UPGRADE AND EXTENSIVE HARDWARE AND SOFTWARE TESTS OF THE CMS ECAL DETECTOR FOR THE UPCOMING LS3

L. Cokić, P. Milenović, M. Mijić (University of Belgrade, Belgrade, Serbia), W. Lustermann, R. Jiménez Estupiñán, G. Dissertori (ETH, Zurich, Switzerland), A. Tsirou (CERN, Geneva, Switzerland), P. Giorgio Verdini (INFN, Pisa, Italy)

SOFTWARE UPGRADE OF THE PLC SYSTEM IN THE TEST LABORATORY

Software migration from Statement List (STL) to Structured Control Language (SCL) inside Siemens PLC

The CMS ECAL Safety System (ESS) software was organized in Functions (FCs) and Function Blocks (FBs) in SIMATIC STEP7. However, the software was written in STL which had a lot of disadvantages: it was not easy to maintain, to troubleshoot and most importantly, it was not written in the spirit of Object-Oriented Programming (OOP). Example of STL can be found in Figure 1, where the part of the old code is shown.

All this led to the consideration of rewriting the software in a more sophisticated way. To begin with, the functionalities are separated into meaningful units (in the form of functions and function blocks), the SCL language is used, which is clean, clear and easy to debug. Example of the SCL can be found in Figure 2, where the logic for reading and processing analogue sensors is shown.

Figure 3 shows the flowchart of the Safety System PLC program located in Organization Block 1 (OB1). OB1 contains all the control logic for entire system (all FCs and FBs). The operating system of the S7 CPU executes OB1 periodically. When OB1 has been executed, the operating system starts it again.

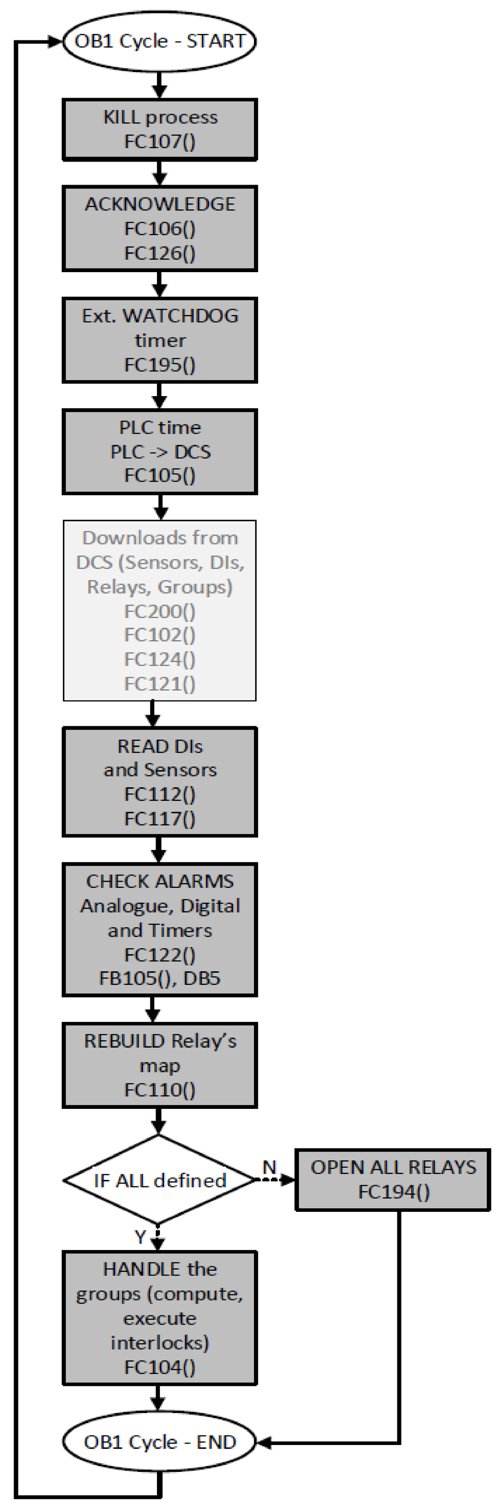

Figure 3: ESS flowchart.

```
//————————————————————————————————————————
//    Get #Cooling_Valve_Byte from the ESS_CONTROL_DB
//————————————————————————————————————————
      CALL  "ESS_get_indirect_address"
       ADDR_BYTES_OFFSET :="ESS_INTERLOCK_DATA_DB".Cooling_Valve_UPOS
       ADDR_STEP_NUMBER  :=0
       ADDR_STEP_SIZE    :=1            // #Cooling_Valve_Byte size in bytes
       ADDR_BYTE_POSITION:=#temp_addr_byte_position
      L     DBB [AR1,P#0.0]
      T     #Cooling_Valve_Byte
//————————————————————————————————————————
// Check #Sensor_Intlk_Byte and #External_Intlk_Byte
//————————————————————————————————————————
      L     #External_Intlk_Byte
      L     #Sensor_Intlk_Byte

      OW
      JZ    end                         // jump to the end if no alarm bit is set in status words
//————————————————————————————————————————
//    Close Cooling Valv if WLD has been detected
//————————————————————————————————————————
      L     #External_Intlk_Byte
      L     B#16#10
      AW
      JZ    nvlv

      CALL  "ESS_status_byte_update"    // write to     #HV_Intlk_Byte
       status_bit      :=TRUE
       bit_num         :=#SM_NUM
       status_mask_byte:=B#16#FF
       status_byte     :=#Cooling_Valve_Byte
//————————————————————————————————————————
//    Set #Cooling_Valve_Byte from the ESS_CONTROL_DB
//————————————————————————————————————————
      CALL  "ESS_get_indirect_address"
       ADDR_BYTES_OFFSET :="ESS_INTERLOCK_DATA_DB".Cooling_Valve_UPOS    // #Cooling_Valve_Byte start byte
       ADDR_STEP_NUMBER  :=0
       ADDR_STEP_SIZE    :=1            // #Cooling_Valve_Byte size in bytes
       ADDR_BYTE_POSITION:=#temp_addr_byte_position
      L     #Cooling_Valve_Byte
      T     DBB [AR1,P#0.0]

      L     2
      ITD
      L     #SM_NUM
      ITD
      +I
      L     4
      ITD
      MOD
      L     4
      *I
      L     W#16#FFFE                   // Interlock "pattern": set LV Int (1), LV Cut (1), HV Int (1), Cool(0)
      ITD
      RLD
      L     QW     0                    // set Cool Vlv
      AW
      T     QW     0                    // set Cool Vlv


nvlv: SET
```

Figure 1: Piece of the old code in STL (cooling interlock logic).

```
FUNCTION FC117: INT

// Reads analog sensors and writes them in the array of sensor structure in DB100.
// The number of sensors to read is reported in DB100.NumbofEntries.
// The arrays to be filled, whose number is reported in DB100.NumbofEntries are consecutive.

CONST
  maxADC    := 27648.0;
  overShoot :=   32512; // Start of the overshoot range for the Analogue Input module
  typeNTC   :=      47; // NTC Thermistor
  typeWLD   :=      48; // Water Leak Detector
  MAXCRIT   :=       8; // Maximum index for sensors generating Critical alarm
END_CONST

 FOR iSensor := 1 TO DB100.NumbofEntries BY 1 DO

    WhichIW := DB100.PLCData[iSensor].SensorID;  // Get the ID of the sensor
    myType  := BYTE_TO_INT(DB100.PLCData[iSensor].SensorType);  // Also the type

    iValue  := WORD_TO_INT(PIW[WhichIW]);  // The Sensor PIW is given by the sensor ID.

    CASE myType OF
    typeNTC :
      rResist := INT_TO_REAL(iValue);  // Do not rescale at all
      // OverShoot protection START
      IF iValue ≥ overShoot THEN
        DB100.PLCData[iSensor].ChannelWireBr :=  TRUE;
        DB400.PLCData[iSensor].ChannelWireBr :=  TRUE;
        IF iSensor > MAXCRIT THEN
          rResist := INT_TO_REAL(DB100.PLCData[iSensor].LowAlThresh - 1);
        END_IF;
      ELSE
        DB100.PLCData[iSensor].ChannelWireBr := FALSE;
        DB400.PLCData[iSensor].ChannelWireBr := FALSE;
      END_IF;

    ELSE:
      rResist := INT_TO_REAL(iValue); // Do not rescale at all for Pt1000.
      IF iValue ≥ overShoot THEN
        DB100.PLCData[iSensor].ChannelWireBr :=  TRUE;
        DB400.PLCData[iSensor].ChannelWireBr :=  TRUE;
      ELSE
        DB100.PLCData[iSensor].ChannelWireBr := FALSE;
        DB400.PLCData[iSensor].ChannelWireBr := FALSE;
      END_IF;

    END_CASE;

    dResist := ROUND(rResist);  // Round to nearest even integer as per  DIN EN 61131-3
    iResist := DINT_TO_INT(dResist);  // ...and make into an INT for WinCC OA

    DB100.PLCData[iSensor].SensorCounts := iResist;
    DB400.PLCData[iSensor].SensorCounts := DB100.PLCData[iSensor].SensorCounts;

    iCount := iCount + 1;
  END_FOR;

  FC117 := iCount;

END_FUNCTION
```

Figure 2: One of the FCs written in SCL (read analogue sensors logic).

HARDWARE UPGRADE OF THE ECAL PLC SAFETY TEST SYSTEM

Installation and upgrade of the PLC system for the supermodule 36 (SM36), cooling unit and dry air system

Figure 4: PLC rack.

The setup in one of our laboratories is updated accordingly in order to satisfy all requirements for the extensive tests. The idea was to place the additional temperature (48 pt1000) and humidity (20 HYT271) sensors all over the supermodule so that we could monitor trends of the temperature, humidity and dew point through the entire space inside the supermodule in order to assess whether lowering the temperature of the supermodule from 18°C to 8°C is safe for the electronics (in the sense that it would not be condensation inside the supermodule) and to adjust high and low thresholds to fulfil the requirements.

Along with the upgrading of the PLC and additional PLC modules (see Figure 4), we also needed a dry air system installed next to the supermodule for the tests. Finally, the cooling system is wired to send and receive interlock signals to ensure that the safety system will react properly in case of the cooling problem.

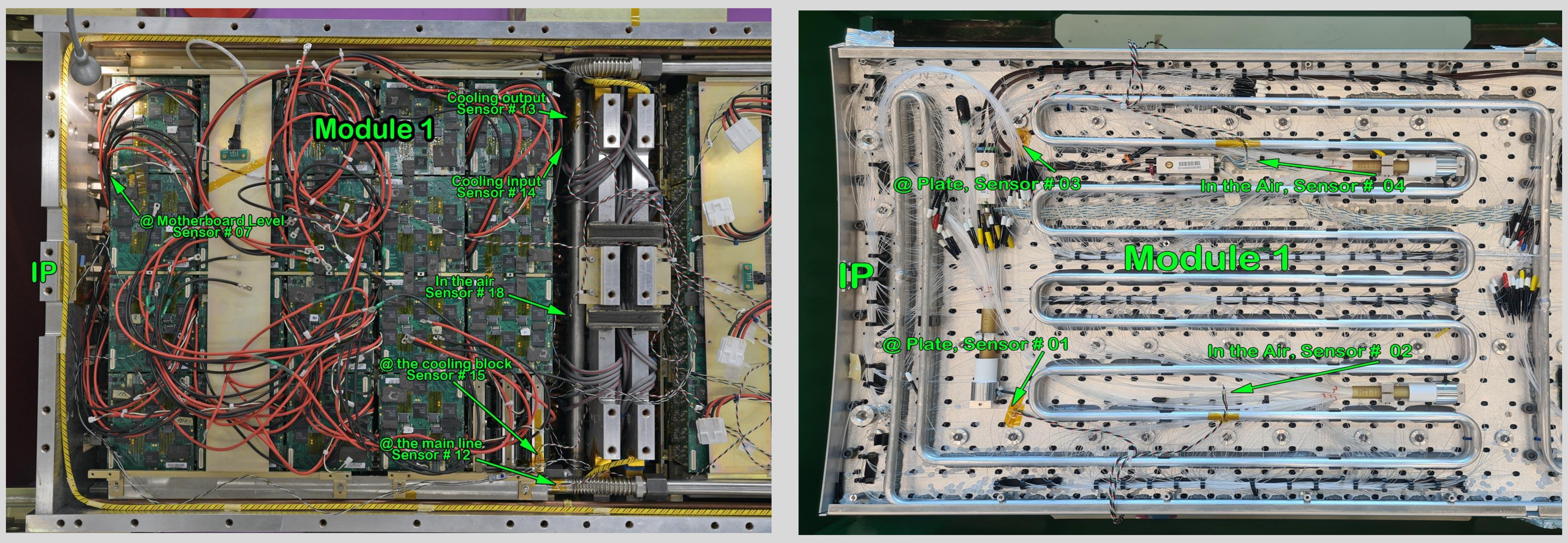

Figure 5: Additional pt1000 temperature sensors: electronics side (left) and thermal screen side (right).

Let's briefly explain how the ECAL safety system is working at the moment. In principle, one supermodule consists of four parts – called "modules", each of which houses the front-end electronics. As the front-end electronics is the main cause of heating, powered by low voltage (LV), there are two NTC temperature sensors in each module directly wired to ESS PLC. If one of the sensors exceeds the critical temperature, the system will react by interlocking the LV and sending the signal that overheating has occurred. This system works independently of any other and its main function is to ensure that in case of any problem, the system will react in the right and safe way.

Figure 5 shows the installation of the additional pt1000 temperature sensors both from the electronics and the thermal screen side in module 1 (the other three modules inside the SM36 are also arranged in a similar manner).

RESULT AND FINAL COMMENTS

Test results and explanations

You can see in Figure 6 the full period of the performed cooling tests, where the following capabilities and limits of the supermodule 36 were tested:

- Cooling of SM36 to 8°C (in small steps of a couple of degrees Celsius)
- Cooling in a single step from 21°C to 9°C and vice versa
- Drying of supermodule 36 using one injection pipe of the dry air
- Safety system reaction in case of different errors
- Monitoring the accuracy and the activation of interlocks for two different temperature sensor types independently – NTC and pt1000

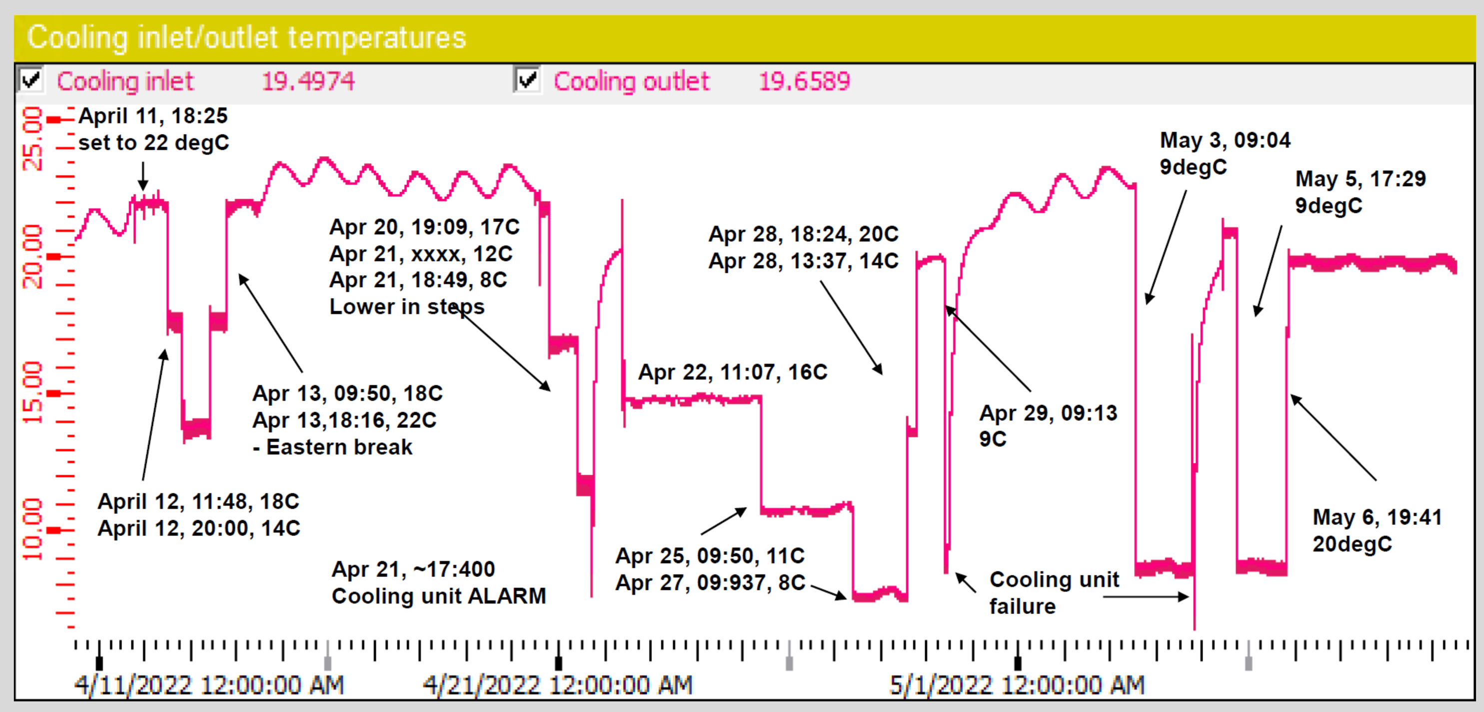

Figure 6: Time scale of the extensive tests on SM36.

The obtained results are different compared to the side of the supermodule 36, which was expected because the electronics side is more "open" than the thermal screen side (due to the fact that the dry air pipes are located on the electronics side) and accordingly the changes in temperature, humidity and dew point are much faster. On the electronics side, it took approx. 2 days to extract all the water from the air, while on the thermal screen side, it took about 10 days. Cooling of SM36 to 8°C and in the single step from 21°C to 9°C and vice versa were successful and the ESS followed all the changes without a single error. During the tests, we had a couple of situations where the cooling system failed, and the temperature was too high or too low, but every time the safety system reacted in the right way.