



Universidad de Oviedo

Memoria del Trabajo Fin de Máster realizado por

Manuel Vázquez Muñiz

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

**Upgrade of the UNICOS Time Stamp Push Protocol
(TSPP) broker to include ultra-fast events**

Junio 2017

INDEX

INDEX	1
FIGURE INDEX	3
1. INTRODUCTION	4
1.1. Project identification.....	4
1.2. Project overview	4
1.3. Document overview.....	4
2. OVERVIEW	5
2.1. CERN.....	5
2.2. PLC.....	6
2.3. Fast Interlocks.....	7
2.4. The project.....	8
3. DESCRIPTION OF THE UNICOS FRAMEWORK	9
3.1. UAB	9
3.2. UNICOS PLC code	10
3.3. UNICOS code structure in Siemens PLC.....	11
3.3.1. TSPP.....	13
3.3.2. Compilation of SCL files in Step 7.....	13
3.4. Templates used by the UNICOS framework	14
3.4.1. DeviceTypes	16
3.4.2. PlcParams.....	16
3.4.3. S7InstanceGenerator folder.....	16
3.4.4. S7LogicGenerator folder	16
3.4.5. SemanticCheckRules folder.....	17
3.4.6. SharedTemplates	17
3.4.7. WINCCOAIInstanceGenerator folder	17
4. OBJECTIVES	18
5. ANALYSIS	20
5.1. UNICOS project	20
5.2. Fast response possibilities in Siemens PLC	20
5.2.1. Interrupts regarding PLC	21
5.2.2. Interrupts regarding the UNICOS Framework.....	22
5.3. TSPP.....	23
5.4. Test requirements.....	23
5.5. Templates.....	24
6. CONCLUSIONS	25
7. FUTURE WORKS	27
8. ACRONYMS	28

9. DOCUMENTS OF THE PROJECT	29
10. BIBLIOGRAPHY	30

FIGURE INDEX

Fig. 1. Machines and experimental areas at CERN	5
Fig. 2. Siemens PLC CPU 317-2 PN/DP [4]	6
Fig. 3. Longest response time for an input signal [5]	7
Fig. 4. UNICOS application builder module selection	10
Fig. 5. SCL code visualized inside the Step 7 software	11
Fig. 6. UNICOS functions and function blocks inside OB 1 in a Siemens application [8]	12
Fig. 7. Compilation order in Step 7 [8]	13
Fig. 8. Folder Tree inside a UNICOS Step7 project	15
Fig. 9. Process image transfer time for CPU 31x family [5]	21
Fig. 10. Time for a byte load instruction using process image or peripheral addressing in CPU 31x [20]	22
Fig. 11. Hardware configuration in Step 7	23

1. Introduction

1.1. Project identification

- Title: Upgrade of the UNICOS Time Stamp Push Protocol (TSPP) broker to include ultra-fast events
- Author: Manuel Vázquez Muñiz
- Advisor: Víctor Manuel González Suárez
- Co-advisor: Jerónimo Ortolá Vidal
- Date: June 2017
- Organization: CERN

1.2. Project overview

The current project objective is to solve the issue of the fast interlocks (or ultra-fast events) by improving the Time Stamp Push Protocol (TSPP) used to communicate the control and supervision layers. This protocol is used in the framework UNICOS, and this framework should also be modified as to support this new feature.

With this new feature, the organization will be able to fulfil the requirements of the internal clients who need this capability as to have a proper use of their equipment.

1.3. Document overview

This document gives a general description of the project and explains the need of this feature, the objectives that the organization wants to achieve and the problematics for its realization. It also describes the conditions for the testing of the project, and explains the conclusions obtained from the realization of the project, as well as some of the possible future works.

2. Overview

2.1. CERN

CERN is a European Organization dedicated to the research of nuclear particles [1]. The purpose of the organization is to deliver the knowledge acquired from different experiments consisting mostly in the analysis of the collision of high speed beams.

The current accelerators (and some of the experiments) existent at the CERN organization are the ones shown in Fig. 1.

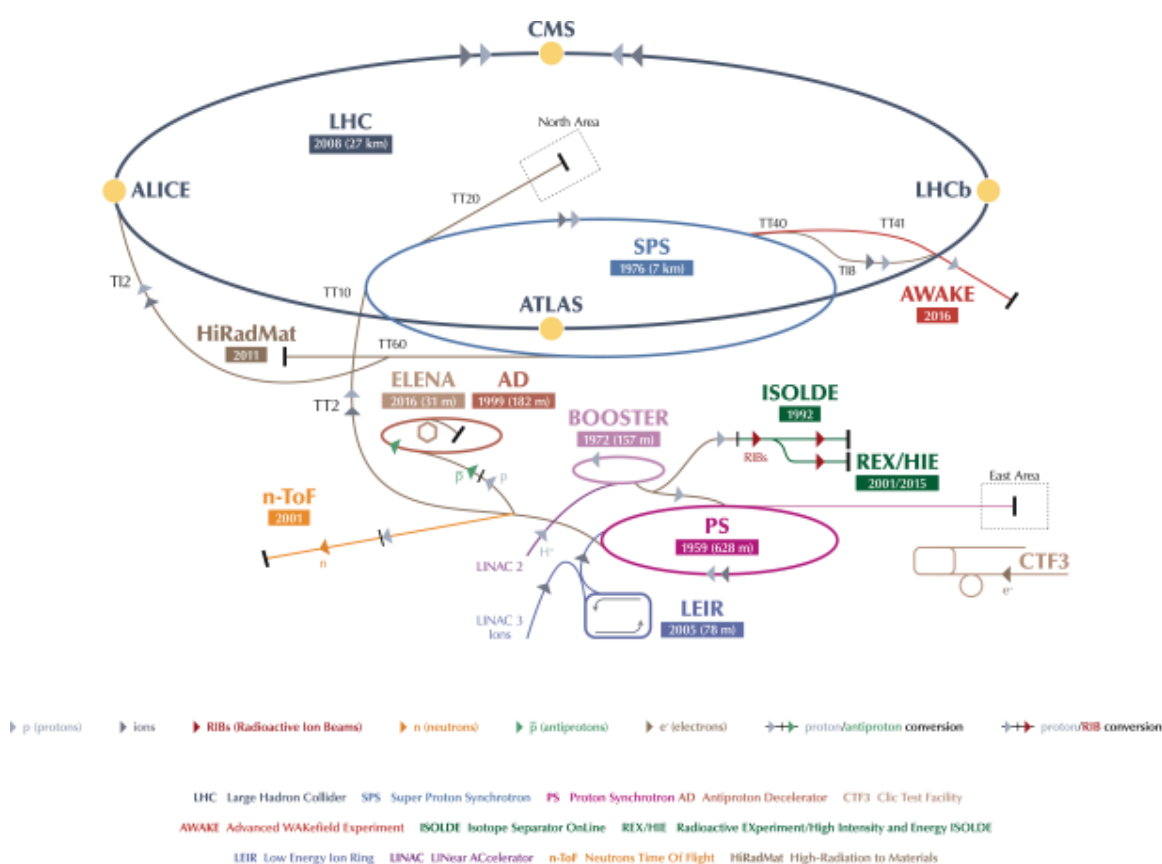


Fig. 1. Machines and experimental areas at CERN

For implementing the control over the experiments and other resources used at CERN (accelerators, cryogenic, vacuum...), the organization has its own section for creating and maintaining the applications for the automation and control of the different (continuous) processes. That section is called the **Process Control Systems** section, included in the **Industrial Control and Safety Systems** group, within the **Beams** department [2] (BE-ICS-PCS).

The PCS section has delivered and maintains a framework called UNICOS [3] for developing applications for different PLCs (basically Siemens and Schneider PLCs in the organization, although some features are also supported in CodeSys for other possible brands).

2.2. PLC

A programmable logic controller, or PLC, is a device used mostly in industrial environments as to control the equipment required for the proper functionality of a process, shown in Fig. 2. The current project is intended to solve the issue of the fast interlocks for the Siemens PLCs that use Step 7 as a programming software, as that is the device used by the client that required this functionality.



Fig. 2. Siemens PLC CPU 317-2 PN/DP [4]

Step 7 is the programming software where the code of the PLC can be generated and uploaded. It has several tools to edit the code, configure the hardware, generate the program, visualize online data, and others.

In a Siemens PLC, programs are structured in organization blocks, which make use of other blocks (function blocks, data blocks and functions) to achieve their purpose. Each organization block has a different purpose and so it is executed by the operating system in a different way. The organization block OB 1 is executed cyclically, which means the PLC executes it and when its code is

processed, it executes it again from the beginning with the new inputs read from the periphery. The update of the inputs is done before the OB 1 is executed and the update of the outputs is done right after its execution.

The time spent in between the start of two consecutive OB 1 executions is called the PLC cycle time. The response time is the time elapsed between the trigger of an input of the PLC and its reaction in the outputs.

2.3. Fast Interlocks

The normal longest response time for an event in a Siemens PLC can be up to almost twice the PLC cycle (as shown in Fig. 3). The cycle time can vary depending on the size of the application, the PLC model and the number of interrupts occurred. The maximum PLC cycle time for the PLCs used at CERN can be around 500 ms, which would imply a maximum of a 1 s response time.

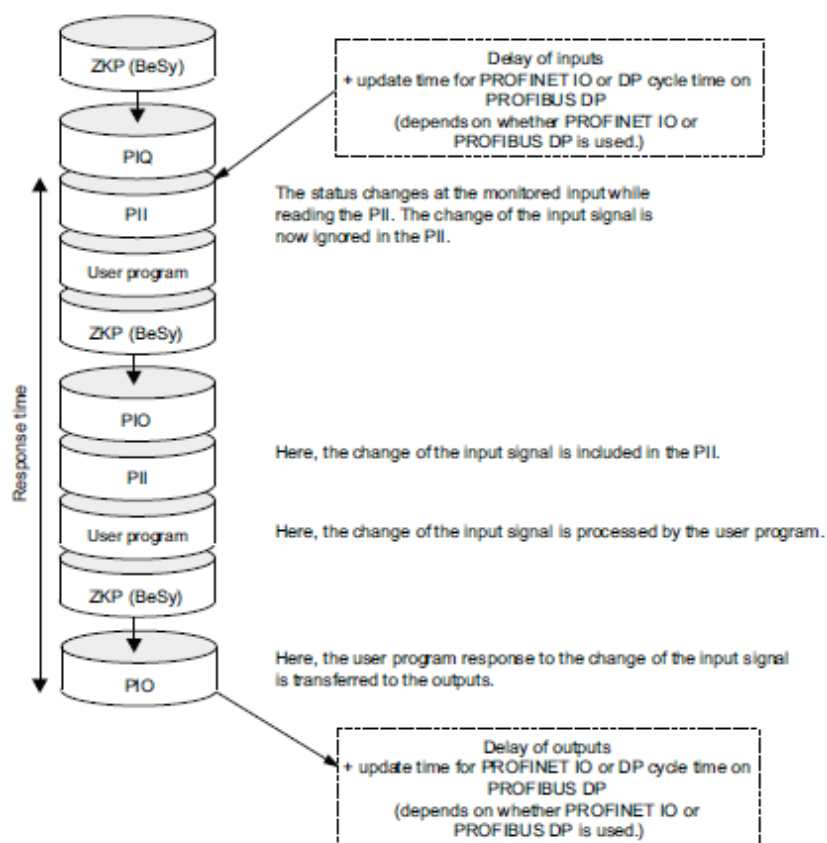


Fig. 3. Longest response time for an input signal [5]

However, there are some events (fast interlock events) that need to be treated much faster. Around 10 ms or less would be a reasonable response time for this kind of events. That makes it non-viable to use the normal processing execution for this kind of task.

These events are expected to be needed for very specific applications related to security of people or machines. This is an important fact that needs to be taken into account for the decisions that are to be made.

2.4. The project

One of the users of the UNICOS framework developed at CERN has requested to integrate a solution in Siemens PLCs for the fast interlock events. The requirement was to respond through one or more digital outputs to one or more digital inputs in an amount of time smaller than the PLC cycle time. The decision of the PCS section has been to implement a solution for this type of events in the UNICOS framework that they support.

The developing of this feature requires a deep understanding of the UNICOS framework and of the TSPP protocol used on it, as well as knowledge on automation systems (more specifically Siemens PLCs and the Step 7 software). For a further implementation using the automatic generation tool, it would also be convenient to have some knowledge of the python and java programming languages.

3. Description of the UNICOS framework

UNICOS [3] [6] is a framework developed by CERN that permits the creation, implementation and maintenance of applications for PLCs in a faster way than creating them one by one from the very beginning.

For creating an application, the UNICOS CPC framework (delivered for Continuous Process Control) requires an excel file filled with the specifications of the different objects used inside the program. From there, it is possible to configure the logic inside the program in some predefined parts of the code generated by the framework as to accomplish the desired functionality of the process. The logic that needs to be implemented should also be explained in a document for that purpose. That logic can either be directly implemented in the SCL code generated or be automatically generated by creating the custom template files for that purpose.

3.1. UAB

The UNICOS Application Builder is a software containing different modules to generate applications for both the control and the supervision layers of a process. The CPC module for continuous process control contained in it generates the necessary files (in SCL code for Step7), along with the PLC project, and the file used to generate the WINCC OA project. For this objective, this software uses a plugin (maintained by the UAB developers) and some templates (maintained by the PCS section) to generate the multiple files used. The selection of the CPC module used inside UAB is shown in Fig. 4.



Fig. 4. UNICOS application builder module selection

3.2. UNICOS PLC code

The code used in the UNICOS framework is mostly generated using the Structured Control Language (SCL) programming language [7]. It is very similar to the structured text (ST) programming language supported by the IEC 61131-3 standard, but with the difference that SCL is a private programming language from Siemens.

A piece of SCL code is shown in Fig. 5.

```

(*EXEC of ONOFF*****
FUNCTION FC_ONOFF : VOID
TITLE = 'FC_ONOFF'
//
// ONOFF calls
//
AUTHOR: 'UNICOS'
NAME: 'Call_OO'
FAMILY: 'OnOff'
VAR_TEMP
    old_status1 : DWORD;
    old_status2 : DWORD;
END_VAR
BEGIN

// -----
// ---- OnOff <1>: OnOff01
// -----
old_status1 := DB_bin_status_ONOFF.OnOff01.stsreg01;
old_status2 := DB_bin_status_ONOFF.OnOff01.stsreg02;
old_status1 := ROR(IN:=old_status1, N:=16);
old_status2 := ROR(IN:=old_status2, N:=16);
OnOff01.ManReg01:=DB_ONOFF_ManRequest.ONOFF_Requests[1].ManReg01;

OnOff01.HFOn := DB_DI_All.DI_SET.UDI01.PosSt;

```

Fig. 5. SCL code visualized inside the Step 7 software

3.3. UNICOS code structure in Siemens PLC

Applications created using the UNICOS framework are developed with a structure and with the necessary predefined UNICOS objects, including input/output objects (such as digital inputs), interface objects (such as analog parameter), field objects (such as OnOff) and control objects (such as PCO).

The main processing of a UNICOS application in a Siemens PLC is done inside the OB 1, which is executed cyclically. The order of execution of the different UNICOS objects inside that OB is shown in Fig. 6.

This kind of applications in some cases also make use of OB 35, 32, 100 and others to fulfil their requirements.

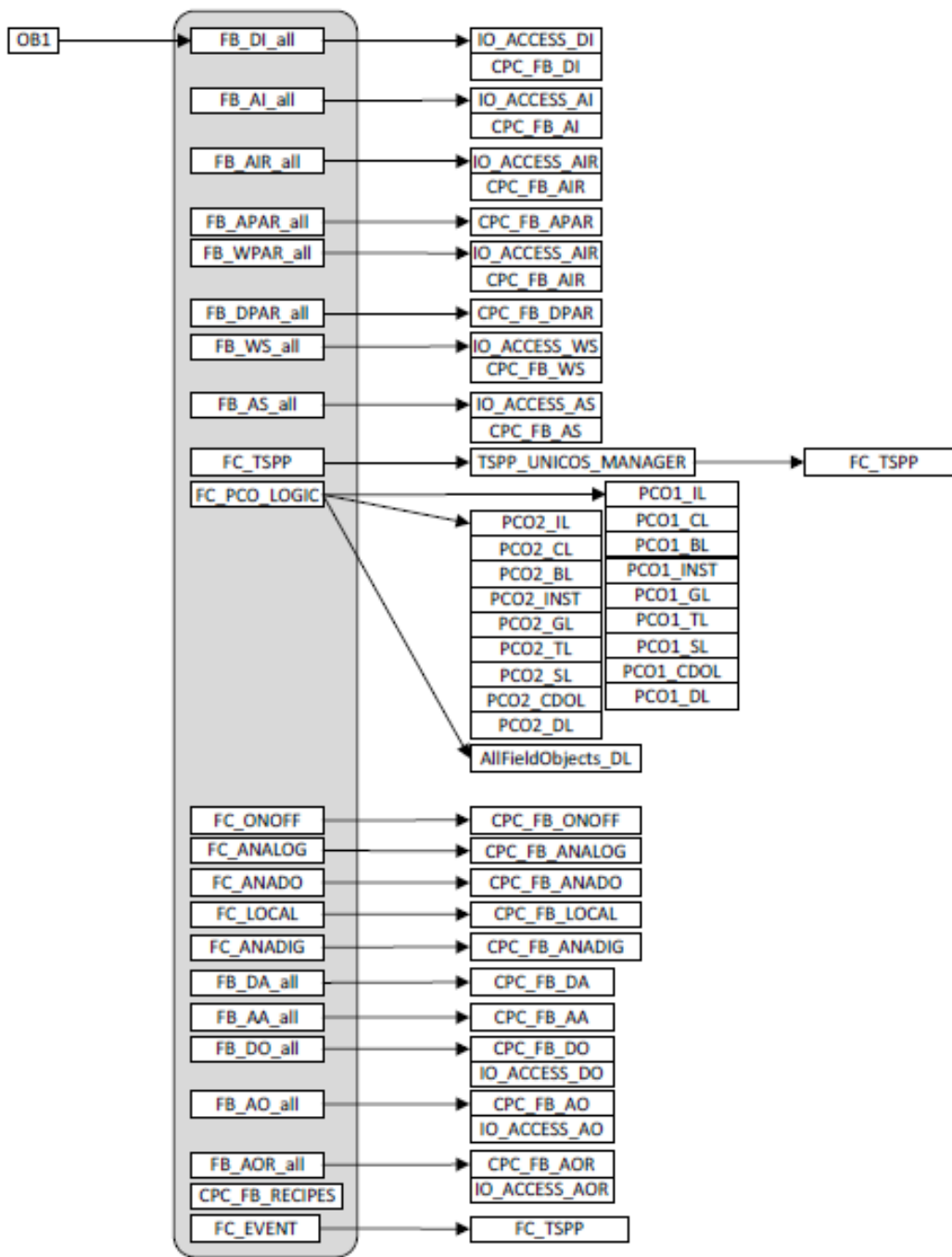


Fig. 6. UNICOS functions and function blocks inside OB 1 in a Siemens application [8]

Inside OB 1, first the input objects are called (DI, AI, AIR), then the interface objects (APAR, WPAR, DPAR, WS, AS), followed by the control (PCO) and the field objects (ONOFF, ANALOG, ANADO, LOCAL, ANADIG), ending with the alarm (DA, AA) and finally the output objects (DO, AO, AOR).

The TSPP is called at the end of the program (FC_Event), although if the PLC cycle is too long, it is also called in the middle of the OB (right before the logic) without using the events processing.

3.3.1. TSPP

The Time Stamp Push Protocol [9] (TSPP) is used by the UNICOS framework to communicate the PLC and the SCADA layers using time stamped events, status tables and a watchdog to determinate if the communication is still alive.

3.3.2. Compilation of SCL files in Step 7

The compilation of the SCL files inside the Step 7 software is done by the INP (SCL compile control file) files, with the exception of the OBs compilation, which is directly written in a SCL file. The order of execution of the INP files and the 4_Compilation_OB.scl file is shown in Fig. 7.

The INP files call other SCL files that they compile. By this process, the different blocks used inside the Step 7 program are generated with the code implemented in those files.

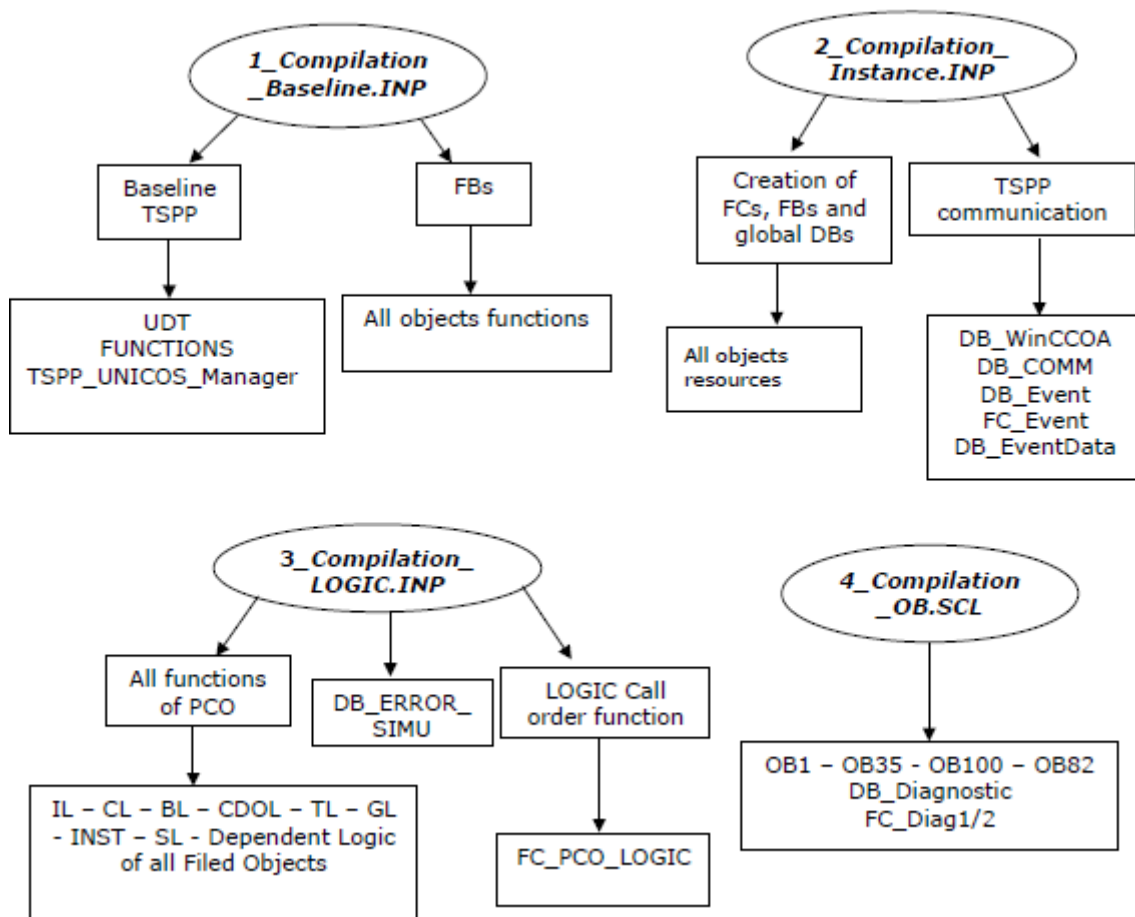


Fig. 7. Compilation order in Step 7 [8]

3.4. Templates used by the UNICOS framework

The templates used by the UNICOS framework are stored in the folder “Resources” of any application built with the UAB, as shown in Fig. 8 for a Step7 application. That folder contains multiple subfolders including the description of the devices that can be used and the templates for generating the instances and the logic inside Step7 and the WINCC OA instances. The folders and files not explained in this section are not relevant for the current project.

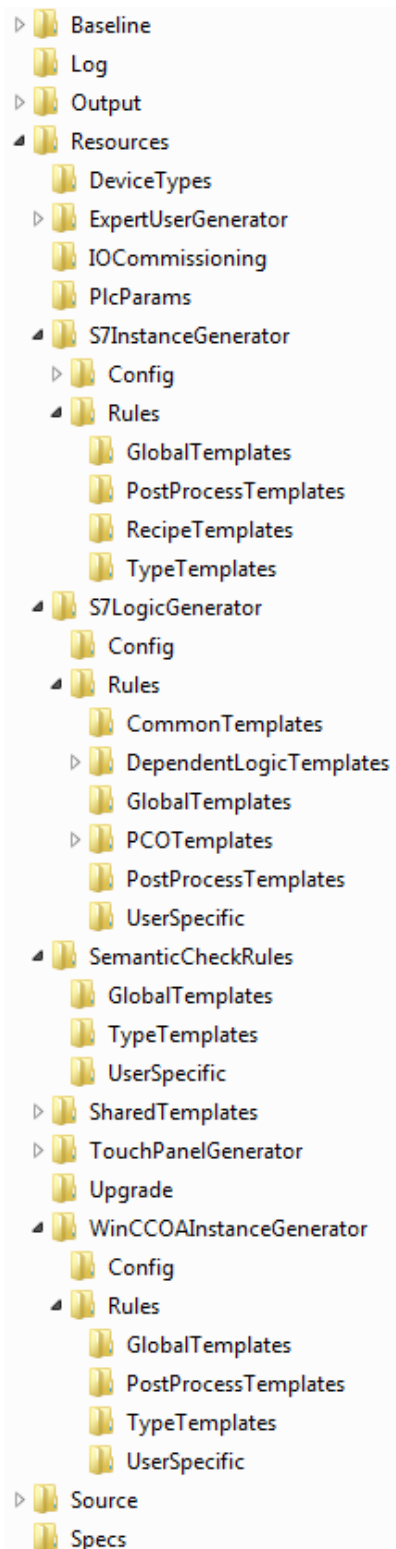


Fig. 8. Folder Tree inside a UNICOS Step7 project

UAB uses the Python files contained in these folders as to generate the files used inside the Step7 project, as well as the WINCC OA file. In order not to write the SCL code directly in those Python files, making the code longer and more difficult to read, the standard, common SCL code for multiple applications is written in the SCL files contained inside those folders as well. Those files are then used

and modified by the Python files to generate the proper, final SCL files used in Step7, customized for the current application.

3.4.1. DeviceTypes

This folder contains an xml file for each UNICOS object type. It defines the different parameters of the objects, as well as the fields that are then filled in the specifications file.

3.4.2. PlcParams

This folder contains the xml files for the different platform configurations (Siemens, Schneider, Codesys and TIA). For the Siemens file, it defines the names that will be used to generate the symbols in the different memory areas that will be used in the project.

3.4.3. S7InstanceGenerator folder

The templates contained inside this folder are used as to generate the instance SCL files of the different UNICOS objects. It contains another two folders. The “Config” folder contains a SCL file that has the skeleton of the TSPP manager function block and its data block. The “Rules” folder contains some more folders with other skeleton of functions used, as well as the python files for generating those files.

- GlobalTemplates: This folder contains the templates for generating the compilation files inside Step7 and the TSPP and COMMUNICATION files.
- TypeTemplates: This folder contains the templates for generating the different UNICOS objects files used in the Step7 application.

3.4.4. S7LogicGenerator folder

The templates contained inside this folder are used as to generate the logic used by the application in some SCL files. It contains the same two folders existent in the instance generator folder. The main python files are contained in the “Rules” folder.

- CommonTemplates: This folder contains the template for generating the code for processing the alarms.
- GlobalTemplates: Among others, this folder contains the template for generating the code for processing the PCOs sections and the dependent logic of the UNICOS objects.

3.4.5. SemanticCheckRules folder

The python files contained inside this folder are used as to check that the specifications file has been filled properly.

- GlobalTemplates: This folder contains the templates for checking the whole specifications file.
- TypeTemplates: This folder contains the templates for checking the different UNICOS objects of the specifications file. Each UNICOS object is contained in a different sheet inside it.

3.4.6. SharedTemplates

This folder contains the python files that are common for all the templates used in the project.

3.4.7. WINCCOAInstanceGenerator folder

The python files contained in this folder are used as to generate the file used by WINCC OA as to generate a project. It contains the same two folders existent in the instance generator folder. The main templates are contained in the “Rules” folder.

- TypeTemplates: The different UNICOS object instances inside WINCC OA are generated in the templates contained in this folder.

4.Objectives

The objective of the current project is to implement a mechanism that responds to fast interlock events as soon as possible inside the UNICOS CPC environment that is used by CERN to create PLC applications. It would be convenient to accomplish the requirements of the UNICOS framework regarding objects treatment and TSPP communication protocol, which includes time stamp of the events occurred in the fast interlock processing.

By fulfilling this requirement, the organization would be able to treat the elements necessary for the fast interlock execution inside the UNICOS framework (which means they would be able to treat the signal by the objects in a similar way that if they were normal ones), and still respond to the fast interlock events in an amount of time that is valid for the client. In this sense, the applications that need this feature would be more easily maintained and upgraded just as any other application currently included in the framework.

Regarding the UNICOS framework, it would be necessary to determinate the objects that could be necessary inside the fast interlock processing, as well as using just the amount of code necessary to permit a fast response in the outputs. The amount of elements of the UNICOS framework used for the fast interlock processing should be as small as possible as not to take an amount of time that could be inviable for the fast interlocks response time requirements.

The need of a modification to support the capability of time stamping inside fast interlock processing in the least amount of time appears, and should be solved in the current project for the TSPP processing. The events occurring during the fast interlocks should have their own time stamp (different and more precise to the time of the fast interlock trigger than the one used during the TSPP processing in the main program). The TSPP processing requires an amount of time which is not possible inside the fast interlocks processing, and may require more than one execution to send the messages, which cannot be done inside the fast interlock processing, as the TSPP processing inside it is just called whenever a fast interlock input triggers it.

The moment a functional code is created and tested in a real PLC, another objective would be to modify the templates used to generate the SCL code of the PLC so that they include the changes made to support the fast interlock feature.

Some other files would also need to be modified to generate the final project for the user. Those files are included in the plugin of UAB, as they are used for some other purposes as well during the application generation.

The objective would be, in summary, to modify the SCL code of a UNICOS project to support the fast interlock feature in Step7, and then to modify the files necessary to generate that code automatically for any application that needs it. In this sense, the objectives could be described and scheduled as the following:

- Research of possible solutions to the problematic described.
- Implementation of the solutions and testing, as to decide the best solution(s) possible.
- Implementation of the solution in the automatic generation tools of UNICOS.
- Validation of the code generated.

5. Analysis

This section describes the different possibilities that could be used to develop the fast interlock feature, the different options that will be implemented, and how they will be tested to select the best solution for the final implementation.

5.1. UNICOS project

Ideally, the processing of the fast interlocks scheme should follow the same schedule used by a normal UNICOS project, but using only the necessary objects for the fast interlock treatment. As that could lead to large response times, a first approach with just the inputs, manually created logic and outputs will be implemented, and from that on, it will be compared with its execution with more UNICOS objects that could also be convenient to treat inside the fast interlock processing.

Finally, a set of UNICOS objects will be allowed to be used in the fast interlock feature, which will depend in the difference of time spent with each set of objects and the best solution to implement regarding simplicity for the user.

5.2. Fast response possibilities in Siemens PLC

As the response time from a normal application execution is not valid to support the fast interlock events treatment, the solution needs to be done by interrupt OBs in a Siemens PLC [10]. The possible solutions are the following:

- Hardware Interrupts

Execution of an OB (OB 40 [11]) related to a hardware interrupt [12] every time a fast interlock signal changes. The OB is executed immediately after the signal change. For the use of this OB, a special hardware that supports this functionality is needed. For example, module 6ES7321-7BH01-0AB0 [13] for S7-300 CPU family.

- Software Interrupts

Execution of an OB (OB 34, since OB 35 is already in use in the framework [11], and the feature needs the highest priority interrupt possible) related to a cyclic interrupt every short amount of time as to detect fast interlock signal changes (cyclic time can be configured as to fit the requirements). If not changes are detected, processing and output update may not be necessary. This solution

consumes more CPU time than the hardware interrupt one, and so it increases PLC cycle time (with the possibility of an OB 80 time error call if the PLC cycle gets longer than the allowed maximum time), but does not need any special hardware to work.

Both options will be tested as to check their viability and select the best solution for the final implementation.

5.2.1. Interrupts regarding PLC

For both hardware and cyclic (software) interrupts, it is necessary to have an updated set of Inputs different from the ones in the process image inputs (as they are not up to date when the OB is called [14]), as well as to write the outputs as soon as possible. The options available for that objective are the following [15] [16]:

- Process Image update inside the OB

Using the SFC 26 (UPDAT_PI) and 27 (UPDAT_PO) [17] it is possible to update the process image of both inputs and outputs every time the related OB is called. Many of those inputs and outputs may not be necessary for the processing though. Time needed for the process image transfer depends on the CPU used. In Fig. 9 the time needed for the CPU 31x family is shown.

Const.	Components	CPU				
		312	314	315	317	319
K	Base load	150 μ s	120 μ s	100 μ s	70 μ s	40 μ s
A	Per byte in rack 0	20 μ s			15 μ s	
B	Per byte in racks 1 to 3	-	30 μ s*		25 μ s*	22 μ s*
D (DP only)	Per word in the DP area for the integrated DP interface	-		0.5 μ s		
P (PROFINET only)	Per word in the PROFINET area for the integrated PROFINET interface	-		0.5 μ s		

* +40 μ s je per rack

Fig. 9. Process image transfer time for CPU 31x family [5]

- Process Image Partition

Using a Process Image Partition [18] [19] for both inputs and outputs it is possible to update just the variables needed for the fast interlock response. It is also possible to attach the Process Image Partition to the related OB so that it is updated at the start and at the end of the OB processing (just like OB1 works with typical process image, only available from S7-400 CPU though). Otherwise, a manual update via SFC 26 and 27 would be necessary. However, inputs and outputs inside Process

Image Partition will not be updated automatically during OB1, so it may be necessary either to update them manually using SFC 26 and 27 in that OB as well, or to access them directly by peripheral addressing.

- Peripheral addressing

It is possible to access the inputs and outputs directly using the peripheral addressing. It takes more time than accessing the process image though, as shown in Fig. 10. By peripheral addressing you also have to read full byte, word or double word (not bit access permitted), so it would be somehow necessary to read the current output before writing it modifying only the proper bit.

In-struction	Address Identifier	Description	Length in Words ²⁾	Typical Execution Time in μ s								
				Direct Addressing				Indirect Addressing ¹⁾				
				312	31x, 147, 151, 154	317	319	312	31x, 147, 151, 154	317	319	
L	IB	a	Load ... Input byte	1/2	0.4	0.2	0.05	0.01	2.7+	1.4+	0.14+	0.01+
	QB	a	Output byte	1/2	0.4	0.2	0.05	0.01	2.7+	1.4+	0.14+	0.01+
	PIB	a	Peripheral input byte for 31x	1/2	70.2	43.3	15.01	13.1	108.4+	44.6+	15.08+	13.1+
	PIB	a	... for 147	1/2	-	50.5	-	-	-	51.8+	-	-
	PIB	a	... for 151-7 (Bus <= 1m)	1/2	-	104.8	-	-	-	105.0+	-	-
	PIB	a	... for 151-7 (Bus > 1m)	1/2	-	136.4	-	-	-	138.2+	-	-
	PIB	a	... for 151-8 (Bus <= 1m)	1/2	-	68.3	-	-	-	69.6+	-	-
	PIB	a	... for 151-8 (Bus > 1m)	1/2	-	88.8	-	-	-	90.5+	-	-
	PIB	a	... for 154	1/2	-	68.3	-	-	-	69.6+	-	-
	PIB	a	Digital Onboard I/O ³⁾	1/2	51.5	48.3	-	-	65.2+	55.6+	-	-
	PIB	a	Analog Onboard I/O ⁴⁾	1/2	-	162.1	-	-	-	169.4+	-	-
	MB	a	Bit memory byte	1/2	0.5	0.2	0.05	0.01	2.6+	1.4+	0.14+	0.01+
	LB	a	Local data byte	2	0.9	0.5	0.05	0.02	3.3+	1.7+	0.13+	0.01+
	DBB	a	Data byte	2	3.0	1.5	0.17	0.02	4.7+	2.5+	0.12+	0.01+
DIB	a	Instance data byte into ACCU1	2	3.0	1.5	0.17	0.02	4.7+	2.5+	0.12+	0.01+	

Fig. 10. Time for a byte load instruction using process image or peripheral addressing in CPU 31x [20]

As the rearrangement of the inputs is not something desired and the amount of fast interlock inputs and outputs is expected to be very low in comparison to the whole application, the peripheral addressing will be the option selected for the testing and the final implementation.

5.2.2. Interrupts regarding the UNICOS Framework

For the implementation of the OB, only digital inputs, logic and digital outputs would be absolutely necessary, as well as TSP management for faster than PLC cycle changes recording. Even for those UNICOS elements, only some of the signals and processing may be necessary, as to make the code short in order to react as fast as possible to the event.

Another option would be processing the user defined logic directly using the inputs and outputs without passing through the UNICOS input, field and output elements making the code even

shorter (similar to the current solution of the client). The user code could still (or not) be contained in a UNICOS logic element. This solution is not convenient, as the processing of the fast interlock signals may still be necessary inside the main program in order to respond to an event, even if the response is not time critical (not a fast interlock processing). Also, it would imply to be unable to visualize and interact with the UNICOS objects in WINCC OA in the supervision layer (at least, not with the conventional UNICOS objects).

5.3. TSP

The TSP normal processing takes too much time for a fast interlock processing, and could imply the loss of messages or the double sending of them. The fast interlock treatment needs to implement another kind of treatment that is compatible with the current TSP implementation but that implies the minimum amount of processing time possible, a time stamp of the fast interlock events (different than the normal processing) and, in general, the avoidance of loss or resending of messages.

5.4. Test requirements

The program for testing will include 3 normal inputs and outputs and 3 fast interlocks inputs and outputs directly connected one by one. This program will be tested in a real PLC, with its input (with hardware interrupt capability) and output module. To determinate the response time of the fast interlock events, an oscilloscope will also be used to test the validity of the solution implemented.

The program needs to be created with the software and hardware configuration according to the real ones. A CPU 317-2 PN/DP [4] will be used, connected via Ethernet to the computer with the program created and the SCADA for monitoring. An input module 6ES7 321-7BH00-0AB0 [21] and an output module 6ES7 322-1BF01-0AA0 will be used as well. This configuration will be established in the hardware configuration, shown in Fig. 11, inside Step 7, which will be the software used.








 CPU 317-2PN/DP	6ES7 317-2EK14-0AB0	V3.1	2		
 MPI/DP			2	8191"	
 PN-DP				8190"	
 Port 1				8189"	
 Port 2				8188"	
 DI16xDC24V, Interrupt	6ES7 321-7BH00-0AB0			0...1	
 DO8xDC24V/2A	6ES7 322-1BF01-0AA0				0

Fig. 11. Hardware configuration in Step 7

The measures will be done with the use of an oscilloscope, taking the signals from the real PLC used. The cycle time of the PLC (when needed) will be taking from the Step 7 software, making use of the online information it provides of the real PLC.

The test of the TSPP changes will be done with the use of the WinCC OA software. For the testing, a custom application will be created for the project generated.

5.5. Templates

The modifications to the templates should generate the SCL files created and modified manually, for all the solutions that are going to be implemented. The modifications will be done trying to be the least invasive possible, respecting the current files configuration, but trying to be easy to maintain and extend. The proper modification of the templates files will be checked comparing them to the ones created manually, using the same specification file. Then it will be implemented in the PLC when the changes that are necessary in the SCL files (because of the application specific requirements) are done, as to check the files are properly generated and have the expected behaviour.

Finally these files will be implemented in the framework as the new fast interlock feature.

6. Conclusions

After the realization of the research part of the current project, it has been confirmed that using either hardware or cyclic interrupts, fast interlock response times are always in between the same ranges regardless of the PLC cycle time, whereas normal response times depend on it.

As expected, cyclic interrupts increase the PLC cycle time depending on the amount of times the OB is called, which depends on the cycle time of the PLC itself. For example, without enlarging the PLC cycle time by a loop, PLC cycle time is between 1 and 3 ms for hardware interrupts and between 2 and 5 ms for cyclic interrupts. When the PLC cycle time is enlarged by the loop to the maximum value measured, the response time is in between 407 to 422 ms for hardware interrupts, whereas it is in between 456 to 477 ms for cyclic interrupts (around 50 ms longer). The amount of time the PLC cycle time is enlarged depends also on the amount of input bytes that need to be read by peripheral addressing, as well as on the cyclic interrupt time defined.

Response time for cyclic interrupts has more spread values than for hardware interrupts, as it depends on the moment of the input change and of the cyclic interrupt trigger, whereas in hardware interrupts the OB 40 is executed when an input change is detected (after the input delay time configured).

Short input delays (0.1 or 0.5 ms) sometimes lead to system failures on the input module, where some interrupts could be lost, with the use of hardware interrupts. Cyclic interrupts do not have this problem, as they are disabled when a change is detected (DIS_IRT) and re-enabled when the processing has ended (EN_IRT), and so, can be used with input delays as short as necessary (although it may be necessary to use longer input delays for proper state change detection). This problem appears when there are multiple changes in a short amount of time, as the processing of the respective OB is slower than the input change detection itself.

The use of the necessary UNICOS objects inside the interrupt does not affect the response time in a big way (up to 439 μ s average measured), and permits a more coherent with UNICOS framework processing. The final decision was to use Digital Input, Digital Output and Digital Alarm objects to have the proper time stamp of the fast interlock occurrence, and OnOff objects to process the logic defined by the user.

Hardware interrupts are in general terms a better solution for processing fast interlock events than cyclic interrupts, especially for long input delays (3 ms or more). Still though, its use requires a specific hardware which needs to be taken in consideration.

The feature developed after the realization of the project implies the possibility for the users to decide between hardware and cyclic interrupts, as both have been implemented. For the user to know the advantages and disadvantages of both type of interrupts, the user manual, which is also included on the documentation of this project, is provided.

An extract of the TSPP has been adapted for the fast interlock processing, and now communicates the changes occurred in the fast interlock processing to the SCADA level with the proper time stamp of the events.

The modifications done to the template files, as well as to the files used in the plugin of UAB, now permit the automatic generation of UNICOS applications with fast interlock support. For the use of the feature, a new column has been added to the specifications file for the fast interlock objects (DI, DA, OnOff and DO). The necessary check of the proper filling of those cells has also been implemented.

7.Future works

The solution implemented is fully functional and fulfils the requirements expected from the client. Still though, some other possibilities could be implemented in the future, as to solve other problems similar to the one solved by the execution of this project. For example:

- Checking the viability and, if appropriate, implementing the fast interlock feature also for Schneider PLCs; as well as for TIA Portal and CodeSys if needed.
- Including other UNICOS objects in the fast interlock processing, such as the PCO, other field and output objects and even analog input triggers.
- Permitting more interactions between fast interlock and non-fast interlock objects, as well as avoiding consistency issues if viable.
- Modifying the general TSPP processing so that it does not need a sending of redundant data (status and events have, in general, the same information), a double check of that data and a double storing of it inside the PLC. This would imply some changes in the control layer, but also a considerable amount of changes in the supervision layer.
- Checking the code generated has been properly implemented with the use of a formal verification tool, both for the SCL code and for the templates code.

8.Acronyms

CERN	European Organization for Nuclear Research
UNICOS	Unified Industrial Control Systems
UAB	UNICOS Application Builder
CPC	Continuous Process Control
BE-ICS-PCS	Beams department, Industrial Controls and Safety group, Process Control Systems Section
FI	Fast Interlock
TSPP	Time Stamp Push Protocol
PLC	Programmable Logic Controller
ST	Structured Text
SCL	Structured Control Language
OB	Organization Block
FB	Function Block
FC	Function
SFB	Standard Function Block
SFC	Standard Function
PII	Peripheral Image of Inputs
PIO/PIQ	Peripheral Image of Outputs
IEC	International Electrotechnical Commission
SCADA	Supervisory Control And Data Acquisition
WINCC OA	WinCC Open Architecture
DI	Digital Input
DA	Digital Alarm
DO	Digital Output
PCO	Process Control Object

9. Documents of the project

The current project has been elaborated in multiple documents that describe a certain part of the project.

1. Report: General description of the project. Objectives and conditions for its test. Conclusion from the realization of the project and future works.
2. Planning and budget: Schedule of the different tasks that compound the project and price of the resources used.
3. Step 7 programmer manual: Modifications to the code of the UNICOS applications to support the fast interlock capability. Results obtained from these modifications.
4. Templates programmer manual: Modifications to the code of the templates and of the plugin used to generate the SCL files used in the PLC.
5. User manual: Steps to create a fast interlock UNICOS application.
6. Templates code: Modified template files inside the resources folder of an application and of the UAB plugin.
7. Datasheets: Datasheets of the devices used to research and test the solution for the fast interlocks issue.

Attachments.

1. Attachment 1: Fast interlock application example.

10. Bibliography

- [1] CERN, "About CERN," 19 January 2012. [Online]. Available: <http://home.cern/about>. [Accessed 11 November 2016].
- [2] CERN, "Beams Department," [Online]. Available: <https://espace.cern.ch/be-dep/default.aspx>. [Accessed 11 November 2016].
- [3] CERN, "UNICOS," [Online]. Available: <https://unicos.web.cern.ch/>. [Accessed 11 November 2016].
- [4] Siemens, "Device 6ES7317-2EK14-0AB0," [Online]. Available: <https://mall.industry.siemens.com/mall/en/WW/Catalog/Product/6ES7317-2EK14-0AB0>. [Accessed 15 December 2016].
- [5] Siemens, "S7-300 CPU 31xC and CPU 31x: Specifications Firmware V2.6," 20 May 2009. [Online]. Available: <https://support.industry.siemens.com/cs/document/36305149/s7-300-cpu-31xc-and-cpu-31x%3A-specifications-firmware-v2-6?dti=0&lc=en-WW>. [Accessed 11 November 2016].
- [6] CERN, "UNICOS CPC Resources Package," 2016. [Online]. Available: <http://ucpc-resources.web.cern.ch/ucpc-resources/1.9.1/index.html>. [Accessed 19 January 2017].
- [7] Siemens, "SIMATIC S7-SCL," [Online]. Available: <http://w3.siemens.com/mcmsg/simatic-controller-software/en/step7/simatic-s7-scl/pages/default.aspx>. [Accessed 15 December 2016].
- [8] J. Ortolá Vidal, "UNICOS CPC6 PLC architecture on Siemens S7," 31 January 2007. [Online]. Available: https://edms.cern.ch/ui/file/1228441/1.8.0/CPC6_S7_PLC_architecture.pdf. [Accessed 15 December 2016].
- [9] J. Brahy, "TSPP UNICOS MANAGER," 14 May 2005. [Online]. Available: https://svn.cern.ch/repos/en-ice-svn/trunk/tools/UNICOS/CPC/TSPP/DriverS7/Doc/TSPP_description.pdf. [Accessed 15 December 2016].
- [10] RAWAT, "Forum topic: what happen when OB1 scanning and Cyclic Interrupt OB come ?," 26 October 2008. [Online]. Available: <https://support.industry.siemens.com/tf/ww/en/posts/what-happen-when-ob1-scanning-and-cyclic-interrupt-ob-come/23086/?page=0&pageSize=10>. [Accessed 11 November 2016].
- [11] Siemens, "SIMATIC Programming with STEP 7 V5.5," May 2010. [Online]. Available: <https://support.industry.siemens.com/cs/document/45531107/simatic-programming-with-step-7-v5-5?dti=0&lc=en-WW>. [Accessed 11 November 2016].
- [12] Siemens, "What is a hardware interrupt and how do they work in the S7-300 system?," 20 December 2010. [Online]. Available:

<https://support.industry.siemens.com/cs/document/23657941/what-is-a-hardware-interrupt-and-how-do-they-work-in-the-s7-300-system-?dti=0&lc=en-WW>. [Accessed 11 November 2016].

- [13] Siemens, "Module 6ES7321-7BH01-0AB0," [Online]. Available: <https://support.industry.siemens.com/cs/pd/254353?ptdi=td&pnid=13751&lc=en-WW>. [Accessed 11 November 2016].
- [14] chiragshah, "Forum topic: Hardware Interrupt," 3 September 2015. [Online]. Available: <https://support.industry.siemens.com/tf/ww/fr/posts/hardware-interrupt/132823?page=0&pageSize=10>. [Accessed 11 November 2016].
- [15] Siemens, "Where and when do you need peripheral addressing?," 25 July 2011. [Online]. Available: <https://support.industry.siemens.com/cs/document/18325417/where-and-when-do-you-need-peripheral-addressing-?dti=0&lc=en-WW>. [Accessed 11 November 2016].
- [16] Gssc1414, "Forum topic: Increasing PIP & PIQ vs. peripheral addressing," 1 September 2011. [Online]. Available: <https://support.industry.siemens.com/tf/ww/en/posts/increasing-pip-piq-vs-peripheral-addressing/62626/?page=0&pageSize=10>. [Accessed 11 November 2016].
- [17] "PLC dev," [Online]. Available: http://www.plcdev.com/s7_library_functions. [Accessed 11 November 2016].
- [18] daad, "Forum topic: Is hardware interrupt update I/Os," Siemens, 14 October 2012. [Online]. Available: <https://support.industry.siemens.com/tf/ww/en/posts/is-hardware-interrupt-update-i-os/81597/?page=0&pageSize=10>. [Accessed 11 November 2016].
- [19] Siemens, "How do you use or update process image partitions for S7-400 CPU modules during an interrupt OB?," 30 November 2012. [Online]. Available: <https://support.industry.siemens.com/cs/document/18325216/how-do-you-use-or-update-process-image-partitions-for-s7-400-cpu-modules-during-an-interrupt-ob-?dti=0&lc=en-WW>. [Accessed 11 November 2016].
- [20] Siemens, "S7-300 Instruction List," June 2008. [Online]. Available: <https://support.industry.siemens.com/cs/document/13206730/s7-300-instruction-list-cpu-31xc-cpu-31x-im-151-7-cpu-im-151-8-cpu-im-154-8-cpu-bm-147-1-cpu-bm-147-2-cpu?dti=0&lc=en-WW>. [Accessed 11 November 2016].
- [21] Siemens, "SIMATIC S7-300 S7-300 Module data," 25 February 2013. [Online]. Available: <https://support.industry.siemens.com/cs/document/8859629/simatic-s7-300-s7-300-module-data?dti=0&lc=en-WW>. [Accessed 11 November 2016].