How the processing of Data from the Smartmeter works

Processing of the data is performed in the Loop of a Arduino-Style application.
There is a Class as a template for reading Data from the Smartmeter via Modbus.
The name of the Class is: Read_2_Inputregisters.h
For each measured value to be read a dedicated instance of this Class must be created.
The instances created in this example are:
readSummedCurrent and
readImportWork
The constructor has two parameters. The first parameter defines the Struct, where the read data are written to. The second parameter is the Register Address of the Smartmeter from where the values are fetched.
In the Setup part of the Arduino Style App each instance of the Read_2_Inputregisters class has to be initialized by calling its .begin method.
The .begin method has two parameters. The first is the used Modbus host, the second is the InitialReleaseTimespan in milliseconds. The instances for reading from the Smartmeter can not be read (unless it is forced as described later) before this time in ms is elapsed.
Read_2_Inputregisters has a method isReleased() to find out by polling in the Loop whether the Sensorvalue can be read.
If the releaseTime time is elapsed, the sensor value is read by calling the method .get_2_InputRegisters.
This method has three parameters. First: Modbus-Address of the Smartmeter, Second: ReleaseTime which shall be used after this call, Third: ForceState, defining wheter the read shall be performed despide the releaseTime is not yet elapsed.

When Sensor values ('Current' and 'Work') are read from the Smartmeter they are passed to a "DataContainer" by calling the setNewValues() method
The Datacontainer sums up the Sensor value for the current in Ampere over the time (AveragingTime is 1 minute) and calculates an average value until certain trigger conditions are met.
When at the end of the AveragingTime a trigger condition is met, the DataContainer signals that Data are ready to be read by setting the flag _hasToBeSent , which can read from the Arduino Loop by calling the Datacontainer.hasToBeSent() method.
The instance of the Class DataContainer is created by calling its constructor with three parameters.
First: the time in ms after which the Datacontainer shall release its Data in any case if no other conditions are met before.
Second: A deviation in percent from the preceding average current. If the deviation is more than i.e. 10% the _hasToBeSent flag is set.
Third: An absolute deviation in Ampere from the preceding average current.

If the call of .hasToBeSent() in the Arduino loop returns true, the method getSampleValuesAndReset() is called, which returns the average Current over the

designated time and the “Work” value in kWh at the end of the time period. This method additionally resets the DataContainer to receive new Sensor Data.

Flowchart Processing of Smartmeter Data