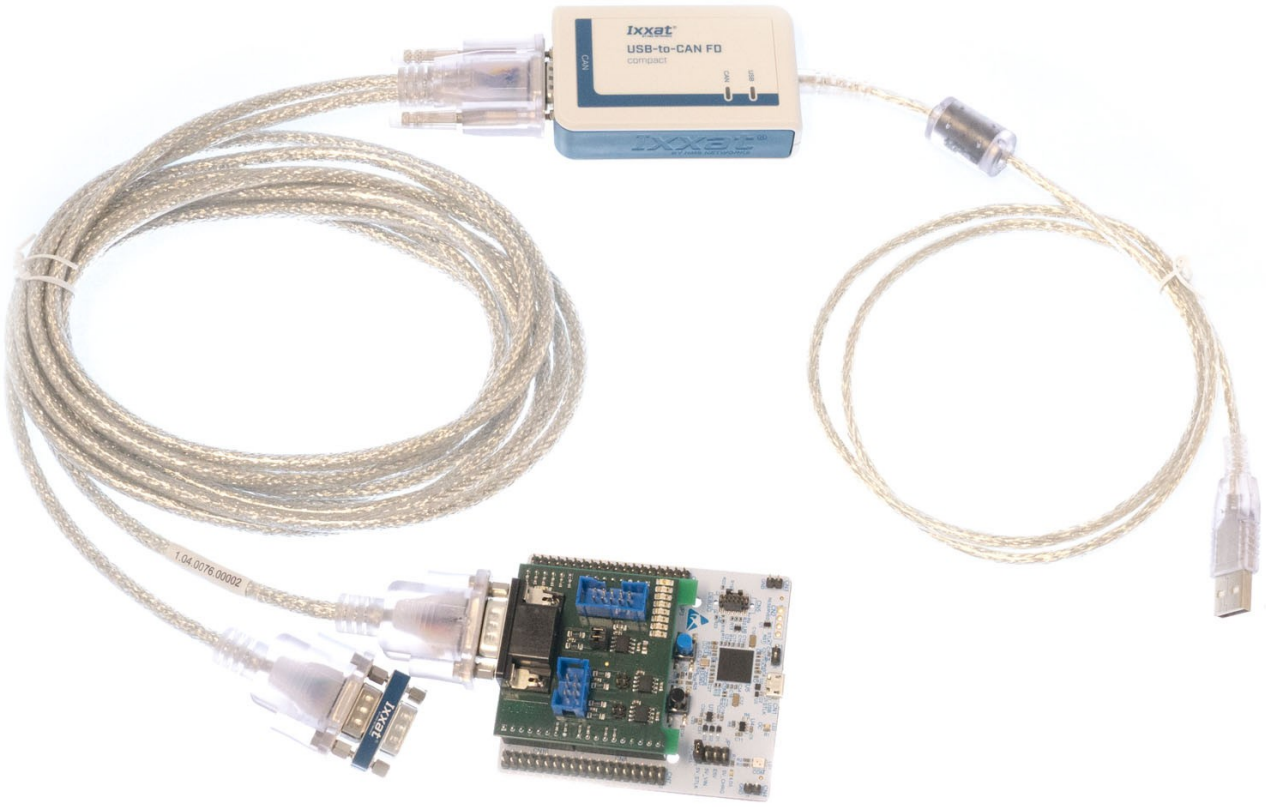
# CANopen FD Starter Kit
# User Manual

1.0.0

# Table of Contents

# 1 Overview




The CANopen FD Starter Kit provides a cost-effective but yet functional solution to evaluate the benefits of CANopen FD.

Therefore it contains of a powerful STM32 Microcontroller board, a CAN FD extension board with several input and output possibilities, and an Ixxat USB-to-CAN FD interface. Other equipment like a Y CAN cable and a Sub-D9 Connector with CAN termination resistor are also included.

The CANopen FD Starter Kit comes with a CANopen FD evaluation library and a STM32CubeIDE example project. The CANopen DeviceDesigner Software can be used to change the CANopen FD Device configuration. For analyzing purposes, the CANopen DeviceExplorer Software is also included.

# 2 Hardware

## 2.1 NUCLEO-STM32G431RB

This Board features a STM32G431RB MCU with 1 CAN FD Controller. It also includes a ST-LINK debugger/programmer.

For more information please visit the STMicroelectronics Website:
Link to the NUCLEO_STM32G431RB Website

## 2.2 CAN FD Shield

The CAN FD Shield provides 1 CAN FD Transceiver as well as 8 LEDs and 4 Input Pins. It connects through the Arduino™ Uno V3 connector to the NUCLEO-STM32G431RB board.

The mapped Arduino pins are:

- FDCAN1: D15 (RX) and D14 (TX)
- Input: A1, A0, D12 and D11.
- LEDs: A5, A4, A3, A2, D9, D8, D6 and D7

## 2.3 Y CAN cable

The 2 meter shielded Y CAN cable connects from a Sub-D9 socket to a Sub-D9socket and a Sub-D9 plug. All Sub-D9 pins on this cable are connected.

For more information please visit the Ixxat Website:
Link to the Ixxat Y cable

## 2.4 Sub-D9 Connector with CAN Termination

Between pin 2 and pin 7 the Sub-D9 Connector contains of a 120 Ohm resistor for High-Speed CAN systems according to ISO 11898-2. It has a 1-to-1 pin assignment and connects from a from a Sub-D9 socket to a Sub-D9 plug.

For more information please visit the Ixxat Website:
Link to the Ixxat Sub-D9 Connector with CAN Termination

## 2.5 Ixxat USB-to-CAN FD Compact

This galvanic isolated CAN FD interface connects via USB 2.0 Hi-Speed to a computer. It allows ISO CAN FD, non ISO CAN FD and CAN High-Speed operation.

It can be driven under Windows via Ixxat's own VCI4 driver:
Link to the Ixxat VCI4 driver

Under Linux it can be used via a socketCAN extension:
Link to the Ixxat socketCAN extension

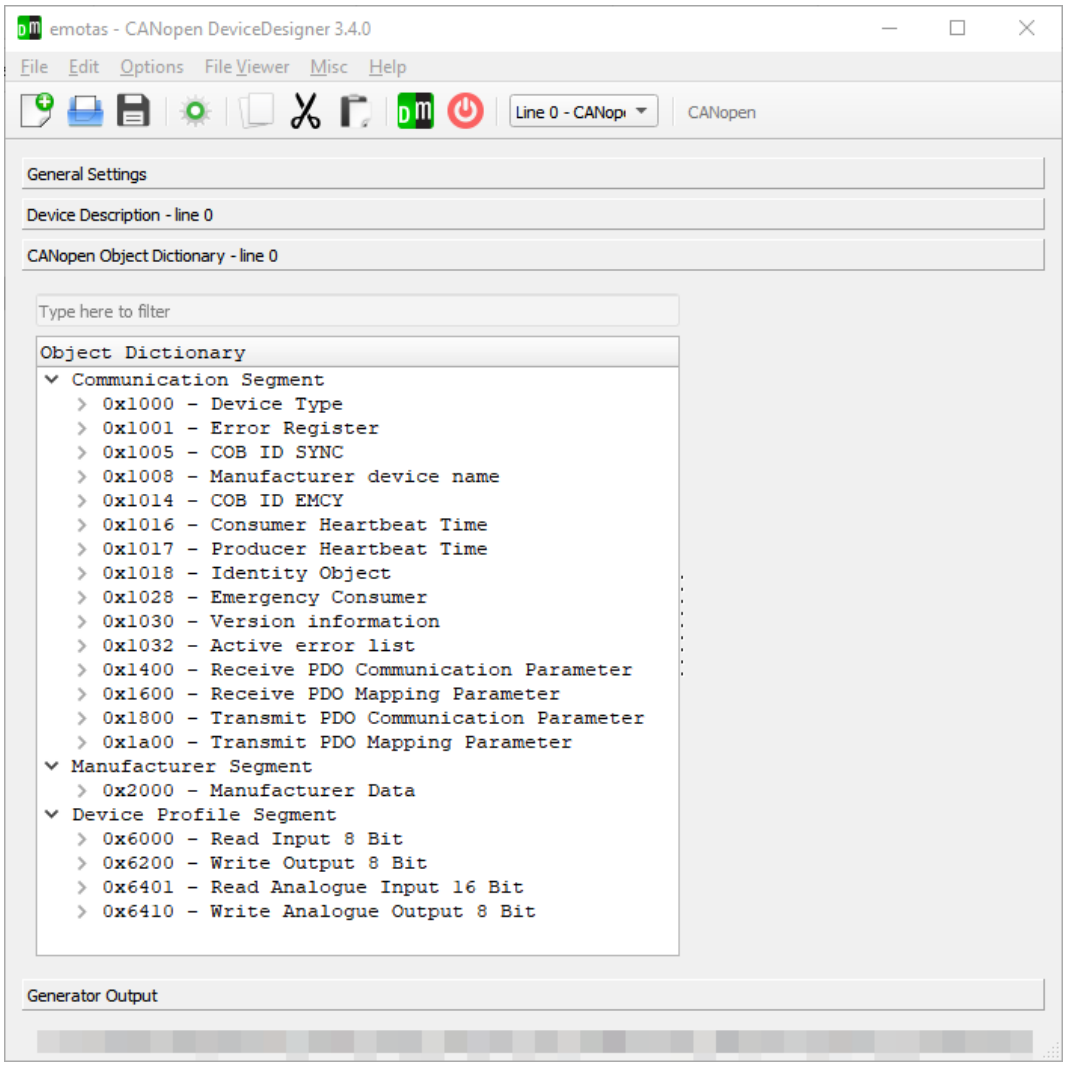
Also under Linux up to kernel version 4.4.0 it can be used with Ixxat's own ECI driver:
Link to the Ixxat ECI driver

Under macOS it can be used with open source driver can4osx:
Link to open source driver can4osx

For more information please visit the Ixxat Website:
Link to the Ixxat USB-to-CAN FD Compact Website

# 3 Software

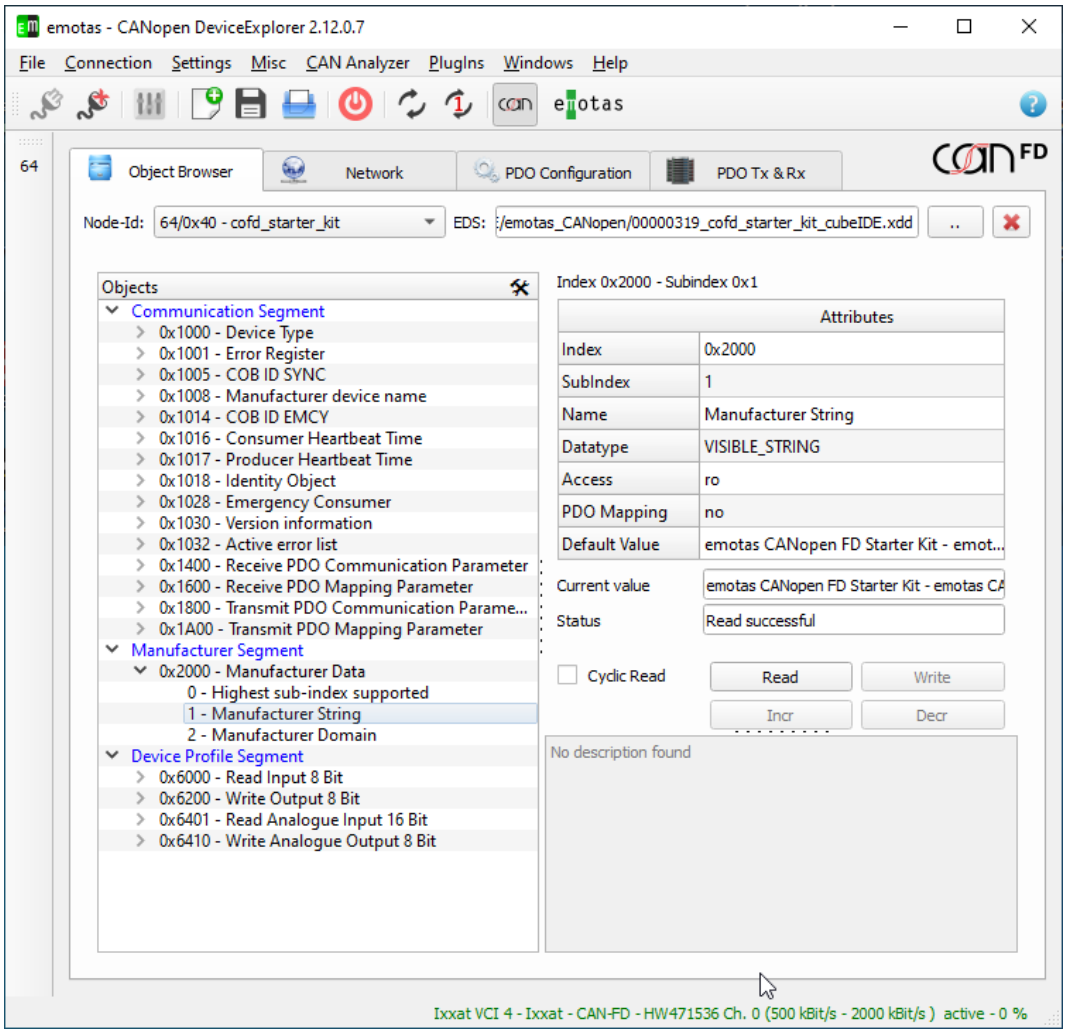
## 3.1 CANopen DeviceDesigner Demo




The CANopen DeviceDesigner is an easy-to-use tool for fast and cost-saving design of CANopen devices. With a few mouse clicks the object dictionary of the device can be created on the basis of predefined profiles. The CANopen DeviceDesigner creates the object dictionary and initializing functions in C, the electronic data sheet in XDD format and a device documentation. Additionally the CANopen DeviceDesigner configures the CANopen stack and CANopen driver under consideration of the device characteristics. So the optimal configuration is given.

More information can be found on the CANopen DeviceDesigner Website:
Link to the CANopen DeviceDesigner Website

## 3.2 CANopen DeviceExplorer Demo




The CANopen DeviceExplorer is a versatile tool for development, testing, diagnostics and service tasks. It provides CANopen master functionalities and allows the analysis and configuration of CANopen devices.

Information about each CANopen device are read from the electronic data sheet of the device, or they can be scanned directly from the device. Using standardized device configuration files (DCF) device configurations can be saved or imported. Additionally, data of entire CANopen networks can be stored in project files. The built-in scripting capability using QtScript allows users to create their own test and control applications with only little effort.

More information can be found on the CANopen DeviceExplorer Website:
Link to the CANopen DeviceExplorer Website

## 3.3 CANopen FD Slave Stack Evaluation Library

The CANopen FD Slave Library has following features:

- NMT Slave
- USDO server with 5 simultaneous connections
- 1 Transmit PDO
- 1 Receive PDO
- SYNC Consumer
- Heartbeat Producer
- 1 Heartbeat Consumer
- Emergency Producer

The Library has a 60 minutes time limitation. After this time the CANopen Library stops working. It was compiled by STM32CubeIDE 1.3.0, which uses GCC 6.3.1.

## 3.4 CANopen FD Slave Example Project

The CANopen FD Starter Kit comes with a STM32CubeIDE project. It contains of the default CANopen FD Library initialization, as well as the registration of some fundamental CANopen FD indication functions such as:

- NMT Indication
- NMT Error Control Indication
- USDO Server Read Indication
- USDO Server Write Indication
- PDO Indication
- CAN State Indication
- Communication Event Indication
- LED Green Indication
- LED Red Indication

The application simulates a digital/analogue IO device.

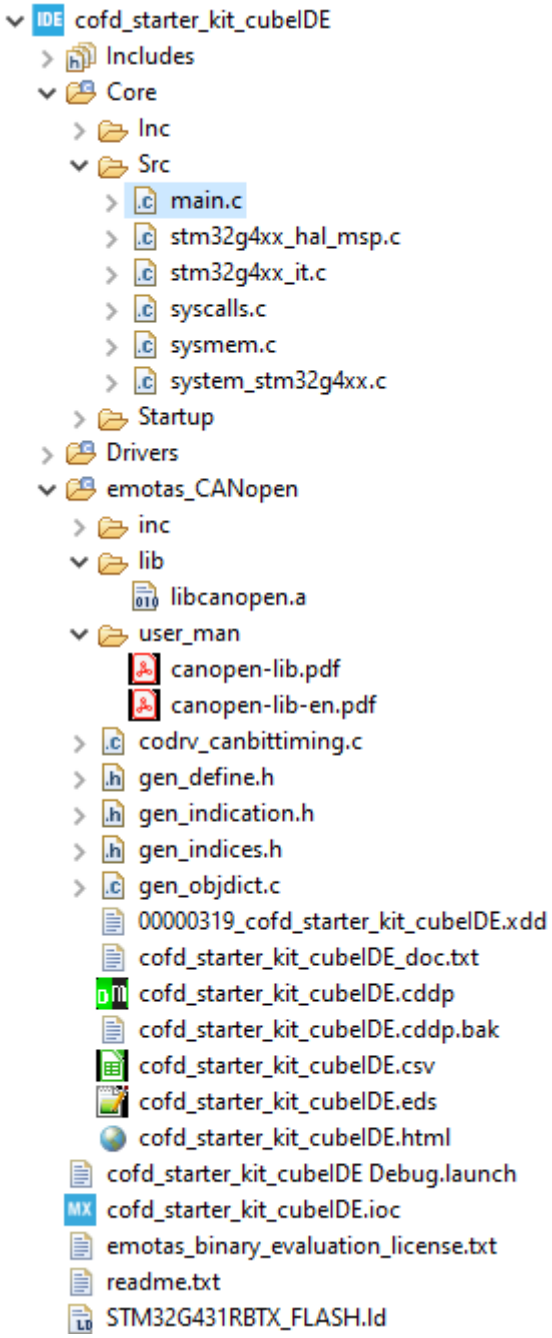
It initializes the 8 LEDs of the CAN FD Shield, 1 of them is used for the CANopen Run LED and 1 is used for the CANopen Error LED. The other 6 LEDs are used for a simple running light.

It also initializes the 4 input pins of the CAN FD Shield, all of them are currently unused.

When pressing the blue user Button (B1) on the NUCLEO-STM32G431RB, the device switches between OPERATIONAL and PRE-OPERATIONAL NMT state. In OPERATIONAL NMT state, the device sends a PDO with a DLC of 14 (48 Bytes) every 500 milliseconds.

The Project folder includes the file cofd_starter_kit_cubeIDE.ioc, which can be used to generate Projects for other IDEs like IAR or Keil by using STM32CubeMX. This Tool by can be downloaded from the STMicroelectronics website:
Link to the STM32CubeMX Website

To use the original project you have to use the STM32CubeIDE. You can download it from the STMicroelectronics Website:
Link to the STM32CubeIDE Website




### 3.4.1 General project structure

The project folder is separated into 4 sub folders. The folders Core and Drivers are generated by the STM32CubeIDE and should not be changed. Otherwise changes in this structure could be overwritten by the STM32CubeIDE, when generating new code, after changing something in the device configuration tool.

The generated file main.c holds the hardware initialization as well as user code like the CANopen FD library initialization code and other application specific code.

User code can be placed between special comment placeholders:

```
/* USER CODE BEGIN xxx */
(void)userFunctionCall()
/* USER CODE END xxx */
```

The emotas_CANopen folder includes the CANopen DeviceDesigner project, as well as the generated CANopen FD object dictionary (gen_objdict.c), the CANopen FD Library configuration (gen_define.h) and the XDD file.

The CANopen FD Library itself is located in the lib sub folder. The associated header files can be found in the sub folder inc. The user manual can be found in the sub folder user_man.

### 3.4.2 The CANopen FD Library example initialization

The following code shows an example initialization of the CANopen FD Library:

```
BOOL_T useCANopenFD = CO_TRUE;

      /* Initialize the CAN FD controller with:
       * arbitration bit rate of 500 kBit/s
       * data bit rate of 2000 kBit/s */
      if (codrvCanFdInit(500u, 2000u) != RET_OK)  {
      Error_Handler();
      }

      /* provide a timer clock (Systick) to the CANopen stack
       * this sets the Systick interval to CO_TIMER_INTERVAL */
      if (codrvTimerSetup(CO_TIMER_INTERVAL) != RET_OK)  {
      Error_Handler();
      }

      /* CANopen FD Initialization */
      /* Initialize the CANopen stack in CANopen FD mode */
      if (coCanOpenStackInit(NULL, &useCANopenFD) != RET_OK)  {
      Error_Handler();
      }

      /* enable CAN controller */
      if (codrvCanEnable() != RET_OK)  {
      Error_Handler();
      }
```

### 3.4.3 The CANopen FD Library service Routine

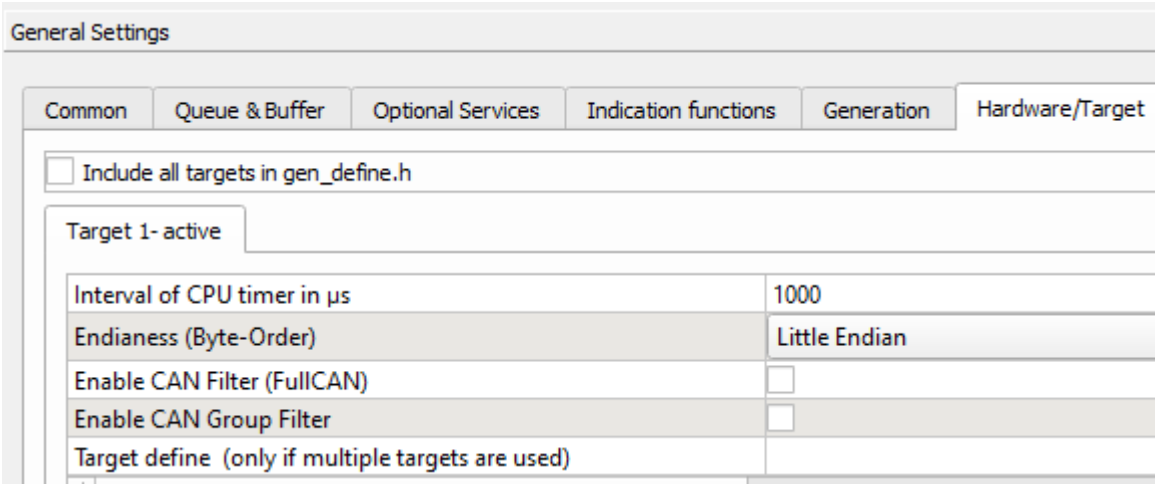
In the following example loop, the CANopen FD Library's service function is called. When returning CO_TRUE another CAN FD message has to be processed. In this case coCommTask() has to be called again.

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
        while (coCommTask() == CO_TRUE)  {
        /* more CANopen FD messages have to be processed
         * coCommTask() is executed again
         */
        }
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

### 3.4.4 The CANopen FD timer

To run with correct timings, the function codrvTimerISR() has to be called cyclic. In the example project, this function call is implemented in the HAL_SYSTICK_Callback() function. The Systick interval frequency has to be shared with the CANopen FD Library by setting the frequency value in microseconds in the CANopen DeviceDesigner:



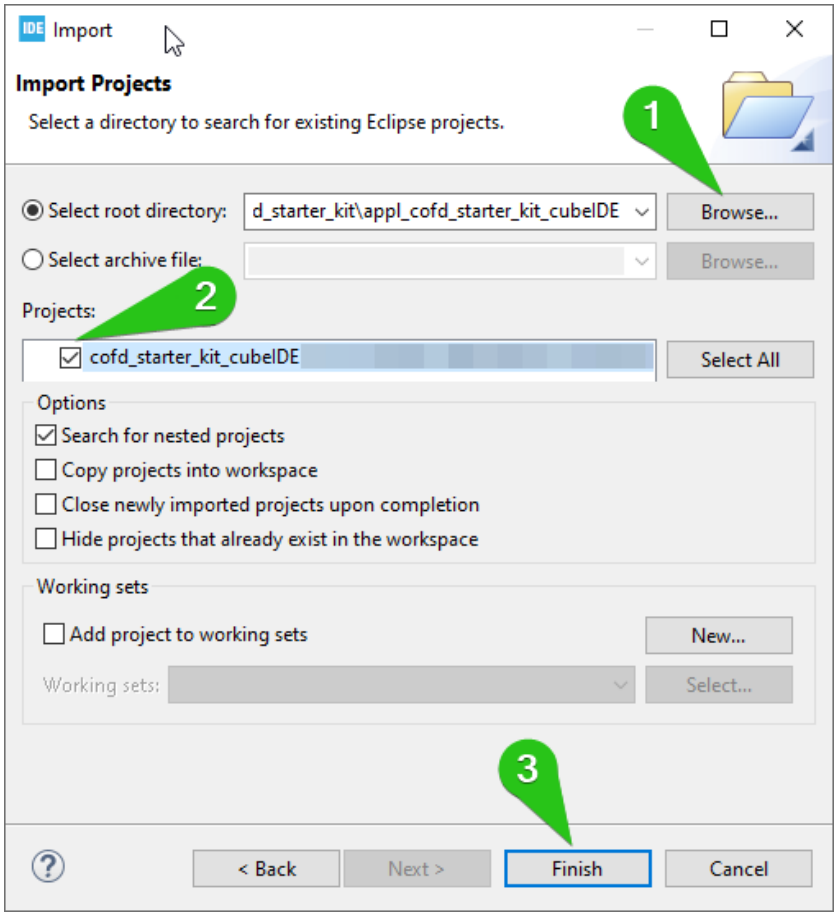

This sets the define CO_TIMER_INTERVAL in the file gen_define.h, which is used to setup the Systick in the example Library initialization.

# 4 Getting Started

After unboxing and assembling the Hardware of the CANopen FD Starter Kit, please go to the CANopen FD Starter Kit Website, and follow the instructions to download and install the required Software components:
[Link to the CANopen FD Starter Kit Website]

## 4.1 Import the Project to the STM32CubeIDE

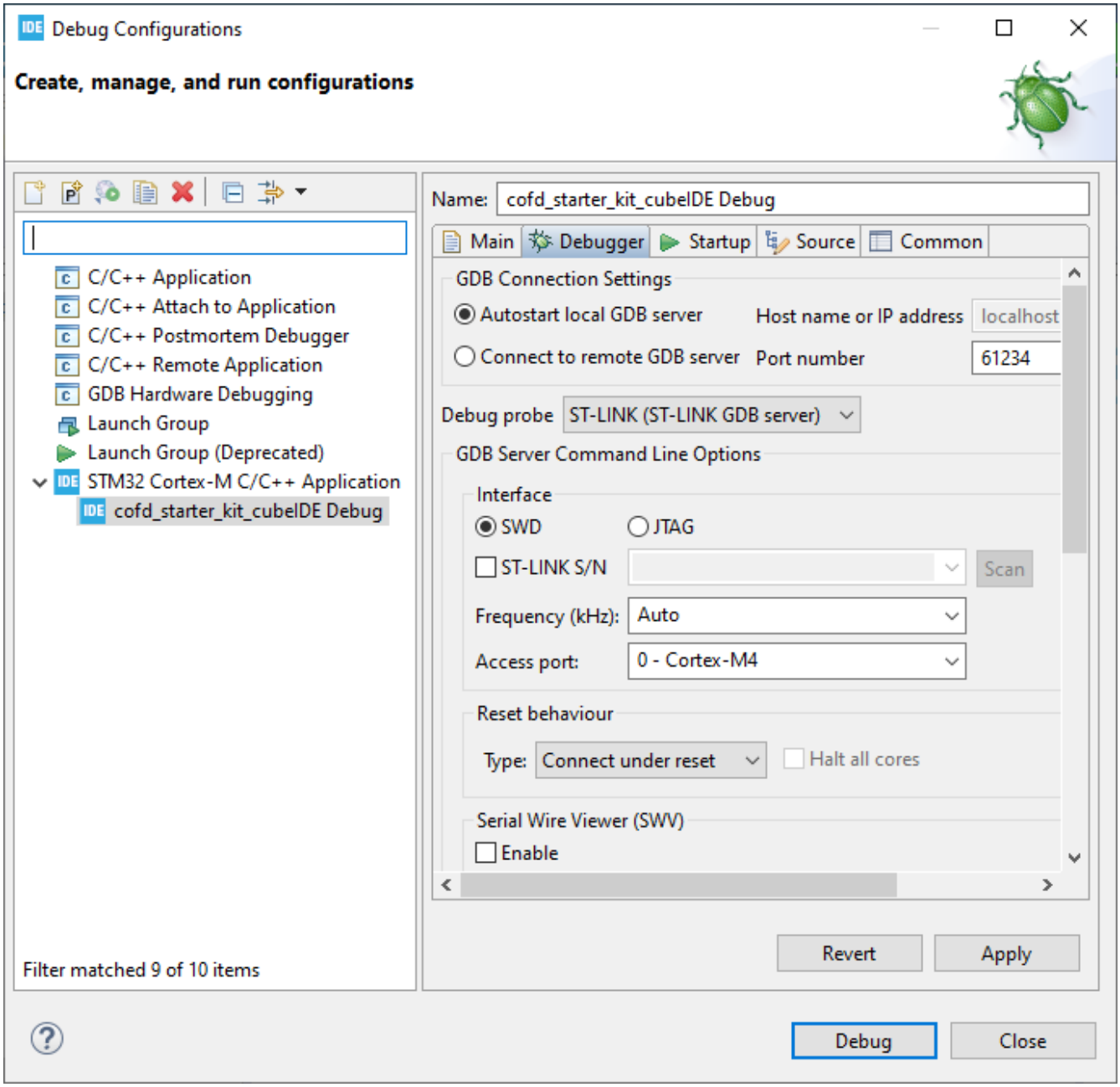
First extract the CANopen FD Starter Kit example zip file to the desired folder on your system. Now open the STM32CubeIDE and navigate to File → Import... In the following Dialog select from the General Category → Existing Projects into Workspace In the next Dialog please provide the path to the CANopen FD Starter Kit Example Path:




By clicking the Finish Button the project is finally imported into the STM32CubeIDE.

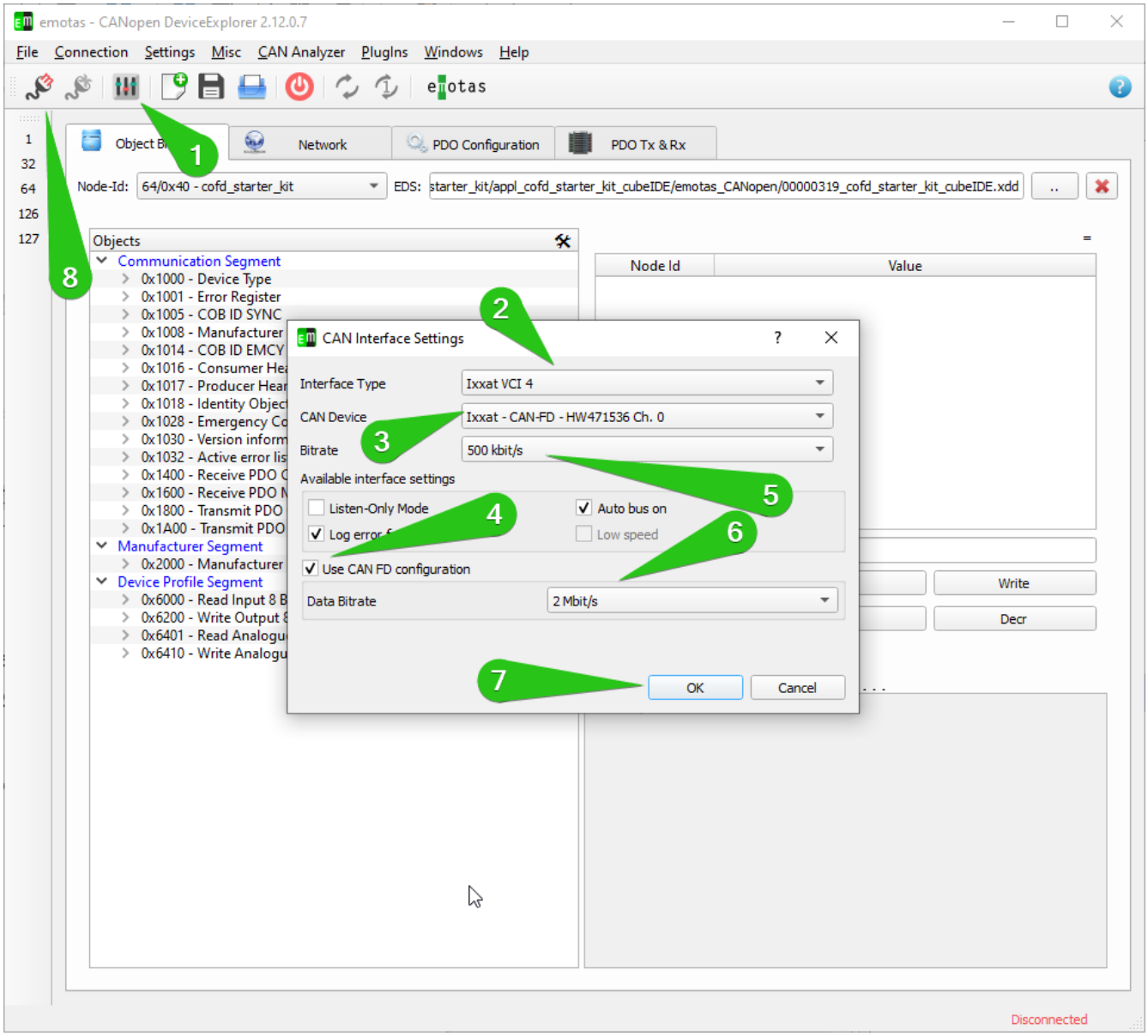
## 4.2 Compiling and Debugging

For Flashing the Program and Debugging, the onboard ST-Link of the NUCLEO-STM32G431RB is used. To start flashing/debugging just click the little bug icon in the STM32CubeIDE: 



## 4.3 Analyzing

For analyzing purposes, the CANopen DeviceExplorer Demo is included in the CANopen FD starter Kit. After opening the CANopen DeviceExplorer, you have to select your Can Interface, that you want to connect to:

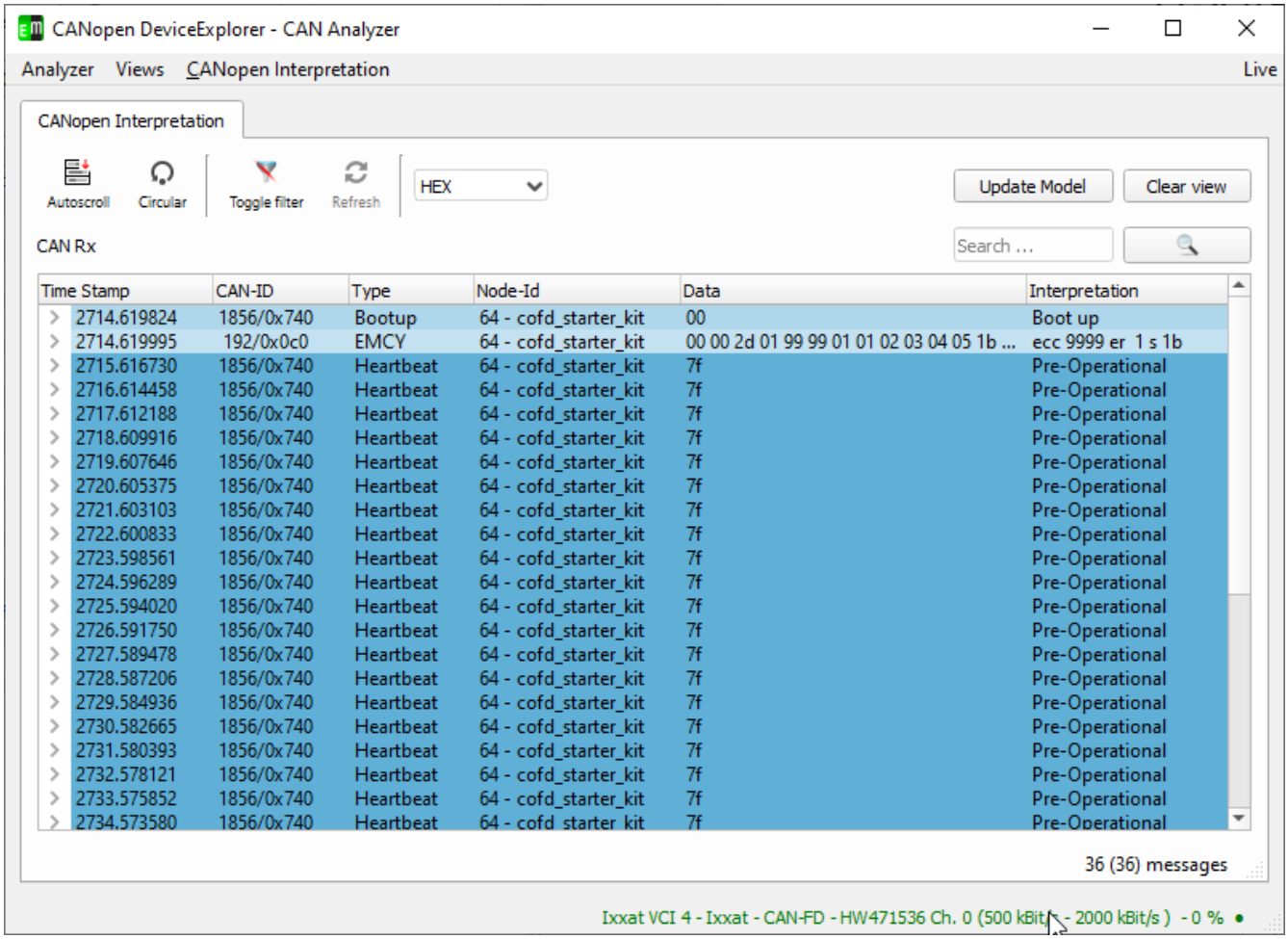



The default arbitration bit rate of the CANopen FD Starter Kit example application is 500 kBit/s. The default data bit rate is 2000 kBit/s. So you have to setup the same values in the Interface settings dialogue. After pressing OK, these settings are saved.

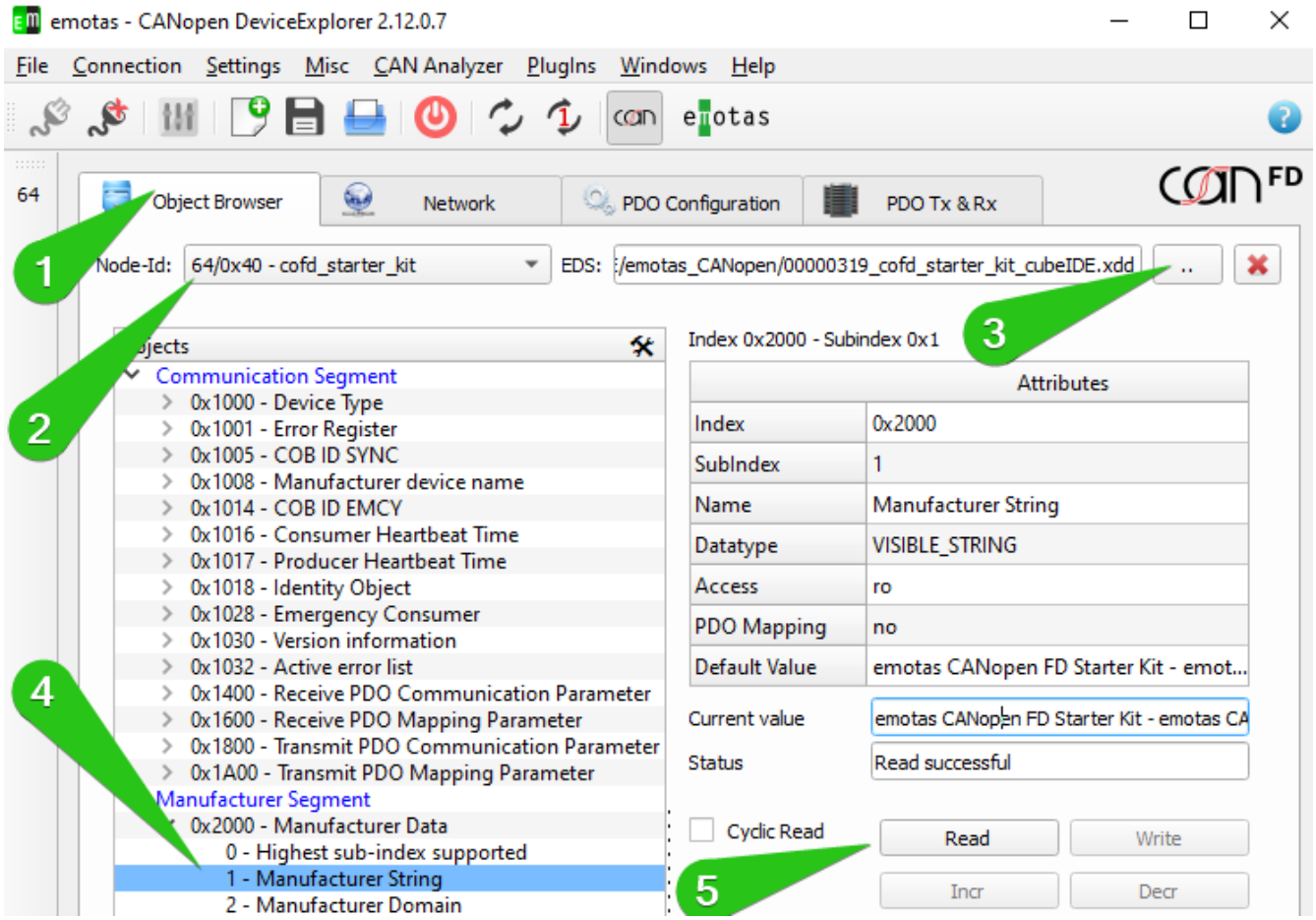
If the CANopen FD Starter Kit software is flashed, the CAN FD Shield is mounted to the NUCLEO-STM32G431RB, both devices are connected with the CAN cable and the Sub-D9 Connector with CAN Termination is fitted at the remaining Sub-D9 Connector of the CAN cable, you can use the Connect

button  to connect to the CAN bus. Now open the Analyzer Window by navigating to CAN Analyzer → CANopen Interpretation. If not connected, connect the NUCLEO-STM32G431RB to power over USB.

Now you should see the CANopen FD Messages in the Analyzer Window:



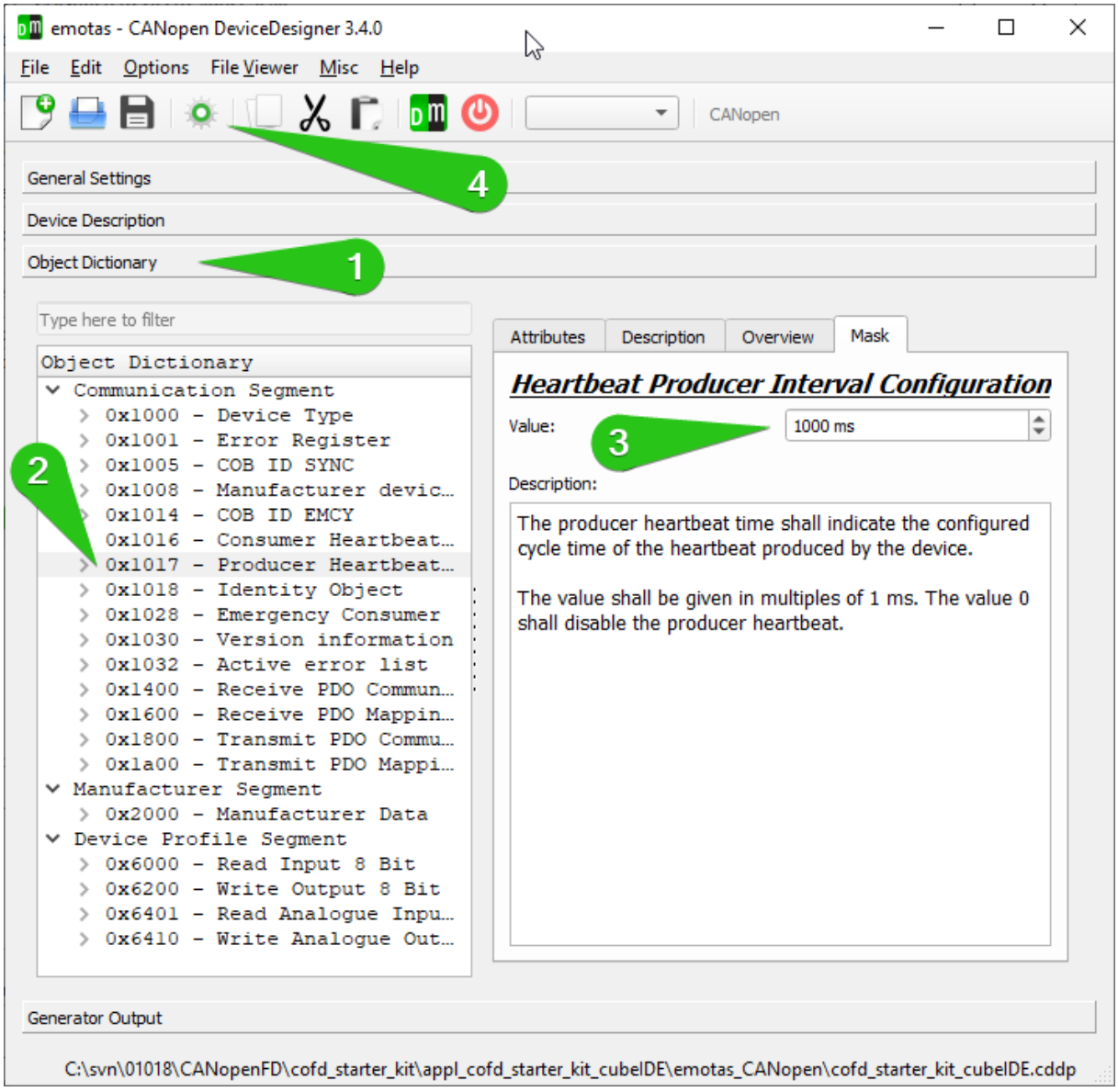

To read data via USDO from the CANopen FD Starter Kit, first load the XDD file into the CANopen DeviceExplorer. Therefore select Node 64 from the Node ID drop down menu, then select the path to the XDD file, which is located in the emotas_CANopen sub folder of the CANopen FD Starter Kit example:




Now you can read from and write to the CANopen FD devices object dictionary.

## 4.4 Changing CANopen FD Object Dictionary Entries

You can make changes to the CANopen FD Object dictionary, by using the CANopen DeviceExplorer Demo. The CANopen DeviceDesigner Project cofd_starter_kit_cubeIDE.cddp, is located in the examples sub folder emotas_CANopen. After opening the CANopen DeviceExplorer Demo, it can be imported by navigating to File → Open Project. For example if you would like to change the Producer Heartbeat time:


emotas - CANopen DeviceDesigner 3.4.0
File Edit Options File Viewer Misc Help
CANopen
General Settings
Device Description
Object Dictionary
1
2
3
4
Type here to filter
Object Dictionary
Communication Segment
0x1000 - Device Type
0x1001 - Error Register
0x1005 - COB ID SYNC
0x1008 - Manufacturer devic...
0x1014 - COB ID EMCY
0x1016 - Consumer Heartbeat...
0x1017 - Producer Heartbeat...
0x1018 - Identity Object
0x1028 - Emergency Consumer
0x1030 - Version information
0x1032 - Active error list
0x1400 - Receive PDO Commun...
0x1600 - Receive PDO Mappin...
0x1800 - Transmit PDO Commu...
0x1a00 - Transmit PDO Mappi...
Manufacturer Segment
0x2000 - Manufacturer Data
Device Profile Segment
0x6000 - Read Input 8 Bit
0x6200 - Write Output 8 Bit
0x6401 - Read Analogue Inpu...
0x6410 - Write Analogue Out...
Attributes
Description
Overview
Mask
Heartbeat Producer Interval Configuration
Value:
1000 ms
Description:
The producer heartbeat time shall indicate the configured cycle time of the heartbeat produced by the device.
The value shall be given in multiples of 1 ms. The value 0 shall disable the producer heartbeat.
Generator Output
C:\svn\01018\CANopenFD\cofd_starter_kit\appl_cofd_starter_kit_cubeIDE\emotas_CANopen\cofd_starter_kit_cubeIDE.cddp