**SPI\_SENSOR LOGIC**

**Using variables**

**Readvc()**

* readvoltage 🡪 it is used to read the voltage value.
* read current 🡪 it is used to read the current value.
* vIndex 🡪 using index
* cIndex 🡪using index
* voltage 🡪 max voltage stored.
* current 🡪 max current stored.

**Readmcp3008()**

* result 🡪 is used to store the result of the ADC conversion.
* Firstbye 🡪 is used to store the first byte received from the MCP3008.
* configBits 🡪 this variable contains the Order of Configuration In Mcp3008

**ReadTemperature()**

* data[2] 🡪 Reading temp value MSB & LSB are stored.
* Temp variable 🡪 This variable contains the result of temp.

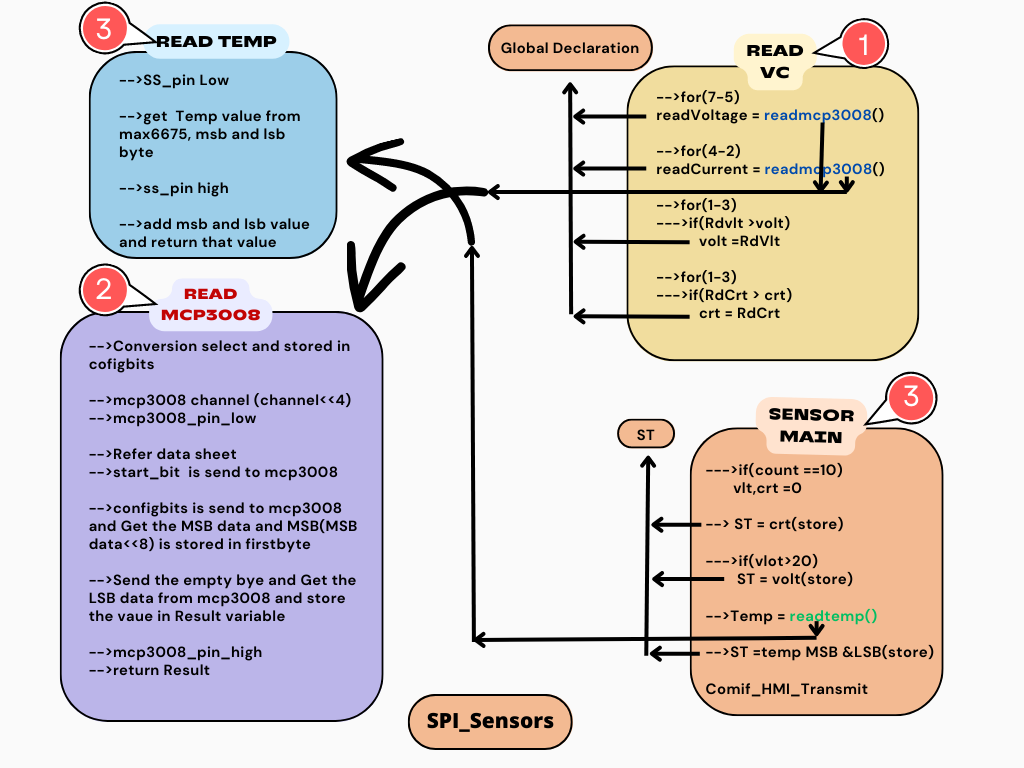
**Flow of States**

🡪ReadVC()(10ms task)

🡪ReadMCP3008()

🡪SensorMain()(1000ms task)

🡪ReadTemperature()

**BLOCK DIAGRAM**

**Functions explainations:**

1. **ReadVc() (10ms task)**

🡪This functions is called every 10ms task,

🡪In this function read data from an MCP3008 Analog-to-Digital Converter (ADC) integrated circuit (IC). It seems to be capturing voltage and current measurements and updating corresponding arrays Voltage and Current

🡪**Voltage and Current Reading Loops:**

🡪There are two loops, one for reading voltage and another for reading current. Both loops iterate through the input channels of the MCP3008.

**For voltage readings**

🡪It starts a loop with i initialized to 7 (the highest voltage channel) and iterates while i is greater than or equal to 5 (the lowest voltage channel).

🡪Inside this loop, it calls the readMCP3008 function with the current channel i and stores the result in the **readVoltage** array at index **Vindex**.

🡪It increments **Vindex** to move to the next index in the **readVoltage** array.

**For current readings**

🡪In this loop is used for reading current, with i initialized to 4 (the highest current channel) and iterates while i is greater than or equal to 2 (the lowest current channel).

🡪Inside this loop, it calls the readMCP3008 function with the current channel i and stores the result in the readCurrent array at index cIndex.

🡪It increments cIndex to move to the next index in the readCurrent array.

**🡪Voltage and Current Comparison and Update**

**🡪**After reading all voltage and current channels, there are two more loops for each (voltage and current) to compare the newly read values with the existing values stored in arrays Voltage and Current.

**🡪if(readVoltage[i] > Voltage[i])**

🡪If the new value is greater than the existing value in Voltage, it updates the value in the **Voltage** array.

🡪**if(readCurrent[i] > Current[i])**

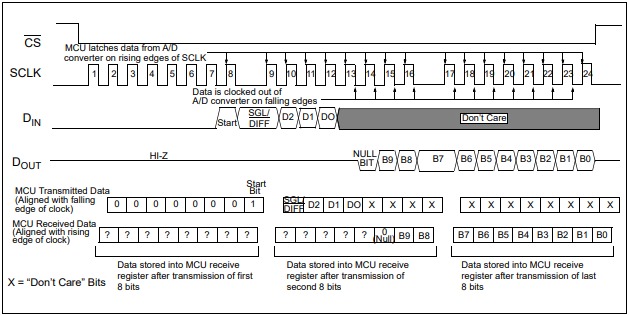
🡪If the new value is greater than the existing value in Current, it updates the value in the **Current** array.

1. **ReadMCP3008()**

* This function get the Voltage & current Data form 3 phase line using SPI protocol(MCP3008).
* Refer the data sheet.

**Data read from MCP3008**

* Refer to the datasheet, and find the configuration method.
* So first we config single-ended (0x80 in data sheet). and add the channel. that value is stored in configbits(spisensor).



* Select the ADC slave pin to LOW, refer to the diagram to get the Raw data from the mcp3008 module
* Start the conversion, like sending the start bit first(look like the diagram), next send the configbits, simultaneously to get the MSB byte data,It stored the local variable (FirstByte).
* Again send the empty byte and get the LSB byte simultaneously. The LSB byte stored the localvariable(result). They add two variables(result | FirstByte) to get the raw data.

1. **SensorMain()(1000ms task)**

**🡪**This Function is Called by every 1000ms Task,

**🡪**That is responsible for collecting and processing sensor data. It seems to work with various sensors, including voltage, current, and temperature sensors.

**🡪Counter Incrementation:**

🡪The variable Count is incremented at the beginning of the function, likely for tracking the number of times this function has been called**.**

**🡪Data Clearing on Count**

**🡪**If Count reaches a value of 10, a loop is used to clear some arrays and variables.

🡪Three elements of the **Voltage** array, **readVoltage** array, and **Current** array are set to 0.

🡪Count is reset to 0.

🡪 if the count variable reaches at the value of 10, the max voltage and current values are stored in the variables.

🡪 Temperature Sensor Reading:

🡪It reads the temperature using the **readTemprature** function.

🡪The temperature is then converted from bits to degrees Celsius by multiplying it by 0.25. The result is stored in ovenTemprature.

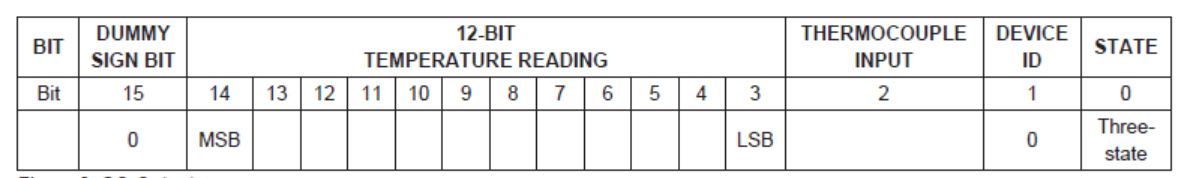
🡪 Sensor data is stored in the sensorData structure.

🡪 The sensor data is then transmitted using the **ComIf\_TransmitFromBuffer\_HMI\_sensorData()** function(shard ST data’s with HMI).

1. **Readtemperature()**

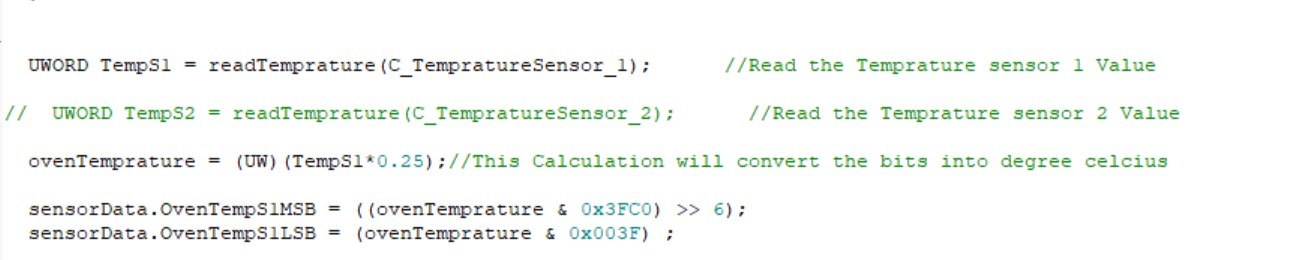
* This function get the OvenTemp Data From Inside oven using SPI protocol(Max6675).
* Refer Datasheet.

**Temp\_data \_Read**

* First, the slave pin set LOW and sends the empty byte to max6675. The temp sensor gives the MSB byte data, Once again sends the Empty data and max6675 returns to send the LSB data. Next, the slave pin is Set HIGH.
* Find the temp Raw data using the tabular column
* After getting the Raw data(bit-3 to bit-14).
* The result is stored in the **Temp** variable as a floating-point value.

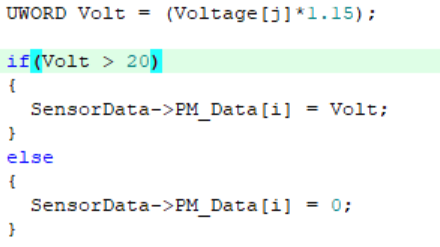
**CALCULATION PART FOR TEMPERATURE SENSOR(MAX6675)**

**This part of code (calculation) is used to convert the bits values that is readed from the sensor is converted into degree celcius. The temperature is then converted from bits to degrees Celsius by multiplying it by 0.25. The result is stored in ovenTemprature**.

****

**CALCULATION PART FOR VOLTAGE AND CURRENT**

* **SensorData->PM\_Data[i] = ((Current[i]\*0.08)\*4); // This part of code (calculation) is used to convert the bits values that is readed from the sensor is converted into current that is multiplied by 0.08 and 4 and stored it in a Data array.**
* **UWORD Volt = (Voltage[j]\*1.15); // this line is used to convert the bits into voltage**

****

* **This part of code (calculation) is used to convert the bits values that is readed from the sensor is converted into voltage that is multiplied by 1.15 and it stored it in a variable volt. And if the volt value is greater than 20 ,it will stored in an array Data.otherwise it will be set as 0.**