU: OK/Error
    - Examples: SF for S7-300, IntF & ExtF for S7-400, Error for S7-1500.
  - 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 with CODAC
    - Type of communication configured: SPSS or OPC-UA
    - Max numbers of connexion available
      Number of connections used
  - Communication with FieldBus or FieldNet
    - Bus Status (Profinet/Profibus)
  - Alive Counter (it is given by default using SPSS and OPC-UA)

- **Board Health Data**
  - General Health Status of the board.
    - Examples: Int/Ext Failure, 24VC Error, Unplugged board.
  - Health Status of every channel on the board (if the board has this feature).
    - Examples: Current outside limits, common mode error, short-circuit

If the ‘I/O Board’ does not have the feature of reporting the status of every channel, the status of the entire board will be reported as status of the channel to the ‘IO Object’.

The report to CODAC of the bolded parameters is mandatory for every single PLC in the Plant System. In the non-advised case that the PLC is not visible to CODAC (because it is connected to a Plant local Network) the ‘health status’ of the PLC must still be reported to CODAC through the closer PLC visible connected to the Plant Local Network.

To know about the currently suitable ITER tools for the automatic generation of the ‘System Monitoring’ refer to the chapter: [Summary of the ITER Development Tools/Libraries].
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, FB, DB and UDT. A number is attributed to each of these blocks. For a good comprehension and standardization of the PLC Code we will number the blocks grouping them in areas. Every area will have its own range of numbers.

Currently, the blocks numbering is divided in **6 Categories or Areas**.

First **3 categories** are directly extracted from main software architecture diagram [Figure 4].

1. CODAC Interface (SPSS)
2. PLC Core Application
3. Hardware Access Layer

(*For the remaining main blocks of the diagram: PLC Interface, Fast Controller Interface and System Monitoring, the block numbering has still to be decided. So, the number of categories could increase if the remaining main blocks of the diagram are assigned to a numbering range).

Moreover for a better understanding and matching with the SIEMENS and CODAC standards, we will still add **3 categories more**:

4. Exchange Data Blocks
5. SIEMENS Internal
6. Basic Software Architecture

In the next paragraph we are going to explain the reason of being for every area or category:

1. **CODAC Interface (SPSS)**:
   SPSS will be used for the PLC to communicate with CODAC, using the ‘Exchange Data DBs’ as a vehicle.
   SPSS is set of blocks created for control and communication purposes, in the scope of the standardization. This set of blocks will have always the same numbering regardless of the ‘S7 project’.
   If OPC-UA is used as a ‘CODAC Interface’ method, these blocks will no longer be necessary, so it will not exist in the code.

   As explained in previous chapters ITER will provide the tools to ease this development job.
2. **PLC Core Application**
   This is the most important part of the code, core of the application.
   All Blocks in the Core Application are produced by the Plant System I&C Developer.
   So, the blocks used will have specific content to the ‘S7 Project’ and the numbering of
   the blocks will, again, specific to the application; always respecting the given range.
   As explained in previous chapters ITER will provide the tools and the libraries to ease
   this development job. Refer to Summary of the ITER Development Tools/Libraries[].

3. **Hardware Access Layer ( Hardware I/O Interface )**:  
   It is the software layer to access the Input/Output boards, so the data of the field.
   The Functions (FB and FC) must follow exactly the same structure regardless of the
   application (S7 Project); but, the size and the data inside the DBs will be, of course,
   different.

   As explained in previous chapters ITER will provide the tools and the libraries to ease
   this development job. Refer to [].

4. **Exchange Data Blocks**:
   These blocks are the vehicle used for the ‘CODAC Interface’ to exchange data with the
   ‘CODAC’.
   The numbering of these blocks will be the same regardless of the ‘S7 Project’ only the
   size, data and structure will be different.
   As explained in previous chapters ITER will provide the tools and the libraries to ease
   this development job. Refer to [].

5. **SIEMENS System**:
   Internal blocks belonging to the SIEMENS PLC.
   They are totally transparent to the user; still their range must be respected.

6. **Basic Software Architecture**:
   It is a set of blocks intended to set the most basic guidelines of the PLC software
   architecture. They are called in OB1.
   So, this few set of blocks is stating the common high-level functions of every ITER
   PLC, so, the numbering must be common too; but the contents will depend on the
   application.

   - CODAC Interface → FC 100 (Only for SPSS) → Refer [CODAC Interface]  
   - Inputs Processing → FC 101 → Refer to [Hardware Access Layer]  
   - Outputs Processing → FC 102 → Refer to [Hardware Access Layer]  
   - Process → FC 103 → Refer to [PLC Core Application]

   The following table makes a great summary of the ITER block numbering convention.
<table>
<thead>
<tr>
<th>Block</th>
<th>Area or Category</th>
<th>Numbering</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OB</td>
<td>SIEMENS System</td>
<td>Siemens Default</td>
<td>Blocks for user programs based on different operating system events</td>
</tr>
<tr>
<td>UDT</td>
<td>Hardware Access Layer</td>
<td>20..29</td>
<td>Access Layer to the hardware</td>
</tr>
<tr>
<td></td>
<td>CODAC Interface (SPSS)</td>
<td>100..299</td>
<td>SPSS Communication and Control</td>
</tr>
<tr>
<td></td>
<td>Exchange Data</td>
<td>300..309</td>
<td>Exchange Data vehicle for the communication with CODAC</td>
</tr>
<tr>
<td></td>
<td>PLC Core Application</td>
<td>310..65535</td>
<td>Core of the PLC Application</td>
</tr>
<tr>
<td>DB</td>
<td>Hardware Access Layer</td>
<td>20..29</td>
<td>Access Layer to the hardware</td>
</tr>
<tr>
<td></td>
<td>CODAC Interface (SPSS)</td>
<td>100..159</td>
<td>SPSS Communication and Control blocks</td>
</tr>
<tr>
<td></td>
<td>Shared Instance</td>
<td>1..19 // 30..49</td>
<td>SPSS Communication and Control blocks</td>
</tr>
<tr>
<td></td>
<td>Exchange Data</td>
<td>100..109</td>
<td>Exchange Data vehicle for the communication with CODAC</td>
</tr>
<tr>
<td></td>
<td>PLC Core Application</td>
<td>300..999</td>
<td>Core of the PLC Application</td>
</tr>
<tr>
<td></td>
<td>Shared Instance</td>
<td>110..299</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>300..999</td>
<td></td>
</tr>
<tr>
<td>FC</td>
<td>SIEMENS System</td>
<td>Siemens Default: 1..99</td>
<td>SIEMENS System Data</td>
</tr>
<tr>
<td></td>
<td>Basic SW Architecture</td>
<td>100..109</td>
<td>Bases of the PLC software architecture</td>
</tr>
<tr>
<td></td>
<td>CODAC Interface (SPSS)</td>
<td>110..159</td>
<td>SPSS Communication and Control blocks</td>
</tr>
<tr>
<td></td>
<td>Hardware Access Layer</td>
<td>160..179</td>
<td>Access Layer to the hardware</td>
</tr>
<tr>
<td></td>
<td>PLC Core Application</td>
<td>200..999</td>
<td>Core of the PLC Application</td>
</tr>
<tr>
<td>FB</td>
<td>SIEMENS System</td>
<td>Siemens Default: 1..99</td>
<td>SIEMENS System Data</td>
</tr>
<tr>
<td></td>
<td>CODAC Interface (SPSS)</td>
<td>100..159</td>
<td>SPSS Communication and Control</td>
</tr>
<tr>
<td></td>
<td>Hardware Access Layer</td>
<td>160..179</td>
<td>Access Layer to the hardware</td>
</tr>
<tr>
<td></td>
<td>PLC Core Application</td>
<td>200..999</td>
<td>Core of the PLC Application</td>
</tr>
<tr>
<td>SFC</td>
<td>Siemens Default</td>
<td>Integrated FB for CPU information</td>
<td></td>
</tr>
<tr>
<td>SFB</td>
<td>Siemens Default</td>
<td>Integrated FC for CPU information</td>
<td></td>
</tr>
</tbody>
</table>
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 have to be considered:

- CBS 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 ( rules will be stated in this document with the terminology NR).

[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 CBS, the CBS identifiers will always be in capital letters, and separated by "_" (underscore character).

Examples:
- UDT : ":_WFC_CIStates",
- FC : "WFC",

5.2.1 Core Application Blocks naming convention

The rules will be illustrated with example taken from the CBS of an imaginary and simplified Cooling Water System.
We will use the case of a Water Flow Process Control Function, WFC. We have simplified the structure of the ‘Process Control Function’, focusing only in the ‘Inputs’ (‘Configuration and Commands’ in the CODAC Interface) and the ‘Status’ (States in the CODAC Interface).

There will be one ["FB + iDB"] in the PLC for the Control of the WFC.
The function WFC is presented in the figure below.

![Figure 22: Control Block of a CBS Level 4 Function.](image-url)
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 the figure above we can see it represented by the ‘Function Block name’ and the ‘DB name’ between brackets.

The FB is named according to the ‘CBS Level’ name of the Function that it is implementing: “WFC”.

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

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

Every ‘Process Control Application Block’ must have a standard interface broken down in the necessary connexions.

In the simplified example, the ‘Process Control Application Block’ will have the Interface broken down in 3 connexions.

- “CIConf”, the Configuration variables sent by the CODAC.
- “CICmd”, the Simple Commands sent by the CODAC.
- “CIStates”, the State Variables sent to the CODAC

[NR 6] In Control Blocks, the connections of the “Control Interface” will be named following the guideline: “CIConf”, “CICmd”, “CIStates”, as example.

Each connexion is defined by a UDT. In the example, 3 UDTs are specific to the Function.

Each UDT will be composed of the variable of the interface it is defining. The name of the UDT is submitted to rules.

To illustrate this CWS example we present the following hypothetic variables:

For “CIConf”:

<table>
<thead>
<tr>
<th>UDT</th>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>_WFC_CIConf</td>
<td>Name</td>
<td>Type</td>
</tr>
<tr>
<td>CWFC</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>HSRQ</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>PT2SP</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>LFSP</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>HFSP</td>
<td>REAL</td>
<td></td>
</tr>
</tbody>
</table>
For “CICmd”:

<table>
<thead>
<tr>
<th>UDT</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;_WFC_CICmd&quot;</td>
<td>Name: Dummy</td>
</tr>
</tbody>
</table>

For “CIStates”:

<table>
<thead>
<tr>
<th>UDT</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;_WFC_CIStates&quot;</td>
<td>Name: STOPWFC</td>
</tr>
<tr>
<td></td>
<td>Name: LFST</td>
</tr>
<tr>
<td></td>
<td>Name: HFST</td>
</tr>
</tbody>
</table>

The examples above are applying the following rule:

[NR 7] UDTS related to the ‘Control Block Interface’ will be named according to the following Pattern:

“_”+<Control Block Name>+”_”+<Connection Type>.

Where <Connection Type> is the name defined in [NR 6].

For instance the above mentioned : ‘_WFC_CICmd’

Inside these UDTs structures, we propose some examples in the tables to name the variables. These names are following the naming rules defined in [RD 1].

[NR 8] In the UDTS defining interfaces involving signals, their names have to follow the rules 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 General functions and subfunctions naming convention

The general functions and subfunctions will not be directly related to Plant System Control Functions. They will address internal organization of the PLC, communication functions, system functions, etc.…

The general functions and subfunctions implement the structure shown in the key Figure 4.

[NR 9] General functions and subfunctions 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.


To order the structure inside a ‘General function’ or ‘subfunction’, many UDTS could be created. As said, in order to structure the information related to these Control Functions. To do that, a part of the name will be related to CBS, and the other part will be related to the scope of the UDT. Both parts will be separated by a “_”.

Examples: “_WFC_HwiConf”, “_WFC_HealthMon”.
5.3 **Variables naming convention**

5.3.1 **Input/Output Variables (signals)**

I/O Variables are the representation in the PLC of the ‘Field Signals’.
Even if inputs and outputs will almost not be used in their raw state, they have to be named.
The guideline to follow is [RD 1].
Still some rules must be adapted to the SIEMENS programming environment.
It is advised to use proposed tools [7.9] in order to create the ‘I/O Variables’ in the PLC:

[NR 10] Inputs and outputs (Field Signals) will be named according to their full PBS name [RD 1]. As the PBS name begin with a number, and it is not accepted by STEP 7, the user must choose previous character, for instance: “p” or “_” . 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;

5.3.2 **Internal Variables**

Internal Variables are enclosed privately inside the functions and don’t see the light out of it.
The user will have freedom to name these internal variables.
Still the creation of these kind variables must follow the boundaries defined in this document.
In the moment of the release of this document, CSD is preparing a ‘PLC Objects Library’ and a ‘Generator Tool’ [7.9] that will simplify the creation of functions and resolve the most of the naming for the internal variables of the functions in an appropriate way.

5.3.3 **Exchange DB variables (PVs)**

Exchanged DB variables are the representation in the PLC of the ‘CODAC Process Variables’.
The guideline to follow is [RD 1].
The standard way to generate the ‘Exchange DBs’ is through the SDD. In SDD, marking the option "full validation" on the level of I&C project will produce an exhaustive check of the naming convention.
5.4 Hardware Naming Convention

The objective of this chapter is to define a standard naming convention for the hardware components of the slow controllers, in other words, to set the bases on the hardware naming of the ‘PLC Project’.

The hardware naming in the ‘PLC Project’ must be consistent with the naming given in the electrical diagrams (the labelling will be glued on the hardware inside the electrical cabinets).

The final aim is to make the ‘electrical drawings’ and the ‘PLC Project’ match.

The complete hardware naming convention of the ‘PLC Project’ is detailed in the document [RD 17].

In this chapter, we will focus on the most important parameter to configure: the name of the hardware interface to the ‘General Networks’ (Plant Local Network or Central Network).

This interface is significant because it stipulates how the PLC will be seen for SIEMENS discovery protocols, as the ones used for the standard software tools like SIMATIC Manager and TIA Portal.

This interface must be unique among all the ITER PLCs.

Nomenclature for the General Network Interface:

Interface Name => I|Simatic Station Name-Rack Number-Type of Net Identifier

I: One Character indicating Interface.

Simatic Station Name: Standard Identifier of the Simatic Station (PBS nomenclature).

Rack Number: The Rack Number where this interface is placed. Normally: R0.

Type of Net Identifier: Identifier indicating the type network:

- PON: Plant Operational Network (central network: CODAC)
- PLN: Plant Local Network (private network)

Examples: I43CCWS-PLC-0001-R0-PON, I43CCWS-PLC-0001-R0-PLN

In the picture above, we can see a clear example of the General Network interfaces.

General Interfaces are framed in orange:

- One interface is connected to a Plant Local Network (internal to the PBS)
- The other interface is connected to PON (Plant Operational Network).
6 Programming Environment Configuration.

SIMATIC Manger and TIA Portal applications will be edited on several workstations. It is important that the Development environment is identically configured when editing the project.

6.1 SIMATIC Manager

6.1.1 General Configuration

SIMATIC Manager has three different configuration windows:

1. LAD/STL/FBD : To configure different types of programming block
2. HW Config : To configure hardware e.g. Insert new CPU,CP, I/Os etc.
3. NetPro : To Configure a Network e.g. Insert New Connection

All the default settings of SIMATIC Manager 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.

![SIMATIC Manager Language Setting](image)

Figure 24 : SIMATIC Manger 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.

   ![Customize Date and Time](image)

   *Figure 25: SIMATIC Manager Date and Time of Day Format.*

3. **LAD/FBD Layout**

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

   ![Customize LAD/FBD Layout](image)

   *Figure 26: LAD/FBD Layout.*
4. Symbol Editor Import.

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

![Figure 27: Symbol Editor Import.](image)

5. Address Priority:

Project settings have 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 ‘Address Priority’.

To set the ‘Address Priority’ properly: Click right on the “Block Folder” of the Program and check “For all accesses”. This setting intends to enforce Symbolic Programming.

![Figure 28: Address Priority.](image)

Note:
This setting is not applicable for TIA Portal.
6.1.2 **Hardware Configuration**

When creating a PLC SIMATIC Manager 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 ITER catalogue for I&C products - Slow controllers PLC PCDH Catalogue. See .

![HW Config Screen for a CPU Stations with 3 Remote IO Racks.](image)

**Figure 29 : 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 29–1)* the first parameters requested is the IP Address on the network.
   - Double-Click on the CPU in your hardware config *(Figure 29–1)* 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 30)*
Choose the Startup tab. Check “Cold Restart” for Startup after Power On. (Figure 31)

Click right on the X5-PN:IO field of the CPU (Figure 29–3) and select “Insert PROFINET IO System” option.

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 address Plan.

In Subnet, Create a New Network, with default parameters.
2. **CP Configuration**

   - Immediately when a CP is inserted in the Rack (*Figure 29–2*), the first parameter 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.
     - Double-Click on the CP in your hardware config (*Figure 29–2*) and choose the “Time-of-Day Synchronization” tab. Check “Activate NTP time-of-day synchronization” (*Figure 32*) and “Forward time of day to station” in NTP mode.
     - Introduce 2 NTP Servers Address. This information is managed by CODAC. (*Figure 32*)
     - Choose Time zone “(GMT) Dublin, Edinburgh, Lisbon, London”
     - Write 60 in “Update Interval”. (*Figure 32*)

![CP Time-of-Day Synchronization](image)

*Figure 32 : CP Time-of-Day Synchronization.*
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. 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 SIMATIC Manager will be applied. It is important to follow this rule because depending of the type of Boards (digital/analog, input/output) SIMATIC Manager is choosing specific Inputs/Outputs areas.

  - The boards in the Remote IO Racks has to be arranged in the following order (Figure 33):
    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 33. It is the default behaviour of SIMATIC Manager, but not if should reschedule boards manually afterwards.

![Figure 33: Remote IO Rack Board organization.](image-url)
4. **End of Topology Discovery**

Double-Click on the CPU/CP Ethernet port and choose to “Options” and select “End of topology discovery”. Do this setting for all the Ethernet port of each device.

*Figure 34: Ethernet Port Config End of Topology.*
6.2 TIA Portal

6.2.1 General Configuration

Similar to SIMATIC Manager, TIA Portal has following configuration windows:

1. Program Blocks: To configure different types of programming block
2. Device Configuration: To configure hardware e.g. Insert new CPU, CP, I/Os etc.
3. Network view: To configure a network e.g. Insert New Connection

All the default settings of TIA will be used except for the following:

1. Language

In TIA Portal menu, choose “Options/Settings/General/General/General Settings”. Choose English as User interface language and choose International for Mnemonic.

![Figure 35: TIA Language Setting.](image)

2. LAD/FBD Layout

In TIA Portal menu, choose “Options/Settings/PLC Programming/LAD/FBD/View”. Choose Wide for Layout.

![Figure 36: LAD/FBD Layout.](image)

Note: Date and Time
6.2.2 Hardware Configuration

1. CPU Configuration

Double Click on the CPU in Device view and choose the “General” section of Properties. In the “Comment” text field, introduce the PBS number of the PLC, the Cubicle PBS and Location.

Choose the “System and clock memory” section. Check “Enable the use of system memory byte” and “Enable the use of clock memory byte” and introduce “101” and “100” in the respective address field.

- Double-Click on the X1 Ethernet port of the CPU (Figure 29 –3) and in Subnet, Create a New Network, with default parameters.
- 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 address Plan.

2. **CP Configuration**
   - This interface is connected to the PON. CODAC is managing the IP Address plan of the PON and will provide the information.
   - Double-Click on the CP in Device view config and choose the “Time Synchronization”. Check “Enable time synchronization via NTP server”. (Figure 39).
   - Introduce 2 NTP Servers Address. This information is managed by CODAC. (Figure 39)
   - Write 60 in “Update Interval”. (Figure 39)

   ![Figure 39: CP Time Synchronization.](image)

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.
     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 TIA Portal will be applied.
4. **End of Topology Discovery**
   - Double-Click on the CPU/CP Ethernet port and choose to “Port options” and select “End of topology discovery”. Do this setting for all the Ethernet port of each device.

![PROFINET Port Config End of Topology](image)

*Figure 40: PROFINET Port Config End of Topology.*
7 Development of the PLC Application

7.1 Introduction

The target of this chapter is to provide to the user an easy, reliable, maintainable and ‘ITER standard’ way to produce the code inside the PLC.

![PLC Software Architecture Diagram](image)

Figure 41: General PLC Software Architecture.

So, this chapter will go through every block of the ‘General PLC Software Architecture’ explaining the proposed methodology for the construction of it.

7.2 CODAC Communication Blocks (Exchanged DBs)

7.2.1 Introduction

As said, the ‘communication protocol’ could be based either SPSS or OPC-UA:

- **SPSS**: From protocol point of view, the ‘CODAC interface built in SPSS’ 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.

- **OPC-UA**: From protocol point of view, the ‘CODAC Interface built in OPC-UA’ is based in a client/server methodology between the PLC and the PSH. Client is the PSH and the Server the PLC. Currently, this methodology is only available for S7-1500.
Only as a summary, as described in [4.2.2.4], this interface will support 5 types of information:

<table>
<thead>
<tr>
<th>States Variables</th>
<th>SPSS</th>
<th>OPC-UA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Configurations Variables</td>
<td>X</td>
<td>*(included in states)</td>
</tr>
<tr>
<td>Simple Commands</td>
<td>X</td>
<td>*(included in configurations)</td>
</tr>
<tr>
<td>Collaborative Data</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Each type of Data will be transmitted in one DB of Maximum 32 kBytes.

For SPSS: States and Configuration will use the same TCP connexion, Simple Commands and Collaborative data will have their respective TCP Connection.

In order to compare the performance between OPC-UA and SPSS please refer to [RD 18].

**7.2.2 Construction**

Each DB will be built with UDTs. Each UDT will represent the Interface Type (States, Configuration, Events or Commands) of one Control Function identified in the CBS of its Plant System.

The diagram in the next page illustrates with an example the construction of the DBs.

It represents the Block organisation and dependencies from the Control Function example: “CWS-DHLT-WFC” in the Cooling Water System.

The blocks in pale Blue belong to the Plant System Functions.

The blocks in orange are generated. They belong to the ‘Exchange DBs’ (CODAC Communication blocks).

The blocks in white belong to the SPSS (so it does not exist in OPC-UA).
Figure 42: UDTs and DBs organisation and dependencies for “CWS-DHLT-WFC”.
In the case of SPSS the **connections** are also generated:

**Figure 43: UDTs and DBs organisation and dependencies for TCP connexion**

### 7.2.2.1 CODAC Interface good practice

The CODAC interface is built upon a concept described in [RD 3] and summarized in this chapter. 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 must not 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 the figure above. 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 44: Closed Loop Control Example**
7.2.3 **Generation**

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 Control System, as well as, the Interface with the I&C Controller, in our case, the PLC, and then, generate automatically the STEP 7 and PSH files required to build this interface.

Practically, The SDD Toolkit generates as set of plant system specific files: Symbol Table (*.sdf) file, STL (*.awl) file and EXCEL file (*.xlsx).

These ‘source files’ after compilation in the ‘SIEMENS Software’ generate the ‘Exchange DBs’ : “CodacStates”, “CodacEvents” (only for SPSS), “CodacConfiguration” and “CodacCommands” (only for SPSS) and initialize the “CodacChannels” (only for SPSS).

The following screenshot shows the example of ‘SDD Editor’ generated files for the example of the Cooling Water System. The use of SDD is described in [RD 7].

![SDD generated files example](image)

*Figure 45: Add plant system specific STL file generated by SDD Editor.*

To generate the ‘Exchange DBs’, it is needed the ‘Symbol Table (*.sdf) file and the AWL File. The process to generate is detailed in the [RD 12], and explained briefly 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.
7.3 Hardware Inputs/Outputs interface (Hardware Access Layer)

7.3.1 Introduction

As said, the Hardware I/O Interface has two basic objectives:

- **I/O HW Access**: Access the ‘field’ in both directions: gather inputs & deliver outputs.
- **Interface Switch**: Operate as a ‘switch’ between ‘real field data’ and ‘simulated data’.

7.3.2 Construction

The construction of the Hardware Access Layer is fully explained in [RD 16] and depicted in the following figure.

![Software Architecture of the Hardware Access Layer](image)

*Figure 46: Software Architecture of the Hardware Access Layer.*
7.3.3 **Generation**

**Objective:** Generate the ‘Hardware Access Layer’ inside the ‘PLC Application’ through an AWL File.

Currently:

**HAL Tool** has been developed by ITER Control System Division in order to generate the ‘Hardware Access layer’ and the ‘System Monitoring’ targeting all ‘ITER Plant Systems’. It is fast and simple to use. It is targeting operational system and also prototypes. To know more about this tool refers to [RD 15].

**ICL Generator** has been developed by ITER Cryogenics Division in order to generate the ‘PLC Core Application’ and the ‘Hardware Access Layer’. To know more about the ‘ICL Generator’ refer to [RD 11].

**UCL Generator** is being developed by ITER Control System Division in order to generate the ‘PLC Core Application’, the ‘Hardware Access Layer’ and the ‘System Monitoring.’

7.4 **PLC interface**

The ‘PLC Interface’ implements the communication between two PLCs inside the same Plant System. The ITER Control System Division names this type of link as PLN: Plant Local Network.

It is strongly recommended to use a standard SIEMENS communication like ‘S7 Connection’ in order to perform this link. To know more about this kind of ‘Hardware Architecture’ between PLCs refer to [RD 9]: chapter 5, Plant System Local Network.

7.5 **Fast Controllers interface**

The ‘Fast Controllers Interface’ implements the communication between one PLC and one Fast Controller inside the same Plant System.

This link is rare.

It is strongly recommended to use standard TCP/IP wired connection and common protocols like MODBUS TCP/IP in order reach a high level of standardization.

7.6 **Simulator interface**

The ‘Hardware Access Layer’ interface is allowing an easy interface to a simulator, as it is shown in [7.3.2].

The simulator device will have only to interact with the two DBs provided; one for the inputs DB23 and the other for the outputs DB24.

It is strongly recommended to place all the communication functions in the external simulator device, in order to avoid any kind of “simulator oriented code” inside the Control System PLC.
7.7 Health System Monitoring

7.7.1 Introduction

The automatic generation of System Health Monitoring is foreseen, and currently, being developed. HAL Tool has been developed by ITER Control System Division in order to generate the ‘Hardware Access layer’ and the ‘System Monitoring’ targeting all ‘ITER Plant Systems’. The information provided, at the time of the approval of this document, is only the status of the CPU. [RD 15]. UCL Generator is under development, and it is foreseen to provide information about the CPU and the associated I/O Boards of the PLC Hardware Configuration.

7.7.2 Construction

Avoid CPU going to stop
In order to avoid unrehearsed crashes of the PLC, the following OBs could be loaded in the PLC:

The three following ‘Organization Blocks’ must trigger a DP Diagnosis functions to feed the Health Monitoring code:

- OB82: I/O Point Fault
- OB85: Organization Block Not loaded Fault.
- OB86: IO-Device failure interrupt.

The two following ‘Organization Blocks’ are not allowed in the final operational code. Still they can be used in phase of test in order to start-up the system, for instance, when some hardware is missing.

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

The content of these OBs, currently, must be programmed for the developer itself.

Report the Data
In order to report the data of the Health monitoring a periodic OB of medium priority should be used as a container of the function extracting the ‘Health Monitoring’ data. For instance; OB30 has been used successfully.

7.7.3 Generation

As said.
Currently, HAL Tool for Hardware Access Layer Generation is generating a basic Health Monitoring System status. Simple but useful. [RD 15].
7.8 **PLC Core Application**

7.8.1 **Introduction**

In this chapter of construction, we are going to explain the development cycle and deliverables of the most important part of the code, the Core Application. This part of the PLC Core is dedicated for every particular Plant System Slow Controller.

7.8.2 **Construction**

7.8.2.1 **Development Cycle and deliverables**

![Figure 47: Core Application Development Life Cycle](image)

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 the documentation to be delivered following the PCDH [RD 10].

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.

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”.

The detailed design of the Plant System I&C will be completed at the time the software development will begin; so, a full set of documents will be available. At the end of the detailed design, this documentation will be broken down in 9 deliverables defined in the PCDH [RD 10]. These deliverables must 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.

7.8.2.1.1 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 [RD 10].

7.8.2.1.2 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 [RD 10]: [D31].
7.8.2.1.3 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.

7.8.2.1.4 Simulated Validation Testing (FAT First phase)

Included in the FAT. The main objective of the Simulator is to be used against the Control System during FAT.

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 completeness even in the case the systems comes from different procurements.
- If Simulation is available, corrective and adaptive 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, in principle, low level functionalities. The development of complex algorithms with multiple couplings is time consuming and it will be up to the Plant System to decide the worthiness of this action. Meanwhile, it is strongly recommended. The strategy retained according simulation has to be described in PCDH [RD 10] deliverables.

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.

In case of a system already deployed, 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. After the simulation, 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. ITER IO could request the Simulator as a deliverable itself after the accomplishment of the FAT.

For this phase, a complete set of testing documentation has to be provided to ITER IO before the beginning of the validation. Refer to [RD 19].
7.8.2.1.5 Integrated Validation Testing (FAT second phase)

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

7.8.2.1.6 Site Acceptance Test

The test will be carried after the installation of the Control System in the final facilities. The Control System will be tested against the real Plant System using a rigorous and systematic battery of tests. For this phase, a complete set of testing documentation has to be provided to ITER IO before the beginning of the validation. Refer to [RD 19].

7.8.2.2 Languages

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

<table>
<thead>
<tr>
<th>IEC61131-3 Language Name</th>
<th>Siemens Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Ladder Diagram (LD)</td>
<td>Ladder Logic (LAD)</td>
</tr>
<tr>
<td>• Function Block Diagram (FBD)</td>
<td>Function Block Diagram (FBD)</td>
</tr>
<tr>
<td>• Structured Text (ST)</td>
<td>Structured Control Language (SCL)</td>
</tr>
<tr>
<td>• Instruction list (IL)</td>
<td>Statement List (STL)</td>
</tr>
<tr>
<td>• Sequential function Chart (SFC)</td>
<td>Sequential Control System (SCS)</td>
</tr>
</tbody>
</table>

About CFC:
Siemens CFC (Flow Charts) is not recommended for the conventional controllers. First it is not defined in IEC 61131-3, second it has the major drawback; CFC cannot be mixed with other languages.
Still, CFC will be used for redundant architectures, deployed in Interlock and Safety. This topic is not covered in this document.

About HiGraph:
Siemens HiGraph (Petri Nets) is not recommended, again, 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 certain type of problems. The following rules have 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 does not make any difference as we can switch the representation from LAD to FBD in STEP7. See later in the paragraph.
- GRAFCET will be used to implement sequences. But outputs will not be written directly in Grafcets.
- SCL will be used to implement complex numerical algorithms, loop algorithms, complex state machines or ‘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 [Coding Rules].

**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 a program is created 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.

S7 Graph can be programmed using a specific graphical editor. 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.
7.8.3 Generation

**Objective:** Generate the ‘PLC Core Application’ inside the ‘PLC Application’ through one or several AWL Files.

Currently:

**ICL Generator** has been developed by ITER Cryogenics Division in order to generate the ‘PLC Core Application’ and the ‘Hardware Access Layer’. To know more about the ‘ICL Generator’ refer to [RD 11].

**UCL Generator** is being developed by ITER Control System Division in order to generate the ‘PLC Core Application’, the ‘Hardware Access Layer’ and the ‘System Monitoring’.
7.9 **Summary of the ITER Development Tools/Libraries**

For the easiness, practicality and speed of the generation of the ‘PLC Application’, ITER Organization owns a set of tools and libraries targeting this development, in an automatized way.

These tools and libraries are open for the use to any contractor.

These tools and libraries are commonly referred as ‘PLC Objects Libraries’ and ‘PLC Code Generators’.

The tools and libraries mentioned are in continuous evolution, the reflected status is the one at the time of the release of this document.

Let’s assess the current situation.

**General description of every tool:**

1. **SDD. Self-description data.**
   
   SDD is a tool provided inside the package of the CODAC Core System (CCS) developed by the Control System Division targeting all ‘ITER Plant System’.
   
   In SDD the user defines the ‘Static configuration data’ which describe the plant system characteristics in a unified way in order to facilitate configuration of the central I&C systems’ software for operation with the given plant system. SDD cover configuration of controllers, PSH and CODAC services

   **Objective:** Generate the ‘Exchange-data DBs’ inside the PLC Application through an AWL File. Exchange-data DBs are: States (DB100), Configurations (DB101), Commands (DB102), Events (DB105). Refer to [RD 14].

   **The CODAC Interface is using the ‘exchange-DBs’ to exchange data.**

   Not to confuse the ‘Exchange-data DBs’ with ‘CODAC Interface itself’ (implemented by SPSS or OPC-UA). The CODAC Interface is independent of SDD.

2. **Industrial Control Library (ICL) + ICL Generator.**

   A ‘PLC Objects Library’ has been developed by the ‘Cryogenic System Section’ targeting the ‘ITER Cryoplant’.

   The library comes associated with a ‘Code Generator’ for the PLC.

   **Objective:** Generate the ‘PLC Core Application’ inside the ‘PLC Application through an AWL File’. ‘ICL Generator’ moreover is generating EPICS Files and HMI (OPI) Files related to the complete CODAC Project. Refer to [RD 11].

3. **HAL (Hardware Access Layer) Tool.**

   HAL Tool has been developed by Control System Division targeting the lowest part of the PLC Application targeting all ‘ITER Plant Systems’.

   It is extremely fast and simple to use. It is targeting operational system and also prototypes.

   **Objective:** Generate the ‘Hardware Access Layer’ and the ‘Health Monitoring System’ inside the ‘PLC Application’ through an AWL File. Refer to [RD 15].
4. **ITER Standard Control Library (UCL) + UCL Generator.**

An ‘ITER Standard PLC Objects Library’ is under development by the ‘Control System division’ targeting all ‘ITER Plant Systems’.

The library will come associated with a ‘Code Generator’ for the PLC.

**Objective:** Generate the ‘PLC Core Application’ inside the ‘PLC Application’ through an AWL File. The ‘Code Generator’ is intended to generate the ‘Hardware Access Layer’ and the ‘System Monitoring’ as well.

**Summary Table:**

<table>
<thead>
<tr>
<th></th>
<th>Tool</th>
<th>Status</th>
<th>Provider</th>
<th>Target</th>
<th>Code Generated</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SDD</td>
<td>Operational</td>
<td>IO-CSD</td>
<td>Any ITER Plant System</td>
<td>CODAC Interface</td>
<td>[RD 14]</td>
</tr>
<tr>
<td>2</td>
<td>ICL + ICL Generator</td>
<td>Operational</td>
<td>IO-CSE</td>
<td>Cryoplant &amp; Cryodistribution</td>
<td>PLC Core App + HAL</td>
<td>[RD 11]</td>
</tr>
<tr>
<td>3</td>
<td>HAL</td>
<td>Testing</td>
<td>IO-CSD</td>
<td>Any ITER Plant System</td>
<td>HAL</td>
<td>[RD 15]</td>
</tr>
<tr>
<td>4</td>
<td>UCL + UCL Generator</td>
<td>Development</td>
<td>IO-CSD</td>
<td>Any ITER Plant System</td>
<td>PLC Core App + System Monitoring + HAL</td>
<td>TBD</td>
</tr>
</tbody>
</table>

Scope of the tools (based on the main software architecture diagram):

**Figure 48: Scope of the ‘Code Generator Tools’**
7.10 Alarms Management

Standard alarms are managed inside the CODAC. No Specific programming is required regarding alarms in the PLC.

In the CODAC side the PV (STATE or EVENT) will be declared as an alarm.

In the PLC side the variable will be declared as STATE, in case of Analog data, and preferably as an EVENT in case of digital data (still could be declared as STATE in a slow system where no cascade of events is foreseen)

If a combination of information is required in order to generate an alarm in the CODAC, it could be handled of two different ways:

- In the CODAC side: Creating a dedicated CALC PV and programming in the PSH the desired behaviour taking in account the status of several STATES and/or EVENTS reported for the PLC.

- In the PLC side: Creating a dedicated STATE variable and programming in the PLC the desired behaviour taking in account the status of the internal PLC variables generated by the ‘PLC Core Application’.

The standard alarm parameters related to Alarms are: level of the analog to trigger, hysteresis, delay time to trigger.

7.11 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 at 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 in case of 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 Global |
Shared DB directly in the Control Block is prohibited. It is not always technically possible so exceptions must be clearly stated in PCDH [D31].

This rule will:

- Make the code portable
- Shorten the length of variables inside blocks, as local variables names do not have to include the name of the block.
- Make easier the Unit Testing of blocks.

**[CR 5]** Use only Shared DB variables, instead of ‘PLC Memory Addresses’ (memento). 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 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 to 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 STL. 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, for instance: “WFC”. PL1-CY”, …

**[CR 13]** No explicit use of Address Register instructions in STL.
8 Software Configuration Management

‘Software Control Version’ per se is not required to be able to develop and run software; however, it has become a ‘de facto’ standard procedure for any software project bigger than a few lines of code. It allows:

- To record and retrieve consistent versions of software.
- To be able to answer on who did when which exactly change and for what reason.
- To clearly identify production software, stable snapshots, deliverables.
- To develop or maintain multiple versions of software at the same time.
- To have controlled source code exchange.
- To enable various QA procedures, including automated ones.

In the industrial control system environment with a long life span it brings important benefits:

- Enable clean handover between I&C production, integration and maintenance contractors and teams.
- Allow reinstallation of the exact I&C system configuration in the case of controller breakdown.
- Allow traceability of faults or incidents back to software, where applicable.

ITER Control System Division is proposing a methodology; please refer to [RD 20].
9 Annexes

9.1 Already Reserved Blocks for CODAC

<table>
<thead>
<tr>
<th>Block Symbol</th>
<th>Block Number</th>
<th>Bloc Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_CodacStatesHeader</td>
<td>UDT 100</td>
<td>UDT 100</td>
<td>CodacStatesHeader</td>
</tr>
<tr>
<td>_CodacStatesFooter</td>
<td>UDT 101</td>
<td>UDT 101</td>
<td>CodacStatesFooter</td>
</tr>
<tr>
<td>_CodacConnection</td>
<td>UDT 110</td>
<td>UDT 110</td>
<td>Codac Connection</td>
</tr>
<tr>
<td>_CodacChannel</td>
<td>UDT 111</td>
<td>UDT 111</td>
<td>Codac Interface States Communications</td>
</tr>
<tr>
<td>CodacStates</td>
<td>DB 100</td>
<td>DB 100</td>
<td>Codac Interface Simple Command Communications</td>
</tr>
<tr>
<td>CodacConfiguration</td>
<td>DB 101</td>
<td>DB 101</td>
<td>Codac Interface Parameters of Codac Interface</td>
</tr>
<tr>
<td>CodacCommands</td>
<td>DB 102</td>
<td>DB 102</td>
<td>Communication Parameters of Codac Interface</td>
</tr>
<tr>
<td>CodacConnections</td>
<td>DB 103</td>
<td>DB 103</td>
<td>Plc Hardware Interface Wired Inputs</td>
</tr>
<tr>
<td>PlcHwiWiredInputs</td>
<td>DB 21</td>
<td>DB 21</td>
<td>Plc Hardware Interface Wired Outputs</td>
</tr>
<tr>
<td>PlcHwiWireOutputs</td>
<td>DB 22</td>
<td>DB 22</td>
<td>Plc Hardware Interface Simulated Outputs</td>
</tr>
<tr>
<td>PlcHwiSimInputs</td>
<td>DB 23</td>
<td>DB 23</td>
<td>Plc Hardware Interface Simulated Outputs</td>
</tr>
<tr>
<td>PlcHwiSimOutputs</td>
<td>DB 24</td>
<td>DB 24</td>
<td>Plc Hardware Interface Simulated Outputs</td>
</tr>
<tr>
<td>PlcHwiInputs</td>
<td>DB 25</td>
<td>DB 25</td>
<td>Plc Hardware Interface Inputs</td>
</tr>
<tr>
<td>PlcHwiOutputs</td>
<td>DB 26</td>
<td>DB 26</td>
<td>Plc Hardware Interface Outputs</td>
</tr>
<tr>
<td>iCodacChannel1</td>
<td>DB 50 FB 110</td>
<td>FB 110</td>
<td>Codac Interface States and Configuration Channel (1)</td>
</tr>
<tr>
<td>iCodacChannel2</td>
<td>DB 51 FB 110</td>
<td>FB 110</td>
<td>Codac Interface Simple Commands Channel (2)</td>
</tr>
<tr>
<td>CodacTimestamp</td>
<td>FB 105</td>
<td>FB 105</td>
<td>Codac TimeStamp assigned to the</td>
</tr>
<tr>
<td>CodacChannel</td>
<td>FB 110</td>
<td>FB 110</td>
<td>Codac Interface Open Communication Control</td>
</tr>
<tr>
<td>CodacInterface</td>
<td>FC 100</td>
<td>FC 100</td>
<td>Codac Interface Communications</td>
</tr>
<tr>
<td>InputsProcessing</td>
<td>FC 101</td>
<td>FC 101</td>
<td>Hardware interface Inputs Processing Block</td>
</tr>
<tr>
<td>OutputsProcessing</td>
<td>FC 102</td>
<td>FC 102</td>
<td>Hardware interface Outputs Processing Block</td>
</tr>
<tr>
<td>Process</td>
<td>FC 103</td>
<td>FC 103</td>
<td>Process Function Blocks</td>
</tr>
<tr>
<td>CodacSetTcpEndPointx</td>
<td>FC 111</td>
<td>FC 111</td>
<td>Codac Interface TCP Endpoint Setting</td>
</tr>
<tr>
<td>CodacConnectionInit</td>
<td>FC 115</td>
<td>FC 115</td>
<td></td>
</tr>
<tr>
<td>ResetDB</td>
<td>FC 116</td>
<td>FC 116</td>
<td></td>
</tr>
</tbody>
</table>
### 9.2 Already Reserved Global Variables for CODAC

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

### 9.3 Brief Example about Cooling Water System

ITER Control System Division is providing an ‘End-to-End Project’ example, [RD 8]. This example is targeting new ITER Control System developers, this document showing the methodology, tools and ‘support documentation’ provided by ITER IO in order to develop a consistent Control System.

As it is fully explained in [RD 8]; the case study example is all about ‘Water Level Control’ of a tank. The function of the WLC is to control the level of water in tank using two ‘On Off valves’ (Supply and Exhaust on/off valve) and two level switches (High level switch and Low level switch) and a level sensor.

![Diagram of Water Level Control System](Figure 49: CWS Case of Study Example)
CBS Level Structure of the example:

<table>
<thead>
<tr>
<th>CBS L1</th>
<th>CBS L2</th>
<th>CBS L3</th>
<th>CBS L4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWS</td>
<td>PHTS</td>
<td>DHLT</td>
<td>WFC</td>
</tr>
</tbody>
</table>

“CWS” identify the Cooling Water Supply Function.
“PHTS” identify the Primary Heat Transfer System.
“DHLT” identify the Divertor Loop Heat Transfer.
“WLC” identify the Water Level Control.

The example proposed is implemented using the Cryo Objects Library [RD 11].

The ‘Hardware Access Layer’ of the example:
The Hardware Access Layer is the interface with the field.

Inputs:

<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
<th>Data Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0.0</td>
<td>VC1_FVY1</td>
<td>BOOL</td>
<td>Open Status of Inlet Valve</td>
</tr>
<tr>
<td>I0.1</td>
<td>VC1_FVY2</td>
<td>BOOL</td>
<td>Close Status of Inlet Valve</td>
</tr>
<tr>
<td>I0.2</td>
<td>VC2_FVY1</td>
<td>BOOL</td>
<td>Open Status of the Exhaust Valve</td>
</tr>
<tr>
<td>I0.3</td>
<td>VC2_FVY2</td>
<td>BOOL</td>
<td>Close Status of the Exhaust Valve</td>
</tr>
<tr>
<td>I0.4</td>
<td>FVY1_ICC</td>
<td>BOOL</td>
<td>High Level Switch Status</td>
</tr>
<tr>
<td>I0.5</td>
<td>FVY2_ICC</td>
<td>BOOL</td>
<td>Low Level Switch Status</td>
</tr>
</tbody>
</table>

Outputs:

<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
<th>Data Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q0.0</td>
<td>VC1_FVZ</td>
<td>BOOL</td>
<td>Command of inlet valve</td>
</tr>
<tr>
<td>Q0.1</td>
<td>VC2_FVZ</td>
<td>BOOL</td>
<td>Command of Exhaust valve</td>
</tr>
<tr>
<td>Q0.2</td>
<td>WLC_AUTOMAN</td>
<td>BOOL</td>
<td>Auto manual status</td>
</tr>
<tr>
<td>Q0.3</td>
<td>WLC_CSTA</td>
<td>BOOL</td>
<td>Connection Status</td>
</tr>
</tbody>
</table>
The ‘CODAC Interface’ of the example:

The Water Flow Control function has the following ‘Communications DBs with the CODAC’; forming what is called ‘CODAC Interface’:

As State Variables:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sDI_HS</td>
<td>Status of high level switch (DI object)</td>
</tr>
<tr>
<td>sDI_LS</td>
<td>Status of Low level switch (DI object)</td>
</tr>
<tr>
<td>sAI_ML</td>
<td>Status of level transmitter (AI object)</td>
</tr>
<tr>
<td>sVLVONOF_Inlet</td>
<td>Status of inlet valve (VLV_ON_OF object)</td>
</tr>
<tr>
<td>sVLVONOF_Exhaust</td>
<td>Status of Exhaust valve (VLV_ON_OF object)</td>
</tr>
<tr>
<td>sProcess</td>
<td>Status of process logic</td>
</tr>
</tbody>
</table>

As commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmDI_HS</td>
<td>Simple command for high level switch (DI object)</td>
</tr>
<tr>
<td>cmDI_LS</td>
<td>Simple command for Low level switch (DI object)</td>
</tr>
<tr>
<td>cmAI_ML</td>
<td>Simple command for level transmitter (AI object)</td>
</tr>
<tr>
<td>cmVLVONOF_Inlet</td>
<td>Simple command for inlet valve (VLV_ON_OF object)</td>
</tr>
<tr>
<td>cmVLVONOF_Exhaust</td>
<td>Simple command for Exhaust valve (VLV_ON_OF object)</td>
</tr>
<tr>
<td>cmProcess</td>
<td>Simple command for process logic</td>
</tr>
</tbody>
</table>

As configurations:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfg_DI_HS</td>
<td>Configuration for high level switch (DI object)</td>
</tr>
<tr>
<td>cfg_DI_LS</td>
<td>Configuration for Low level switch (DI object)</td>
</tr>
<tr>
<td>cfg_AI_ML</td>
<td>Configuration for level transmitter (AI object)</td>
</tr>
<tr>
<td>cfgVLVONOF_Inlet</td>
<td>Configuration for inlet valve (VLV_ON_OF object)</td>
</tr>
<tr>
<td>cfg_VLVONOF_Exhaust</td>
<td>Configuration for Exhaust valve (VLV_ON_OF object)</td>
</tr>
</tbody>
</table>

Please refer to [RD 8] to completely understand and develop this example.