IFM O2I100, IFM O2D220 & Siemens S7-1200

The O2I100 and O2D220 are both connected to the S7-1200 via the internal profinet interface of the CPU. In the S7-1200 program, TIA Portal, the program block O2Ix_PN and O2Dx_PN libraries allow an easy way to activate a parameter set saved in the sensors, and to receive the responsive data from the triggered sensors.

1.1.1 Installation

Both sensors are connected to the internal Profinet interface of the S7-1200 CPU using an Ethernet cable, M12/RJ45, 4 poles connector. If the sensors are directly connected to the interface then a crossover cable is recommended. The next step is to connect TIA Portal to the S7-1200 by changing to [Device view] register in Device configuration. Next open the properties of the Profinet interface and in the IP protocol section set the IP address and Subnet mask of the interface.

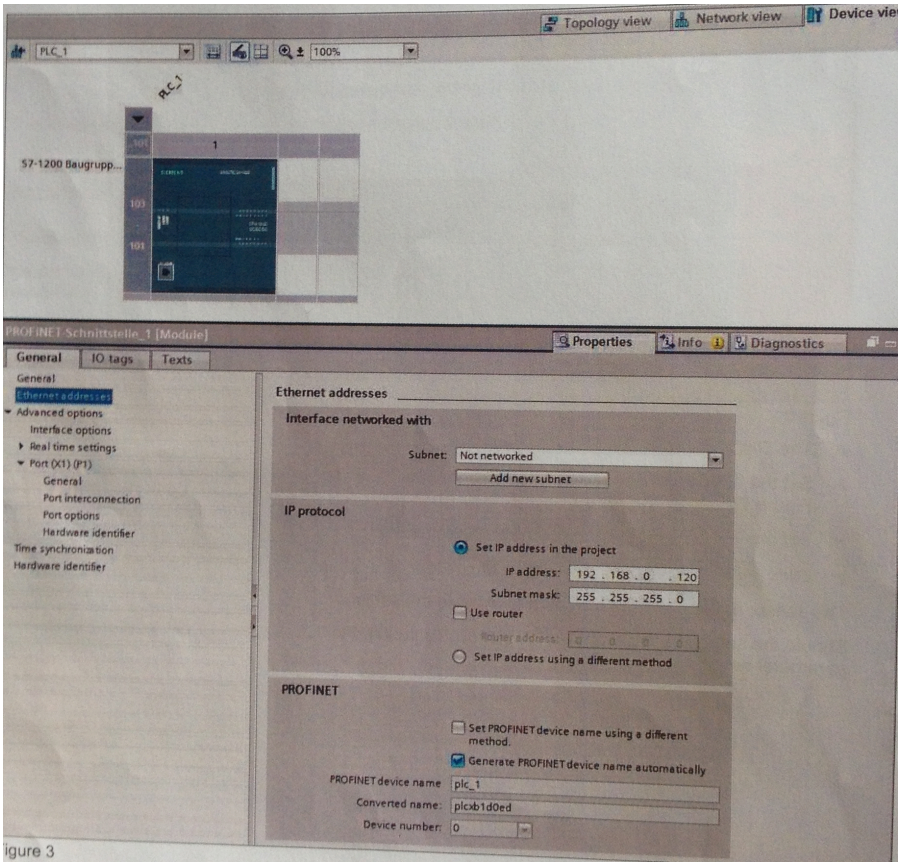


Figure 1 Profinet Interface

1.1.2 Programming

To be independent of existing TIA Portal software, the libraries are delivered as SCL source files, which we downloaded from an IFM support page http://www.ifm.com/ifmus/web/pdownloads020_010_050.htm, and from these two files the program

blocks can be generated. In the external source files tab, click [Add new external file], browse to the source and add it to the project. Then right click on the added source file and click [Generate blocks from source]. The program blocks will be generated by TIA Portal. Each library contains a PLC data type, a function and a function block.

The function blocks are the core of the program blocks in the library. It contains the main application and provides the use of functions like riggers to the sensors. The function uses Siemens function blocks TSEND, TRCV and TCON.

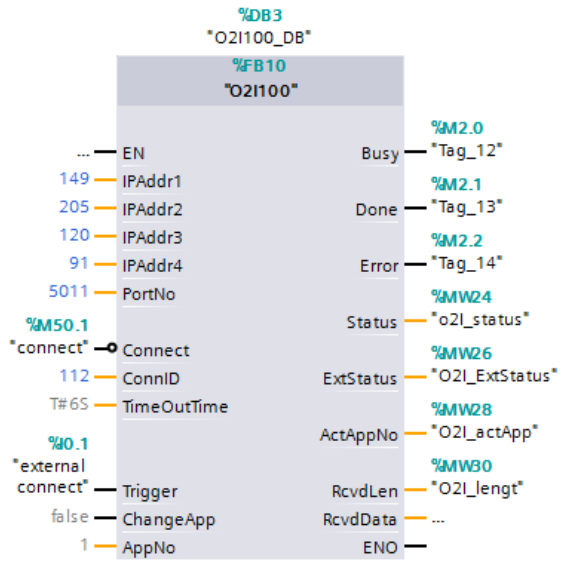


Figure 2 O2I100 Function Block

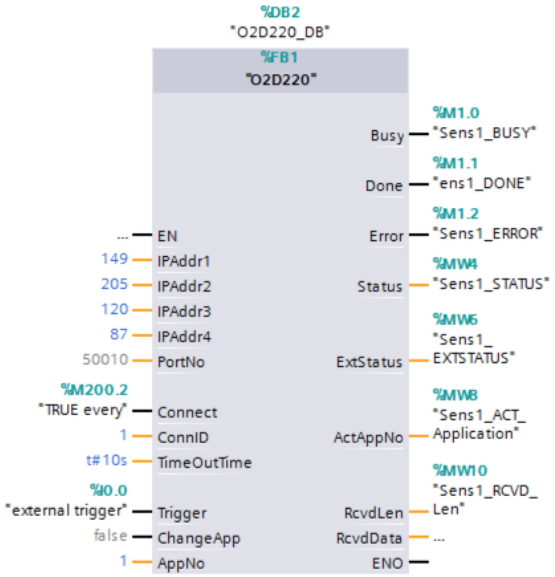


Figure 3 O2D220 Function Block

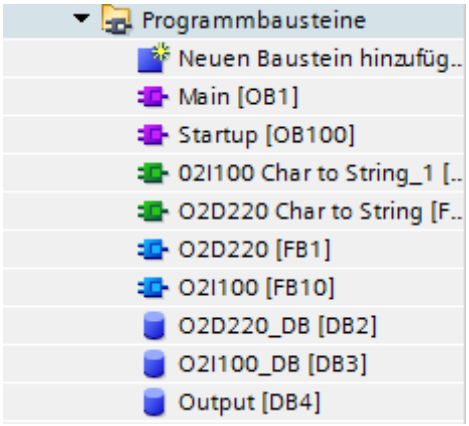


Figure 4 Function Blocks In TIA Portal

Once the output of a sensor is received it is held in a data block in char format. From here it must first be converted to a string and saved to a memory bit. Now it is snipped so we are left with the information we desire. For the O2I100, we only leave the contents of the QR codes which is the number of the tile which is then converted into an integer. And finally because we are left with a

number between 01 and 16 we must subtract one from this final outcome each time the sensor has been triggered so that we can use a
4-bit binary output (0-15) to control the RV-E2 robotic arm.

```
1  // CHAR to STRING
2  Chars_TO_Strg(Chars := "O2I100_DB".RcvdData,pChars := 0,Cnt := 64,Strg => "Output"."O2I100 String");
3
4  // Find Model
5  "Output"."O2I100 Model" := MID(IN := "Output"."O2I100 String", L := 2, P := 6);
6
7  // String to Int
8  STRG_VAL(IN := "Output"."O2I100 Model",
9           FORMAT := 00,
10          P := 1,
11          OUT => "Output"."O2I100 Out");
12
13 #werd1:= "Output"."O2I100 Out";
14 STRG_VAL(IN := "Output"."O2I100 Model",
15          FORMAT := 00,
16          P := 1,
17          OUT => "Output"."O2I100 Out2");
18
```

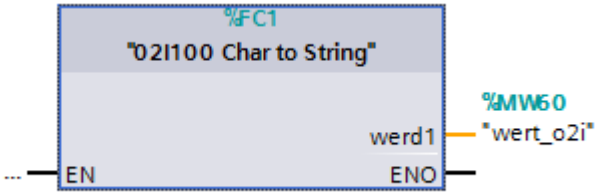


Figure 5 Char to String SCL & FB O2I100 Example

The same is done with the information in the O2D220 data block, first the tile number is converted into an integer, one subtracted to allow binary counting 0-15 instead of 1-16. Next the percentage match is converted to a float or a decimal integer as TIA Portal refers, and finally the full string is stored so that all the data can be displayed on a HMI later.

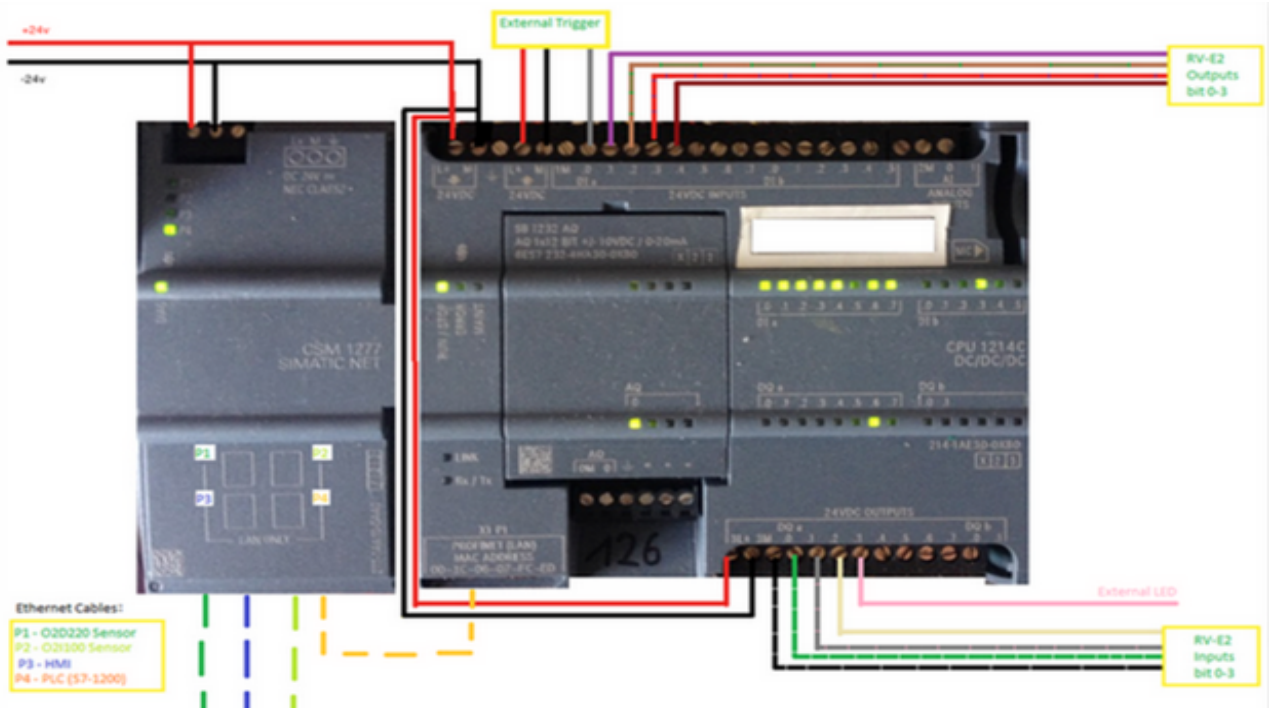
After both strings have been received the converted and trimmed down the two tile number integers are converted once more into a word and the two are compared as they should match.

```
// CHAR to STRING
Chars_TO_Strg(Chars := "O2D220_DB".RcvdData,
              pChars := 0,
              Cnt := 64,
              Strg => "Output"."O2D220 String");

//Find Model
"Output"."O2D220 Model" := MID(IN := "Output"."O2D220 String", L := 2, P :=21);
// Percentage Error
"Output"."O2D220 Percentage error" := MID(IN := "Output"."O2D220 String", L := 4, P := 12);

//String to Integer
STRG_VAL(IN := "Output"."O2D220 Model",
         FORMAT := 00,
         P := 1,
         OUT => "Output"."O2D220 Out2");
//output = output2 - 1, to fit binary
"Output"."O2D220 Out" := "Output"."O2D220 Out2" -1;

#werd2 := "Output"."O2D220 Out";

//String to Integer
STRG_VAL(IN := "Output"."O2D220 Percentage error",
         FORMAT := 00,
         P := 1,
         OUT => "Output"."O2D220 Percentage error int");

```

Figure 6 Char to String SCL O2D220 Example

1.2 Siemens S7-1200 & Mitsubishi RV-E2 & CR-E116

From previously saving the output, tile number, of the sensor as a 4-bit word in a memory bit, after receiving a match from the comparison, we now set these as the outputs of the plc using 4-bit binary, using an internal conversion method, storing the words in MW%60 and MW%62. There are 4 inputs of the RV-E2 linked to 4 outputs of the S7-1200, and also 4 outputs of the RV-E2 which are are connected to 4 inputs of the the S7-1200. The outputs of the RV-E2 are switched internally by the program that the arm is running, these outputs of the RV-E2 have been set, one to trigger each sensor when in the correct position, internally within the PLC. A third output bit was used from the RV-E2 and was set to control the external LED used for the lighting of the IFM O2D220 object recognition



sensor.

Figure 7 S7-1200 Wiring Diagram

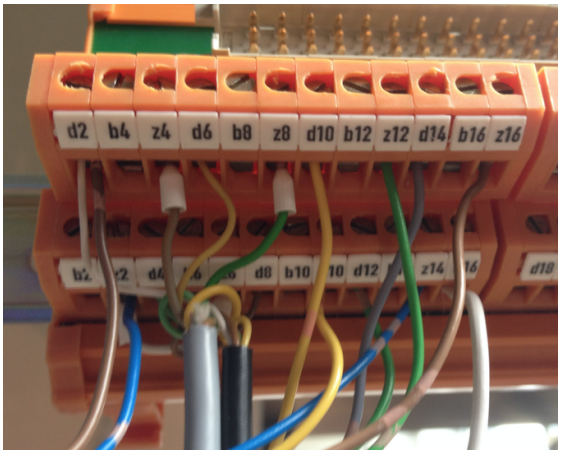


Figure 8 CR-E116 Controller Box Connection Block

For our final program for the RV-E2 Arm and CR-E116 Controller Box we decided, as the output of the S7-1200 PLC is a 4 bit binary number signalled by 4 output cables connected to the 4 input signal bits of the controller box, to use a series of loops to decode the binary input.

There are 16 cycles in total, one for each tile. Each cycle, while picking up each tile we drop the Z axis a further 3mm to reach further into the card holder. This is done with the use of a counter. Binary decoding worked as, first all inputs were read, then in order of 0-3, each bit was checked if it was positive, and if so the program was ordered to jump to a new line number which would have commands to move to a particular position restarting the cycle, upon the last cycle the arm returned to the next position.

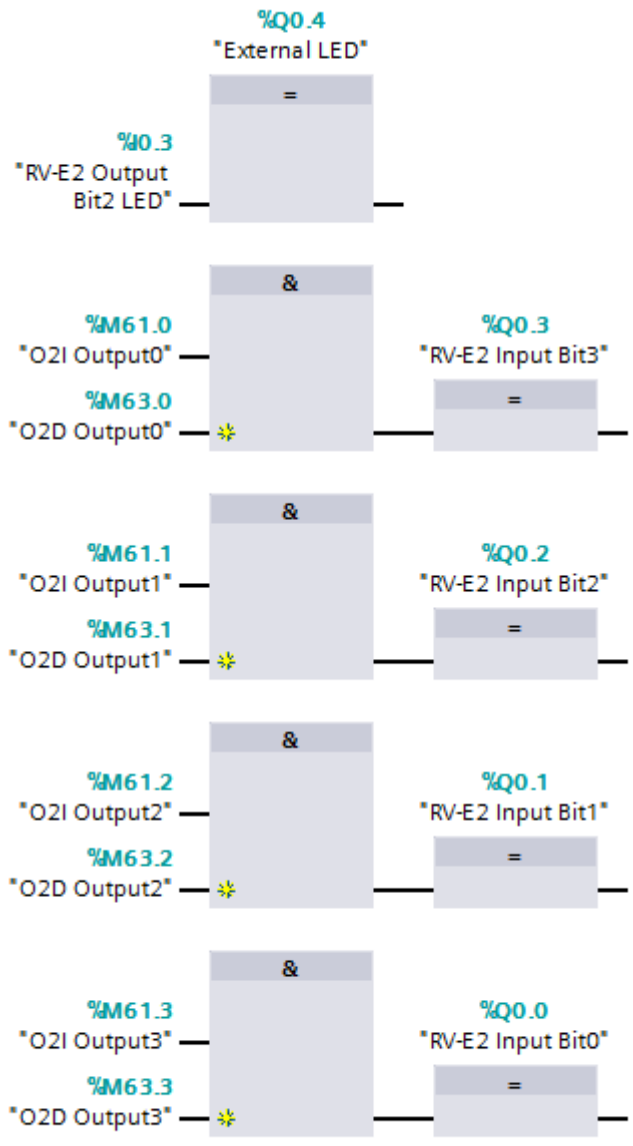


Figure 9 Matching The Outputs