



User Manual

Doc Rev 1.0.13 - 30.01.2014
Kickdrive Zero V1.9.80

www.kickdrive.de

Table of Contents

1 Copyright	3
2 About This Manual	4
1 Symbols Used	4
2 Terms Used	4
3 Safety	5
1 Intended Use	5
2 Safety Notices	6
4 Overview	6
5 Installation	6
6 Using Kickdrive Zero	7
1 Getting Started	7
2 Main Screen and Toolbar	8
3 User Level	9
4 Projects and Templates	9
7 Kickdrive Zero Modules	11
1 CAN Interface	11
CAN Monitor	12
CAN Sender	14
2 CANopen Drive	15
Setup	16
Object Editor	16
Script	18
Firmware	18
3 CANopen Node	19
CANopen Node - Edit Mode	20
CANopen Node - UI Format	20
4 Network Manager	22

1 Copyright

Copyright 2014 Flachmann und Heggelbacher GbR (www.fuh-edv.de) and fullmo GmbH (www.fullmo.de)

All rights reserved. No parts of this work may be reproduced in any form or by any means without the written permission of the publisher.

Trademarks

Products that are referred to in this document may be either trademarks and/or registered trademarks of their respective owners. The publisher and the author make no claim to these trademarks.

Microsoft and *Windows* are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Disclaimer

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Contact

Technical Support for Kickdrive Zero :

support@kickdrive.de

Flachmann & Heggelbacher GbR

Waldkirchbogen 27

D-82061 Neuried, Germany

www.fuh-edv.de

Any information or support related to fullmo or ebmpapst drive products (positioning drives, linear drives, automation applications):

info@fullmo.de

Fullmo GmbH

Robert-Bosch-Straße 12

D-88677 Markdorf, Germany

www.fullmo.de

2 About This Manual

2.1 Symbols Used

The following symbols are used in this document:



Warning

As defined by these operating instructions and the danger notices on the products, death, severe bodily injury or considerable damage to property could result if the appropriate safety measures are not taken.



Notice

Important information about the product or part of the operating instructions.

2.2 Terms Used

Qualified personnel

Personnel who are familiar with the installation, assembly, commissioning and operation of the product and are qualified to carry out the respective activities.

CAN (CAN-Bus)

Controller Area Network. Field bus design according to ISO 11898-1.

CANopen

CAN-based higher layer protocol designed for use in motion-oriented machine control networks, as maintained by <http://www.can-cia.org>.

CAN Interface

Accessory to connect the PC with the CAN bus network. Usually this is a CAN-USB interface, but other options (e.g. CAN-NET via Ethernet TCP) can be available.

Node

CANopen device, identified by a **Node ID** in the range of 1..127.

CANopen Drive

EC position drive that can be configured and operated using Kickdrive Zero, e.g. ebm-papst ECI series.

Python

A programming and scripting language under OSI Open Source License, maintained by <http://www.python.org>. Kickdrive Zero currently uses the [Python 2.6.7 implementation](#).

Data Object

A device parameter or value that can be read out and/or modified using Kickdrive Zero. For CANopen devices a data object is referenced by the **Node ID**, and the object **index** and **subindex**. Kickdrive Zero uses the following standard format for a Data Object ID:

127.1000h.00h (here: Node ID = 127, object index 1000h and subindex 0h)

CANopen object data values are obtained using **SDO** or **PDO** transfer.

SDO Service Data Object

CANopen SDO transfers are used for low priority reading and writing of device parameters and objects. **SDO Upload** (read from device) and **SDO Download** (write to device) specifications define different standard methods of reading and writing different types of data, including optimized transfers for large blocks of data.

PDO Process Data Object

CANopen devices produce or consume PDO CAN frames. They are used for fast data transfer of critical process data. Before using PDO transfer, the CANopen device must be configured, i.e. the content of the PDO object must be defined (PDO Mapping). Two types of PDOs are defined for CANopen:

Transmit PDO (TPDO) - process data produced by the device node, e.g. status and position information.

Receive PDO (RPDO) - process data consumed by the device node, e.g. control command.

XDD / XDC

XML Device Description File (**.xdd**) / XML Device Configuration File (**.xdc**).

XDD files contain a dictionary of device functions and parameters and supersede the classic .eds files (Electronic DataSheet).

XDC file in addition contain parameter values for device configuration and supersede the .dcf file format (Device Configuration File).

3 Safety

3.1 Intended Use

This software is intended for the commissioning and service of position drives and other motion-oriented machine control units. This software accesses the devices via a high-level application layer protocol such as CANopen.

3.2 Safety Notices



Warning

For all commissioning and service tasks, please refer to the technical documentation of your drive unit or automation device controlled by this software. All works must be performed by Qualified Personnel familiar with the drive units or devices.

4 Overview

Kickdrive Zero is a freeware tool that can be used for a variety of configuration, commissioning and service tasks for CAN and CANopen networks and devices.

With Kickdrive Zero you can:

- Configure and operate your CANopen network and devices.
- Read or modify parameters for any device you have an .xdd/xdc (device description) file for.
- Monitor and trace your CANopen network data. Filter your data, log to a file or load an existing trace file.
- Perform firmware updates.
- Store your configuration as a project file you can reuse.

For advanced applications such as oscilloscope and data logging, creating high-quality user interfaces, Python script automation, generating one-click service tools and more, please visit www.kickdrive.de and try our full-featured product, **fullmo Kickdrive**.

5 Installation

Kickdrive Zero does not require installation. Just double click the **Kickdrive Zero.exe** file.



Notice

Additional device driver installation can be required for hardware accessories like the CAN Interface (USB-CAN). Driver installation on *Windows 8*, *Windows 7* and other recent *Windows* versions is fully automatic when connecting the device for the first time. Please follow the *Windows* instructions.

6 Using Kickdrive Zero

6.1 Getting Started



Warning

Do not operate Kickdrive Zero within an automation network where any of the Kickdrive Zero CAN or CANopen communication could possibly disrupt the normal operation.

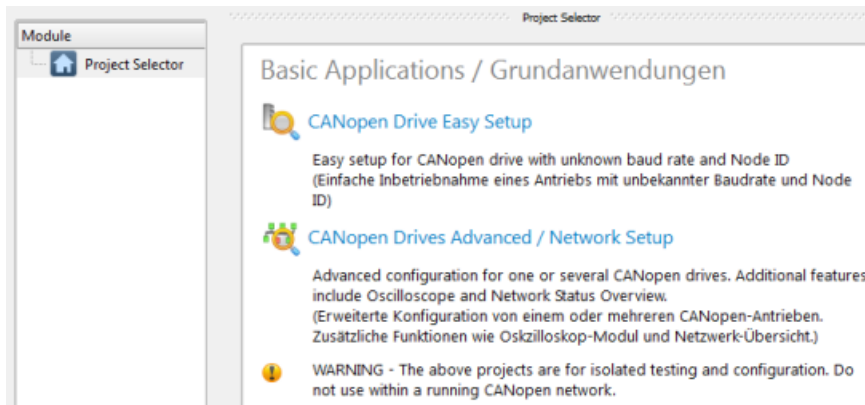


Warning


Incorrect wiring or wrong cabling can damage electronic components, including the CAN Interface, CANopen devices and the PC connected with the CAN Interface.

- Connect the **CAN Interface** to your equipment first, *before* connecting it to the PC.
- Switch on power supply for the CAN nodes.
- Connect the CAN Interface to your PC using the USB cable supplied.
- Start the Kickdrive Zero software application.

Kickdrive Zero presents a home screen that allows you to pick your specific application or project.

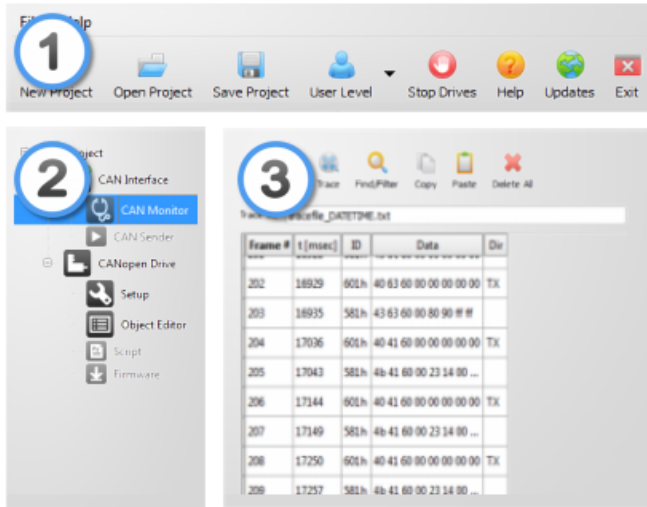


Kickdrive Zero Main Screen and Project Selector

- Pick a project or an application from the **Project Selector** list on the right.
- Alternatively, use  **Open Project** to load a Kickdrive Zero Project File (**.kickzip** or **.kickpro**) that you received for your specific application.

6.2 Main Screen and Toolbar

The Kickdrive Zero main screen is divided into three sections:



Kickdrive Zero Main Screen Layout

1 - Menu and Toolbar Area

The menu and toolbar area stays the same for all Kickdrive Zero project.

▣ Main Toolbar Functions

📄 New Project

Opens the Kickdrive Zero Home Project template. Usually, it is a project selector that allows you to pick your specific application.

📁 Open Project

Opens an existing Kickdrive Zero [project](#) (.kickzip or .kickpro) or [template](#) (.kicktpl).

💾 Save Project

Saves the currently open [project](#) including all configurations and additional files used within the project.

👤 User Level

Changes the [User Level](#). A password may be required.

2 - Modules Area

This area shows the [Kickdrive Zero Modules](#) available with the currently opened project. Some modules might be hidden by default and require a higher [User Level](#) for access.

6.3 User Level

User levels can be used in Kickdrive Zero projects to limit access to modules that are not needed for standard operation, or that need to be used with extra precautions.

The password protection is not intended to prohibit access to certain users, but rather as an indication that you are now operating on a more advanced level. The default user levels and passwords are:

User Level	Password
0 - Demo	
1 - Standard	
2 - Expert	expert



Notice

Different versions of the Kickdrive Zero software may use different user levels and passwords.



Warning

The highest User Level tasks can require additional expert knowledge about the controlled devices, or could possibly render a device unusable or cause other damage.

Changing the required User Level for a module

The minimum User Level required for a specific module is stored in the [Kickdrive Zero project file](#). To modify these settings, perform the following steps:

- Store your project as a **.kickpro** Uncompressed Project File as described in [Projects and Templates](#).
- Edit the **.kickpro** file using a standard text editor or a specialized XML editor. Search for the **userLevel="..."** attributes and change accordingly.
- To convert your modified project file back into a **.kickzip** or **.kicktpl** file, please follow the instructions in [Projects and Templates](#).

6.4 Projects and Templates

Everything you see in the [modules and content area](#) of Kickdrive Zero is defined in a Kickdrive Zero template or project file. The following file formats are used:

.kickpro Uncompressed Project Files

A Kickdrive Zero **.kickpro** file defines everything you can see in the Modules and Content area of Kickdrive Zero: [CAN Interfaces](#), [CANopen Drive](#) units, Scope modules, a.s.o.

The **.kickpro** file is a standard XML file that can be edited by experienced users to build specific applications (e.g. work with three drive units instead of one, modify the user levels required for

each module, create additional HTML documentation).

**Notice**

When manually editing **.kickpro** files, make sure you keep a copy of the original files. Check carefully that your modified file is a valid **.xml** file. A good XML editor can help validate the XML format and prevent invalid changes that could break the structure.

The **.kickpro** file is often accompanied by additional files that are necessary to run the project's predefined tasks. These files can include:

- **.css** files for documentation formatting.
- **.py** files for [Python](#) script code that is executed by Kickdrive Zero or is compiled and loaded into a drive unit.
- **.bin** firmware files for updating drive unit application firmware.

☐ **.kickzip Compressed Project Files**


.kickzip files are compressed file archives that contain:

- One **.kickpro** file of the same name as the **.kickzip** archive.
- Additional helper files (**.css**, **.py**, **.bin**, ...) as described above.

When opening a **.kickzip** project, Kickdrive Zero creates a temporary folder and extracts all files to this location.

When saving a **.kickzip** project, Kickdrive Zero first creates an updated **.kickpro** file in the temporary folder, and then creates the compressed **.kickzip** file containing all files from the temporary folder, including sub-directories.

**Notice**


You can extract the **.kickpro** file and all helper files by using menu **File >**  **Save As...** and choosing **Kickdrive Zero Uncompressed Project (.kickpro)** instead of the default **.kickzip** file format.

**Notice**

To create a **.kickzip** compressed file archive, first create a standard **.zip** file archive that contains the **.kickpro** file and all required helper files. Then change the file extension from **.zip** to **.kickzip**.

☐ **.kicktpl Template Files**

A Kickdrive Zero installation comes with a number of predefined projects for specific tasks, stored as **.kicktpl** template files. The Project Selector and the projects defined in [Getting Started](#) are examples for such templates.

Opening a **.kicktpl** file is the same as opening a **.kickzip** file, except that no file name for saving the project is chosen. If you are saving the project for the first time using  **Save Project**, you are prompted to provide a location and name of the new project - which will be saved as a **.kickzip** file.

**Notice**

You can create your own templates by using a **.kickzip** file and simply changing its extension to **.kicktpl**.

7 Kickdrive Zero Modules

7.1 CAN Interface

The **CAN Interface** module controls the communication interface to the CAN network. This is usually a CAN-USB interface with the following parameters:

Communication Port

The default setting (**first found**) will use the first fullmo **USB2Drive** or **USB-CAN** device found. It is the standard setting for CAN / CANopen applications.

Third-Party Hardware Support

The **CAN Interface** module also supports several third-party hardware configurations. The following interfaces can be selected in the **Communication Port** box:

Communication Port	Device	Device Drivers and Technical Support
vscom or COM5, COM6 etc.	VSCOM USB-CAN. (The COMx syntax is an alternative which allows multiple CAN interfaces for special applications.)	http://www.vscom.com
ixxat	IXXAT CAN devices (VCI 3 API)	http://www.ixxat.com
lawicel	Lawicel CANUSB	http://www.canusb.com
peak	Peak PCAN-USB products	http://www.peak-system.com
kvaser	All Kvaser CAN products supported by the Kvaser CANLIB API	http://www.kvaser.com
ems	CAN PC Interfaces from EMS Thomas Wünsche	http://www.ems-wuensche.de
castor	Movimento Castor USB	http://www.movimentogroup.com
mhs	MHS Elektronik Tiny-CAN	http://www.mhs-elektronik.de
zanthic	Zanthic CAN4USBFX	http://www.zanthic.com

**Notice**

Third-party hardware usually requires the installation of additional drivers available from the vendor. Please consult the documentation supplied by the vendor to make sure your CAN interface is properly installed and configured. For any issues or questions concerning specific devices, please contact our [Technical Support](#) for assistance.

**Notice**

Kickdrive Zero only provides basic support for third-party hardware interfaces and does not intend to support their capabilities to a full extend. Usually only one CAN channel is available though Kickdrive Zero, and time stamping is based on PC time stamps, not on more accurate hardware time stamps that a specific interface might provide.

Baud Rate

The (`scan`) setting is used when you are connecting to a CAN network or device with an unknown baud rate. The baud rate scan will actively reset the nodes and try to access any existing CAN node.

**Warning**

The baud rate scan uses a CANopen NMT communication reset which will might will interfere with any running application. Never use baud rate scan in an automation network or in a drive application which is in normal operation mode.

**Notice**

The baud rate scan or any previous access attempt with mismatching baud rate can render the network inaccessible. I.e. one or all CAN nodes, or the CAN interface's controller itself have entered a "bus off" state and are no longer communicating. If the baud rate scan fails for no obvious reason, please power cycle all CAN nodes and then the CAN Interface itself.

7.1.1 CAN Monitor

Use the **CAN Monitor** module to display the current communication on the CAN Bus, or log it to a text file using the **Record** function.

The **Find/Filter** function allows to search for specific CAN telegrams, resp. display a filtered version of the CAN bus traffic. The drop-down list in the **Find/Filter** contains some presets for useful filtering options. demonstrating how the filter works.

Filter Conditions

`id = <MyCobID>` - show only CAN frames with a specific COB ID

`id != <MyCobID>` - show only frames that do NOT contain this COB ID

`dir = tx` - show only frames transmitted by Kickdrive Zero

`index = <ObjIdx>, subindex = <ObjSubIdx>` - show only CANopen frames related to a SDO with the specified `ObjIdx` and `ObjSubIdx`

`data = 80*` - show only frames where the CAN data starts with hex 80

Multiple Conditions

You can use more than one filter condition in one filter, e.g.

`dir = tx, index = <ObjIdx>, subindex = <ObjSubIdx>` - show only frames from Kickdrive Zero that contain `ObjIdx` and `ObjSubIdx`. Only CAN frames that match ALL filter conditions (`dir`, `index` and `subindex`) will be displayed.

Inverting your filter

Starting your filter with the keyword **exclude** will hide anything that matches the filter condition, e.g.

`exclude id = 581h, index=606c, index=6041` - show only SDO traffic for object 606c and 6041 for CANopen node 1

Comments

Use a **#** to add additional free text comments about your filter, e.g.

`id != 81-ff # hide emergency msg`

Auto Filter

Pressing **Auto Filter** will add a filter condition to suppress the CAN frame type with the most frequent appearance. Repeated use of **Auto Filter** is a convenient way to get an almost steady display where only rare CAN frames (e.g. control PDOs or emergencies) are shown.

Recording Monitored Frames

Use the **Record** button to start recording CAN frames to a text file. By default, the traced file will be stored in the projects' folder with the date-time stamp of the beginning of recording. Make sure you save the project to have the recorded file accessible in the future.

To load a recorded file, use the **Open** button - this will load all CAN frames from the recorded file into the CAN monitor. Optionally, use **Delete All** to clear the monitor before opening the file.



Notice

Filtering only affects the screen display, not the actual CAN frame buffer or trace file recordings. After deactivating **Find/Filter**, the full CAN traffic is shown again.



Notice

The CAN Monitor display will only hold a limited amount of CAN frames in memory. Please use the **Record** function to record an unlimited amount of CAN frames to a standard text file (**.txt**). You can re-open a CAN trace file using any textual editor, or by using the **Open** button. This will load all frames into the monitor.

7.1.2 CAN Sender

The **CAN Sender** module can be used for simple CAN testing applications which require sending one or several fixed CAN frames.

It is not a full scripting engine (for that, use the Automation module) but a simple macro sequence for quick and easy testing.

Available commands:

```
send <ID>,<data Size>,<data Hex>
```

```
delay <milliseconds>
```

```
wait <ID>,<data size>,<data Hex>[,<timeoutMillisec>]
```

ID and data size can be provided as hex expressions:

```
send 0x7e5,2,04 01
```

or

```
send 7e5h,0x2,04 01
```

wait supports string wildcards * and ? for the <data hex> parts. An additional timeout can be specified.

Example:

```
wait 581h,8,4b 41 60 00 ?? ?4 *,1000
```

(waits for a status word where the highest four bits are "4". gives up after one second)

The following macros are supported inside the *send* and *wait* commands:

canid(<expression>) - calculates the expression and returns a CAN ID in hex format

canunsigned32(<expression>) or *caninteger32*(<expression>)

- calculates the expression and returns a 4-byte CAN data block

canunsigned16(<expression>) or *caninteger16*(<expression>)

- as above, but 2 byte

canunsigned8(<expression>) or *caninteger8*(<expression>)

- as above, but 1 byte

Example:

```
nodeId = 1
```

```
velocity = 1000
```

```
send canid(0x600 + nodeId),8,23 ff 60 00 canunsigned32(velocity)
```

A simple *do...loop* syntax can be used for endless loops. Or use *do...loop until* found in combination with a *wait* macro.

Example:

```
do
  do
    ; request status
    send canid(0x600 + nodeId),8,40 41 60 00 00 00 00 00
    ; wait for target reached (status word upper four bits = 4)
    ; using 0.5 seconds timeout
    wait canid(0x580 + nodeId),8,4b 41 60 00 ?? ?4 ?? ??,500
  loop until found

  ; target has been reached. wait until drive moves again

do
  send canid(0x600 + nodeId),8,40 41 60 00 00 00 00 00
  wait canid(0x580 + nodeId),8,4b 41 60 00 ?? ?0 ?? ??,500
loop until found
loop
```

7.2 CANopen Drive

The **CANopen Drive** module offers basic configuration, testing and service tasks for CANopen positioning drives supported by this Kickdrive Zero application. Details on the actual drive models supported can be found in the [Kickdrive Zero Projects / Templates](#) included in your specific Kickdrive Zero software version.



Notice

The following sections on CANopen Drive panels provide a basic overview of the available functions. Details can depend on the actual drive model used and are documented in the [Kickdrive Zero Projects / Templates](#) section of the application itself.

7.2.1 Setup

The **Setup** panel is used for various configuration and simple testing tasks:



Notice

Use initial setup functions with care. It is strongly recommended to perform initial setup functions with only one drive unit connected to Kickdrive Zero, and not within a full CANopen installation with several drive nodes.

Initial Setup

Assign or change the CAN Node ID or CAN Baud Rate.

The following panels are available after starting a drive node using the **Start Drive** button:

Test Velocity

Switch the drive to operation mode "Profile Velocity Mode" and perform basic velocity control (set target velocity, acceleration and deceleration).

Test Position

Switch the drive to operation mode "Profile Position Mode" and perform basic positioning control. (Set target position and enable positioning; set additional profile position parameters like velocity.)

Homing


Test basic homing methods, including Homing Mode 35 to reset the actual position to zero (new reference position) without moving the drive.

7.2.2 Object Editor

The **Object Editor** module is used for reading, editing and writing device parameters. It is also where you can define any [Data Object](#) that can be used in other modules like the [CANopen Node](#).

The Object Editor panel consists of two parts:

Object Dictionary (Device Description)

The upper part of the Object Editor panel is used to display a dictionary of parameters available for a device. The parameter descriptions are defined by a [XDD / XDC](#) configuration file. Use the  **Open Dict.** function to load a different / new configuration file.

Object Editor


Drag one or several parameters from the Object Dictionary to the Object Editor table to make them available for reading, editing and writing.

Object Editor Basic Functions

Read

Read the selected values from the device (CANopen Upload). If nothing is selected in the Object Editor table, all values are read.


Edit

Use this function or double click on an element in the Value / Editor column. The values are not written to the device immediately, unless  **Synch Mode** is used.

Write

Write the selected values to the device (CANopen Download). If nothing is selected in the Object Editor table, all writable values are downloaded.

Store

Store the written parameters permanently. I.e. the CANopen device will continue to use these values after a voltage reset. This function is used after writing modified values using the  **Write** function.


Reset

Reset all device parameters to the factory settings.

Synch Mode

Read read-only parameters periodically. Writable parameters are read once. After that, every modification entered by the user will be downloaded to the device immediately.

Default

Copy the value from the dictionary's **Default** column to the **Value** column, for the selected rows. If no rows are selected, all objects will be updated to the default values. To write these defaults into the device, use  **Write**.

Scaling

When operating at [User Level 2](#), you can use the **Scaling** column to define how values should be scaled before showing them in the **Value** column.

This allows scaling and processing the raw data values. You can use a simple factor or more complex formulas, including offsets or even non-linear transformations. Examples:

Scaling	Description
0.25	Scale the value by 0.25 (i.e. the displayed value is four times smaller than the raw data value)
100	Scale the value by 100 (i.e. the displayed value is multiplied by 100)
$= x / 65536 + 10$	Divide raw data value by 65536, then add 10

$= 1 / x$	Non-linear expressions like this are possible, too. However, watch out for invalid results, e.g. "division by zero". Kickdrive Zero will display these values as "0"
-----------	--

**Notice**

for **Read/Write** values, only simple scaling factors like **0.25** are permitted. On **Write**, the entered value will be transformed back to the original raw `unscaled` value. When using scaling expressions like $=x/20+10$, the object value becomes read-only and cannot be edited or written to the device.

7.2.3 Script

The **Script** Module is used to edit, compile and download Embedded [Python](#) scripts into a drive unit that supports it.

**Notice**

[User Level 2](#) is required to access the **Script** module in the basic CANopen Drive Setup template.

**Notice**

For details on the Embedded Python engine, sample code and applications, please refer to the documentation available for your CANopen drive unit or contact Applications Support.

7.2.4 Firmware

Using the **Firmware** module you can update your device's application firmware with a new software version provided in a **.bin** file format.

**Notice**

[User Level 2](#) is required to access the **Firmware** module in the basic CANopen Drive Setup template.

**Notice**

Before starting a Firmware download, make sure the Node ID is correct and the device is accessible, e.g. using the [Drive Setup](#) panel.

**Warning**

Please make sure that you are using a correct **.bin** file intended for your specific device. Kickdrive Zero does not perform additional checks before downloading the file. Using a wrong or mismatching **.bin** file can damage your drive unit and make it impossible to download a correct **.bin** file in a second attempt.

7.3 CANopen Node

The **CANopen Node** module offers a basic configuration screen for any type of CANopen node, e.g. a third-party servo drive, I/O module or sensor device.

The **Parameter Editor** table contains a list of CANopen objects and their values, similar to the [Object Editor](#).

For the selected object, additional display and editing controls are shown below the table. These elements are configurable by changing the [UI Format](#) column in **Design Mode**.

**Notice**

The appearance and behavior of the **Parameter Editor** depends on the [Edit Mode](#) setting. Some of the functionality described below is only relevant to specific **Edit Mode** settings.

+ Parameter Editor functions


 **Read**

Read the value of a parameter, or of the entire list, depending on the [Edit Mode](#).

 **Write**

Write to the device the value of a selected parameter, or of the entire list, depending on the [Edit Mode](#).

 **Store**

Store the written parameters permanently. I.e. the CANopen device will continue to use these values after a voltage reset. This function is used after writing modified values using the  **Write** function.

 **Reset**

Reset all device parameters to the factory settings.

 **Design Mode**

Allows adding and removing parameters from the list, as well as editing the **UI Format** and **Comment** columns, to modify the display and editing controls used for this element.

In **Design Mode** you can:

- Add more objects: Drag data objects from the [Object Editor](#) and drop them on the Parameter

Editor. Alternatively, **Copy** data objects from the Object Editor and **Paste** them into the parameter table.

- Modify the parameter's **Name**, **UI Format** and **Comment**. The **UI Format** configures the UI elements that are displayed for the parameter, as described in the [UI Format](#) documentation. The **Comment** defines the additional description text shown below the name in the UI elements' section.
- Apply [Scaling](#) factors and formulas, in the same way as for the [Object Editor](#).
- Modify the [Edit Mode](#).



Notice

[User Level 2](#) is required for accessing **Design Mode**.

7.3.1 CANopen Node - Edit Mode

The **Edit Mode** affects the appearance and behavior of the [CANopen Node](#) module. It can be modified from **Design Mode**.



Notice

[User Level 2](#) is required to switch to **Design Mode**.

Name	Description
0 - Standard	The read/write behavior is similar to the Object Editor , Read / Write works on the currently selected object only.
1 - List Read	No Write / Edit functionality, Read always reads the entire list of parameters, not only a single object.
2 - List Read/Write	No Store / Reset functionality. Read / Write always affect the entire list of parameters.
3 - Auto Store	Same as the EditMode = 2 but Write will always execute a Store action after writing the values, to make the parameter change permanent.

7.3.2 CANopen Node - UI Format



Notice

[User Level 2](#) is required to switch to **Design Mode**, which makes the **UI Format** column available for editing.

The **UI Format** column in the **Parameter Editor** table configures the editing control available for

each data object.

Custom **UI Format** settings for each object allow a clean and intuitive on-the-fly visualization of its value. For example, a dial and a slider for integer data within a well-defined min/max range, or a combo box for multiple-choice parameters.

The **UI Format** value is a list of key-value attribute pairs, delimited by semi-colons (“;”). The following is a description of the available attributes:

type

Defines the type of data to be visualized.

type	Description
integer	Integer values are displayed as a textbox, a dial and a slider. See the range attribute to define min/max values.
text	Texts are displayed using a textbox only.
combo	A combo box. This is useful for objects that accept a limited set of different values with different meanings. The values available and their descriptions are defined using the options attribute. See below for more information and an example.

range

Defines the minimum and maximum values. For example:

```
range:0,100
```

```
range:-100,100
```

options

Used with `type:combo`. This specifies a list of available options and their corresponding values.

Format:

```
options:<value_1>,<text _1>|<value_2>,<text_2> | ... (etc.)
```

For example, here is a combo box to choose between 4 different values (0, 1, 3 or 6):

```
type:combo; options:0,Reset|1,Profile Position|3,Velocity Mode|6,Homing Mode
```

Example UI Format Definitions

UI Format	Description
<code>type:text</code>	A simple label or a text box. This can be used to display or edit (if writable) any value.
<code>type:integer; range:0,100</code>	A textbox, a slider and a dial, accepting values between 0 and 100.



<pre>type:combo; options:0,Reset 1,Profile Position 3,Velocity Mode 6,Homing Mode</pre>	<p>A combo box with the specified options (Reset, Profile Position, Velocity Mode, Homing Mode) and corresponding values (0, 1, 3, 6).</p>
--	--

7.4 Network Manager

Use the **Network Manager** to perform a baud rate and node scan for an unknown network, and build a Kickdrive Zero [setup and operation](#) module for every [supported drive node](#) found.

Network Scan

Use the following procedure for a successful scan:

- Connect the [CAN Interface](#) to the CANopen network
- Connect the CAN Interface to your PC via USB.
- Switch on power supply for the CANopen nodes
- Make sure in the **CAN Interface** module, **baud rate** is set to either (**Scan**), or the actual baud rate of your network
- Choose  **Scan Network** to find out about:
 - Scanned Node IDs** - all existing node IDs within this network
 - Configurable Node IDs** - [supported drive nodes](#) available for [setup and operation](#) through Kickdrive Zero
- Choose  **Build Project** to create a Kickdrive Zero module for each of the IDs listed in the **Configurable Node IDs** text field.




Warning


Never use baud rate scan or similar configuration tasks in running machinery or in a CANopen application in normal operation. See also the additional information in the [CAN Interface](#) module description.

Node Monitor

You can use the Node Monitor for a simple network overview which displays some basic information on every Node ID that is listed in the **Scanned Node IDs** text field.

- Make sure the **Scanned Node IDs** list is the list of nodes you want to monitor.
- Click the  **Node Mon. Config** to display a list of parameters that should be read for every node. You can drag additional [Data Objects](#) from the [Object Editor module](#), and drop them on the **Node Monitoring Configuration** table.
- In the **UI Format** column, use the **updates:** keyword to specify how the values should be obtained, e.g.
 - updates:r** - read only once when starting the **Node Monitor**
 - updates:r2000** - read every 2000 milliseconds

(See UI Panel / UI Format Definition section for additional details on the **updates:** specifier)

- Click on  **Node Monitor** to start the network overview / node monitoring.

**Notice**

Node Monitor uses [SDO](#) transfer for reading out the values from the network nodes. Please make sure this is not conflicting with other operations on the CANopen network, e.g. service operation tasks such as [Firmware downloads](#).