Document Title:	BakeOut documentation
Project Name:	
Description:	An documentation about the bakeout system at GSI
Division/Organization:	ACO-IND
Field of application:	Project FAIR@GSI

Template History:

Version	Prepared by:	Checked, Date	Released, Date	Comment
V 1.0				

Document History:

Version	Prepared by:	Checked, Date	Released, Date	Comment
V 1.0				

TABLE OF CONTENTS

1. Introduction

1.1. Purpose

The purpose of this document is to describe the hardware and software architecture of the bakeout system that is used for CryRing project and for SIS100 / HEBT at FAIR

1.2. Abbreviations

Abbreviation	Description
CERN	European Organization for Nuclear Research Organization
UNICOS	UNified Industrial Control System
UAB	UNICOS Application Builder
TC	Thermocouple
PID	Proportional Integral Derivative Controller
PLC	Programmable Logic Controller
HW	Hardware
PROFINET	Process Field Net
GRAPH	Graphical Programming Language
PWM	Pulse Width Modulation
GUI	Graphical User Interface

Table 1: Abbreviations

1.3. References

Reference	Revision
[1] https://fair-center.eu/	
[2] https://unicos.web.cern.ch/	
[3] BO_PLC_01_v1_Output BO_PLC_01_v1_Output.xlsx	V1.0

Table 2: References

2. System Overview

The bakeout system used for CryRing, SIS100 and HEBT project, which is part of FAIR [1].

The main purpose is to create a special heating procedure that releases impurities and other contaminants, from components in a closed system which enables that high vacuum can be reached.

It possible to create a maximum configuration for bakeout projects of max 3 SSR racks. In every SSR rack can be supplied with 6x3 terminal boxes. Every terminal box can be connected to 8 PID circles.

2.1. Terminal box

All terminal boxes will be permanently installed at the frames of the accelerator magnets where all connections the terminal boxes such as sensors and heating jackets will be done permanently. Terminal boxes are mounted on the right and left side under the Dipole magnet. Each terminal box will provide all required connection points for 24 (or min. 18) heater channels where each heating channel will be equipped with one heating jacket, one thermocouple (TC) and min. one thermoclick/thermoswitch. Additionally, each terminal box provides one internal PT100/PT1000 in order to measure the inside temperature of terminal box, in order to execute the temperature compensation for the connected thermocouples.

Components that belong to one terminal box are listed in Table 3.

Quantity	Terminal Type	Description
1x Compensation TC	Compensation thermocouple	PT100 sensor for measuring the inside temperature of terminal box. Temperature is used as compensation temperature for the connected thermocouples. Compensation thermocouple sensor is connected to the analog input card. **Quantity:** 1x/terminal box
24x HCS	Thermocouple	PT100 sensor for temperature readout of the specific HTC Thermocouple sensor is connected to analog input card. **Quantity:** Up to 24x/terminal box
	Heating jacket	Heating jacket used for heating of the components. Heating jacket is connected to digital PWM output of the Siemens HCS4200 module. **Quantity:** Up to 24x/terminal box
	2x Thermoclick/thermoswitch	Thermoclick/thermoswitch used as an overheating protection. **Quantity:** Up to 24x/terminal box

Table 3: Terminal box components

2.2. Electrical Cabinet

The electrical cabinets are includes all PLC components and the powering of the cabinets. For every project, it is possible to create max 3 SSR racks. Due to the fact that the SCADA and Industrial Network connector is not connected on the slave cabinets this would lead that the PLC in slave cabinets is not powered so the master PLC has control over the other slave cabinets periphery. The examples of configuration with one electrical cabinet is shown in the following figure:

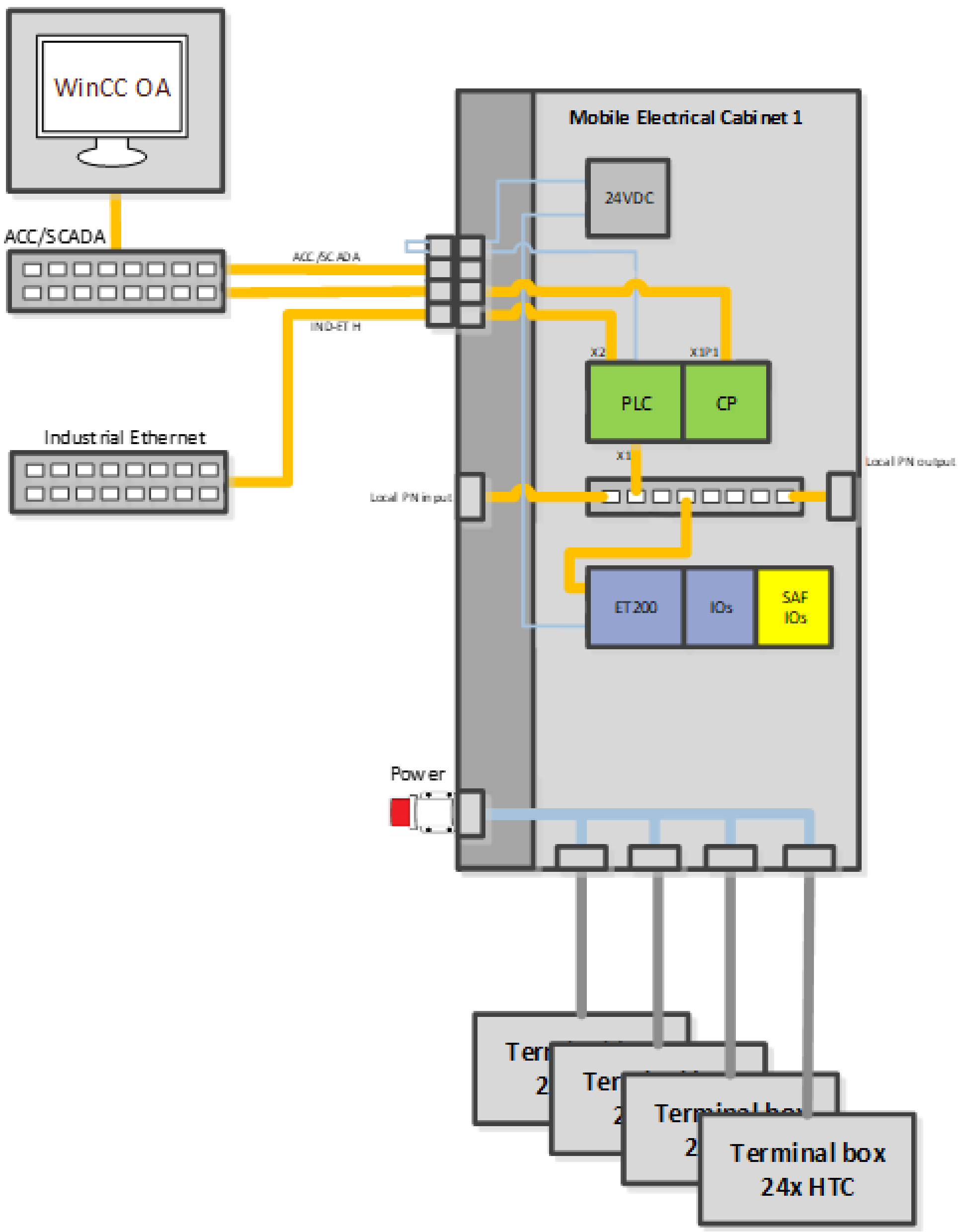

Figure 1: Electrical cabinet

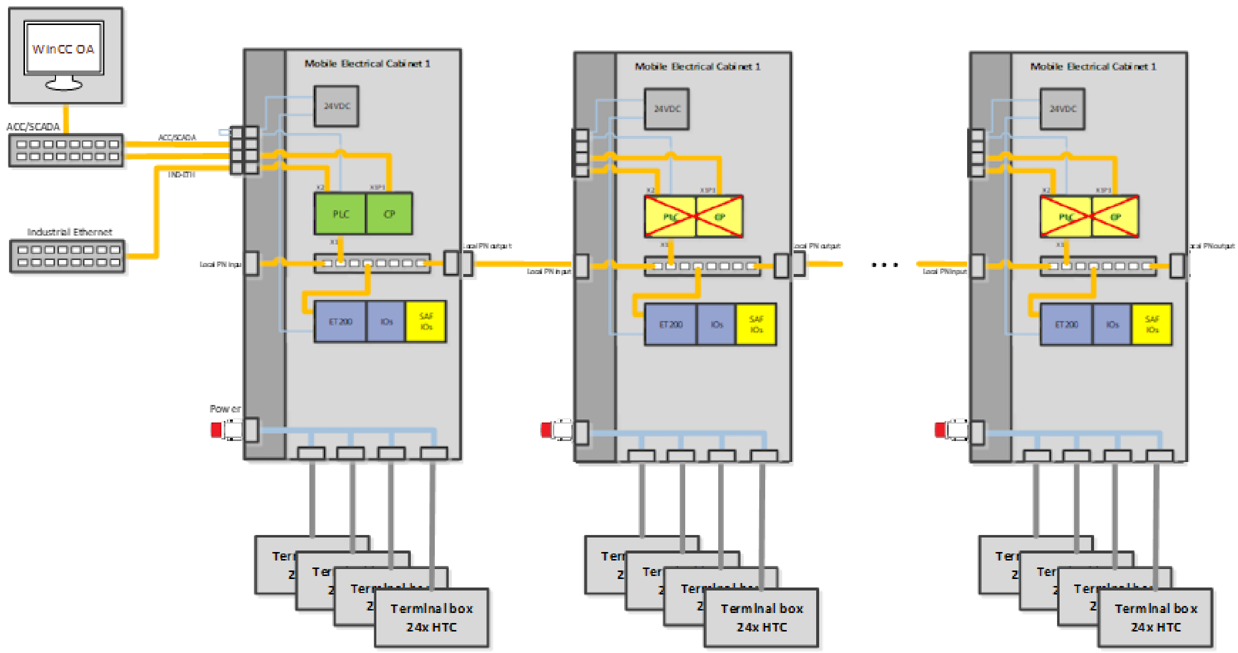

Figure 2: Multiple electrical cabinet

2.3. Hardware Used

The EC uses several different types of equipment, which communicate, over Siemens PROFINET interface.

EC hardware is composed of the following:

- Control PLC Unit
- Distributed IO Unit

2.4. Control PLC Unit

The PLC is the core of the system and it holds the logic needed for correct system operation.

The Siemens safety CPU together with the TIA portal automation software is used as the control unit of the HESR system. The upstream PLC connections are connected to SCADA and Industrial Network and its downstream connection to distributed IO units and HCS4200 units are achieved through PROFINET.

The control part consists of Siemens components, which are shown in Figure 3 and are listed below:

- Control PLC unit (1518F-4 PN/DP)
  - Type: 6ES7 518-4FP00-0AB0
  - Safety PLC used for normal and safety logic.
- Communication module (CP1543-1).
  - Type: 6GK7 543-1AX00-0XE0
  - Module that enables one additional port to the ACC/SCADA network.

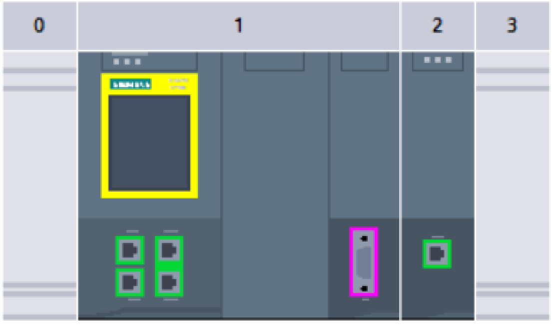

Figure 3: Control PLC Unit

2.5. Distributed IO Units

The bakeout have has a distributed architecture in which distributed IO units are connected to the main PLC over the PROFINET communication protocol.

Each distributed IO unit consist of a Siemens ET200SP coupler and several different types of modules that are shown in Figure 6 and are listed below:

- ET200SP coupler (IM 155-6 PN ST).
  - Type: 6ES7 155-6AU01-0BN0
  - Distributed IO unit that supports connection of the input/output modules.
- Digital input terminals (DI 16x24VDC ST).
  - Type: 6ES7 131-6BH01-0BA0
  - Used for reading thermoclick/thermoswitch, coding, circuit breaker...
- Analog input terminals (PT100; AI 8xRTD/TC 2-wire HF)
  - Type: 6ES7 134-6JF00-0CA1
  - Used for reading TCs.
- Analog input terminals (PT100; AI 4xRTD/TC 2-,3-,4-wire HF)
  - Type: 6ES7 134-6JD00-0CA1
  - Used for reading RTDs.
- Digital output terminals (DQ 16x24VDC/0.5A ST).
  - Type: 6ES7 132-6BH01-0BA0
  - Used for activating circuit breaker and for the LEDs.
- Safety digital input terminals (F-DI 8x24VDC HF).
  - Type: 6ES7 136-6BA00-0CA0
  - Used for reading E-Stop buttons, contactor feedback and E-Stop acknowledgment.
- Safety digital output terminals (F-DQ 4x24VDC/2A PM HF).
  - Type: 6ES7 136-6DB00-0CA0
  - Used for activating contactor power.
- Safety digital output terminals (F-DQ 8x24VDC/0.5A PP HF).
  - Type: 6ES7 136-6DC00-0CA0
  - Used as reserve.

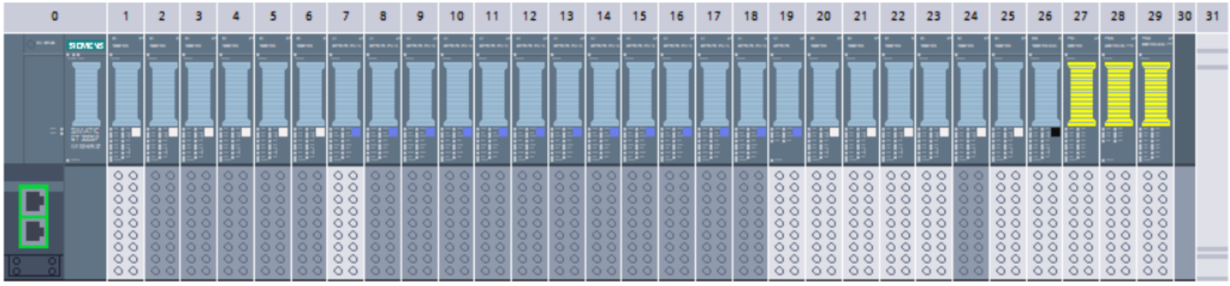

Table 4: ET200SP

2.6. Safety program

The safety consists of Stop buttons and safety actuators that are connected to safety I/O modules at the ET200SP units that is located on each terminal box.

The safety consists of:

- Stop buttons – Redundant Stop input buttons.
- Stop acknowledge buttons – Buttons for acknowledging Stop logic.
- Feedback from the contactors – Redundant feedback over which the safety logic is monitoring if the safety components are working correctly.

PLC gather the status of all Stop Buttons from all used ECs and when all buttons are ok unpressed and the Stop is acknowledged it triggers safety outputs, which enables the power contactors in every MEC used in the PLC project.

The example of the Stop Buttons that are located on each of the EC is shown in the Figure 4.

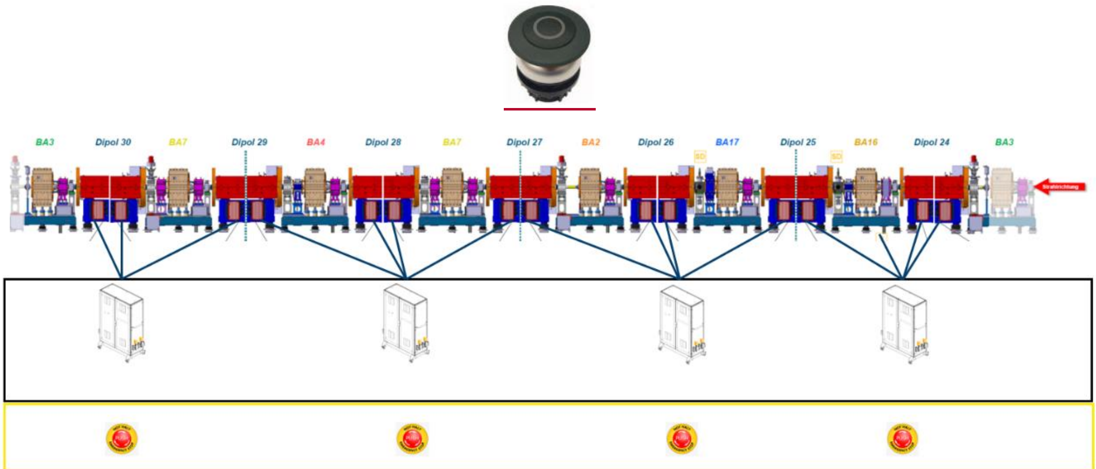

Figure 4: Safety Overview

3. Software architecture

The software is programmed in the UNICOS [2] framework developed by CERN and the tool used for generating sources for PLC and SCADA is called UAB.

The logic behind a bakeout can be nicely explained with the stepper machine that is used for every section as shown in Figure 5.

When operator start section PCO, stepper machine is started. It first goes to »Error check« step where its cheeks that all HTCs are error free and if not, the stepper machine is stopped. In case there is no error the stepper machine goes over 15 heating steps for which temperature profiles are prepared in advance by vacuum people. If the temperature profile uses e.g. 4 steps this would mean that after »Heating step 4« the stepper machine would be stopped.

In every heating step the reglation follows average setpoint value where some derogations are allowed. In case any value deviates too much the heating step jumps to »Error step«. If the error is removed and if the operator confirms the error in specific time the stepper machine will go into the »Recover step« otherwise the stepper machine will stop. The stepper machine moves from »Recover step« back to normal »Heating step« when measured values and the setpoint values for will be in the range of average measure value and average setpoint value calculated from the section. The logic is then repeated for all configured heating steps.

When the heating sequence is finished, the stepper machine sets the status »Heating finished« and stepper machine is stopped.

Stepper machine logic is shown in the following Figure 5.

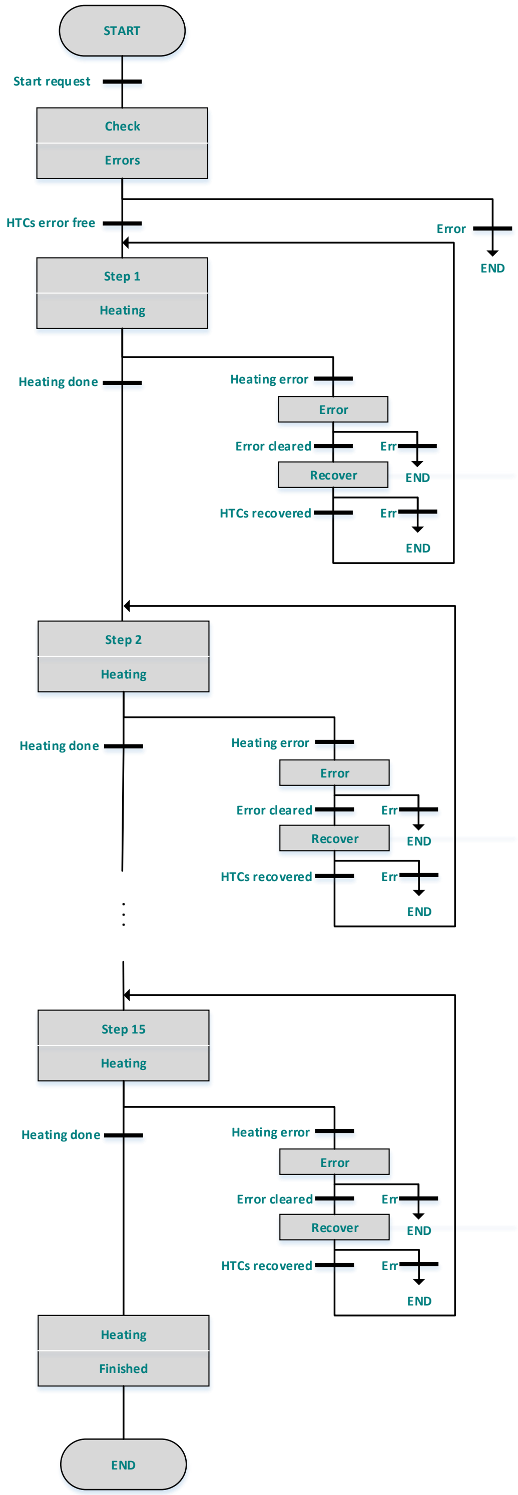

Figure 5: Section stepper machine

3.1. Project Generation

Sources are generated through UAB that uses input files (specFile, python scripts, TIA user sources, TIA Portal base project) which are used to generate output files (TIA instances, TIA logic, TIA Portal project, WinCC OA source file).

The overview of input files and generated output files that are needed/generated during UAB generation for bakeout project is shown on the Figure 6.

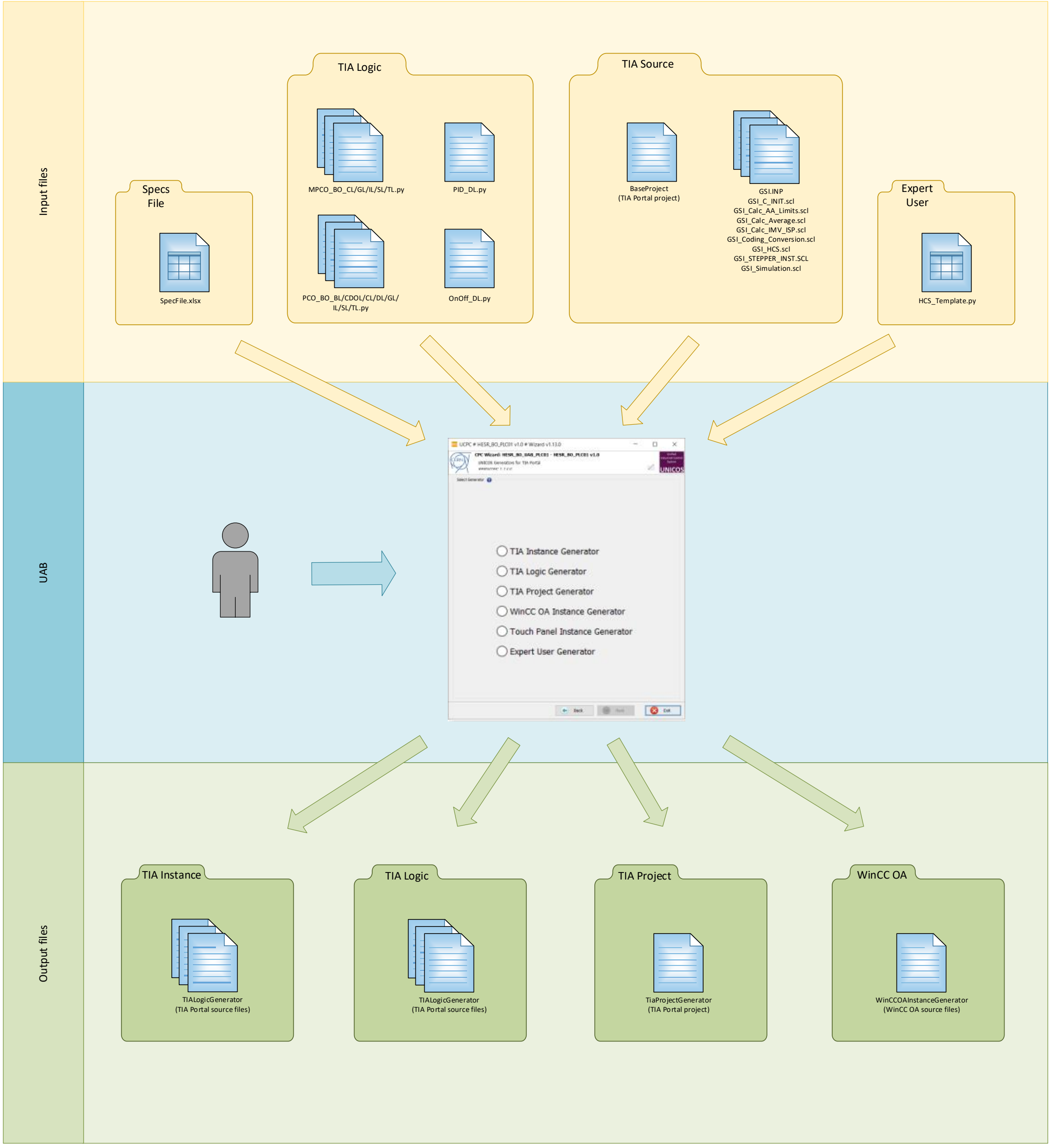

Figure 6: UAB project overview

The UAB software can be logically split into the following sections:

- Input Files
- UAB
- Output Files

3.2. Input Files

For the bakeout project generation input files need to be prepared which are then used during UAB project generation. The needed input files are listed below and explained in details in the following sections:

- Specification SpecFile where all UNICOS object and their relations are specified.
- Python Scripts – Python files needed for generating TIA Portal logic.
- User Sources – Predefined TIA Portal base project and files (.scl and .INP).
- Expert User – Expert user files needed for expert logic

3.3. Specification File

Specification file (specFile) is the UAB input file where the UNICOS objects, object parameters and their relations are specified. Due to the fact that specFile uses a lot of repetitive combinations of objects, "SpecFileCreator" tool is used in order to fill out specFile.

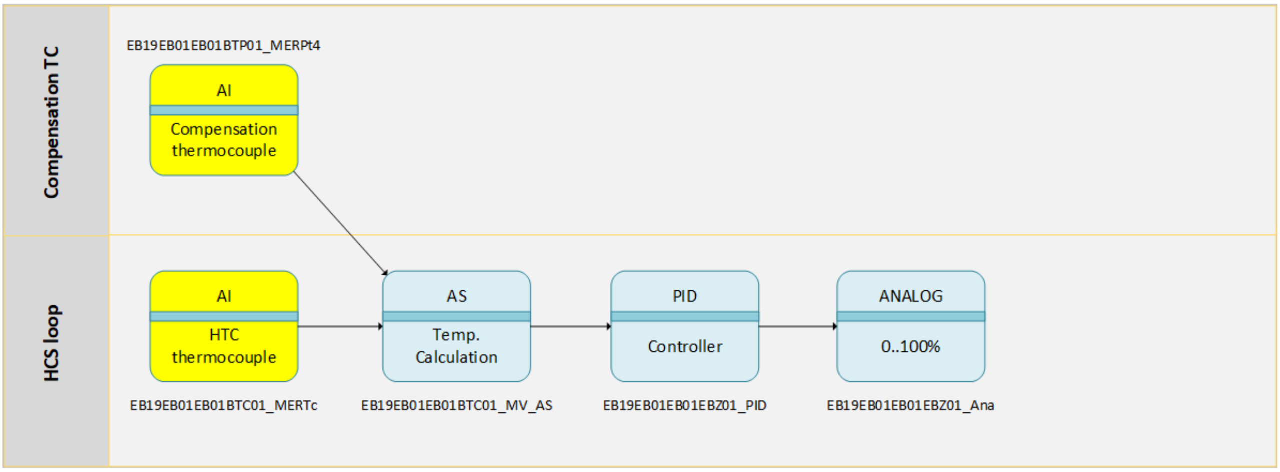

Figure 7: UNICOS objects

All object used for PLC01 project with their descriptions and other details can be found in specification file for PLC01

3.4. Python Scripts

The python scripts are in the library of BO_Library.py and located in:

…\UAB_Project\Resources\TIALogicGenerator\Rules\UserSpecific Resources\

The functions needs to be set from SpecFile:

Table 5: Python scripts for MPCO

Part of	Script name	Description
Master PCO	MPCO_CL(All Sec)CL	Master PCO configuration logic. The logic includes: • **Master PCO feedback On/Off** – Logic for setting master PCO feedback to "On" or "Off". • **Master PCO control stop finished** – Logic for setting master PCO control stop finished status.
	MPCO_GL(All Sec)GL	Master PCO global logic. The logic includes: • **Coding check** – Logic compares coding setpoint and actual values and set the digital outputs object (PLC01_M_UC_I_SPR, PLC01_S1_UC_I_SPR) of the mobile

		cabinets in case the coding is ok. Digital output is connected on the LED located on the mobile cabinet. • **HCS warning –** Logic checks if there is warning on any of the HCS channel of the HCS4200 system and set the digital input object in case the warning is active. • **Sector PCOs On/Off request** – Logic over which master PCO can trigger section PCO to “On” or “Off”.
	ilk Ext(All Cabinet)CL	Master PCO interlock logic. The logic includes: • **External Digital Alarm** – Logic adds conditions for external error object (PLC01_Ext_DA) as many MECs cabinet is configured in the project (NOT PLC01_M_AExt OR NOT PLC01_S1_AExt). • **Coding Digital Alarm** – Logic adds conditions for coding error object (PLC01_UC_DA) as many MECs cabinet is configured in project (NOT PLC01_M_UC_I_SPR OR NOT PLC01_S1_UC_I_SPR). • **E-Stop Digital Alarm** – Logic adds conditions for emergency stop error object (PLC01_ESTOP_DA) as many MECs cabinet is configured in project (NOT PLC01_M_FQ01FN02_ANoFail OR NOT PLC01_S1_FQ01FN02_ANoFail). • **Analog alarm limits (GSI_Calc_AA_Limits)** – Logic calls function for calculating warning and error limits for every HTC.
Section PCO	PCO_CL()CL	Section PCO configuration logic. The logic includes: • **Section PCO feedback On/Off** – Logic for setting section PCO feedback to “On” or “Off”. • **Section PCO control stop finished** – Logic for setting section PCO control stop finished status.
	PCO_GL(Sec Num)GL	Section PCO global logic. The logic includes: • **Transfer step values regarding step NR** – Logic transfers the values regarding in which heating step the heating sequence is currently in. The following values are transferred: “step time”, “NEG value”, “STD value”, “LOW value”, “MID value”, “H alarm limit” and “L alarm limit”. • **Step delay (FC_Cond_Delay)** – Call of the logic for checking how much time the heating sequence need to stay in certain step. The function takes “step time” setpoint from SCADA (EB19_TS_WS) and returns to SCADA status of remaining minutes (EB19_RMTS_WS) and status of remaining seconds (EB19_RTS_WS). • **Transfer statuses to SCADA** – Logic transfers the statuses of the heating sequence to the SCADA. The following statuses are transferred: “Heating step active”, “Recover step active”, “Error step active” and “Bakeout finished successfully”.
	PCO_IL(Sec name, All PIDs)IL	Section PCO interlock logic. The logic includes:

		• **Error status** – Logic checks all interlocks per section and transfer the section error status to digital input object (EB19_AErr). The error status is composed from the following interlocks: o "PLC01_Ext_DA" o And the following errors from the HTC: ▪ "EB19EB01EB01EBZ01_Diff_AA" ▪ "EB19EB01EB01EBZ01_Max_AA" ▪ "EB19EB01EB01EBZ01_TC_DA" • **Control stop after time out** – Logic for checking if there is any unacknowledged interlocks (interlocks listed under "Error status") in section and transfer the status to digital input object (EB19_TO). In case the timer for control stop elapsed the stepper machine will jump to the end. • **Maximal high limit (GSI_Calc_AA_Limits)** – Logic calculated warning and error limit for every controller in the section regarding HTC actual temperature value and belonging HTC group (NEG, STD, LOW, MID) setpoint.
	PCO_DL(All PIDs)DL	Section PCO dependent logic. The logic includes: • **Average value (GSI_Calc_Average)** – Logic calculates the average setpoint and average measured value regarding to which HTC group (NEG, STD, LOW, MID) group it belongs. • **Average SP** – Logic transfers the average PID controller setpoint to the belonging HTC group (NEG, STD, LOW, MID). The average setpoints are transferred to the following objects: "EB19_NEG_ASP", "EB19_STD_ASP_AS", "EB19_LOW_ASP_AS" and "EB19_MID_ASP_AS". • **Average MV** – Logic transfers the average PID controller measured values to the belonging HTC group (NEG, STD, LOW, MID). The average measured values are transferred to the following objects: "EB19_NEG_AMV", "EB19_STD_AMV_AS", "EB19_LOW_AMV_AS" and "EB19_MID_AMV_AS".
	instanceStp(Seq name)SL	Section PCO sequencer logic. The logic includes: • **Section stepper logic (Stp_EB19_Seq)** – Function call of the section stepper machine which is in advance prepared in the TIA base project.
	PCO_TL(All PIDs, Sec num)TL	Section PCO transition logic. The logic includes: • **Calculating IMV/ISP (GSI_Calc_IMV_ISP)** – Logic calculates the following digital input objects per HTC: o "EB19EB01EB01EBZ01_IMV_OK" indicates that the difference between PID controller measures value and average measured value is smaller than the required offset value ("EB19_SOff_AP").

		◦ "EB19EB01EB01EBZ01_ISP_OK" indicates that the difference between PID controller setpoint value and average setpoint values is smaller than "0.01". • **Step Condition** – Logic checks if we have conditions from all of the PID controllers to move to the next heating step. The logic checks for the following conditions: ◦ IMV and ISP status of every HTC controller in case the HTC controller is not disabled. ("EB19EB01EB01EBZ01_IMV_OK" and "EB19EB01EB01EBZ01_ISP_OK") • **Stepper Step Transitions** – Logic with defined conditions for ever steps in the sector stepper machine logic (Stp_EB19_Seq). The basics steps and conditions can be found in **Fehler! Verweisquelle konnte nicht gefunden werden.**.
PID	PID_DL.py	PID dependent logic. The logic includes: • **PID setup** – Logic transfer the PID settings regarding which HTC group is selected and what is the current status of the HTC (error step, sync step, heating step). The logic transfers the following values to the PID controllers: PID mode, PID setpoint, increase speed and decrease speed.
OnOff	OnOff_DL.py	OnOff dependent logic. The logic includes: • **OnOff Acknowledge** – Logic for acknowledging 24V circuit breaker.

3.5. User Files

The user source holds the static TIA portal logic that is needed for the HESR project.

The user source logic is located in:

… \Resources\TIALogicGenerator\Rules\UserSpecific\Temp

with the function in the SpecFile it is possible to take and add the tempates to the project and to the INP file:

addTemplate()GL

addSimpleDB(EB050_DB,rEB01_MW_NEG;Word,rEB01_MW_STD;Word,rEB01_MW_LOW;Word,rEB01_MW_MID;Word,rEB02_MW_NEG;Word,rEB02_MW_STD;Word,rEB02_MW_LOW;Word,rEB02_MW_MID;Word,rEB03_MW_NEG;Word,rEB03_MW_STD;Word,rEB03_MW_LOW;Word,rEB03_MW_MID;Word,rEB04_MW_NEG;Word,rEB04_MW_STD;Word,rEB04_MW_LOW;Word,rEB04_MW_MID;Word,rEB05_MW_NEG;Word,rEB05_MW_STD;Word,rEB05_MW_LOW;Word,rEB05_MW_MID;Word,rEB06_MW_NEG;Word,rEB06_MW_STD;Word,rEB06_MW_LOW;Word,rEB06_MW_MID;Word,iEB01_MW_NEG;Word,iEB01_MW_STD;Word,iEB01_MW_LOW;Word,iEB01_MW_MID;Word,eEB01_MW_NEG;Word,eEB01_MW_STD;Word,eEB01_MW_LOW;Word,eEB01_MW_MID;Word)GL

Table 6: Python scripts for PCO

Script name	Description Table 7
GSI.inp	**GSI** – The file includes the list of all ".scl" files (listed below) which is used during TIA portal project generation. The file can be used for automatic generation of the user sources during TIA project generation.
GSI_C_INIT.scl	The file includes common functions as listed below: **FC_COND_DELAY** – Function for calculating time. Logic checks how much time the heating sequence need to stay in certain step. The function takes "step time" setpoint from SCADA (EB19_TS_WS) and returns to SCADA remaining minutes status (EB19_RMTS_WS) and remaining seconds status (EB19_RTS_WS).
GSI_Calc_AA_Limits.scl	The file includes: **GSI_Calc_AA_Limits** – Function calculates warning and error limits for every HTCs regarding to which heating group the HTC belongs. It returns calculated warning (EB19EB01EB01EBZ01_Max_AA.H) and error limit (EB19EB01EB01EBZ01_Max_AA.HH) which are used in analog alarm object.
GSI_Calc_Average.scl	The file includes: **GSI_Calc_Average** – Function calculates the average measured value (AMV) and the average setpoint value (ASP) of the certain heating group in certain section. The average values are calculated from all of the HTC values included in the section.
GSI_Calc_IMV_ISP.scl	The file includes: **GSI_Calc_IMV_ISP** – Function calculates when the HTC measured (IMV) value and HTC setpoint value (ISP) is reached. The values are then transferred to digital output object (EB19EB01EB01EBZ01_IMV_OK and EB19EB01EB01EBZ01_ISP_OK). The objects are true when the following conditions are fulfilled: • **IMV = Absolute (AMV – MV) < EB19_SOff_AP** Where "EB19_SOff_AP" is the offset setpoint. • **ISP = Absolute (ASP – SP) < 0.01**

3.6. Base Project

The base project is TIA portal base project that includes HW configuration of the component together with some predefined tags needed for later logic.

The base project is located in:

`…\UAB_Project\Source\SkeletonName`

What needs to be included in the base project for bakeout project is listed and described in the following in Table 8.

Table 8: Description of the base project

Script name	Description
HW configuration	The HW configuration includes: • PLC + CP module • ET200SP + input/outputs modules **P.S.** The HW configuration for ET200SP and HCS4200 need to be set as many MEC instances we have in the project.
Safety	Predefined safety logic regarding how many HCS are used.

3.7. UAB

UAB is a UNICOS tool used for generating sources which uses input files (specFile, python scripts, TIA user sources, TIA Portal base project) which are used to generate output files (TIA instances, TIA logic, TIA Portal project, WinCC OA source file).

Details about UAB software can be found here [2].

UAB has many functionalities but for HESR project, the following functionalities are needed:

- TIA Instance Generator – Used for generating object instances.
- TIA Logic Generator – Used for generating object logic which are prepared in advance in Python scripts.
- TIA Project Generator – Used for generating TIA Project. During prototype project, we didn't use UABs TIA Project Generator for generating TIA Project but we are using tool developed by GSI instead.
- WinCC OA Instance Generator – Used for generating source for WinCC OA.
- Expert User Generator – Used for generating expert logic. Expert logic add HCS calls used in organization blocks to TIA Portal project. The following TIA Portal organization blocks are modified for the HCS4200 operation: OB1, OB82, OB86 and OB100.

3.8. Output Files

During UAB generation project sources are generated which are needed for TIA Portal and for WinCC OA project.

The UAB outputs are stored to the following outputs:

- TIA Instance Generator: …\UAB_Project\Outputs\TIAInstanceGenerator
- TIA Logic Generator: …\UAB_Project\Outputs\TIALogicGenerator
- Expert User Generator: …\UAB_Project\Outputs\ExpertUserGenerator
- TIA Project Generator: …\UAB_Project\Outputs\TIAProjectGenerator
    - TIA Project generator uses outputs generated in all above folder in order to generate necessary source for TIA Portal project.
- WinCC OA Instance Generator: …\UAB_Project\Outputs\WinCCOAInstance Generator

3.9. Project Configuration

Before the project is started it needs to be configured over the GUI. Before the heating is started the following needs to be done:

- PID Configuration – Selection of which PIDs are used.
- PID Tuning – Setting PIDs parameters.
- Heating Configuration – Setting heating sequences.

3.10. PID Configuration

Before the first heating process can be started, PIDs setting needs to be done. The PIDs need to be allocated to the certain heating group and they need to be disabled in case they are not used.

PID allocation is done with word parameter objects, e.g. "EB19EB01EB01EBZ01_PType_WP" and can set to the following heating groups:

- 0 = None heating group
- 1 = NEG, foreseen for NEG coated parts, default max. temperature is 320°C
- 2 = STD, foreseen for standard parts, default max. temperature is 220°C
- 3 = LOW, foreseen for temperature-sensitive parts, default max. temperature is 120°C
- 4 = MID, foreseen for parts between STD and LOW, default max. temperature is 180°C

Attention: All HTCs of a heating group will always follow the same temperature profile

Additionally, if the PID is not in use it must be disabled with one of the following options:

- Setting heating group to "0" or
- Block termoclick digital alarm "EB19EB01EB01EBZ01_TC_DA".

3.11. PID Tuning

PID tuning is needed due to optimal operation of the PID loop. This must be done with support of UNICOS PID auto tuning functionality, through experts of the system and the tuning functionality.

3.12. Heating Configuration

Heating sequence need to be set within the recipe in order that heating can be started.

The parametrization needs to be done by vacuum people. It needs to be done for the following parameters for as many instances as many heating steps we would like to have (from 1 to 15 heating steps):

- Neg SP – Heating setpoint for NEG-group
- Std SP – Heating setpoint for STD-group
- Low SP – Heating setpoint for LOW-group
- Mid SP – Heating setpoint for MID-group
- Time – Step duration setpoint
- Limit H – High limit setpoint used for calculating PIDs warnings and errors.
- Limit L – Low limit setpoint used for calculating PIDs warnings and errors.

4. Project Operation - GUI

This chapter provides information about how to operate the HESR project via the GUI.

The HESR GUI is made of the following screens:

- Overview – This is the high-level overview GUI. This is startup GUI over which operator can jump into specific PLC project.
- Project Overview – Provides an overview of sections that are available in specific PLC project and their high-level statuses.
- Hierarchy – Overview about MPCO and PCO hierarchy inside one project.
- Cabinets – List of all cabinets over which operator can see HCSs details.

Their functionality is explained in the sections below.

4.1. Overview GUI

In this section the individual parts of the HESR Overview GUI are described as marked in Figure 8 and listed in Table 9.

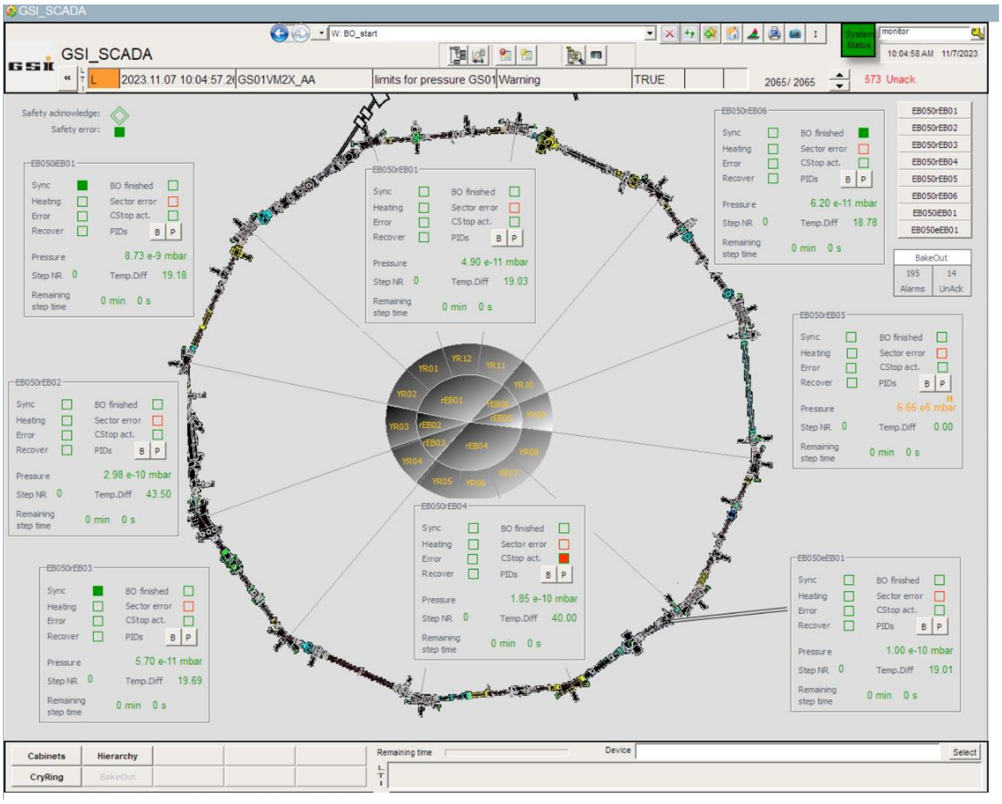

Figure 8: Overview GUI

Table 9: Description of the HESR Overview GUI

Component number	Component	Description
1.1	Sector text	Text indicates (mouseover) which PLC project overview **Fehler! Verweisquelle konnte nicht gefunden werden.** (which sections) will open when you click left mouse click on specific area. When the mouse is on no specify area, the text will change to "Overview".
1.2	Square around the section	When the mouse is moved over a specific section the white square around the sections is drawn.
1.3	Sec button	Button over which section 17 and 18 PLC project overview GUI **Fehler! Verweisquelle konnte nicht gefunden werden.** is opened.
1.4	Sec0 button	Button over which section 19 and 20 PLC project overview GUI **Fehler! Verweisquelle konnte nicht gefunden werden.** is opened.

Table 10: Description of the Project Overview GUI

Component number	Component	Description
2	Section status Overview	High-level status information about the section. Details information are in explained in 4.2.
3	Marked section	When the mouse is moved over a specific section data or moved over section area the section is coloured orange.
4.1	EB240EB19 button	Button over which the section operation GUI 4.3 is opened.
4.2	EB240EB20 button	Button over which the section operation GUI 4.3 is opened.
4.3	Hierarchy button	Button over which the hierarchy GUI 4.8 is opened.

4.4	Cabinet button	Button over which the cabinets GUI 4.9 is opened.
4.4	Coding button	Button over which the coding GUI **Fehler! Verweisquelle konnte nicht gefunden werden.** is opened.
5	HESR button	Button which returns us back to the HESR overview GUI 4.1.

4.2. Section Status GUI Area

In this section the individual parts of the section status GUI area are marked in Figure 9 and listed in Table 11 in the section 4.3.

Figure 9: Section Status

4.3. Section Operation GUI

In this section the individual parts of the section operation GUI are described as marked in Figure 10 and listed in Table 11.

Figure 10: Section Operation GUI

Table 11: Description of the Section Operation GUI

Component number	Component	Description
1.1	Sync status	Status indicates that the stepper machine is in synchronization step.
1.2	Heating/Cooling status	Status indicates that the stepper machine is in Heating/Cooling step.
1.3	Error indicator	Status indicates that the stepper machine is in Error step. When the PIDs are error free the operator can jump to “Recover” step with pressing on “Recover” button.
1.4	Recover status	Status indicates that the stepper machine is in Recovery step.

1.5	Recover button	Button over which stepper machine jumps into “Recover” step. The condition for this is that all PIDs are error free.
2.1	Sector BO finished status	Status indicates that the heating sequence is finished successfully.
2.2	Sector error status	Status indicates that there is an error in the section.
2.3	CStop act. status	Status indicates that the control stop is activated.
2.4	PCO buttons	Quick buttons over which the following GUIs can be opened: •  Button over which the GUI with HTCs bar status opens. •  Button over which the GUI with list of all objects that belongs to certain section opens. •  Button over which the GUI with stepper machine steps opens.
2.5		Section 19 PCO Object.
3.1	Step NR. status	Status shows in which step the stepper machine is currently is.
3.2	Step time status	Status shows how much time the current heating step in the stepper machine will last.
3.3	Delayed controlled stop setpoint	Setpoint for setting how much time the stepper machine waits in the error step before it jumps to end and sopped the heating.
3.4	Remaining step time status	Status shows what is the remaining heating step time (minutes, seconds) of the stepper machine.
4.1	Pressure status	Vacuum pressure value. This value comes from vacuum project.

4.2	Temp. Diff status	
5.1	Recipe table button	Button over which the heating table GUI 0 is opened.
5.2	Starting Step NR setpoint	Setpoint for setting in which heating step the stepper machine will start when operator starts the stepper machine.
5.3	Step MV offset setpoint	Setpoint for setting the measured value offset needed for calculating IMV status which is needed as one of the steps conditions to jump into the next heating step.
5.4	Recipe Overview button	Button over which the recipes overview for the actual section opens.
5.4	Heating recipe button	Button over which the recipe with the heating parameters opens. Operator select heating configuration and activate the selection which will configure heating parameters.
5.4	PIDs recipe button	Button over which the recipe with the PID parameters opens. Operator select PIDs configuration and activate the selection which will configure PIDs parameters.
6	Actual Heating parameters	Settings of the heating parameters used in the actual heating step. For every step, there are different heating parameters and the actual parameters are shown with respect to which heating step the stepper machine is currently in.

4.4. Heating Table GUI

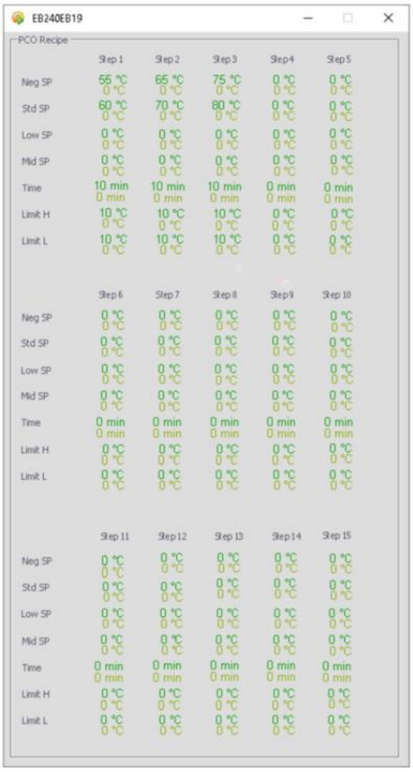

Figure 11: Heating Table

Show the matrix of the heating configuration for total 15 steps of all 4 heating types.

4.5. Stepper Step Status GUI

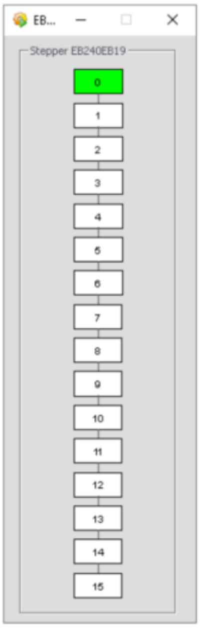

Figure 12: Stepper Step Status

Show the status of the sequence step of the sector.

4.6. Section Hierarchy GUI

Figure 13: Section Hierarchy

Show the hierarchy of the sector with all PIDs, heater, outputs and feedback.

4.7. Section Bar Status GUI

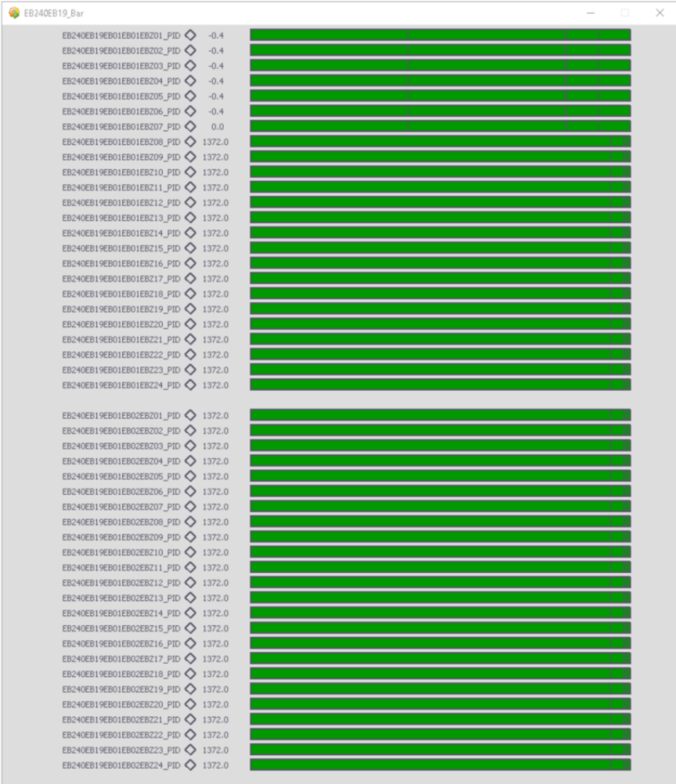

Figure 14: Section Bar Status

Show the sector bars and the range of the heatings. In case of overheat, the bars change the color to red and over cool to blue.

4.8. Hierarchy GUI

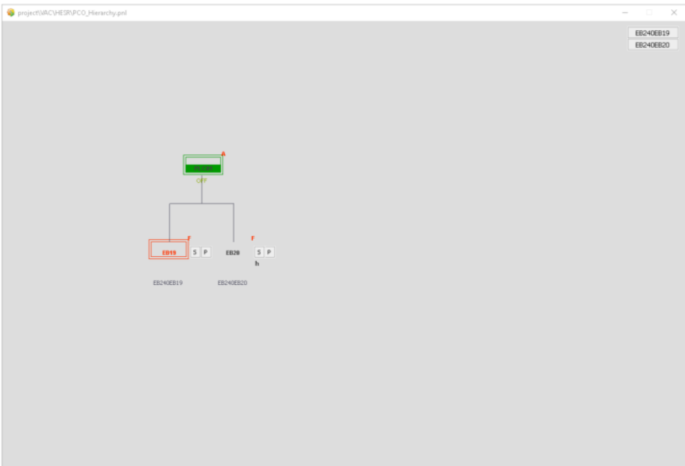

Figure 15: Hierarchy

Show the PCO hyrarchie.

4.9. Cabinets GUI

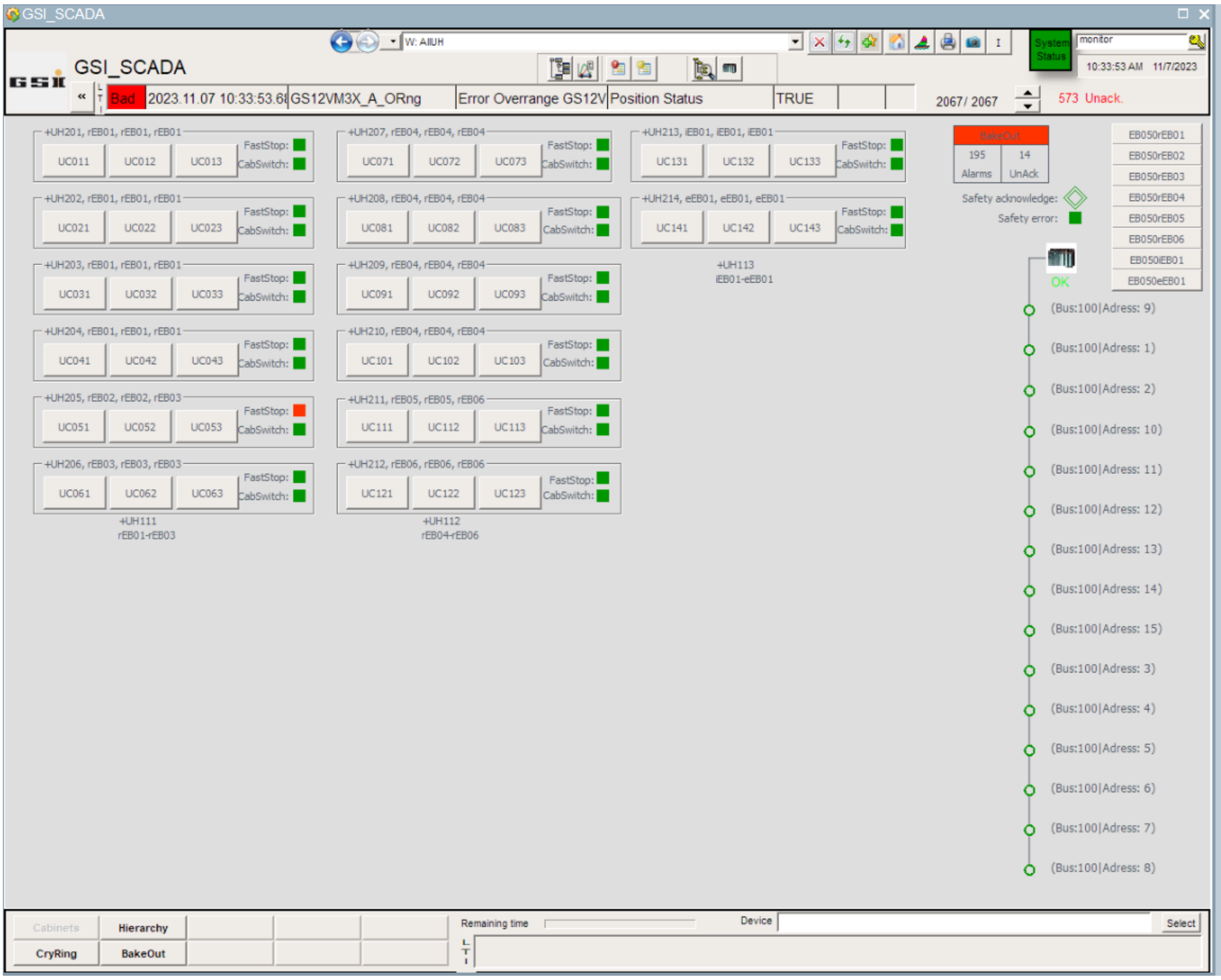

Figure 16: Cabinet status

Show the heating groups of all cabinets. In addition, it is hear shown the status of the safety and the status of the PLC periphery.

The “safety error” bit show the status of the safety system. In safety error case the bit start blinking red and the status of the cabinet switch show, if the main switch has been turn off.

The fastStop status shows, where someone pressed the faststop.

If all cabinet switches where turn on again, the safety must be acknowledge by the UNICOS object “Safety acknowledge”.

4.10. Channels

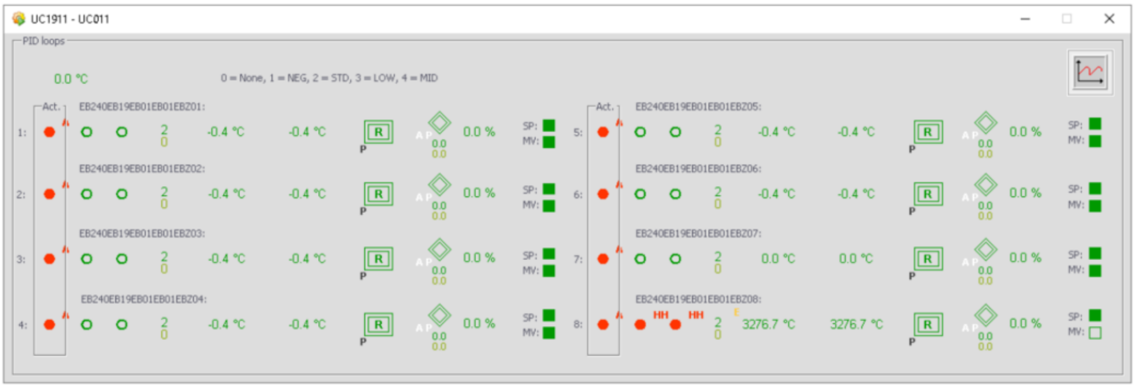

Figure 17: Channel status

Show all eight heating PIDs of the heater group.