Universidad de Oviedo

Manual del programador de Templates 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

FIGURE INDEX

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 explains the changes made to the python code of the templates (as well as the java code of the plugin of UAB) and other files as to generate the SCL code used to make a functional fast interlock treatment using the UNICOS framework.

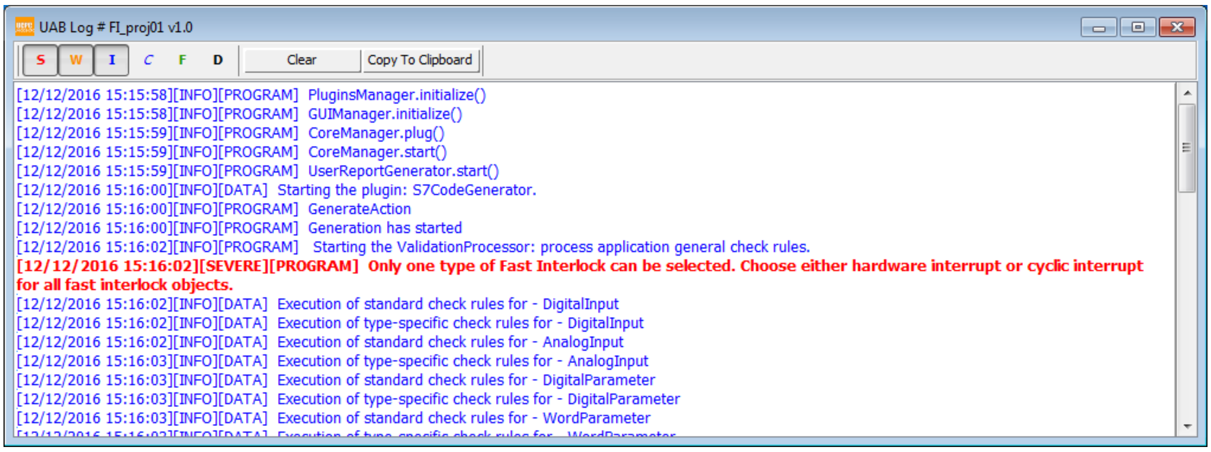
For an insight of the full code of the template files that have been modified, see the Templates code document. The plugin files of UAB that have been modified are also included in that document.

2. Templates code

As to generate the new SCL code necessary, some of the template files have been modified. They are described in this section.

The modifications of the files were tested making use of the UNICOS cpc wizard, as the resources it uses are the ones from the resources folder (in most of the cases). The execution time errors were displayed in the UAB log, and used as to correct the mistakes or misbehaviours of the files modified.

A screenshot of the UAB log is shown in Fig. 1.



```
[12/12/2016 15:15:58][INFO][PROGRAM]  PluginsManager.initialize()
[12/12/2016 15:15:58][INFO][PROGRAM]  GUIManager.initialize()
[12/12/2016 15:15:59][INFO][PROGRAM]  CoreManager.plug()
[12/12/2016 15:15:59][INFO][PROGRAM]  CoreManager.start()
[12/12/2016 15:15:59][INFO][PROGRAM]  UserReportGenerator.start()
[12/12/2016 15:16:00][INFO][DATA]  Starting the plugin: S7CodeGenerator.
[12/12/2016 15:16:00][INFO][PROGRAM]  GenerateAction
[12/12/2016 15:16:00][INFO][PROGRAM]  Generation has started
[12/12/2016 15:16:02][INFO][PROGRAM]   Starting the ValidationProcessor: process application general check rules.
[12/12/2016 15:16:02][SEVERE][PROGRAM]  Only one type of Fast Interlock can be selected. Choose either hardware interrupt or cyclic interrupt
for all fast interlock objects.
[12/12/2016 15:16:02][INFO][DATA]  Execution of standard check rules for - DigitalInput
[12/12/2016 15:16:02][INFO][DATA]  Execution of type-specific check rules for - DigitalInput
[12/12/2016 15:16:02][INFO][DATA]  Execution of standard check rules for - AnalogInput
[12/12/2016 15:16:03][INFO][DATA]  Execution of type-specific check rules for - AnalogInput
[12/12/2016 15:16:03][INFO][DATA]  Execution of standard check rules for - DigitalParameter
[12/12/2016 15:16:03][INFO][DATA]  Execution of type-specific check rules for - DigitalParameter
[12/12/2016 15:16:03][INFO][DATA]  Execution of standard check rules for - WordParameter
[12/12/2016 15:16:03][INFO][DATA]  Execution of type-specific check rules for - WordParameter
```

Fig. 1. Fast interlock error in UAB log

2.1. Files modified

This chapter contains a list of the files created or modified in the "Resources" folder.

- DeviceTypes (all files included in it)
- PlcParams
  - Siemens.xml
- S7InstanceGenerator
  - Config
    - CPC_TSPP_UNICOS_FI.scl
  - Rules
    - GlobalTemplates
      - S7Inst_Communication_Template.py

            - S7Inst_Communication_Template.scl
            - S7Inst_Communication_Template_FI.scl
            - S7Inst_CompilationOB_Template.py
            - S7Inst_CompilationOB_Template.scl
            - S7Inst_TSPP_UNICOS_Template.py
        - TypeTemplates
            - S7Inst_DigitalAlarm_Template.py
            - S7Inst_DigitalAlarm_Template.scl
            - S7Inst_DigitalAlarm_Template_DB.scl
            - S7Inst_DigitalAlarm_Template_optimized.scl
            - S7Inst_DigitalInput_Template.py
            - S7Inst_DigitalInput_Template.scl
            - S7Inst_DigitalInput_Template_DB.scl
            - S7Inst_DigitalInput_Template_optimized.scl
            - S7Inst_DigitalOutput_Template.py
            - S7Inst_DigitalOutput_Template.scl
            - S7Inst_DigitalOutput_Template_blocks.scl
            - S7Inst_OnOff_Template.py
            - S7Inst_OnOff_Template.scl
            - S7Inst_OnOff_Template_DB.scl
- S7LogicGenerator
    - Rules
        - CommonTemplates
            - S7Logic_DefaultAlarms_Template.py
        - GlobalTemplates
            - S7Logic_FC_PCO_Logic_Template.py
- SemanticCheckRules
    - GlobalTemplates
        - ApplicationGeneralCheckRules.py
    - TypeTemplates
        - DigitalAlarm_SemanticCheckRules.py
        - DigitalInput_SemanticCheckRules.py
        - DigitalOutput_SemanticCheckRules.py
        - OnOff_SemanticCheckRules.py

- SharedTemplates
    - FI_Functions.py
- WinCCOAInstanceGenerator
    - Rules
        - TypeTemplates
            - WinCCOA_DigitalAlarm_Template.py
            - WinCCOA_DigitalInput_Template.py
            - WinCCOA_DigitalOutput_Template.py
            - WinCCOA_OnOff_Template.py

2.2. General functions

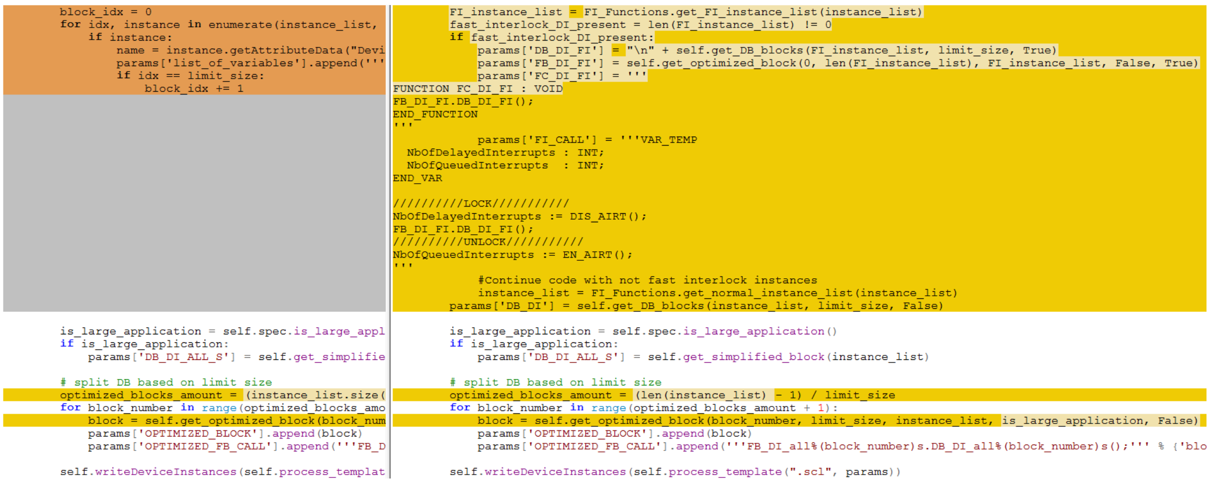
A couple of functions have been created to get the fast interlock and the normal instances from an instance list (“get_FI_instance_list” and “get_normal_instance_list” respectively) and added to a new file “FI_Functions.py” contained inside the “Shared_Templates” folder, which contains the files shared by all the templates.

2.3. Digital Input

The files used to generate the digital input SCL file have been modified to include the fast interlock code generation. The common code is still contained in the “S7Inst_DigitalInput_Template.scl”, whereas the code for the data blocks that are similar for normal and fast interlock objects has been included in the “S7Inst_DigitalInput_Template_DB.scl” file. The code for the function block, function and data block of the digital inputs is contained in the “S7Inst_DigitalInput_Template_optimized.scl” or in “S7Inst_DigitalInput_Template_simplified.scl” for the simplified objects (used when the application is defined as large).

The processing of these files is done in the “S7Inst_DigitalInput_Template.py” and includes the modification of the code for generating the normal or the fast interlock blocks. All these files are included in the “S7InstanceGenerator\Rules\TypeTemplates” folder.

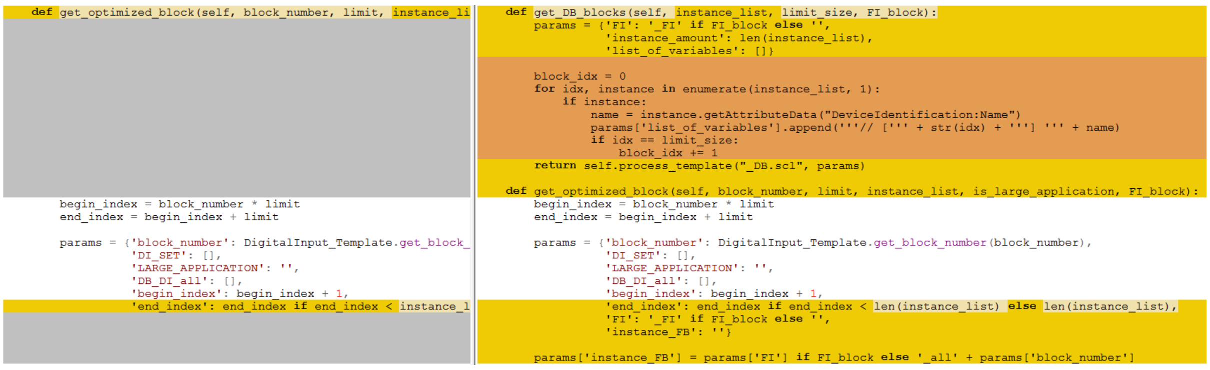
In Fig. 2 is shown an extract of the difference between the original and the new python template used to generate the SCL DI file. Note that the function “get_optimized_block” has a new parameter to indicate if the block is or not fast interlock. That’s why the function is called once for the fast interlock objects with that parameter to “True” and once for the non-fast interlock objects with that parameter to “False”. The “get_optimized_block” function with the new parameter is shown in Fig. 3.



```
        block_idx = 0
        for idx, instance in enumerate(instance_list,
            if instance:
                name = instance.getAttributeData("Devi
                params['list_of_variables'].append('''
                if idx == limit_size:
                    block_idx += 1

        is_large_application = self.spec.is_large_appl
        if is_large_application:
            params['DB_DI_ALL_S'] = self.get_simplifie

        # split DB based on limit size
        optimized_blocks_amount = (instance_list.size(
        for block_number in range(optimized_blocks_amo
            block = self.get_optimized_block(block_num
            params['OPTIMIZED_BLOCK'].append(block)
            params['OPTIMIZED_FB_CALL'].append('''FB_D

        self.writeDeviceInstances(self.process_templat
```

```
        FI_instance_list = FI_Functions.get_FI_instance_list(instance_list)
        fast_interlock_DI_present = len(FI_instance_list) != 0
        if fast_interlock_DI_present:
            params['DB_DI_FI'] = "\n" + self.get_DB_blocks(FI_instance_list, limit_size, True)
            params['FB_DI_FI'] = self.get_optimized_block(0, len(FI_instance_list), FI_instance_list, False, True)
            params['FC_DI_FI'] = '''
FUNCTION FC_DI_FI : VOID
FB_DI_FI.DB_DI_FI();
END_FUNCTION
'''
            params['FI_CALL'] = '''VAR_TEMP
  NbOfDelayedInterrupts : INT;
  NbOfQueuedInterrupts  : INT;
END_VAR

//////////LOCK///////////
NbOfDelayedInterrupts := DIS_AIRT();
FB_DI_FI.DB_DI_FI();
//////////UNLOCK///////////
NbOfQueuedInterrupts := EN_AIRT();
'''
            #Continue code with not fast interlock instances
            instance_list = FI_Functions.get_normal_instance_list(instance_list)
        params['DB_DI'] = self.get_DB_blocks(instance_list, limit_size, False)

        is_large_application = self.spec.is_large_application()
        if is_large_application:
            params['DB_DI_ALL_S'] = self.get_simplified_block(instance_list)

        # split DB based on limit size
        optimized_blocks_amount = (len(instance_list) - 1) / limit_size
        for block_number in range(optimized_blocks_amount + 1):
            block = self.get_optimized_block(block_number, limit_size, instance_list, is_large_application, False)
            params['OPTIMIZED_BLOCK'].append(block)
            params['OPTIMIZED_FB_CALL'].append('''FB_DI_all%(block_number)s.DB_DI_all%(block_number)s();''' % {'blo

        self.writeDeviceInstances(self.process_template(".scl", params))
```

Fig. 2. Difference between original (left) and new (right) implementation of the python file used to generate the digital input (DI) SCL file



```
    def get_optimized_block(self, block_number, limit, instance_li
        begin_index = block_number * limit
        end_index = begin_index + limit

        params = {'block_number': DigitalInput_Template.get_block_
                  'DI_SET': [],
                  'LARGE_APPLICATION': '',
                  'DB_DI_all': [],
                  'begin_index': begin_index + 1,
                  'end_index': end_index if end_index < instance_l
```

```
    def get_DB_blocks(self, instance_list, limit_size, FI_block):
        params = {'FI': '_FI' if FI_block else '',
                  'instance_amount': len(instance_list),
                  'list_of_variables': []}

        block_idx = 0
        for idx, instance in enumerate(instance_list, 1):
            if instance:
                name = instance.getAttributeData("DeviceIdentification:Name")
                params['list_of_variables'].append('''// [''' + str(idx) + '''] ''' + name)
                if idx == limit_size:
                    block_idx += 1
        return self.process_template("_DB.scl", params)

    def get_optimized_block(self, block_number, limit, instance_list, is_large_application, FI_block):
        begin_index = block_number * limit
        end_index = begin_index + limit

        params = {'block_number': DigitalInput_Template.get_block_number(block_number),
                  'DI_SET': [],
                  'LARGE_APPLICATION': '',
                  'DB_DI_all': [],
                  'begin_index': begin_index + 1,
                  'end_index': end_index if end_index < len(instance_list) else len(instance_list),
                  'FI': '_FI' if FI_block else '',
                  'instance_FB': ''}

        params['instance_FB'] = params['FI'] if FI_block else '_all' + params['block_number']
```

Fig. 3. Difference between original (left) and new (right) implementation of the “get_optimized_block” function and the new “get_DB_blocks” in the python file used to generate the digital input (DI) SCL file

2.4. Digital Output

Same modifications have been done for the digital output code as for the digital input one, with the exception that no simplified nor optimized blocks are needed, so the whole blocks that are similar for fast interlock and normal processing are included in the “S7Inst_DigitalOutput_Template_blocks.scl” file.

2.5. Digital Alarms

Same modifications have been done for the digital alarm code as for the digital input one, with the exception that no simplified blocks are needed.

2.6. OnOff

The modification of the code for the OnOff is similar to the one of the digital input in the sense that the common code and the data blocks that are similar are processed the same way. The different part, which is the function that calls the objects and that are not included in a multiinstance DB but in one DB for each object, is created inside the python file “S7Inst_OnOff_Template.py” as plain text. As such, that file has been modified as to include the function used to call the fast interlock objects.

2.7. Communication

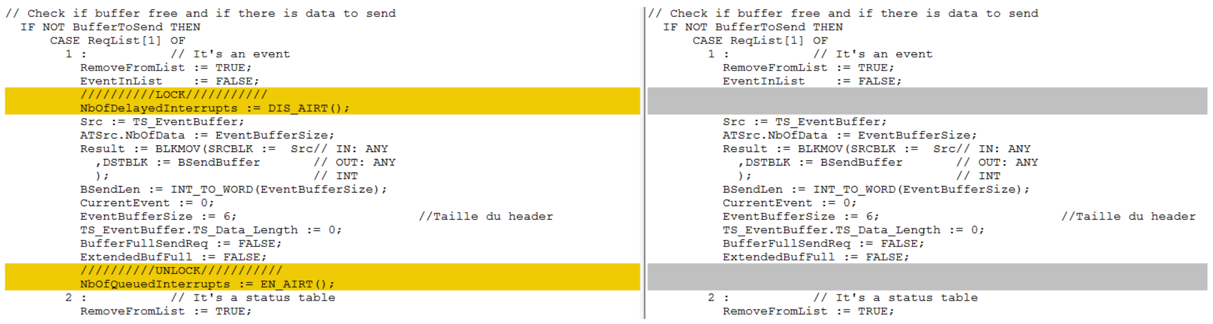
The files to generate the communication file have been modified as to include the new code necessary for the fast interlock treatment. A new file S7Inst_Communication_Template_FI.scl has been included as to generate the code of the blocks used for the fast interlock processing (“DB_FIEvent” and “FC_Event_FI”). The files “S7Inst_Communication_Template.scl” and “S7Inst_Communication_Template.py” have also been modified as to include the fast interlock code in the generated SCL file and to modify the normal blocks as to include the code necessary for the fast interlock objects processing in the normal PLC processing and the data blocks of the fast interlock objects in the normal TSPP processing.

The files for the communication file generation are included in the “S7InstanceGenerator\Rules\GlobalTemplates” folder.

2.8. TSPP

The file “S7Inst_TSPP_UNICOS_Template.py”, which is included in the “S7InstanceGenerator\Rules\GlobalTemplates” folder, has been modified as to select the “CPC_TSPP_UNICOS_FI.scl” file instead of the “CPC_TSPP_UNICOS.scl” file whenever a fast interlock is present in the project. It also includes now the amount of status tables used inside the fast interlock processing depending on the objects used (although it should always be 4 more status tables if a fast interlock is present). The new file “CPC_TSPP_UNICOS_FI.scl” included in the “S7InstanceGenerator\Config” folder has the new function block and data blocked defined as to support the TSPP treatment for the fast interlocks.

A sample of the interrupts established in the new “CPC_TSPP_UNICOS_FI.scl” file as to avoid the concurrency issues in shown in Fig. 4.



```
// Check if buffer free and if there is data to send
  IF NOT BufferToSend THEN
      CASE ReqList[1] OF
        1 :           // It's an event
          RemoveFromList := TRUE;
          EventInList    := FALSE;
          //////////LOCK///////////
          NbOfDelayedInterrupts := DIS_AIRT();
          Src := TS_EventBuffer;
          ATSrc.NbOfData := EventBufferSize;
          Result := BLKMOV(SRCBLK :=  Src// IN: ANY
            ,DSTBLK := BSendBuffer       // OUT: ANY
            );                           // INT
          BSendLen := INT_TO_WORD(EventBufferSize);
          CurrentEvent := 0;
          EventBufferSize := 6;                         //Taille du header
          TS_EventBuffer.TS_Data_Length := 0;
          BufferFullSendReq := FALSE;
          ExtendedBufFull := FALSE;
          //////////UNLOCK///////////
          NbOfQueuedInterrupts := EN_AIRT();
        2 :           // It's a status table
          RemoveFromList := TRUE;
```

```
// Check if buffer free and if there is data to send
  IF NOT BufferToSend THEN
      CASE ReqList[1] OF
        1 :           // It's an event
          RemoveFromList := TRUE;
          EventInList    := FALSE;


          Src := TS_EventBuffer;
          ATSrc.NbOfData := EventBufferSize;
          Result := BLKMOV(SRCBLK :=  Src// IN: ANY
            ,DSTBLK := BSendBuffer       // OUT: ANY
            );                           // INT
          BSendLen := INT_TO_WORD(EventBufferSize);
          CurrentEvent := 0;
          EventBufferSize := 6;                         //Taille du header
          TS_EventBuffer.TS_Data_Length := 0;
          BufferFullSendReq := FALSE;
          ExtendedBufFull := FALSE;


        2 :           // It's a status table
          RemoveFromList := TRUE;
```

Fig. 4. Difference between “CPC_TSPP_UNICOS.scl” (right) and “CPC_TSPP_UNICOS_FI.scl” (left) files.

2.9. OB Compilation

The files used for the generation of the organization blocks and some other type of blocks have been modified as to include the new organization block used to treat the fast interlock (either OB 40 for hardware interrupts or OB 34 for cyclic interrupts) and the function, function block and data block used inside them for the peripheral updates.

Although it’s necessary to have the full chain of objects for the processing of the fast interlocks, the check of existence of at least a fast interlock object of each kind is done as to add the function call inside the OB. That’s done that way in concordance with the generation of the OB 1, which is the reference that has been used as to create these OBs.

The files used for the OB compilation file generation are included in the “S7InstanceGenerator\Rules\GlobalTemplates” folder.

2.10. Logic

The “S7Logic_FC_PCO_Logic_Template.py” used as to generate the code of the “FC_PCO_Logic” where the logic of the UNICOS processing is called, both for the PCO objects and the field ones, has been modified as to include the new “FC_FI_LOGIC” function where just the fast interlock logic processing is done.

The file mentioned is included in the “S7LogicGenerator\Rules\GlobalTemplates” folder.

The alarm logic processing is generated with the use of the “S7Logic_DefaultAlarms_Template.py” contained in the “S7LogicGenerator\Rules\CommonTemplates” folder. The file has been modified as to include the fast interlock alarms which are contained in a different data block (“DB_DA_FI.DA_SET”) than the ones implemented already.

2.11. WINCC OA

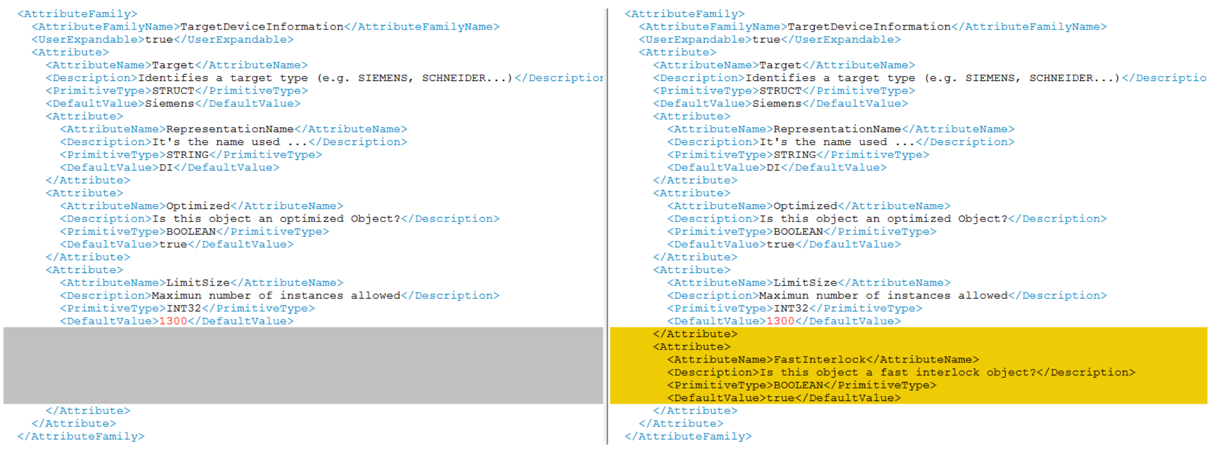
The different files for generating the wincc_oa_db_file_"project name".txt file used to create the WINCC OA project are included in the "WinCCOAInstanceGenerator". The files that have been modified are the ones included in the "Rules\TypeTemplates" inside that folder which correspond to the objects used inside the fast interlock processing (for example, "WinCCOA_DigitalInput_Template.py" for the digital input processing). The files have been modified as to call the "getAddressSCADA" function included in the plugin with the proper object name ("$Alias$_StsReg01_FI" for the fast interlock status register 01 of the object called "Alias") where necessary.

2.12. Specifications File Generation

The specifications file generation is done with the xml files described in the "Resources\DeviceType" folder for the different UNICOS objects. This generation is done with the use of the baseline code, so the modification needs to be implemented inside the UAB as to be tried.

The files contained in the directory mentioned (for example, "DigitalInputDeviceType.xml" for digital input) were modified as to include a new column ("Fast Interlock Type", included in the "LogicDeviceDefinitions attributefamily") where the fast interlock objects are indicated by selecting the type of interrupt that is going to be used ("Hardware Interrupt" or "Cyclic Interrupt"). These files are also used later by the UAB as to check the proper data filling inside the excel file.

They also have been modified as to indicate whether the objects are used or not in the fast interlock processing (for the symbols generation), as shown in Fig. 5 for digital inputs.



```
<AttributeFamily>
  <AttributeFamilyName>TargetDeviceInformation</AttributeFamilyName>
  <UserExpandable>true</UserExpandable>
  <Attribute>
    <AttributeName>Target</AttributeName>
    <Description>Identifies a target type (e.g. SIEMENS, SCHNEIDER...)</Descriptior
    <PrimitiveType>STRUCT</PrimitiveType>
    <DefaultValue>Siemens</DefaultValue>
    <Attribute>
      <AttributeName>RepresentationName</AttributeName>
      <Description>It's the name used ...</Description>
      <PrimitiveType>STRING</PrimitiveType>
      <DefaultValue>DI</DefaultValue>
    </Attribute>
    <Attribute>
      <AttributeName>Optimized</AttributeName>
      <Description>Is this object an optimized Object?</Description>
      <PrimitiveType>BOOLEAN</PrimitiveType>
      <DefaultValue>true</DefaultValue>
    </Attribute>
    <Attribute>
      <AttributeName>LimitSize</AttributeName>
      <Description>Maximun number of instances allowed</Description>
      <PrimitiveType>INT32</PrimitiveType>
      <DefaultValue>1300</DefaultValue>
    </Attribute>
  </Attribute>
</AttributeFamily>
```

```
<AttributeFamily>
  <AttributeFamilyName>TargetDeviceInformation</AttributeFamilyName>
  <UserExpandable>true</UserExpandable>
  <Attribute>
    <AttributeName>Target</AttributeName>
    <Description>Identifies a target type (e.g. SIEMENS, SCHNEIDER...)</Descriptio
    <PrimitiveType>STRUCT</PrimitiveType>
    <DefaultValue>Siemens</DefaultValue>
    <Attribute>
      <AttributeName>RepresentationName</AttributeName>
      <Description>It's the name used ...</Description>
      <PrimitiveType>STRING</PrimitiveType>
      <DefaultValue>DI</DefaultValue>
    </Attribute>
    <Attribute>
      <AttributeName>Optimized</AttributeName>
      <Description>Is this object an optimized Object?</Description>
      <PrimitiveType>BOOLEAN</PrimitiveType>
      <DefaultValue>true</DefaultValue>
    </Attribute>
    <Attribute>
      <AttributeName>LimitSize</AttributeName>
      <Description>Maximun number of instances allowed</Description>
      <PrimitiveType>INT32</PrimitiveType>
      <DefaultValue>1300</DefaultValue>
    </Attribute>
    <Attribute>
      <AttributeName>FastInterlock</AttributeName>
      <Description>Is this object a fast interlock object?</Description>
      <PrimitiveType>BOOLEAN</PrimitiveType>
      <DefaultValue>true</DefaultValue>
    </Attribute>
  </Attribute>
</AttributeFamily>
```

Fig. 5. Difference between original (left) and new (right) implementation of the Digital Input Device Type file with fast interlock attribute

2.13. Semantic Rules

The files for checking the semantic rules have been modified as to check the new rules existent for the fast interlock treatment. The files are contained in the “Resources\SemanticCheckRules” folder, in the “GlobalTemplates” folder for global checking and in the “TypeTemplates” folder for the different UNICOS objects.

2.14. PLC Parameters

The symbols of the Step 7 program are generated with the use of the UnicosApplication.xml file contained in the main directory of any application. This file is generated at the UNICOS application creation making use of the correspondent file in the “Resources\PlcParams” folder (“Siemens.xml” for Siemens Step 7 applications).

As this file cannot be generated with the use of a modified “Siemens.xml” file from the program folder because it generates it in the program creation only, that file is modified and included in a jar file existent in the UAB installation folder for every cpc version. That file contains the resources for generating any project, including the Resources folder.

The functions, function blocks and data blocks that can be generated for each type of fast interlock object are shown in Fig. 6.

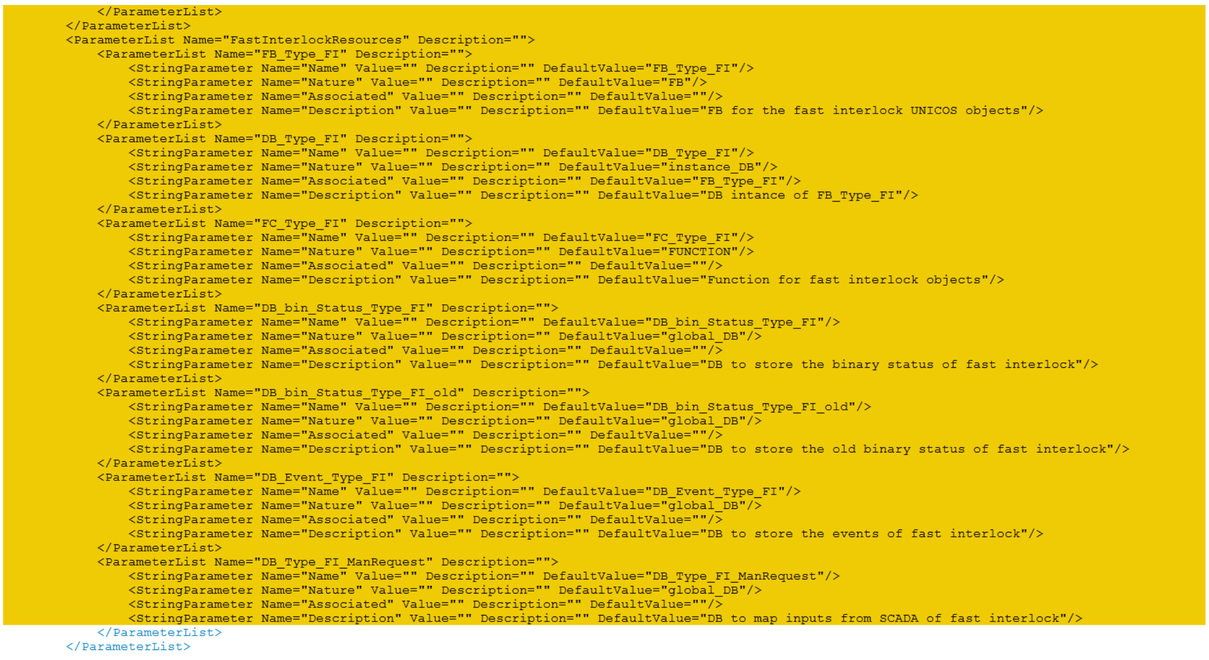
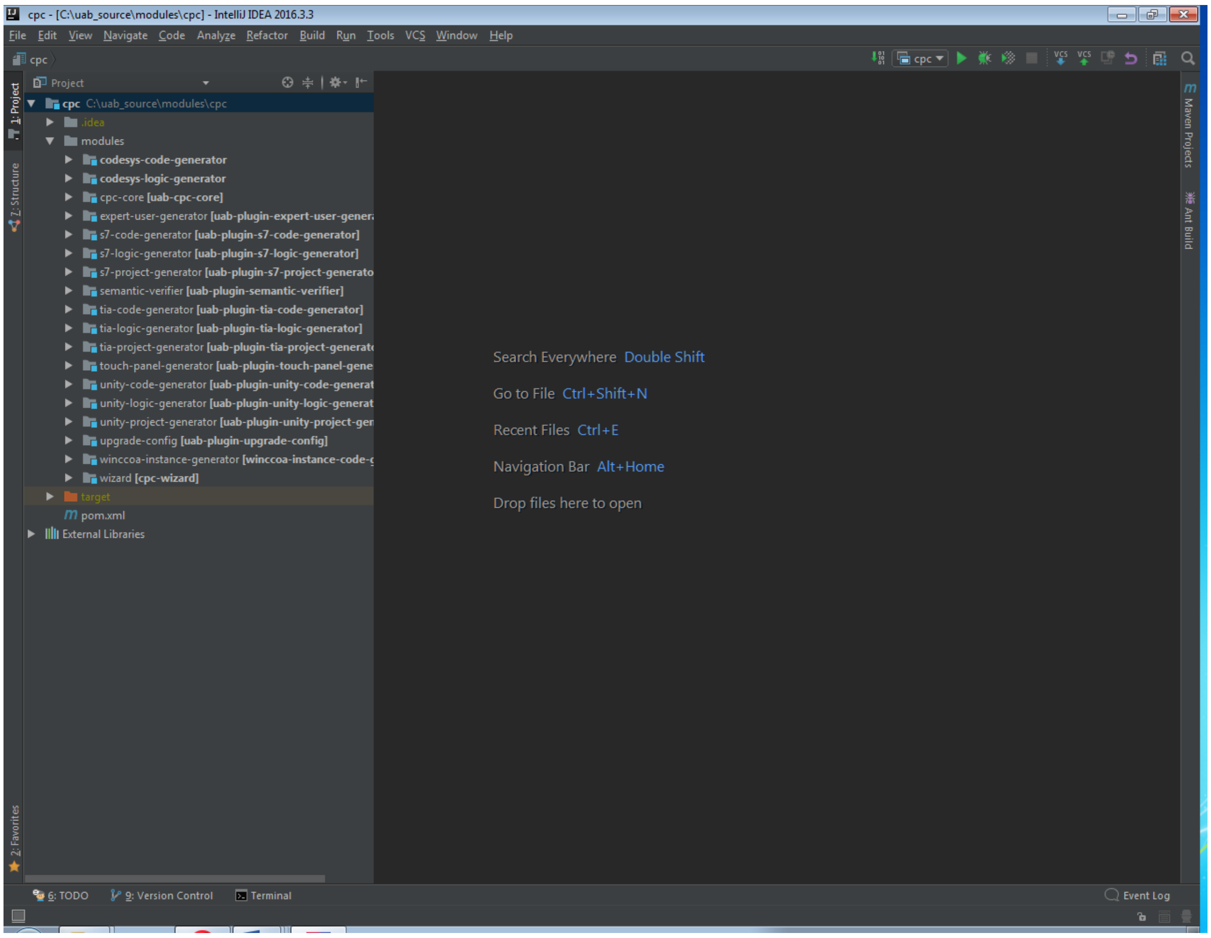

```
            </ParameterList>
        </ParameterList>
        <ParameterList Name="FastInterlockResources" Description="">
            <ParameterList Name="FB_Type_FI" Description="">
                <StringParameter Name="Name" Value="" Description="" DefaultValue="FB_Type_FI"/>
                <StringParameter Name="Nature" Value="" Description="" DefaultValue="FB"/>
                <StringParameter Name="Associated" Value="" Description="" DefaultValue=""/>
                <StringParameter Name="Description" Value="" Description="" DefaultValue="FB for the fast interlock UNICOS objects"/>
            </ParameterList>
            <ParameterList Name="DB_Type_FI" Description="">
                <StringParameter Name="Name" Value="" Description="" DefaultValue="DB_Type_FI"/>
                <StringParameter Name="Nature" Value="" Description="" DefaultValue="instance_DB"/>
                <StringParameter Name="Associated" Value="" Description="" DefaultValue="FB_Type_FI"/>
                <StringParameter Name="Description" Value="" Description="" DefaultValue="DB intance of FB_Type_FI"/>
            </ParameterList>
            <ParameterList Name="FC_Type_FI" Description="">
                <StringParameter Name="Name" Value="" Description="" DefaultValue="FC_Type_FI"/>
                <StringParameter Name="Nature" Value="" Description="" DefaultValue="FUNCTION"/>
                <StringParameter Name="Associated" Value="" Description="" DefaultValue=""/>
                <StringParameter Name="Description" Value="" Description="" DefaultValue="Function for fast interlock objects"/>
            </ParameterList>
            <ParameterList Name="DB_bin_Status_Type_FI" Description="">
                <StringParameter Name="Name" Value="" Description="" DefaultValue="DB_bin_Status_Type_FI"/>
                <StringParameter Name="Nature" Value="" Description="" DefaultValue="global_DB"/>
                <StringParameter Name="Associated" Value="" Description="" DefaultValue=""/>
                <StringParameter Name="Description" Value="" Description="" DefaultValue="DB to store the binary status of fast interlock"/>
            </ParameterList>
            <ParameterList Name="DB_bin_Status_Type_FI_old" Description="">
                <StringParameter Name="Name" Value="" Description="" DefaultValue="DB_bin_Status_Type_FI_old"/>
                <StringParameter Name="Nature" Value="" Description="" DefaultValue="global_DB"/>
                <StringParameter Name="Associated" Value="" Description="" DefaultValue=""/>
                <StringParameter Name="Description" Value="" Description="" DefaultValue="DB to store the old binary status of fast interlock"/>
            </ParameterList>
            <ParameterList Name="DB_Event_Type_FI" Description="">
                <StringParameter Name="Name" Value="" Description="" DefaultValue="DB_Event_Type_FI"/>
                <StringParameter Name="Nature" Value="" Description="" DefaultValue="global_DB"/>
                <StringParameter Name="Associated" Value="" Description="" DefaultValue=""/>
                <StringParameter Name="Description" Value="" Description="" DefaultValue="DB to store the events of fast interlock"/>
            </ParameterList>
            <ParameterList Name="DB_Type_FI_ManRequest" Description="">
                <StringParameter Name="Name" Value="" Description="" DefaultValue="DB_Type_FI_ManRequest"/>
                <StringParameter Name="Nature" Value="" Description="" DefaultValue="global_DB"/>
                <StringParameter Name="Associated" Value="" Description="" DefaultValue=""/>
                <StringParameter Name="Description" Value="" Description="" DefaultValue="DB to map inputs from SCADA of fast interlock"/>
            </ParameterList>
        </ParameterList>
```

Fig. 6. New PLC params for each kind of object

3. Plugin code

Some of the files containing the code of the plugin used in UAB have been modified and are described in this section. The changes of these files have been tested with the use of the IntelliJ software [1], with the proper configuration as to be able to execute the cpc wizard with the local data of the computer, so that the changes are not directly included in the live version.

A screenshot of the IntelliJ application is shown in Fig. 7.



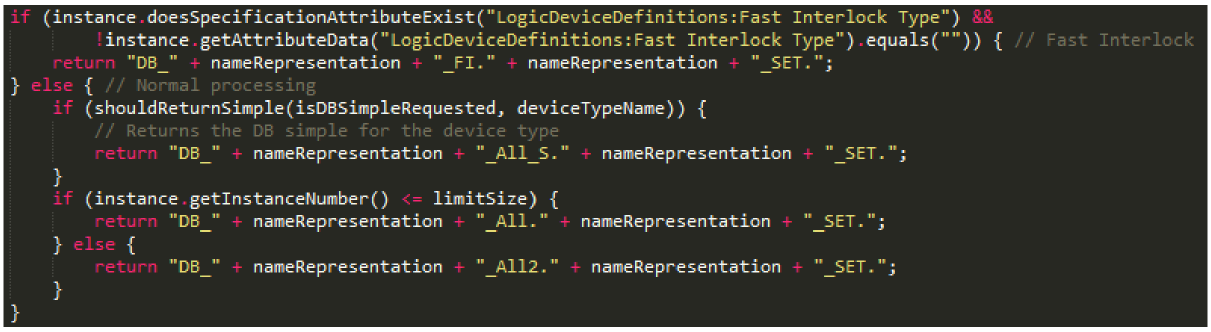
cpc - [C:\uab_source\modules\cpc] - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
cpc
Project
1: Project
Z: Structure
2: Favorites
Maven Projects
Ant Build
6: TODO
9: Version Control
Terminal
Event Log

```
cpc C:\uab_source\modules\cpc
  .idea
  modules
    codesys-code-generator
    codesys-logic-generator
    cpc-core [uab-cpc-core]
    expert-user-generator [uab-plugin-expert-user-gener
    s7-code-generator [uab-plugin-s7-code-generator]
    s7-logic-generator [uab-plugin-s7-logic-generator]
    s7-project-generator [uab-plugin-s7-project-generato
    semantic-verifier [uab-plugin-semantic-verifier]
    tia-code-generator [uab-plugin-tia-code-generator]
    tia-logic-generator [uab-plugin-tia-logic-generator]
    tia-project-generator [uab-plugin-tia-project-generat
    touch-panel-generator [uab-plugin-touch-panel-gene
    unity-code-generator [uab-plugin-unity-code-generat
    unity-logic-generator [uab-plugin-unity-logic-generat
    unity-project-generator [uab-plugin-unity-project-ger
    upgrade-config [uab-plugin-upgrade-config]
    winccoa-instance-generator [winccoa-instance-code-g
    wizard [cpc-wizard]
  target
  pom.xml
External Libraries
```

```
Search Everywhere  Double Shift
Go to File  Ctrl+Shift+N
Recent Files  Ctrl+E
Navigation Bar  Alt+Home
Drop files here to open
```

Fig. 7. IntelliJ configured for the cpc wizard execution

The function “s7db_id” contained in the file “S7Functions.java”, and included in the package “research.ch.cern.unicos.cpc.utilities.siemens” has been modified as to return the proper DB for the fast interlock objects of the different kinds (for example, “DB_DI_FI.DI_SET” for the digital inputs) instead of the normal ones. An extract of the code is shown in Fig. 8.

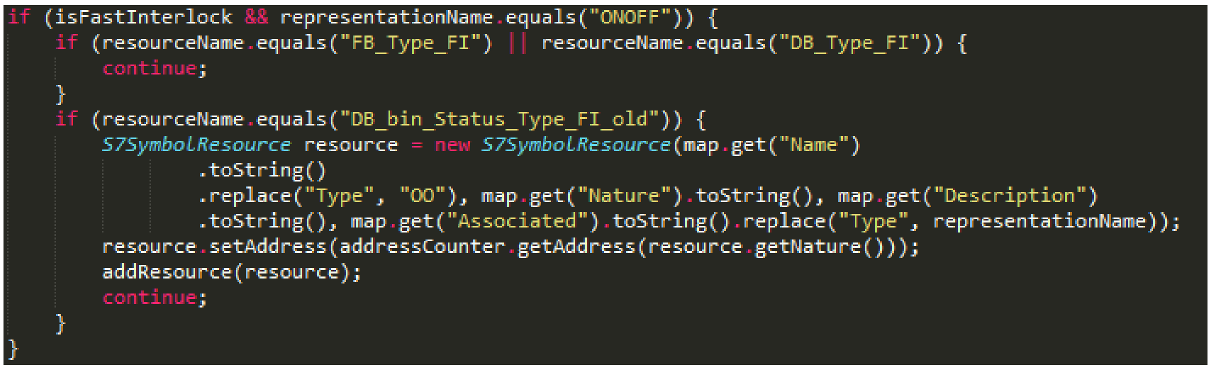


```
if (instance.doesSpecificationAttributeExist("LogicDeviceDefinitions:Fast Interlock Type") &&
        !instance.getAttributeData("LogicDeviceDefinitions:Fast Interlock Type").equals("")) { // Fast Interlock
    return "DB_" + nameRepresentation + "_FI." + nameRepresentation + "_SET.";
} else { // Normal processing
    if (shouldReturnSimple(isDBSimpleRequested, deviceTypeName)) {
        // Returns the DB simple for the device type
        return "DB_" + nameRepresentation + "_All_S." + nameRepresentation + "_SET.";
    }
    if (instance.getInstanceNumber() <= limitSize) {
        return "DB_" + nameRepresentation + "_All." + nameRepresentation + "_SET.";
    } else {
        return "DB_" + nameRepresentation + "_All2." + nameRepresentation + "_SET.";
    }
}
```

Fig. 8. Extract of the "s7db_id" function used in the plugin modified for the fast interlock DBs

The function "createFullAddressMap" contained in the file "SiemensPLCMemoryMapper.java" and included in the package "research.ch.cern.unicos.cpc.utilities.siemens" has been modified as to return the proper DB number for the fast interlock objects that is used as to generate the WINCC OA file in UAB.

The function "createDeviceInstanceResources" of the same file has also been modified as to generate the symbols of the device types used in the fast interlock processing (taking care of the OnOff differences with the rest of the objects used in the fast interlocks, as shown in Fig. 9).



```
if (isFastInterlock && representationName.equals("ONOFF")) {
    if (resourceName.equals("FB_Type_FI") || resourceName.equals("DB_Type_FI")) {
        continue;
    }
    if (resourceName.equals("DB_bin_Status_Type_FI_old")) {
        S7SymbolResource resource = new S7SymbolResource(map.get("Name")
                .toString()
                .replace("Type", "OO"), map.get("Nature").toString(), map.get("Description")
                .toString(), map.get("Associated").toString().replace("Type", representationName));
        resource.setAddress(addressCounter.getAddress(resource.getNature()));
        addResource(resource);
        continue;
    }
}
```

Fig. 9. Extract of the "createDeviceInstanceResources" function used in the plugin modified for the fast interlock symbols

4. 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

5. 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.

6. Bibliography

[1] Jet Beans, “IntelliJ IDEA,” [Online]. Available: https://www.jetbrains.com/idea/. [Accessed 24 January 2017].