High Pressure Water Rinsing machine in blg. 118

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 High Pressure Water Machine is designed to rinse RF cavities different types with water under high pressure.

The unit is located in building 118.

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 diagram of the device is presented below on the figure (Fig.1).

2.2 Principle of operation

The lower part of the head, which is supplied with water and nitrogen gas is moved by two synchronous motors 1FK7 SINAMICS S120 made by Siemens. One of them rotates the head in the range from 0 to 180 degrees while the second one moves it up or down with the roller mechanism with toothed belt (Fig.2, 3, 4, 5). The mechanism has a counterbalance in order to offload the vertical (up and down) drive.

Above there is a nitrogen-bearing, whose task is to minimize the resistance of movement of the rinsing head while ensuring the integrity of the system and drain used rinse water out (Fig.6).

At the top there is mounted a cavity for rinsing. Just below it there are two valves that remain open during the process and after its completion shall be sealed in order to ensure hermetic sealing of machine and eluted cavity (Fig.7). The cavity after the removal is transported to the destination place together with the upper valve.
Fig. 1. Topology of HPWR machine.
Fig. 2. Roller mechanism with toothed belt counterbalance and end position sensor.

Fig. 3. Head base up and down
Fig. 4. Head base with drives up and down

Fig. 5. Head base with drives and mountable operating mechanism with rollers and toothed belt
Fig. 6. Water-bearing with filter and cross-section drawing.

Fig. 7. Sample RF test cavity with the valves.
3. **Control system**

3.1 **Diagram of the control system**

Fig.8. The diagram of the control system.

3.2 **Equipment and hardware contents**

From the standpoint of automation the equipment of control system (fig.8.) 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.

Place for the operator have been made as ergonomic and compact as possible (Fig. 9).

![Fig.9. A general view of station for the operator.](image-url)
Machine control system was built in one rack 19” and combined with controlled object.

### 3.3 Software and drivers 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 presented below have been written for the test cavity with dimensions shown in the figure 10.

![Figure 10. A sample of the cavity with the dimensions A, B, C](image-url)

Figure 10. A sample of the cavity with the dimensions A, B, C
In the Control Panel, there was made letters indicators for the cavity dimensions in mm: A, B and C, and valves which end the machine HPWR: E and D (bottom valve E, which is closed at the end of the process, in order to protect it from external contamination and valve D that protects interior cavity against contamination during its transport to the final destination at the end of the treatment).

Machine cycle consists a few modes:
- **Base Cleaning** – a program for cleaning (washing) machine up to the first (lower) valve, which is an integral part of the machine design,
- **Pre-Rinsing** – a program for cleaning (washing) machine up to the start of the cavity (a few millimeters above the first valve),
- **Main Rinsing** – the cycle due to design of cavity is divided into three parts: A, B and C; the length of each segment and the speed of the move of rinsing head defines an operator on the operator panel,
- **Homing** – mode is used to set zero of head positioning (zero vertical and zero rotation),
- **Manual** – manual control of movement and rotation of the rinsing head.

The pre-rinsing cycle consists of the Base Cleaning subprogram, and then continued cleaning up to the top of the machine (rinsing head goes to Home position when the top position of the machine (the end of E area) has been reached).

The purpose of the Pre-Rinsing cycle is to avoid dirt from the machine to get into the cavity.
The algorithm of the program for **Base Cleaning** cycle is as follows:

- **START**
  - if Option Mode is nr: 1.0 (“BASE CLEANING”)?
    - **NO**
    - **YES**
      - go 2.5 mm up with turning a head
        - wait 4 seconds
      - go 2.5 mm up with turning a head
        - wait 2 seconds
      - go back “Home” position

**STOP**
The algorithm of the program for **Pre-Rinsing** cycle is as follows:

```
START

- if Option Mode is nr: 2.0 ("PRE-RINSING")?
  NO
  YES

  Repeat 5 times the loop:

  - if position of the head is below E size declared on control panel?
    NO
    YES
    - go 2 mm down with turning a head
      - wait 2 seconds
    
    - go 2 mm up with turning a head
      - wait 2 seconds

  - if the speed of the head is bigger than 0?
    NO
    YES
    - turn the head 180 degrees clockwise
      - wait 1 second
      - turn the head 180 degrees anti-clockwise

STOP
```

Settings in Traversing Blocks in Siemens STARTER program for Servo_Up motor

Settings in Traversing Blocks in Siemens STARTER program for Servo_Turn motor
The algorithm of the program for **Main Rinsing** cycle is as follows:

1. **START**
2. - if Option Mode is nr: 3.0 ("MAIN RINSING")?
   - NO
   -YES
3. - go 2 mm down with turning a head until position: D+E
4. - go 2 mm up with turning a head to the top (position: A+B+C+D+E)
5. - if the position of the head is bigger then: D+E?
   - NO
   -YES
6. - turn the head 180 degrees clockwise
   - wait 1 second
   - turn the head 180 degrees anti-clockwise
7. **Repeat 2 times the loop:**
8. - it was already done in two cycles?
   - NO
   -YES
9. - go 2 mm down with turning a head until position: D+E
10. - go 2 mm up with turning a head to the top (position: A+B+C+D+E)
11. **STOP**
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”)

Figure 11. A General view of the Traversing Block for FC_Servo_Up (motor lifting the head in vertical)
in binary. For example: to jump to position 13 is needed to set bits of: the 1st, 3rd and 4th:

"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:
- **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 (funcionality 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 HPWR in Siemens STEP 7 program](image-url)
The program (function FC_Servo_Turn) is divided into five sections for the implementation of the machine cycles:

- **Base Cleaning**  -> Option Mode 1
- **Pre-Rinsing**   -> Option Mode 2
- **Main Rinsing**  -> Option Mode 3
- **Homing**        -> Option Mode 4
- **Manual**        -> Option Mode 5

are used Option Modes of PCO object in Unicos-PVSS. During choosing a duty cycle value of \texttt{HPWR\_B118\_PCO\_OpMoX} is set (for 1,2,3,4,5 or 8).

If the value of \texttt{HPWR\_B118\_PCO\_OpMoX} will be set to 8.0, the machine is stopped.

```plaintext
//********************Servo logic******************************************
//Servo_Turn

FUNCTION FC_Servo_Turn : VOID

VAR

loop : DINT;
clock : BOOL;
anticlock : BOOL;
up : BOOL;
down : BOOL;
a : DINT;  // cavity A size in mm
b : DINT;  // cavity B size in mm
c : DINT;  // cavity C size in mm
d : DINT;  // cavity D size in mm
e : DINT;  // cavity E size in mm

END_VAR

a := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Size_Cav_A.CurValSt);
b := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Size_Cav_B.CurValSt);
c := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Size_Cav_C.CurValSt);
d := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Up_Val_Lgth.CurValSt);
e := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Pre_Rins_Area.CurValSt);

TOF.DB_TOF_T1(IN:=("PZD_Turn".Receive.XIST_A < 180),PT:=T#5s);
TOF.DB_TOF_T2(IN:=("PZD_Turn".Receive.XIST_A > 0),PT:=T#5s);
```
IF (HPWR_B118_PCO.OpMoX = 8.0) THEN
   //init
   "PZD_Turn".Send.STW1.OFF2_CoastStop := TRUE;
   "PZD_Turn".Send.STW1.OFF3_QuickStop := TRUE;
   "PZD_Turn".Send.STW1.EnOp := TRUE;
   "PZD_Turn".Send.OVERRIDE := 1000;
   "PZD_Turn".Send.STW1.OFF1 := FALSE;
   "PZD_Turn".Send.STW1.IntStop := TRUE;
   "PZD_Turn".Send.STW1.RejTrvTask := TRUE;
   "PZD_Turn".Send.POS_STW1.TrvBit_0 := FALSE;
   "PZD_Turn".Send.POS_STW1.TrvBit_1 := FALSE;
   "PZD_Turn".Send.POS_STW1.TrvBit_2 := FALSE;
   "PZD_Turn".Send.POS_STW1.TrvBit_3 := FALSE;
   "PZD_Turn".Send.POS_STW1.TrvBit_4 := FALSE;
   "PZD_Turn".Send.POS_STW1.TrvBit_5 := FALSE;
   "PZD_Turn".Send.STW1.ExtBlkChange := FALSE;
   "PZD_Turn".Send.STW1.TrvTStart := FALSE;
END_IF;
loop := 1;
clock := DB_DOC_ALL.DOC_SET.HPWR_B118_DOC1.PosSt;
anticlock := DB_DOC_ALL.DOC_SET.HPWR_B118_DOC2.PosSt;

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 Base Cleaning---------------------
   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;
      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;

CERN | Documentation of High Pressure Water Rinsing machine
```plaintext
END_IF;
//-----------------------------for mode 2 Pre Rinsing---------------------

IF (HPWR_B118_PCO.OpMoX = 2.0) THEN

IF GlobalVariables.cycleNumber < 5 THEN

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

ELSIF "PZD_Turn".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 Main Rinsing---------------------

IF (HPWR_B118_PCO.OpMoX = 3.0) THEN

IF (GlobalVariables.cycleNumber < 2 AND "PZD_Up".Receive.XIST_A > 1000*(e+d)) THEN

"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_Turn".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.POS_STW1.TrvBit_1 := TRUE;
"PZD_Turn".Send.POS_STW1.TrvBit_2 := TRUE;
"PZD_Turn".Send.POS_STW1.TrvBit_3 := TRUE;
"PZD_Turn".Send.STW1.ExtBlkChange := TRUE;
"PZD_Turn".Send.STW1.TrvTStart := TRUE;

END_IF;

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

IF (HPWR_B118_PCO.OpMoX = 5.0) THEN

IF "PZD_Turn".Receive.XIST_A >= -1 AND "PZD_Turn".Receive.XIST_A <= 181 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.OFF1 := FALSE;
"PZD_Turn".Send.STW1.Jog1 := FALSE;
"PZD_Turn".Send.STW1.Jog2 := FALSE;

END_IF;

END_IF;

END_IF;

END_FUNCTION

//Servo_Up
FUNCTION FC_Servo_Up : VOID

VAR

loop : DINT;
clock : BOOL;
anticlock : BOOL;
up : BOOL;
down : BOOL;
a : DINT; // A size defined in mm
b : DINT; // B size defined in mm
c : DINT;       // C size defined in mm
d : DINT;       // D size defined in mm
e : DINT;       // E size defined in mm
f : INT;        // E speed
g : INT;        // D speed
h : INT;        // C speed
i : INT;        // B speed
j : INT;        // A speed
k : INT;        // speed in Manual mode
l : INT;        // speed in Home mode
returnFlag : BOOL;

END_VAR

a := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Size_Cav_A.CurValSt);
b := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Size_Cav_B.CurValSt);
c := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Size_Cav_C.CurValSt);
d := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Up_Val_Lgth.CurValSt);
e := REAL_TO_DINT(DB_APAR_all.APAR_SET.HPWR_B118_Pre_Rins_Area.CurValSt);
f := REAL_TO_INT(DB_APAR_all.APAR_SET.HPWR_B118_Velo_in_E.CurValSt);
g := REAL_TO_INT(DB_APAR_all.APAR_SET.HPWR_B118_Velo_in_D.CurValSt);
h := REAL_TO_INT(DB_APAR_all.APAR_SET.HPWR_B118_Velo_in_C.CurValSt);
i := REAL_TO_INT(DB_APAR_all.APAR_SET.HPWR_B118_Velo_in_B.CurValSt);
j := REAL_TO_INT(DB_APAR_all.APAR_SET.HPWR_B118_Velo_in_A.CurValSt);
k := REAL_TO_INT(DB_APAR_all.APAR_SET.HPWR_B118_Velo_Manual.CurValSt);
l := REAL_TO_INT(DB_APAR_all.APAR_SET.HPWR_B118_Velo_Home.CurValSt);

IF (HPWR_B118_PCO.OpMoX = 8.0) THEN

// init
"PZD_Up".Send.STW1.OFF2_CoastStop := TRUE;
"PZD_Up".Send.STW1.OFF3_QuickStop := TRUE;
"PZD_Up".Send.STW1.EnOp := TRUE;
"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;
END_IF;

//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;
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 Base Cleaning---------------------

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 Pre Rinsing---------------------

IF (HPWR_B118_PCO.OpMoX = 2.0) THEN

"PZD_Up".Send.OVERRIDE := 260*f;

IF GlobalVariables.cycleNumber < 5 THEN

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

IF ("PZD_Up".Receive.XIST_A < e*1000 AND GlobalVariables.returnFlag = FALSE) THEN

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

"PZD_Up".Send.MDI_VELOCITY := 1000;
"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
    "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;

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 <= 1000*e THEN
      "PZD_Up".Send.OVERRIDE := 260*f;
    END_IF;
  END_IF;
END_IF;

//-------------------------------for mode 3 Main Rinsing--------------
//-----------------------------for mode 3 Main Rinsing-----------------

IF "PZD_Up".Receive.XIST_A > 1000*e AND "PZD_Up".Receive.XIST_A <= 1000*(e+d) THEN
  "PZD_Up".Send.OVERRIDE := 260*g;
END_IF;

IF "PZD_Up".Receive.XIST_A > 1000*(e+d) AND "PZD_Up".Receive.XIST_A <= 1000*(e+d+c)
THEN
  "PZD_Up".Send.OVERRIDE := 260*h;
END_IF;

IF "PZD_Up".Receive.XIST_A > 1000*(e+d+c) AND "PZD_Up".Receive.XIST_A <=
1000*(e+d+c+b) THEN
  "PZD_Up".Send.OVERRIDE := 260*i;
END_IF;

IF "PZD_Up".Receive.XIST_A > 1000*(e+d+c+b) AND "PZD_Up".Receive.XIST_A <=
1000*(e+d+c+b+a) THEN
  "PZD_Up".Send.OVERRIDE := 260*j;
END_IF;

IF ("PZD_Up".Receive.XIST_A < 1000*(d + e) + 1000*(a + b + c) AND GlobalVariables.returnFlag = FALSE) THEN // if not on the top and not return
  IF ("PZD_Up".Send.STW1.RejTrvTask = TRUE) AND "PZD_Up".Receive.XIST_A < 1000*(d + e) - 4000 THEN // if below cavity go to the top
    GlobalVariables.returnFlag := FALSE;
    GlobalVariables.resetFlag := FALSE;
    "PZD_Up".Send.POS_STW1.TrvBit_1 := TRUE;
    "PZD_Up".Send.POS_STW1.TrvBit_2 := TRUE;
    "PZD_Up".Send.POS_STW1.TrvBit_4 := 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 >= 1000*(d + e) - 4000 THEN // if in cavity cyclic go at 2 mm up
    "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 // if on the top and resetFlag = FALSE -> stop all
"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

 IF "PZD_Up".Receive.XIST_A >= -2000 AND "PZD_Up".Receive.XIST_A < 1000*(a + b + c + d + e)
 THEN

 "PZD_Up".Send.OVERRIDE := 260*k;
 "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;

////////////////////////////////////////////////////////////////////////
//------------------------------- 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)).
2. Turn the Main Switch 3-phase power:
3. Turn on the computer and logging as:
   Login: \textit{hpwr118}
   Password: \textit{118HPWRsystem082013}

4. Turn on \textit{PVSS Project Administration} (the icon on Desktop)
   and choose \textit{HPWR} project and then \textit{Start Project} icon:
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).

### 3.8 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.

Remotely turn-on the valves: Gas Cavity, Cavity Blow Gas and Gas Lower Level.

Before using the system, you must to define cavity dimensions in millimeters: size A, B, C, and the length of valves: upper valve - D size and lower valve - E size. For each area of work: A, B, C, D and E of the machine, you must specify the speed of head in mm/s (by double clicking on the green number for the currently set speed and enter a new value).
To run the system, it is necessary to select the mode of operation: Base Cleaning, Pre-Rinsing, Main Rinsing, Homing or Manual. 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.

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
- Technical drawings of mechanical equipment (only electronic version; *.hpgl files)

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