|  |  |
| --- | --- |
|  | Minervastraat 10-12  B-1930 Zaventem  Phone: +32 2 723 98 11  www.phoenixcontact.be |

**Creating a Code Creator package**

|  |  |
| --- | --- |
| Subject: | Creating a Code Creator package |
| Document code | Documentation |
| Date | 17/07/2023 |
| Status | First version |
| Revisienr document: | 01 |
| Prepared by: | beot01 |

Table of contents

[1 Document data 3](#_Toc140498229)

[1.1 Revision Overview 3](#_Toc140498230)

[2 Introduction to Code Creator 4](#_Toc140498231)

[2.1 Prior knowledge 4](#_Toc140498232)

[2.2 Supplies 4](#_Toc140498233)

[3 Structure of the device package 5](#_Toc140498234)

[3.1 Source files 5](#_Toc140498235)

[3.1.1 DataConversionFunctionBlocks 6](#_Toc140498236)

[3.1.2 DataTypeSheets 6](#_Toc140498237)

[3.1.3 SelectFunctionBlocks 6](#_Toc140498238)

[4 Description xml 7](#_Toc140498239)

[4.1 Device Parameters 7](#_Toc140498240)

[4.2 Description of the data points 7](#_Toc140498241)

[5 Functionblocks en Datatype sheet 9](#_Toc140498242)

[5.1 Starting point 9](#_Toc140498243)

[5.2 Datatype sheet 10](#_Toc140498244)

[5.3 Functions and function blocks 10](#_Toc140498245)

[5.3.1 ConversionMethod\_FC3 10](#_Toc140498246)

[5.3.2 SelectFunctionBlocks 11](#_Toc140498247)

# Document data

## Revision Overview

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Description | Status | Author |
| 01 | 17/07/2023 | First version |  | beot01 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Introduction to Code Creator

With the Code Creator, IEC 611131 code for PLCnext Engineer can be generated for setting up a modbus RTU or Modbus TCP connection.

Currently, the FC3, FC6, and FC16 features are supported.

In the Code Creator, devices are made available with the help of device packages. In the code creator the desired registers only need to be checked and the Code Creator itself will generate the necessary code to read this data from the device and convert it into the a standard data type, unit and format order.

The device packages contain all the necessary information about the device and the transformation methods that are needed to generate the necessary code with the Code Creator.

## Prior knowledge

To create a device package it is assumed that there is already some prior knowledge in the use of PLCnext Engineer and Code Creator.

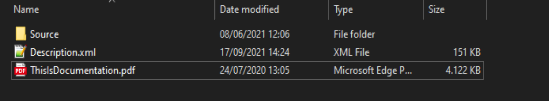
## Supplies

* PLCnext Engineer 2022.0.4 LTS
* Code creator
* Xml editor (notepad, notepadd++, ...)

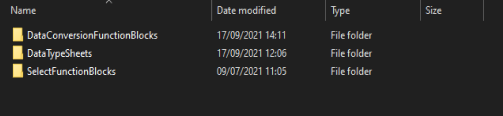
# Structure of the device package

A code creator device package consists of a zipped folder that contains a description xml, a pdf with documentation and a folder with the source files.

The pdf with documentation will usually be the user manual or the modbus registry table of the device. This must be present, but there is no obligation here about what the content of this pdf should be.

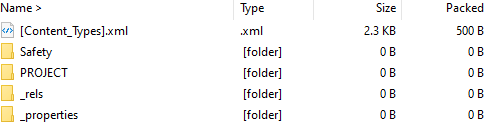


The source folder contains 3 child folders: the DataConversionFunctionBlocks folder, the DataTypesheets folder and the SelectFunctionBlocks folder. In these folders the necessary function blocks, data type sheets and methods must be placed to generate the Code Creator code. The DataConversionFunctionBlocks folder contains the FB and methods for converting the read in modbus data (FC3). The SelectFunctionBlocks folder contains the FB’s for sending the data via modbus (FC6 and FC16). This folder must also be present even if no data is sent.

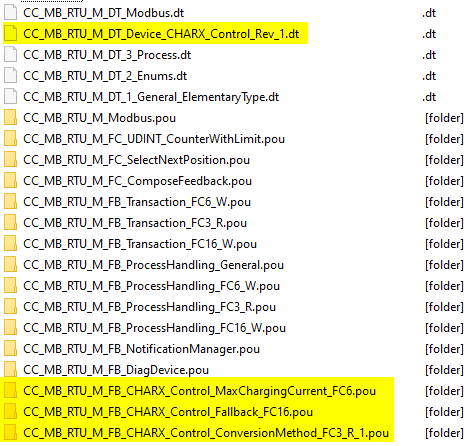


## Source files

The source files that have to be put in the above folders should be extracted from a PLCnext project, in the next chapter we will delve deeper in how this project can be set up. You can open a PLCnext Engineer project as an archive with the following result:



Under PROJECT > Logical%20Elements you can then select the different data types and function blocks to place in the source files. The files in the folder of the required function block must be zipped in order to be used in the source package.



The above example of a Charx controller shows the files to be copied into the source package. This consists of a data type sheet, 1 conversion function block and 2 selectfunctionblocks to send 2 registers via FC6 to the charging station. For FC 6, 1 selectfunctionblock is required per register to be forwarded. For FC16, multiple registers can be combined in 1 selectfunctionblock.

### DataConversionFunctionBlocks

This folder contains the necessary function blocks for reading a device via modbus and converting the data according to the values defined in the description xml.

### DataTypeSheets

This folder contains the data type sheets used that the project needs to build the modbus connection and transform the data.

### SelectFunctionBlocks

This folder contains the necessary function blocks to convert data to the correct format and forward it to a device via modbus.

# Description xml

The description xml describes the modbus module. The first part consists of a number of device parameters for display in the Code Creator software. Then the different data points to be read and written are described.

## Device Parameters

Line 3 in the screenshot below should be a unique UUID for the module.

Line 4 is a self-determined name for the module

Line 5 is the name that will be visible in the Code Creator software

Line 6 is the Manufacturer, which is the parent directory for the module in Code Creator

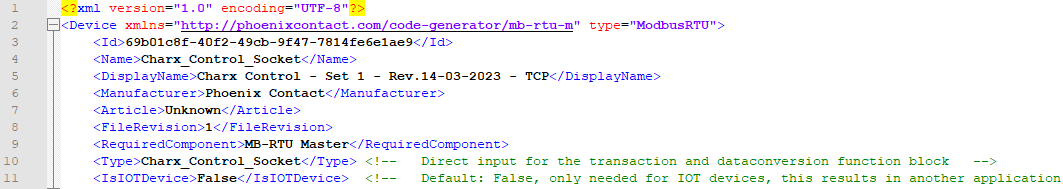
Line 7 is the article number of the device (if applicable, otherwise write “unknown”)

Line 8 is file revision number.

Line 9 is which RTU component is needed, here "MB-RTU Master" must be entered.

Line 10 is the type of device, this name is used for variable names in the generated code.

Line 11 does not apply and must be False.



## Description of the data points

The first part describes the function blocks that should be used for reading/reading the data and converting the data.

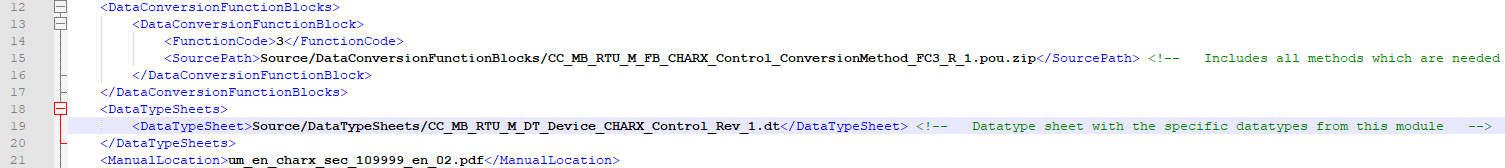
In the screenshot below:

Line 14 is the function code to be used

Line 15 is the conversion function block to be used.

Line 19 is the data type sheet to be used.

Line 21 is the documentation pdf of the package.



Then the data groups with them in the different transactions and data points are defined one by one.

Lines 22 in the screenshot below is the data group. Here you can insert the display name in Code Creator and the name of the struct in the generated code.

Each of these data groups contains some parameters (function code, start register, number of registers) and the data points.

Line 24 is a description of the data group

Line 26 contains the function code to be used and the suffix of the variables for that group, which must also be unique to this module.

Line 27 is the modbus starting address

Line 28 is the number of registers to be read.

Line 30 shows whether the data point is checked by default in the Code creator software.

Line 31 is a description of the data point, which is the display name in Code Creator

Line 32 is the variable name in PLCnext Engineer for this data point

Line 33 displays the data type of the data point

Line 35 is the convert method, it should be "ConvertDatapoint"

Line 36 is the method that the generated code should use to convert the value of the data point

Lines 38,39,40 and 41 are the parameters to be given to the conversion method for this data point. These consist of the type of conversion this is a self-made number corresponding to the conversion method, the offset of the start register to read out, by which factor the result should be multiplied and the unit that should have the result.

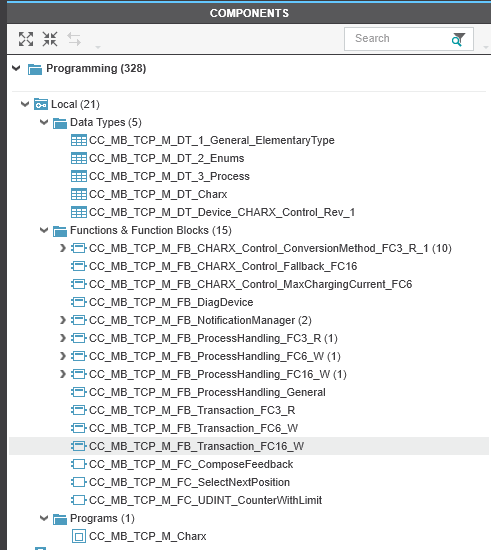


From line 45, the exercise is repeated for a second data point in the same transaction. It is also possible to make multiple datagroups and transactions. In this case you will have to make sure that the Structnames of the datagroups and the datagroupsuffixes of the transactions are unique for that device.

# Functionblocks en Datatype sheet

## Starting point

The easiest way to create a new device package is to start from an existing device package. This contains a number of data type sheets, a number of functions and function blocks and at least 1 program.



Most of these functions, function blocks and data type sheets are the same for each project that is generated and therefore device independent. These should therefore not be put in a device package.

For example: in the above example to create a new device package, the objects to be modified are:

* Datatype sheets
  + CC\_MB\_TCP\_M\_DT\_Device\_CHARX\_Control\_Rev\_1
* Features and function blocks:
  + CC\_MB\_TCP\_M\_FB\_CHARX\_Control\_ConversionMethod\_FC3\_R\_1
    - And the underlying methods
  + CC\_MB\_TCP\_M\_FB\_CHARX\_Control\_MaxChargingCurrent\_FC6
  + CC\_MB\_TCP\_M\_FB\_CHARX\_Control\_Fallback\_FC16
* The FC6 and FC 16 function blocks can be adjusted depending on whether you also need to write to the device.

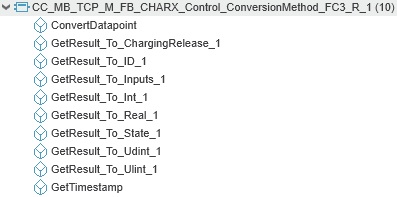
## Datatype sheet

In the datatype sheet, the data types are defined to store and display the read-in values after conversion.

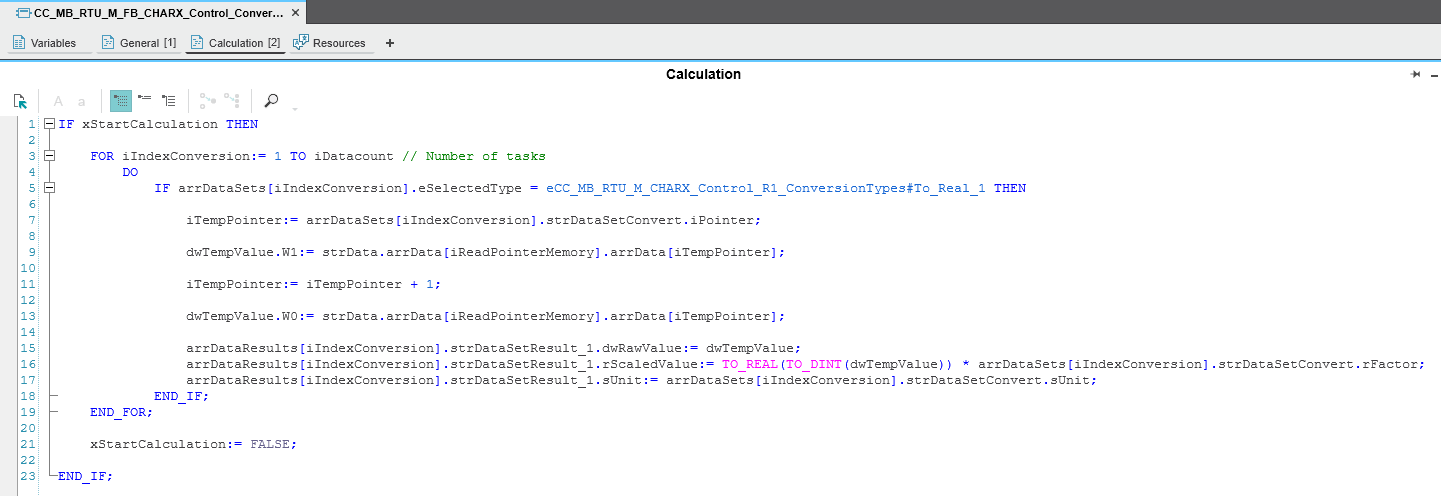
## Functions and function blocks

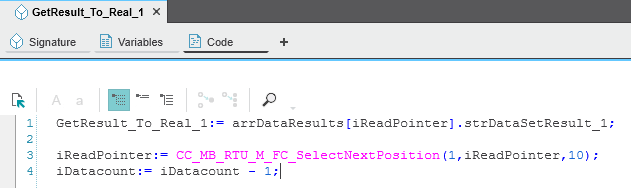
### ConversionMethod\_FC3

In the conversionMethod, the read-in modbus registers are transformed and scaled to the required value. For this, a calculation must be done in the second tab "Calculation" depending on the conversion type for each type. These types are device dependent and need to be defined for each device. For each calculation type, a method must also be made to pass the data to the calculation.

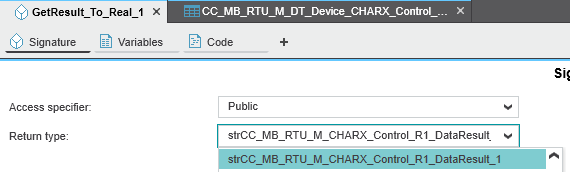


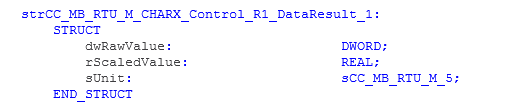
In the example below, only a transformation from a value to a real is defined. For this, the calculation type below is used and the data is transformed into a REAL data type via the GetResult\_To\_Real\_1 method.





This is then returned via the data type below that is defined in the data type sheet.





### SelectFunctionBlocks

To write to a device, a function block must be written for each FC6 and FC16 call. The structure to forward data is not fixed and can be built up at will.

The example below writes a maximum current value every minute or if it changes.

