HIE Isolde cavity high pressure water rinsing machine in blg. 252

Aleksander Skala
Table of contents

1. Introduction

2. Topology of the machine
   2.1 Diagram of the machine
   2.2 Principle of operation

3. Control system
   3.1 Diagram of the control system
   3.2 Equipment and hardware contents
   3.3 Software and drivers on the PC
   3.4 Algorithms of control process
   3.5 Using of Traversing Block functionality in Siemens’s Starter program
   3.6 Main functions motors control with comments
   3.7 Manual for the operator
   3.8 Using of control panel

4. Attachments
1. **Introduction**

The HIE-ISOLDE superconducting linac at CERN will be based on 101.28MHz niobium sputtered copper Quarter Wave Resonators (QWRs), which will be installed downstream of the present REX-ISOLDE linac. The current design considers two basic cavity geometries (geometric $\omega$ of 0.063 and 0.103).

CERN has designed and prepared new facilities for the surface treatment and niobium sputter coating of the HIEISOLDE superconducting cavities.

The HIE Isolde pressure water machine is designed to rinse new types of RF cavities (fig. 1) with water under high pressure.

![Fig.1. General view of new type RF cavity dedicated for HIE Isolde project.](image)

The unit is located in building 252.

The device is constructed in the form of the rinsing head and by means of a system of two drives and automation system is raised and rotated.
2. **Topology of the machine**

2.1 **Diagram of the machine**

The general view of the machine is presented below on the figures (fig. 2 and 3).

![Fig. 2. General view of the machine during construction works.](image1)

![Fig. 3. General view of the machine with motors and automation system.](image2)
2.2 Principle of operation

The machine is made in the form of a metal table on which they are installing cavity in horizontal position. Support points are four rubber rollers that rotate with the cavity.

The purpose of automation system is to control the cavity pressure washing process, which is performed by rotating the lying cavity at the set speed and move the washing head while process along the length of the internal cavity.

The movement of the rinsing head and rotating the cavity are responsible two synchronous motors controlled by the system Sinamics S120 shown on fig. 5.

Communication of Sinamics with the running motors is done by using orange cables (presented on the image) and using the protocol Drive CLIQ. Green cables are the power cables for motors which are powered from the power converter called: Double Motor Module 3A (fig. 5).

![Image of installed motors]

Fig. 4. The view of installed motors.
3. Control system

3.1 Diagram of the control system

Fig. 5. The diagram of the control system.

3.2 Equipment and hardware contents

From the standpoint of automation the equipment of control system (fig. 5) consist of hardware:

1. **PLC controller CPU 314C-2PN/DP** - stores the programs/algorithms that performed by system as a whole (detailed description in chapter 3.4 and 3.5).

2. **Ethernet/Profi-Net Switch module** – mediates in communication by switching communication devices (each element of the system has its own IP address).

3. **Drive control unit CU320-2PN/DP** – motor controller, which "translates" the individual lines of code in the main program contained in the PLC (with the settings for a particular type of motor) to electrical signals which
control the Double Motor Module 3A, which is an advanced inverter that changes the frequency of the output current and voltage, powered motors.

4. **Smart Line Module 5kW** - uncontrolled rectifier which feed the inverter (Motor Module). Output voltage: 560V DC.

5. **Double motor module** – changing frequency inverter with adjustable output current and voltage for powered motors. Frequency changes cause changes in motor speed and the change in the value of the output current varies motor torque.

6. **Computer PC** – configured as a SCADA server, is used to program the PLC, process visualization and archiving of process data.

Machine control system was built in one rack 19” and combined with controlled object.

Place for the operator have been made as ergonomic and compact as possible (Fig. 6).

---

**Fig. 6. A place for the operator of HIE Isolde rinsing machine**

3.3 **Software and drives on the PC**
The program which controls system as a whole is contained in the PLC. It was written using Siemens Step7 v.11 sp1 program.

The drive control system consists of: Drive Control Unit CU320-2PN/DP, Smart Line Module 5kW and Double Motor Module is configured using the Starter V4.2 HF1 Siemens.
Visualization of the process (operator panel) was created with the PVSS ver. 3.8 SP2 using Unicos-PVSS-5.1.2.

3.4 Algorithms of control process

Algorithms and programs implemented in the control system were prepared for the standard cavity of a desired size of the active length of 732 mm internally. Further cycles of the machine have been prepared in a way that ensures maximum flexibility and versatility of the device in the event of having to modify its functionality.

Machine cycle consists a few modes:
- **Mode 732 mm** – basic washing program executing within the cavity along the internal length of 732 mm. It contains two rinsing cycles of the movement the head from the base of cavity to the top and back again. At the same time the cavity is rotated all the time. Realization of whole the mode takes about 40 minutes.
- **Mode Universal** – carries out a program as mode above, but for a length specified by the operator,
- **OUT Position** – mode eject pipe out of the machine (from the washing space) for mounting the nozzle at its end,
- **Homing** – mode is used to set zero of head positioning (zero vertical),
- **Manual** – manual control of movement and rotation of the rinsing head.
- **Parking Position** – program sets the maximal washing head deep inside the washing chamber in order to minimize the dimensions of the machine while not in use.
The algorithm of the program for **Mode 732 mm** cycle is as follows:

1. **START**
   - if Option Mode is nr: 2.0 (“Mode 732 mm”)?
     - **NO**
     - **YES**
       - go 2 mm up until position: 732 mm
       - turn the cavity 180 degrees clockwise
       - wait 1 second
       - turn the cavity 180 degrees anti-clockwise

2. **Repeat 2 times the loop:**
   - it was already done in two cycles?
     - **YES**
     - **NO**
       - if the position of the head is less than: 732 mm?
         - **NO**
         - **YES**
           - go 2 mm up until position: 732 mm
           - turn the cavity 180 degrees clockwise
           - wait 1 second
           - turn the cavity 180 degrees anti-clockwise

3. **STOP**
   - go to Home position (zero position)

Settings in Traversing Blocks in Siemens STARTER program for Servo_Turn and Servo_Up motor
3.5 Using the functionality of Siemens’s Starter program

Program Siemens STARTER is most used in three cases described below:

✓ **To achieve a controlled move** is used the functionality called: “Traversing Block”

Each motor has a separately defined sequence of movement (Traversing Block table), which contains:

- defined sequence of positions generated for the motor,
- definition of the motion: is relative or absolute,
- the velocity between positions (motor position is read by the encoder integrated inside the motor),
- waiting time between successive movements.

To start using of Traversing Block in STEP7 program for the motor Servo_Up, this involves in indicating the position in the array (“Traversing Block”)
in binary. For example: to jump to position 13 is needed to set bits of: the 1\textsuperscript{st}, 3\textsuperscript{rd} and 4\textsuperscript{th}:

"PZD_Up".Send.POS_STW1.TrvBit_0 := TRUE;
"PZD_Up".Send.POS_STW1.TrvBit_2 := TRUE;
"PZD_Up".Send.POS_STW1.TrvBit_3 := TRUE;

because the number thirteen written in binary is: 1011.

After jumping to position nr 13, the sequence of events is as follows (Siemens’s Starter):

1. System waits 2000ms (2 sec.)
2. Movement in relative mode (-2000 length units = 2 mm from last position) with a speed 4000 length units/s (4 mm/sec.)
3. Go to position number 13 in Traversing Block and starts a loop again.

In addition, the control bits have to be set in STEP7 program:

"PZD_Up".Send.STW1.RejTrvTask := TRUE; // rejection the previous task
"PZD_Up".Send.STW1.ExtBlkChange := TRUE;  // permission to run Traversing Block by external signals
"PZD_Up".Send.STW1.TrvTStart := TRUE;  // initiate action of Traversing Block

These commands stored in a control function FC_Servo_Up for the motor in the Step7, mean that for the object PZD_Up will be sent by the PLC using the protocol called the Process Data Telegram, control data to the drive control unit CU320-2PN/DP.

Stopping the motor requires setting the control bits in the inverted position, means FALSE:

"PZD_Turn".Send.STW1.RejTrvTask := FALSE;
"PZD_Turn".Send.STW1.ExtBlkChange := FALSE;
"PZD_Turn".Send.STW1.TrvTStart := FALSE;
Finding the cause of the error in the system:

If for some reason the control system error occurs (for example: due to interruption of ongoing pre-selected mode) it is shown in the Control Panel:

The occurrence should then to be confirmed (double click on the red square) by the operator by "Acknowledge" and then reset:
If this does not help, then open used project in Siemens STARTER. Next find the Control Panel for the appropriate motor:

To look into the error code and using the HELP to find a description of the error:

F07490 EPOS: Enable signal withdrawn while traversing V4.4

Valid as of version: 0.30

Drive objects: SERVO_S110_CAN, SERVO_S110-OP, SERVO_S110-PN, VECTOR, VECTOR_0330/S110

Response: OFF1 (OFF2, OFF3)

Acknowledgment: IMMEDIATELY

Cause: - for a standard assignment, another fault may have occurred as a result of withdrawing the enable signals.
- the drive is in the "switching on inhibited" state (for a standard assignment).

Remedy: - set the enable signals or check the cause of the fault that first occurred and then reset it (for a standard assignment).
- check the assignment to enable the basic positioning function.
✓ **Calibration of zero for each motor:**
   It is used when the machine design was changed and that is required or the encoder lost for some reason actual position.

3.6 **Manual movement**

Siemens STARTER program is also used to carry out manual control of motors. Due to it is used to the functionality of the **Control Panel:**
It is needed first select which drive will be used, and then approve the acquisition of motor control by the program STARTER:

The PLC bit responsible for controlling the system Sinamics will no longer be active:

After doing the above, **Control Panel**, should look like this:

Then choose among the following options:
- *n setpoint specification*
- *Basic positioner*
Functionality **Basic positioner** has three modes:

- **Jog** – action: running as long as you are pressing "I". Ceases to work after pressing "0":

- **Positioning** – to define the distance to be covered:
  - **Positioning absolute** mode - the distance to be covered by the drive in LU with respect to zero (length units, 1000 LU is 1 mm) and the speed needs to be set to,
  - **Positioning relative** mode - the distance of the drive in LU compared to the last position reached (length units, 1000 LU is 1 mm) and the speed needs to be set also,
  - **Traversing block** - the movement sequence is read from Traversing Block (functionality described in section 3.5).
- **Homing** – drive back to zero with the set speed.

- Functionality **n setpoint** has only one function:
  - the motor runs at the set speed in rev / min.

### 3.7 Main functions motors control with comments

The main part of the application for PLC are functions which control the motors: **FC_Servo_Turn** (motor rotating the head) and **FC_Servo_Up** (motor lifting the head in vertical).
Figure 12. A General view of the source code of control application for the project HIE Isolde in Siemens STEP 7 program

The program (function FC_Servo_Turn) is divided into six sections for the implementation of the machine cycles:

- **OUT Position** -> Option Mode 6
- **Mode 732 mm** -> Option Mode 2
- **Mode Universal** -> Option Mode 3
- **Homing** -> Option Mode 4
- **Manual** -> Option Mode 5
- **Parking Position** -> Option Mode 1

are used Option Modes of PCO object in Unicos-PVSS. During choosing a duty cycle value of $\text{HPWR_B118_PCO.OpMoX}$ is set (for 1,2,3,4,5,6 or 8).

If the value of $\text{HPWR_B118_PCO.OpMoX}$ will be set to 8.0, the machine is stopped.
FUNCTION FC_Servo_Turn : VOID

VAR
loop : DINT;
clock : BOOL;
anticlock : BOOL;
up : BOOL;
down : BOOL;
a : DINT;
b : DINT;
c : DINT;
END_VAR

a := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Size_Cav_A.CurValSt); // cavity A size in mm
b := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Size_Cav_B.CurValSt); // cavity B size in mm
c := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Size_Cav_C.CurValSt); // cavity C size in mm

IF (HPWR_B118_PCO.OpMoX = 1.0) OR (HPWR_B118_PCO.OpMoX = 2.0) OR (HPWR_B118_PCO.OpMoX = 3.0) OR (HPWR_B118_PCO.OpMoX = 4.0) THEN
    "PZD_Turn".Send.STW1.OFF1 := TRUE;
END_IF;

IF (HPWR_B118_PCO.RunOSt = TRUE) THEN
//-------------for mode 1 Parking Position-----------------------------------------------

IF (HPWR_B118_PCO.OpMoX = 1.0) THEN

IF "PZD_Up".Receive.XIST_A > 0 THEN

"PZD_Turn".Send.POS_STW1.TrvBit_0 := TRUE;
"PZD_Turn".Send.STW1.ExtBlkChange := TRUE;
"PZD_Turn".Send.STW1.TrvTStart := TRUE;
"PZD_Turn".Send.POS_STW2.SetRefPt := TRUE;
ELSE

"PZD_Turn".Send.STW1.RejTrvTask := FALSE;
"PZD_Turn".Send.POS_STW1.TrvBit_0 := FALSE;
"PZD_Turn".Send.STW1.ExtBlkChange := FALSE;
"PZD_Turn".Send.STW1.TrvTStart := FALSE;
END_IF;
END_IF;
//-------------------------------for mode 2 Mode 732 mm-------------------------------------

IF (HPWR_B118_PCO.OpMoX = 2.0) THEN

IF GlobalVariables.cycleNumber < 2 AND "PZD_Up".Receive.XIST_A > 0 THEN

"PZD_Turn".Send.POS_STW1.TrvBit_1 := TRUE;
"PZD_Turn".Send.POS_STW1.TrvBit_3 := TRUE;
"PZD_Turn".Send.STW1.ExtBlkChange := TRUE;
"PZD_Turn".Send.STW1.TrvTStart := TRUE;
ELSIF "PZD_Up".Receive.XIST_A <= 0 THEN

"PZD_Turn".Send.STW1.RejTrvTask := FALSE;
"PZD_Turn".Send.POS_STW1.TrvBit_3 := FALSE;
"PZD_Turn".Send.STW1.ExtBlkChange := FALSE;
"PZD_Turn".Send.STW1.TrvTStart := FALSE;
END_IF;
END_IF;

//-------------for mode 3 Mode Universal-----------------------------------------------

IF (HPWR_B118_PCO.OpMoX = 3.0) THEN

IF (GlobalVariables.cycleNumber < 2 AND "PZD_Up".Receive.XIST_A > 0) THEN

"PZD_Turn".Send.POS_STW1.TrvBit_1 := TRUE;
"PZD_Turn".Send.STW1.TrvBit_3 := TRUE;
"PZD_Turn".Send.STW1.ExtBlkChange := TRUE;
"PZD_Turn".Send.STW1.TrvTStart := TRUE;
END_IF;
END_IF;

CERN | Documentation of HIE Isolde rinsing machine


//"PZD_Turn".Send.STW1.IntStop := TRUE;
//"PZD_Turn".Send.STW1.RejTrvTask := TRUE;
"PZD_Turn".Send.POS_STW1.TrvBit_3 := TRUE;
"PZD_Turn".Send.STW1.ExtBlkChange := TRUE;
"PZD_Turn".Send.STW1.TrvTStart := TRUE;

ELSIF "PZD_Up".Receive.XIST_A <= 0 THEN

"PZD_Turn".Send.STW1.RejTrvTask := FALSE;
"PZD_Turn".Send.POS_STW1.TrvBit_3 := FALSE;
"PZD_Turn".Send.STW1.ExtBlkChange := FALSE;
"PZD_Turn".Send.STW1.TrvTStart := FALSE;
END_IF;

END_IF;

//-------------------------------------------------------------------------------------------------------------
//--------------------------for mode 4 Homing----------------------------------------------------------
IF (HPWR_B118_PCO.OpMoX = 4.0) THEN

"PZD_Turn".Send.POS_STW1.TrvBit_0 := TRUE;
"PZD_Turn".Send.STW1.ExtBlkChange := TRUE;
"PZD_Turn".Send.STW1.TrvTStart := TRUE;
"PZD_Turn".Send.STW1.OFF1 := TRUE;

END_IF;

//-------------------------------------------------------------------------------------------------------------
//--------------------------for mode 5 Manual-----------------------------------------------------------
IF (HPWR_B118_PCO.OpMoX = 5.0) THEN

"PZD_Turn".Send.STW1.OFF1 := TRUE;
"PZD_Turn".Send.STW1.Jog1 := HPWR_B118_ONOFF01.OutOV;
"PZD_Turn".Send.STW1.Jog2 := HPWR_B118_ONOFF02.OutOV;
ELSE
"PZD_Turn".Send.STW1.Jog1 := FALSE;
"PZD_Turn".Send.STW1.Jog2 := FALSE;
END_IF;

END_IF;

//-------------------------------------------------------------------------------------------------------------
END_FUNCTION

//-------------------------------------------------------------------------------------------------------------
FUNCTION FC_Servo_Up : VOID

VAR
loop : DINT;
clock : BOOL;
anticlock : BOOL;
up : BOOL;
down : BOOL;
a : DINT;
b : DINT;
c : DINT;
returnFlag : BOOL;
END_VAR

a := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Size_Cav_A.CurValSt); // cavity A size in mm
b := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Size_Cav_B.CurValSt); // cavity B size in mm
c := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Size_Cav_C.CurValSt); // cavity C size in mm

//init
"PZD_Up".Send.STW1.OFF2_CoastStop := TRUE;
"PZD_Up".Send.STW1.OFF3_QuickStop := TRUE;
"PZD_Up".Send.STW1.EnOp := TRUE;
"PZD_Up".Send.OVERRIDE := 300;
"PZD_Up".Send.STW1.OFF1 := FALSE;
"PZD_Up".Send.STW1.IntStop := TRUE;
"PZD_Up".Send.POS_STW1.TrvBit_0 := FALSE;
"PZD_Up".Send.POS_STW1.TrvBit_1 := FALSE;
"PZD_Up".Send.POS_STW1.TrvBit_2 := FALSE;
"PZD_Up".Send.POS_STW1.TrvBit_3 := FALSE;
"PZD_Up".Send.POS_STW1.TrvBit_4 := FALSE;
"PZD_Up".Send.POS_STW1.TrvBit_5 := FALSE;
"PZD_Up".Send.STW1.ExtBlkChange := FALSE;
"PZD_Up".Send.STW1.TrvTStart := FALSE;

//to reset counter when we are not in Mode_2
IF (HPWR_B118_PCO.OpMoX <> 2.0) AND (HPWR_B118_PCO.OpMoX <> 3.0) THEN
GlobalVariables.cycleNumber := 0;
GlobalVariables.returnFlag := FALSE;
GlobalVariables.resetFlag := FALSE;
END_IF;

IF (HPWR_B118_PCO.OpMoX = 1.0) OR (HPWR_B118_PCO.OpMoX = 2.0) OR (HPWR_B118_PCO.OpMoX
= 3.0) OR (HPWR_B118_PCO.OpMoX = 4.0) THEN

"PZD_Up".Send.STW1.OFF1 := TRUE;
END_IF;

IF (HPWR_B118_PCO.RunOSt = TRUE) THEN

//---------------------------for mode 1 Parking Position---------------------------------------------

IF (HPWR_B118_PCO.OpMoX = 1.0) THEN

"PZD_Up".Send.STW1.RejTrvTask := TRUE;
"PZD_Up".Send.POS_STW1.TrvBit_0 := TRUE;
"PZD_Up".Send.STW1.ExtBlkChange := TRUE;
"PZD_Up".Send.STW1.TrvTStart := TRUE;

END_IF;

//---------------------------------------------------------------------------------------------------------------------------------

//--------------------------for mode 2 Mode 732 mm---------------------------------------------------------------

IF (HPWR_B118_PCO.OpMoX = 2.0) THEN

IF GlobalVariables.cycleNumber < 2 THEN

TON.DB_TON_T1(IN:="PZD_Up".Send.STW1.RejTrvTask = FALSE),PT:=T#5s);

IF (GlobalVariables.returnFlag = FALSE) THEN

IF ("PZD_Up".Send.STW1.RejTrvTask = TRUE) THEN

"PZD_Up".Send.POS_STW1.TrvBit_1 := TRUE;
"PZD_Up".Send.POS_STW1.TrvBit_3 := TRUE;
"PZD_Up".Send.STW1.ExtBlkChange := TRUE;
"PZD_Up".Send.STW1.TrvTStart := TRUE;

END_IF;

IF (DB_TON_T1.Q = 1) THEN

"PZD_Up".Send.STW1.RejTrvTask := TRUE;

END_IF;

ELSIF (GlobalVariables.resetFlag = FALSE) THEN

"PZD_Up".Send.STW1.RejTrvTask := FALSE;
GlobalVariables.returnFlag := TRUE;
GlobalVariables.resetFlag := TRUE;
"PZD_Up".Send.POS_STW1.TrvBit_1 := FALSE;
"PZD_Up".Send.POS_STW1.TrvBit_3 := FALSE;
"PZD_Up".Send.STW1.ExtBlkChange := FALSE;
"PZD_Up".Send.STW1.TrvTStart := FALSE;

ELSE

IF ("PZD_Up".Send.STW1.RejTrvTask = TRUE) THEN

END_IF;
"PZD_Up".Send.POS_STW1.TrvBit_0 := TRUE;
"PZD_Up".Send.POS_STW1.TrvBit_2 := TRUE;
"PZD_Up".Send.POS_STW1.TrvBit_3 := TRUE;
"PZD_Up".Send.STW1.ExtBlkChange := TRUE;
"PZD_Up".Send.STW1.TrvTStart := TRUE;
END_IF;

IF (DB_TON_T1.Q = 1) THEN
  "PZD_Up".Send.STW1.RejTrvTask := TRUE;
END_IF;

IF ("PZD_Up".Receive.XIST_A <= 0) AND (GlobalVariables.returnFlag = TRUE) THEN
  GlobalVariables.returnFlag := FALSE;
  GlobalVariables.resetFlag := FALSE;
  GlobalVariables.cycleNumber := GlobalVariables.cycleNumber + 1;
  "PZD_Up".Send.STW1.RejTrvTask := FALSE;
  "PZD_Up".Send.POS_STW1.TrvBit_0 := FALSE;
  "PZD_Up".Send.POS_STW1.TrvBit_2 := FALSE;
  "PZD_Up".Send.POS_STW1.TrvBit_3 := FALSE;
  "PZD_Up".Send.STW1.ExtBlkChange := FALSE;
  "PZD_Up".Send.STW1.TrvTStart := FALSE;
END_IF;
END_IF;

END_IF;

END_IF;

END_IF;

//-------------------------------for mode 3 Mode Universal-----------------------------------------------------------
IF (HPWR_B118_PCO.OpMoX = 3.0) THEN
  IF GlobalVariables.cycleNumber < 2 THEN
    TON.DB_TON_T1(IN:="PZD_Up".Send.STW1.RejTrvTask = FALSE),PT:=T#5s);
  IF ("PZD_Up".Receive.XIST_A < 330000 + 1000*(a + b + c) AND GlobalVariables.returnFlag = FALSE) THEN
    IF ("PZD_Up".Send.STW1.RejTrvTask = TRUE) AND "PZD_Up".Receive.XIST_A >= 320000 THEN
      "PZD_Up".Send.POS_STW1.TrvBit_1 := TRUE;
      "PZD_Up".Send.POS_STW1.TrvBit_3 := TRUE;
      "PZD_Up".Send.STW1.ExtBlkChange := TRUE;
      "PZD_Up".Send.STW1.TrvTStart := TRUE;
    END_IF;
  IF ("PZD_Up".Send.STW1.RejTrvTask = TRUE) AND "PZD_Up".Receive.XIST_A < 320000 THEN

"PZD_Up".Send.POS_STW1.TrvBit_1 := TRUE;
"PZD_Up".Send.POS_STW1.TrvBit_4 := TRUE;
"PZD_Up".Send.STW1.ExtBlkChange := TRUE;
"PZD_Up".Send.STW1.TrvTStart := TRUE;
END_IF;

IF (DB_TON_T1.Q = 1) THEN
  "PZD_Up".Send.STW1.RejTrvTask := TRUE;
END_IF;

ELSIF (GlobalVariables.resetFlag = FALSE) THEN

  "PZD_Up".Send.STW1.RejTrvTask := FALSE;
  GlobalVariables.returnFlag := TRUE;
  GlobalVariables.resetFlag := TRUE;
  "PZD_Up".Send.POS_STW1.TrvBit_1 := FALSE;
  "PZD_Up".Send.POS_STW1.TrvBit_3 := FALSE;
  "PZD_Up".Send.POS_STW1.TrvBit_4 := FALSE;
  "PZD_Up".Send.STW1.ExtBlkChange := FALSE;
  "PZD_Up".Send.STW1.TrvTStart := FALSE;
END_IF;

ELSE

  IF ("PZD_Up".Send.STW1.RejTrvTask = TRUE) THEN
    "PZD_Up".Send.POS_STW1.TrvBit_0 := TRUE;
    "PZD_Up".Send.POS_STW1.TrvBit_2 := TRUE;
    "PZD_Up".Send.POS_STW1.TrvBit_3 := TRUE;
    "PZD_Up".Send.STW1.ExtBlkChange := TRUE;
    "PZD_Up".Send.STW1.TrvTStart := TRUE;
  END_IF;

  IF (DB_TON_T1.Q = 1) THEN
    "PZD_Up".Send.STW1.RejTrvTask := TRUE;
  END_IF;

  IF ("PZD_Up".Receive.XIST_A <= 330000) AND (GlobalVariables.returnFlag = TRUE) THEN
    GlobalVariables.returnFlag := FALSE;
    GlobalVariables.resetFlag := FALSE;
    GlobalVariables.cycleNumber := GlobalVariables.cycleNumber + 1;
    "PZD_Up".Send.STW1.RejTrvTask := FALSE;
    "PZD_Up".Send.POS_STW1.TrvBit_0 := FALSE;
    "PZD_Up".Send.POS_STW1.TrvBit_2 := FALSE;
    "PZD_Up".Send.POS_STW1.TrvBit_3 := FALSE;
    "PZD_Up".Send.STW1.ExtBlkChange := FALSE;
    "PZD_Up".Send.STW1.TrvTStart := FALSE;
  END_IF;

END_IF;

END_IF;
END_IF;

END_IF;

//------------------------------------------for mode 4 Homing---------------------------------------------

IF (HPWR_B118_PCO.OpMoX = 4.0) THEN

"PZD_Up".Send.STW1.RejTrvTask := TRUE;
"PZD_Up".Send.POS_STW1.TrvBit_3 := TRUE;
"PZD_Up".Send.STW1.ExtBlkChange := TRUE;
"PZD_Up".Send.STW1.TrvTStart := TRUE;

END_IF;

//------------------------------------------for mode 5 Manual---------------------------------------------

IF (HPWR_B118_PCO.OpMoX = 5.0) THEN

"PZD_Up".Send.STW1.OFF1 := TRUE;
"PZD_Up".Send.STW1.Jog1 := HPWR_B118_ONOFF03.OutOV;
"PZD_Up".Send.STW1.Jog2 := HPWR_B118_ONOFF04.OutOV;

ELSE
"PZD_Up".Send.STW1.OFF1 := FALSE;
"PZD_Up".Send.STW1.Jog1 := FALSE;
"PZD_Up".Send.STW1.Jog2 := FALSE;
//END_IF;

END_IF;

//------------------------------------------for mode 6 OUT Position---------------------------------------------

IF (HPWR_B118_PCO.OpMoX = 6.0) THEN

"PZD_Up".Send.STW1.RejTrvTask := TRUE;
"PZD_Up".Send.POS_STW1.TrvBit_0 := TRUE;
"PZD_Up".Send.STW1.ExtBlkChange := TRUE;
"PZD_Up".Send.STW1.TrvTStart := TRUE;

END_IF;
//END_FUNCTION
3.8 Manual for the operator

In order to start working with, the operator should do the following:
1. Turn on the power 230 VAC for the control system (multiple socket is placed on the left side of the rack (rear)) – Fig. XXX
2. Turn the Main Switch 3-phase power – Fig. XXX

![General view the switches of power in the automation system.](image)

3. Turn on the computer and logging as:
   - Login: hieisold
   - Password: 252Hiesystem

4. Turn on **PVSS Project Administration** (the icon on Desktop) and choose **HIE-ISOLDE** project and then **Start Project** icon:

---

CERN | Documentation of HIE Isolde rinsing machine 29
The operator panel will open.

5. To start **Step7** program which contains the source files for PLC, click the icon on Desktop: ![SIMATIC Manager](image) and choose **HPWR** project.

6. To start Siemens’s **STARTER** program (which contains settings of the motors and Traversing Blocks for controlled move of each motor) click the icon on Desktop: ![STARTER](image).

### 3.9 Using of control panel

Control panel presented below, shows a general view of the machine and includes:
Status of the motors: Servo_Turn (motor rotating the head) and Servo_Up (motor lifting the head in vertical): active the icon (green trapezoid) means the motor runs in defined direction.

Relative position of the head (relate to home position) in degrees for Servo_Turn in millimeters for Servo_Up and speed for both motors.

To run the system, it is necessary to select the mode of operation: OUT Position, Mode 732mm, Homing, Manual or Parking Position. Resignation from the selected mode is done by pressing the STOP button.

After you select the mode and run it, the rest of the options is disabled (it blocks the ability to run several operating modes at the same time).

At the time of the appearance of any fault in the motor control system, it appears the Drive Fault. In the vast majority of cases, these errors do not require the intervention, but only to confirm their understanding by the operator and reset the fault.
Set the zero (zero position) for each axis is possible by using the buttons in field: "Drive set to zero." This operation is performed for unforeseen reasons if the system lost zero (reference position) or the machine will be modified.

4. **Attachments**

- Demande Achat Interne (DAI) nr.: 4664445 - list of ordered components from Siemens
- SIMOTION – Frequently Asked Questions - the sample project using Profinet
- Siemens Sinamics S120 Synchronmotoren 1FK7
- Siemens Sinamics S120 Commissioning Manual
- Siemens Sinamics S120/S150 List Manual
- Siemens Sinamics S120/S150 Function Manual
- Siemens Sinamics S120 Control Units and additional system components – manual

**NOTE!**
All these attachments are stored in electronic files (*.pdf) on the DVD included with the printed version of this document.