Application Note

FESTO

CMMT EtherCat controlled by C++

This document describes how to control the CMMT EtherCat version with the programming language C++, using the SOEM (Simple Open Source EtherCat Master) library.

CMMT-AS-..-..-EC,
CMMT-ST-..-..-EC

100337

Title	CMMT EtherCat with C++
Version	1.10
Document no.	100337
Original	en
Author	Festo
Last saved	19.04.2021

Table of content

1 Components/Software used 5

1.1 Topology of the tested system 5

2 Introduction 6

2.1 Preparation of the software (Windows) 6

2.2 Install WinPcap 7

3 Sample program 8

3.1 Open the target view 8

3.2 Select the slaveinfo program 8

3.3 Learn your network device name 9

4 Units of movement 10

5 The Master program 11

5.1 Open the main source file 11

5.2 Paste the correct network adapter 11

5.3 Run the test 12

6 Supported commands 13

7 Multiple Drives 14

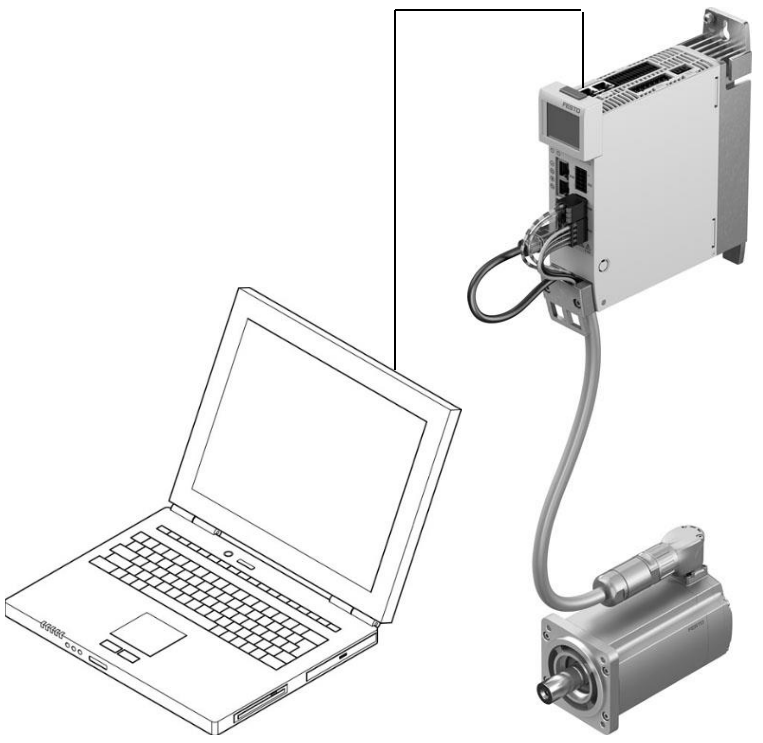
1 Components/Software used

Type/Name	Version Software/Firmware	Date of manufacture
CMMT-AS-C2-3A-EC-S01	17.0.8.48_release	
Visual Studio	2019	
Visual Studio tools for Cmake	1.0	
Visual C++ 2019	00435-60000-00000-AA969	
SOEM library (included)	1.3.3	
WinPcap	4.1.3	
C++ Sample Program (included)	0.0.2	
Automation Suite	1.3.2.4	

Table 1.1: 1 Components/Software used

1.1 Topology of the tested system

The Network adapter from the EtherCat Master (controlling system), is directly connected to the XF1 IN port on top of the CMMT.



2 Introduction

The examples presented in this document are based on the hardware and software mentioned in the previous chapter. This does not mean that they will not work on other versions, or other hardware. It only means they were successfully tested on the ones mentioned before. To make this Application Note more platform independent we used CMake a building platform supported on both Windows and Linux.

2.1 Preparation of the software (Windows)

Install CMake and C++ from the Tools and Features menu in Visual Studio.

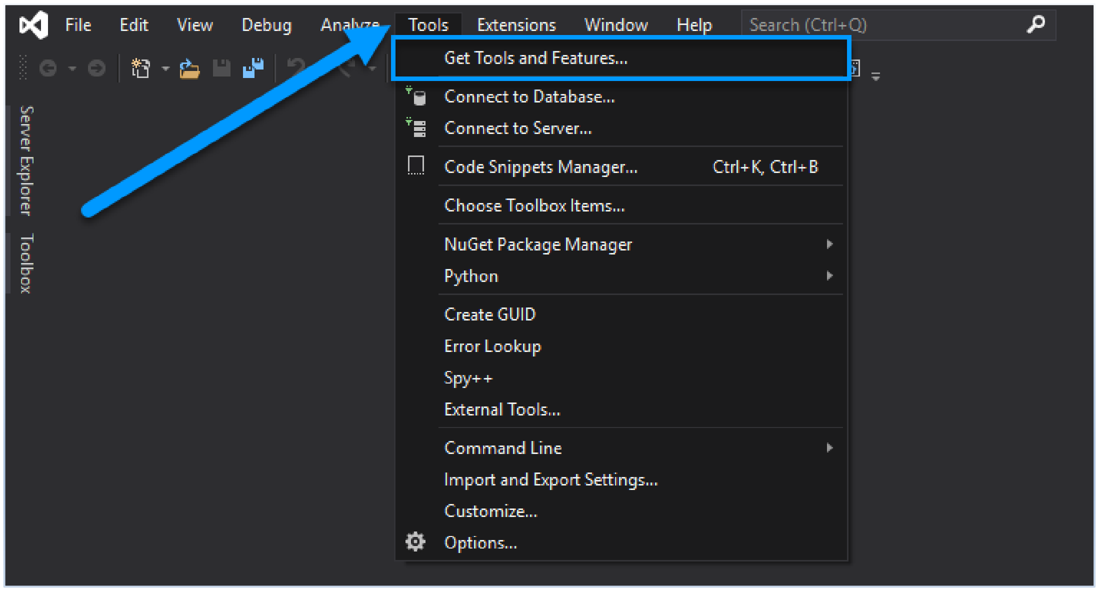
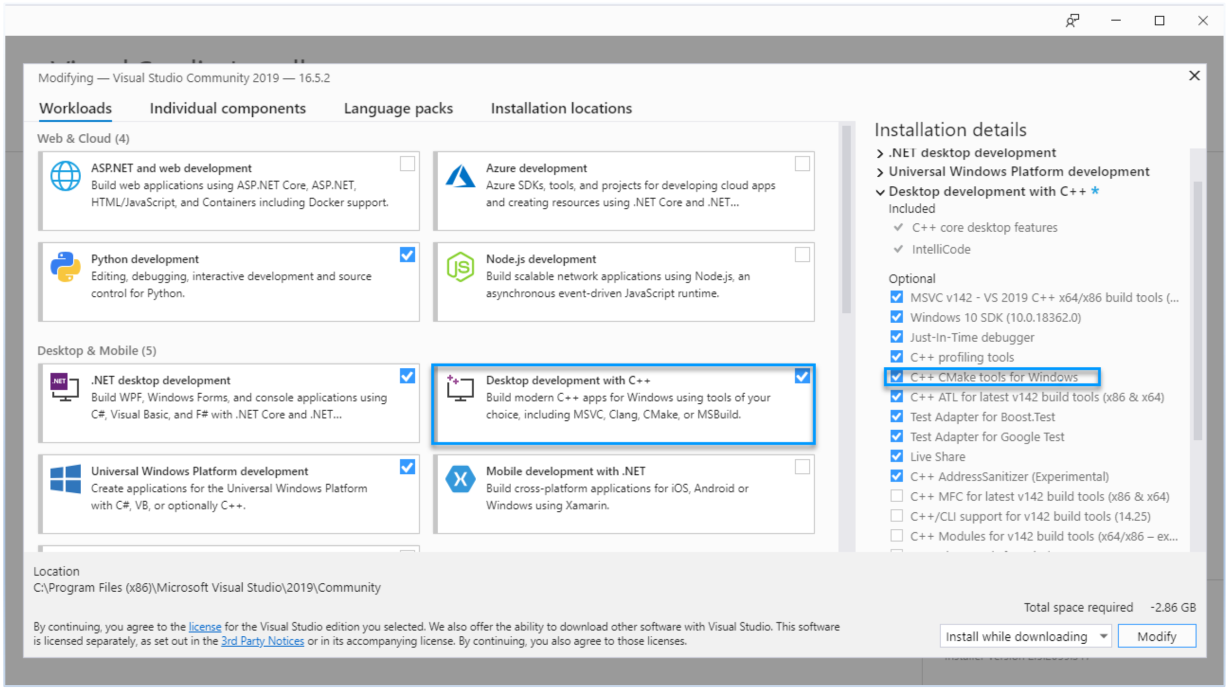


Figure 2-1 Where to find the Tools and Features menu



Modifying — Visual Studio Community 2019 — 16.5.2

Workloads | Individual components | Language packs | Installation locations

Web & Cloud (4)

- [ ] **ASP.NET and web development** Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker support.
- [ ] **Azure development** Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET Core and .NET...
- [✓] **Python development** Editing, debugging, interactive development and source control for Python.
- [ ] **Node.js development** Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

- [✓] **.NET desktop development** Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET Core and .NET...
- [✓] **Desktop development with C++** Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.
- [✓] **Universal Windows Platform development** Create applications for the Universal Windows Platform with C#, VB, or optionally C++.
- [ ] **Mobile development with .NET** Build cross-platform applications for iOS, Android or Windows using Xamarin.

Installation details

- .NET desktop development
- Universal Windows Platform development
- Desktop development with C++ *
  - Included
    - ✓ C++ core desktop features
    - ✓ IntelliCode
  - Optional
    - [✓] MSVC v142 - VS 2019 C++ x64/x86 build tools (...
    - [✓] Windows 10 SDK (10.0.18362.0)
    - [✓] Just-In-Time debugger
    - [✓] C++ profiling tools
    - [✓] C++ CMake tools for Windows
    - [✓] C++ ATL for latest v142 build tools (x86 & x64)
    - [✓] Test Adapter for Boost.Test
    - [✓] Test Adapter for Google Test
    - [✓] Live Share
    - [✓] C++ AddressSanitizer (Experimental)
    - [ ] C++ MFC for latest v142 build tools (x86 & x64)
    - [ ] C++/CLI support for v142 build tools (14.25)
    - [ ] C++ Modules for v142 build tools (x64/x86 – ex...

Location
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total space required -2.86 GB

Install while downloading | Modify

Figure 2-2 Tools and Features menu.

2.2 Install WinPcap

EtherCat is not a default part of Windows or Linux. To get RAW access to the Ethernet Adapter we use the PCAP driver interface. This interface is also included in programs like WireShark, to monitor network traffic.

Download and install the program from : https://www.winpcap.org/

On most systems a restart is required after installation. All configuration of this software is done automatically.

3 Sample program

The sample program is included in this Application note. After the instructions in chapter 2, it is now possible to open a CMake folder.

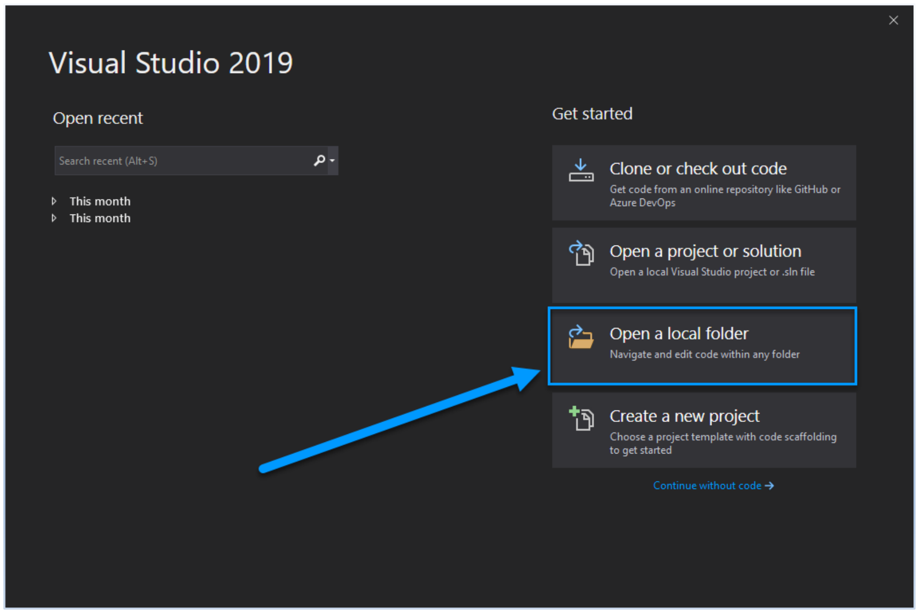

Visual Studio 2019
Open recent
Search recent (Alt+S)
This month
This month
Get started
Clone or check out code
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →

Figure 3-1 Open the sample program

3.1 Open the target view

The sample program includes multiple sample programs. The general programs to get more information about your topology and a program made by Festo to show you how to control your drive. On the right side of your screen we can change the default view, to the target view. This will display all individual programs.

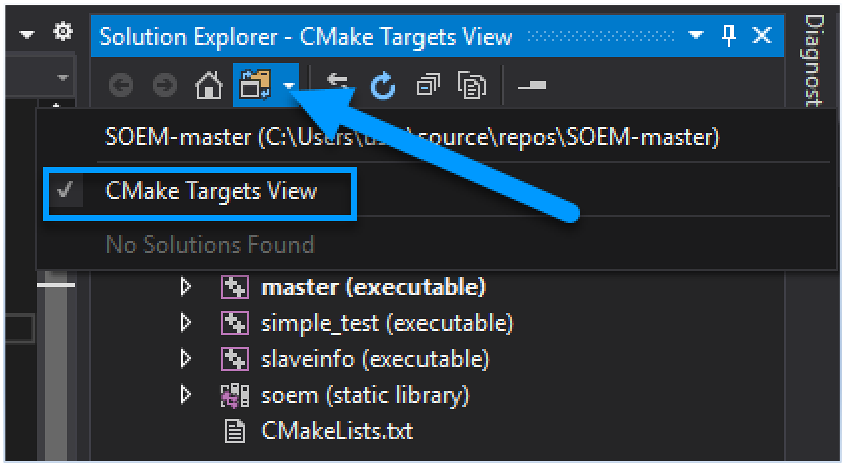


Figure 3-2 Target View

3.2 Select the slaveinfo program

The “slaveinfo” program is a very useful standard program in de SOEM library to get information about your EtherCat master and slaves. To run this program, right click on the slaveinfo (executable) and

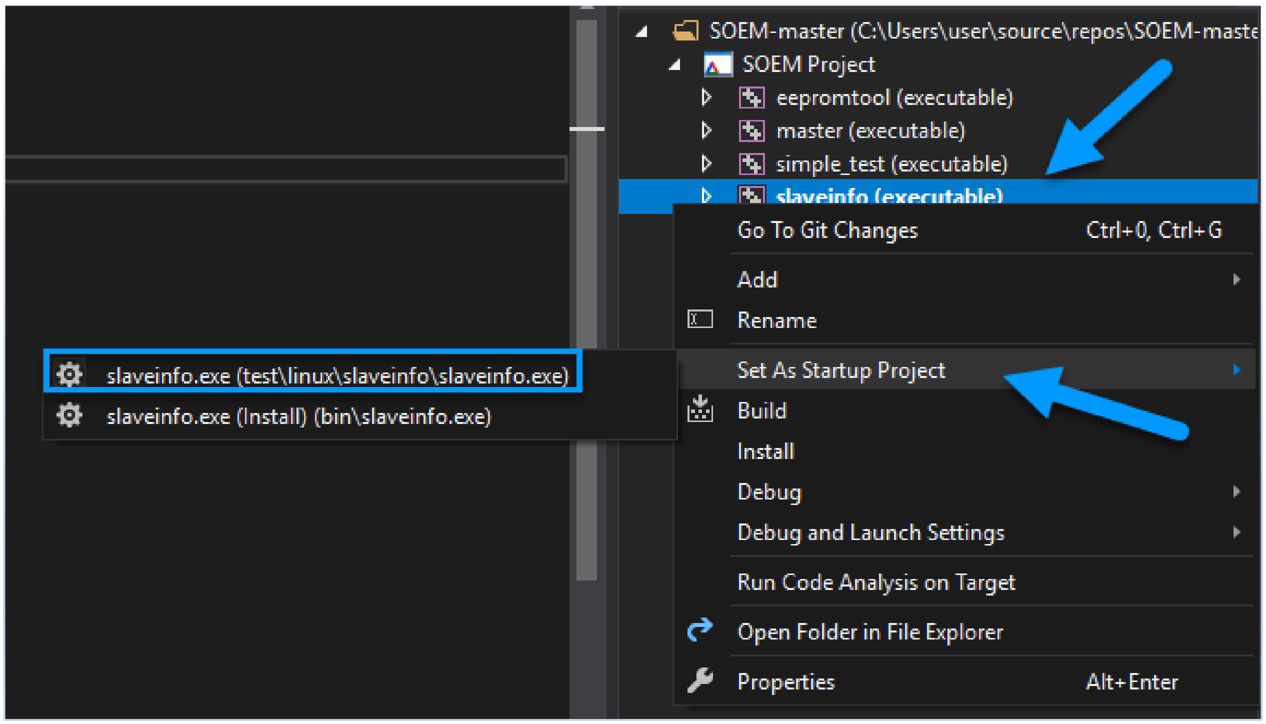

SOEM-master (C:\Users\user\source\repos\SOEM-maste
SOEM Project
eepromtool (executable)
master (executable)
simple_test (executable)
slaveinfo (executable)
Go To Git Changes Ctrl+0, Ctrl+G
Add
Rename
Set As Startup Project
Build
Install
Debug
Debug and Launch Settings
Run Code Analysis on Target
Open Folder in File Explorer
Properties Alt+Enter
slaveinfo.exe (test\linux\slaveinfo\slaveinfo.exe)
slaveinfo.exe (Install) (bin\slaveinfo.exe)

Figure 3-3 Select Startup project

3.3 Learn your network device name

The program needs to know on where to setup the EtherCat master. For this we need to have the unique identifier of your network adapter. On Linux this can be “eth0” for example. On Windows this is a longer unique description number. We can find this number with the “slaveinfo” sample program.

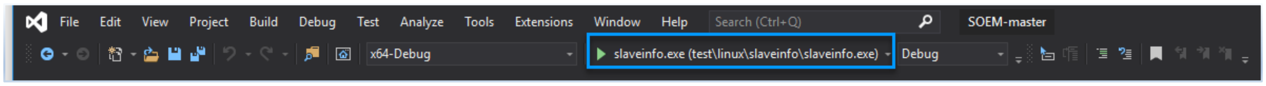
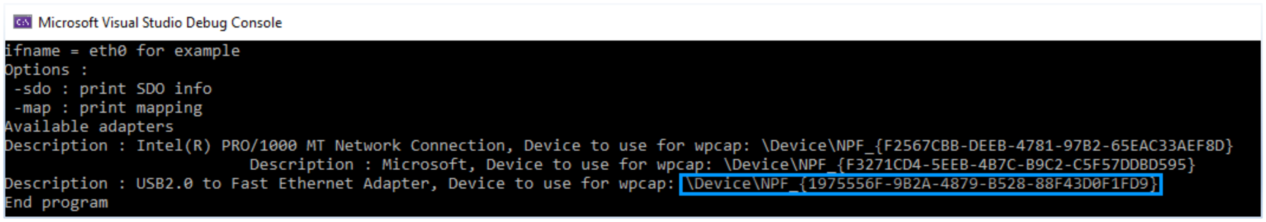


Figure 3-4 Start the program form the top bar

The output of this program will show us the unique identifying number of each connected network interface.



Microsoft Visual Studio Debug Console

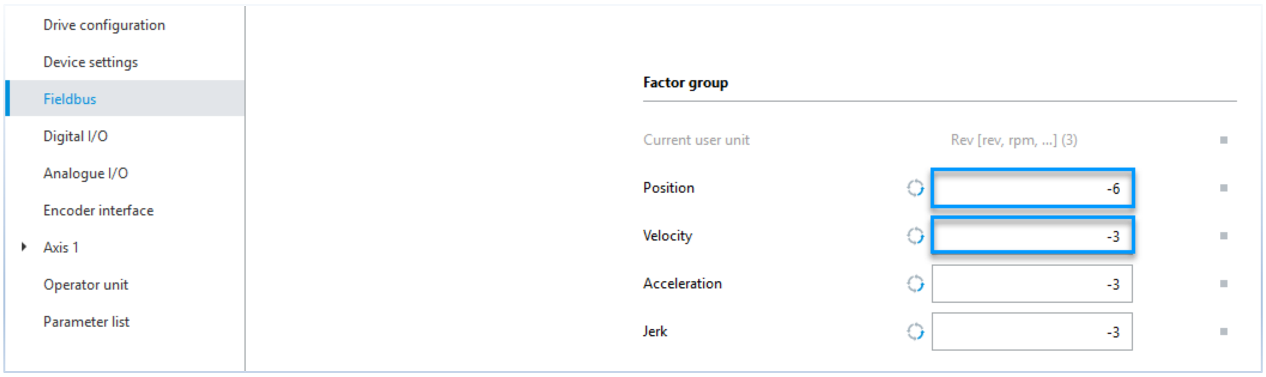
```
ifname = eth0 for example
Options :
 -sdo : print SDO info
 -map : print mapping
Available adapters
Description : Intel(R) PRO/1000 MT Network Connection, Device to use for wpcap: \Device\NPF_{F2567CBB-DEEB-4781-97B2-65EAC33AEF8D}
                          Description : Microsoft, Device to use for wpcap: \Device\NPF_{F3271CD4-5EEB-4B7C-B9C2-C5F57DDBD595}
Description : USB2.0 to Fast Ethernet Adapter, Device to use for wpcap: \Device\NPF_{1975556F-9B2A-4879-B528-88F43D0F1FD9}
End program
```

Figure 3-5 List of interfaces

In this example the interface number is "\Device\NPF_{1975556F-9B2A-4879-B528-88F43D0F1FD9}". The slash in the programming language C is however a special character. We need to modify this value to make it correctly readable by our programming language. To do this we simply add an extra “\” to every “\”. The new value now becomes \\Device\\NPF_{1975556F-9B2A-4879-B528-88F43D0F1FD9}.

4 Units of movement

The formatting of data is dependent on how it is configured in Festo Automation Suite. In this example an unlimited rotative drive is used. We configured a unit of revolution/rotation, with a precision of -6.



Factor group

Current user unit	Rev [rev, rpm, ...] (3)
Position	-6
Velocity	-3
Acceleration	-3
Jerk	-3

This means that when we want to move to a position of 20 rotations, this is transmitted over the fieldbus with this extra precision of -6. Resulting in a setpoint of 20 000 000 to achieve this.

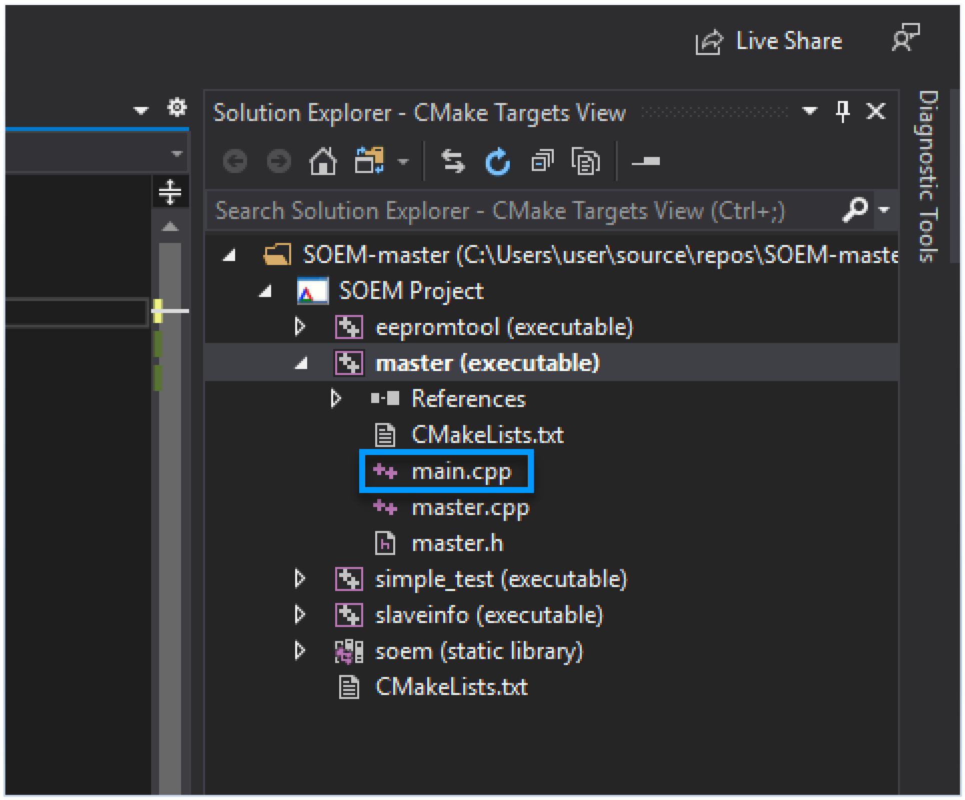
The same is done for the Velocity, in this case we are using a precision of -3, resulting in a setting of 10 000 if we want to achieve 10 rpm.

5 The Master program

Copy the device name and select the master program, like done in chapter 3.2.

5.1 Open the main source file

Double click to open the “main.cpp” source

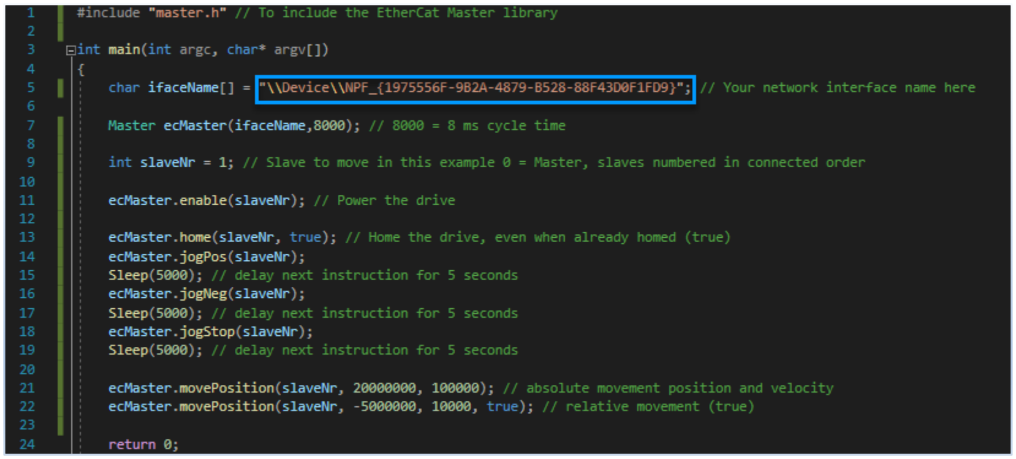

Live Share
Solution Explorer - CMake Targets View
Search Solution Explorer - CMake Targets View (Ctrl+;)
Diagnostic Tools

```
SOEM-master (C:\Users\user\source\repos\SOEM-maste
  SOEM Project
    eepromtool (executable)
    master (executable)
      References
      CMakeLists.txt
      main.cpp
      master.cpp
      master.h
    simple_test (executable)
    slaveinfo (executable)
    soem (static library)
    CMakeLists.txt
```

Figure 5-1 The sample program

5.2 Paste the correct network adapter

Use your unique network identifier from chapter 3.3

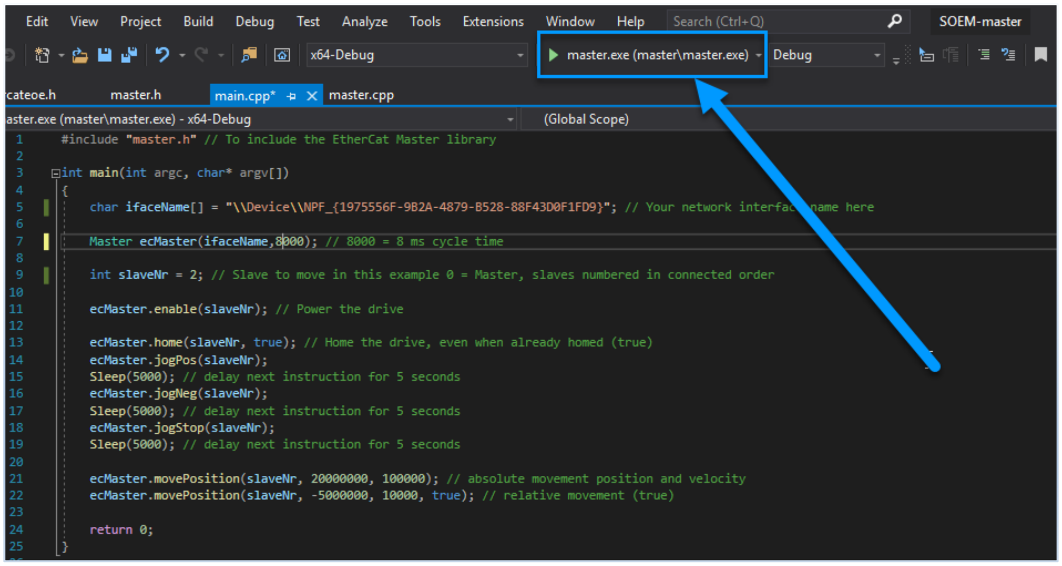


```
#include "master.h" // To include the EtherCat Master library

int main(int argc, char* argv[])
{
    char ifaceName[] = "\\Device\\NPF_{1975556F-9B2A-4879-B528-88F43D0F1FD9}"; // Your network interface name here

    Master ecMaster(ifaceName,8000); // 8000 = 8 ms cycle time

    int slaveNr = 1; // Slave to move in this example 0 = Master, slaves numbered in connected order

    ecMaster.enable(slaveNr); // Power the drive

    ecMaster.home(slaveNr, true); // Home the drive, even when already homed (true)
    ecMaster.jogPos(slaveNr);
    Sleep(5000); // delay next instruction for 5 seconds
    ecMaster.jogNeg(slaveNr);
    Sleep(5000); // delay next instruction for 5 seconds
    ecMaster.jogStop(slaveNr);
    Sleep(5000); // delay next instruction for 5 seconds

    ecMaster.movePosition(slaveNr, 20000000, 100000); // absolute movement position and velocity
    ecMaster.movePosition(slaveNr, -5000000, 10000, true); // relative movement (true)

    return 0;
```

Figure 5-2 Selecting the correct network card

5.3 Run the test

If you work from a virtual machine, your network adapter latency is higher than normal. A cycle time of 8 ms or 8000 nanoseconds should be safe on most recent computers.


Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
SOEM-master
x64-Debug
master.exe (master\master.exe)
Debug
cateoe.h
master.h
main.cpp*
master.cpp
aster.exe (master\master.exe) - x64-Debug
(Global Scope)

```cpp
#include "master.h" // To include the EtherCat Master library

int main(int argc, char* argv[])
{
    char ifaceName[] = "\\Device\\NPF_{1975556F-9B2A-4879-B528-88F43D0F1FD9}"; // Your network interfac name here

    Master ecMaster(ifaceName,8000); // 8000 = 8 ms cycle time

    int slaveNr = 2; // Slave to move in this example 0 = Master, slaves numbered in connected order

    ecMaster.enable(slaveNr); // Power the drive

    ecMaster.home(slaveNr, true); // Home the drive, even when already homed (true)
    ecMaster.jogPos(slaveNr);
    Sleep(5000); // delay next instruction for 5 seconds
    ecMaster.jogNeg(slaveNr);
    Sleep(5000); // delay next instruction for 5 seconds
    ecMaster.jogStop(slaveNr);
    Sleep(5000); // delay next instruction for 5 seconds

    ecMaster.movePosition(slaveNr, 20000000, 100000); // absolute movement position and velocity
    ecMaster.movePosition(slaveNr, -5000000, 10000, true); // relative movement (true)

    return 0;
}
```

Figure 5-3 Run the program, test your configuration

If you are getting the correct amount of slaves and no errors, than the connection is successful.

6 Supported commands

For every project you need to create a master and enable the drive. The enable command automatically also resets all errors on the drive.

Create your EtherCat master:

- Master name(ifaceName,8000); // 8000 = 8 ms cycle time

We select the created master “name” than the command you want to use, followed by the slave number

- name.enable(slaveNr); // Power On the drive
- name.disable(slaveNr); // Power Off the drive
- name.home(slaveNr, true); // Home the drive, even when already homed (true)
- name.movePosition(slaveNr, 20000000, 100000); // absolute movement position and velocity
- name.movePosition(slaveNr, -5000000, 10000, true); // relative movement (true)
- name.getPos(slaveNr); // Returns the actual position
- name.getError(slaveNr); // Retruns 0 if no error
- name.disable(slaveNr); // Disables the drive

These movement functions are all blocking commands, that means the function quits when the movement or action finishes. The return value for a successful operation is 0. It is possible to stack these functions to make a sequential program.

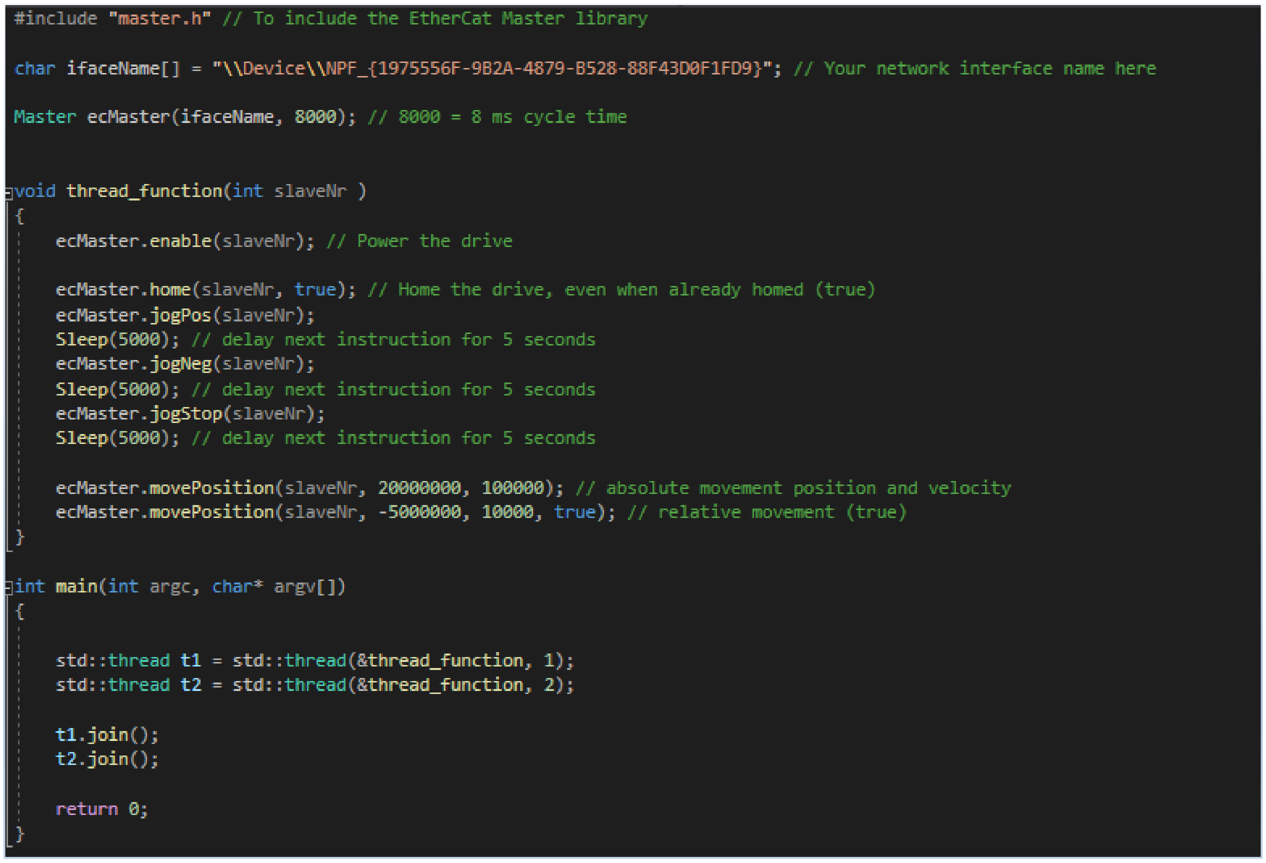
The next commands are non-blocking, this means that the movement will continue after the commands finishes.

- ecMaster.jogPos(slaveNr); // Moves slave in the positive direction
- ecMaster.jogNeg(slaveNr); // Moves slave in the negative direction
- ecMaster.jogStop(slaveNr); // Stop jogging

That means the drive will keep jogging until a new command or the jogStop(slaveNr); is issued. If there is no sleep command between the jogPos()/Neg() and jogStop() then the drive will stop immediately after it’s start.

7 Multiple Drives

We can use multithreading to control multiple drives together. We need to start the master only a single time and then running the different slave commands in different threads.



```cpp
#include "master.h" // To include the EtherCat Master library

char ifaceName[] = "\\Device\\NPF_{1975556F-9B2A-4879-B528-88F43D0F1FD9}"; // Your network interface name here

Master ecMaster(ifaceName, 8000); // 8000 = 8 ms cycle time


void thread_function(int slaveNr )
{
    ecMaster.enable(slaveNr); // Power the drive

    ecMaster.home(slaveNr, true); // Home the drive, even when already homed (true)
    ecMaster.jogPos(slaveNr);
    Sleep(5000); // delay next instruction for 5 seconds
    ecMaster.jogNeg(slaveNr);
    Sleep(5000); // delay next instruction for 5 seconds
    ecMaster.jogStop(slaveNr);
    Sleep(5000); // delay next instruction for 5 seconds

    ecMaster.movePosition(slaveNr, 20000000, 100000); // absolute movement position and velocity
    ecMaster.movePosition(slaveNr, -5000000, 10000, true); // relative movement (true)
}

int main(int argc, char* argv[])
{

    std::thread t1 = std::thread(&thread_function, 1);
    std::thread t2 = std::thread(&thread_function, 2);

    t1.join();
    t2.join();

    return 0;
}
```