LECO Explicit MPC Software Overview

Michael Ralea: mralea99@gmail.com

Table of Contents

Software Overview

There are several software tools you will need to develop code for any PLCs from the Beckhoff family. The focus of this document will be on the CX9020 PLC, but the steps for configuration *should* work for any other Beckhoff PLC. This section will detail how to set up a development environment to write PLC programs and run them locally without any hardware necessary. Connecting hardware to the development environment will be covered in the hardware overview document. This section will also cover MATLAB / Simulink and the toolboxes needed within them for PLC development and Explicit MPC.

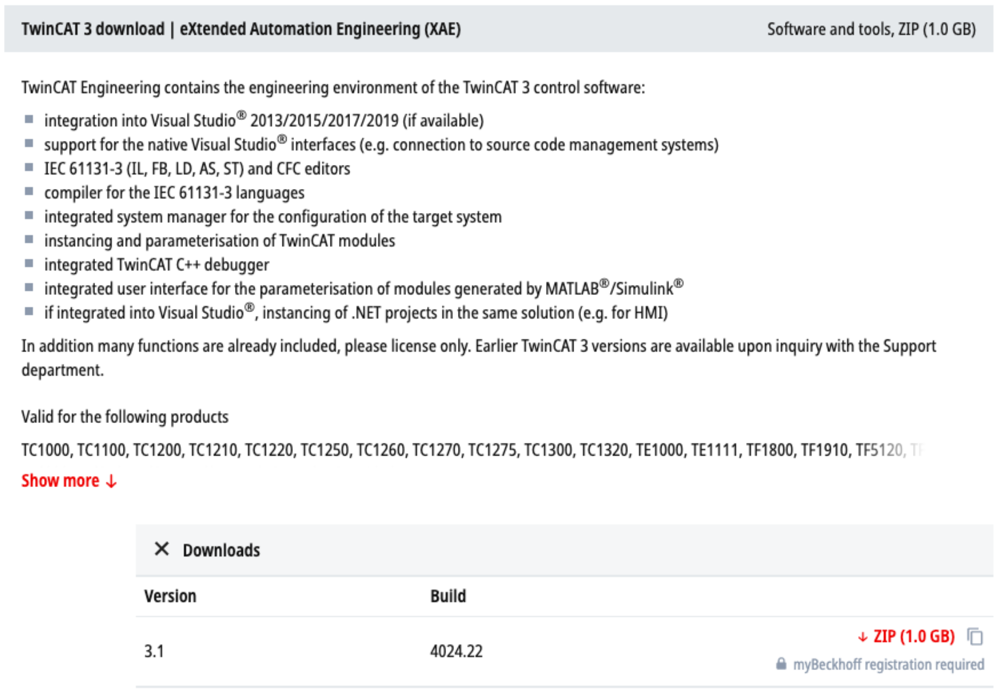
TwinCAT 3 Installation

The first step in setting up your development environment is getting Beckhoff's integrated development environment (IDE), called TwinCAT 3. You will have two options when you get this software product; you can download it as a "shell" or integrate it with Visual Studio. I recommend integrating with Visual Studio for the most user-friendly experience. You will want to check this link to see what versions of Visual Studio are supported at the time and for more detailed installation options. Once you have a compatible Visual Studio version installed, you can download the TwinCAT 3 XAE installer which will walk you through the steps to integrate with Visual Studio. If you install Visual Studio after, you will need to uninstall TwinCAT 3 and reinstall. Visual Studio Community is free and works fine.

Visual Studio Community Download Link

TwinCAT 3 XAE Download Link

The above link will show all options for software that Beckhoff offers. Although other options may be suitable for what you need, the XAE runtime is not excessively large and will have everything you need and more. At the time of writing, the below version is available for download.



TwinCAT 3 download | eXtended Automation Engineering (XAE) — Software and tools, ZIP (1.0 GB)

TwinCAT Engineering contains the engineering environment of the TwinCAT 3 control software:

- integration into Visual Studio® 2013/2015/2017/2019 (if available)
- support for the native Visual Studio® interfaces (e.g. connection to source code management systems)
- IEC 61131-3 (IL, FB, LD, AS, ST) and CFC editors
- compiler for the IEC 61131-3 languages
- integrated system manager for the configuration of the target system
- instancing and parameterisation of TwinCAT modules
- integrated TwinCAT C++ debugger
- integrated user interface for the parameterisation of modules generated by MATLAB®/Simulink®
- if integrated into Visual Studio®, instancing of .NET projects in the same solution (e.g. for HMI)

In addition many functions are already included, please license only. Earlier TwinCAT 3 versions are available upon inquiry with the Support department.

Valid for the following products

TC1000, TC1100, TC1200, TC1210, TC1220, TC1250, TC1260, TC1270, TC1275, TC1300, TC1320, TE1000, TE1111, TF1800, TF1910, TF5120,

Show more ↓

✕ Downloads

Version	Build	
3.1	4024.22	↓ ZIP (1.0 GB) myBeckhoff registration required

It is possible that no download option will be available on that screen for you. On occasion, when Beckhoff finds a bug in a released build, no downloads are publicly available from their site until it gets resolved. I ran into this issue and simply contacted their technical support team through their support form and they emailed me an older release link.

An important thing to note about TwinCAT is that it requires a license to build / run code, but this license can be generated as a 7 day trial, indefinitely. TwinCAT 3 Environment Setup details how to renew the license and gives a more detailed introduction to the software. It is part of a series I highly recommend.

TwinCAT Introduction

Structured text is an IEC 61131-3 language (specific to programmable logic controllers). Not every PLC environment supports it, although most support Ladder Logic. TwinCAT 3 *does* support structured text which is what makes it an appealing option for a low-medium budget. Structured text is similar to C code, at least visually, but is really much simpler. There is no super efficient way to introduce it via a text medium like this document so I've linked several resources below to get you started.

Learning PLCs with Structured Text (Youtube Playlist)

- Highly recommended as a starting point. This playlist walks through configuring your environment, license renewal, data types, PLC tasks, and how scan-time works. Here you will learn how to build and run code without any hardware.

PID Tutorial (Youtube Playlist)

- A slightly more advanced tutorial (also highly recommended) that covers some hardware configuration as well. You can skip to episode PID21 to get to the Beckhoff specific material. Once you have a good theoretical grasp on structured text and the PLC, this tutorial will walk you through a more "real-world" application and also show you how to use the scope features to capture data. This will also show you how to use the data capture options.

Data capture is a very important part of the process and doing it within TwinCAT is described in the playlist above, but you will likely want to export that data and make it pretty in MATLAB. I can't take screenshots of TwinCAT as I no longer have access to the school computers and don't have room to install on my personal machine but there are easy to find options to export to .csv.

You can see my final .csv files and how I plot them in MATLAB here

Data Capture Examples

MATLAB / Simulink Toolbox Installation

If you don't have the most recent versions of MATLAB / Simulink, you can obtain them here.

The primary configuration steps in regards to MATLAB are simply installing the necessary toolboxes. At the time of writing this document, both of these toolboxes were available for free to students due to COVID but that may no longer be the case in the future.

Model Predictive Control Toolbox

Simulink PLC Coder

Any other dependencies of these toolboxes should also be free, and you will be prompted to install them as well if they are not already there.

MPC in MATLAB / Simulink

This section will point you to important links for examples and further clarification. In general, I would recommend you get a full grasp of how MPC works in MATLAB before you move onto EMPC because you will always need to create an implicit controller first, then extract an explicit controller from that.

There are several example projects on the mathworks site:

[Mathworks MPC Examples]
[Mathworks EMPC Examples]

My own code specific to the Quad-Tank can be found here:

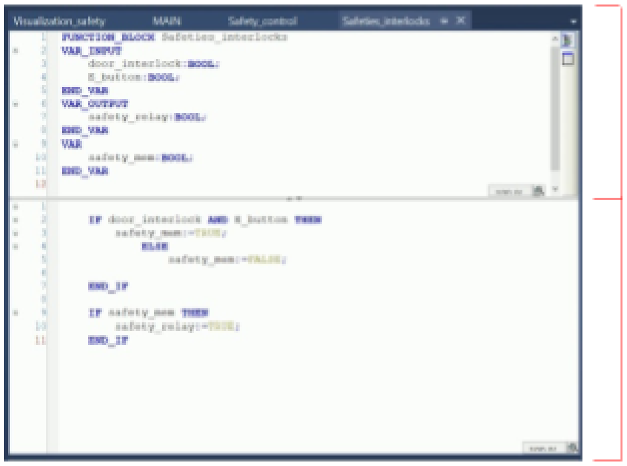
[Quad-Tank MATLAB Files]

If for some reason that is no longer up when you are using this document, feel free to email me.

Exporting Structured Text with Simulink PLC Coder

Unfortunately at the time of writing this document I cannot take screenshots to walk you through this process since I no longer have access to MATLAB, but luckily there is not much to it! The main things to keep in mind are:

1. Make sure your controller is in an "atomic subsystem"
    a. This just means you can't just drag the controller into Simulink, and right click it then export… that would be too easy. Just select the control block you want to generate code for, right click, and select "treat as atomic unit" to be true. This should already be done in my existing files on github linked above.
2. Set your target IDE to TwinCAT 3 and your language to structured text
    a. There should be a clear option to configure these settings when you right click while in Simulink
3. There is a little manual work to be done after exporting
    a. The way TwinCAT is set up, there is a section for variables separate from the code



```
FUNCTION_BLOCK Safeties_interlocks
VAR_INPUT
    door_interlock:BOOL;
    E_button:BOOL;
END_VAR
VAR_OUTPUT
    safety_relay:BOOL;
END_VAR
VAR
    safety_mem:BOOL;
END_VAR
```

```

    IF door_interlock AND E_button THEN
        safety_mem:=TRUE;
            ELSE
                safety_mem:=FALSE;

    END_IF

    IF safety_mem THEN
        safety_relay:=TRUE;
    END_IF
```

which makes it a little inconvenient when copying and pasting your generated code over. Look at my [example code] for clarification. If you open the project within TwinCAT it will be very clear, but if you just look at the files on github it is understandable but less easy. Anything between the "declaration" identifiers is that variable section, and between the “ST” identifiers is code. GVLs stand for global variable lists and POUs are program organizational units (or rather, just code). This is all handled in more detail in the youtube playlists I linked earlier

b. The single worst part about this entire process… **The declaration portion of the code that is generated is not syntactically perfect**. Arrays need to have [ ] around them but do not when you export. You need to manually go and put brackets around each array. The good news is there aren't a whole ton of them, they’re just large.