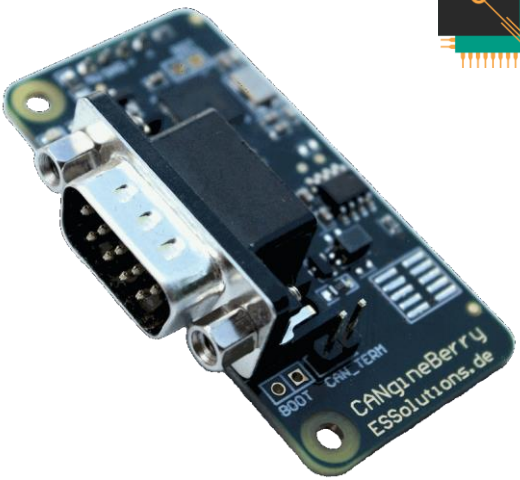
CANgineBerry

Active CAN and CANopen interface for embedded computers

for revision 1.0 or higher



Rev. 1.00 of 1st February 2018

Published by
Embedded Systems Academy GmbH
Bahnhofstraße 17
D-30890 Barsinghausen, Germany
www.esacademy.com

CANgine products by
Embedded Systems Solutions GmbH
Industriestraße 15
D-76829 Landau, Germany
www.essolutions.de

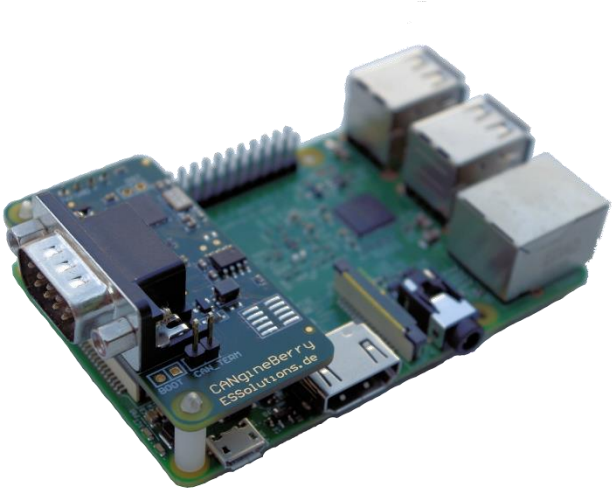
1 Contents

2 Module Overview 3
3 Hardware 6
3.1 Dimensions 6
3.2 Technical Data 7
3.3 Pin assignment header socket 7
3.4 Pin assignment CAN connector 7
3.5 Jumpers 7
4 Using CANgineBerry with a Raspberry PI® 3 9
5 Loading an application Firmware 10
5.1 Using the COIAUpdater utility 10
5.2 Custom applications? 12
6 CANgineBerry Applications 13
6.1 CANopenIA-MGR: Minimal CANopen Manager 13
6.2 CANopenIA-BEDS: CANopen Device (Slave) 14
7 CANopenIA Command Line Utility 15
7.1 Functionality 15
7.2 Command Line Interface usage 15
7.3 Command Line Parameters 15
-p, --port=<name or number> 18
-m, --monitor --limited-monitor=<milliseconds> 18
-w, --write=<index>,<subindex>,<bytesize>,<value> -r, --read=<index>,<subindex> 18
--node-write=<nodeid>,<index>,<subindex>,<bytesize>,<value> --node-read=<nodeid>,<index>,<subindex> 18

2 Module Overview

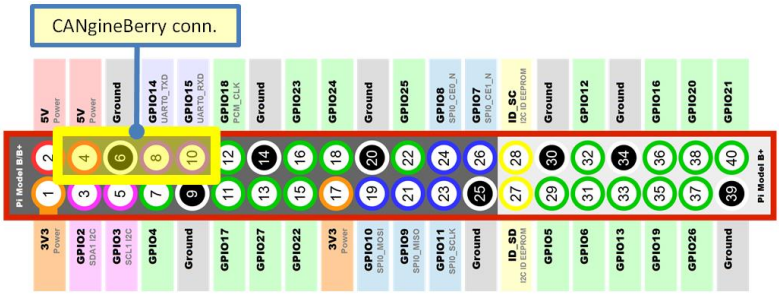
CANgineBerry is an active CAN module intended for the Raspberry Pi®. It has a CAN (DB9) connector and uses only four pins as connection to the Raspberry Pi: +5V, GND and the 3.3V Rx/Tx signals of the serial channel. Applications use a green and a red LED to indicate system states and errors.

The form factor of the module is designed to fit directly on top of the Raspberry Pi® 3, connecting directly to the UART in the GPIO header row.



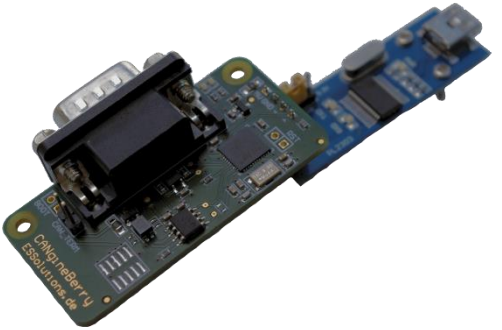
CANGINEBERRY ON RASPBERRY PI® 3

Be careful when plugging the CANgineBerry onto the Pi to use the right pins. **Never plug or unplug the CANgineBerry while the Raspberry Pi® is connected to power!**



CANGINEBERRY CONNECTOR ON RASPBERRY PI® GPIO

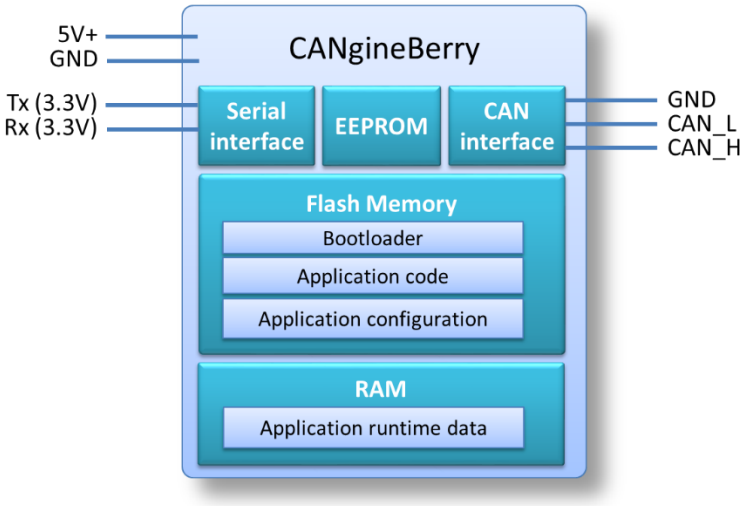
The module can also be used with any other hosts as long as they provide the four required signals. If the host has only USB, then a USB-to-serial converter chip or module can be used to interface to CANgineBerry.



CANgineBerry with a serial (TTL UART) to USB conversion module

Watch out!
The power supply to CANgineBerry is +5V, while the TTL serial pins are low-voltage at 3.3V max.!

The module comes with command line tools for Windows and the Raspberry Pi® running Linux.



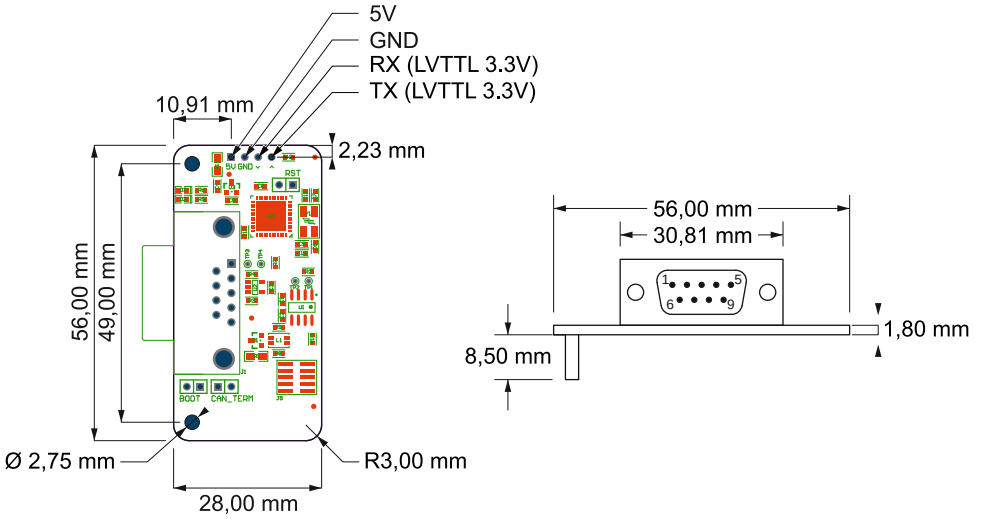
CANGINEBERRY OVERVIEW

CANgineBerry has an internal bootloader to load one of the provided CANgineBerry application firmware images for CANopen, CANcrypt or other CAN applications. The application firmware is loaded from the Raspberry Pi or other host with the COIAUpdater command line utility. Some applications use a separate configuration area in Flash that can be programmed individually. The EEPROM is typically used to store settings like the CAN bitrate or a node ID.

3 Hardware

The CANgineBerry is made in Germany. The intended use is as active CAN or CANopen interface for the Raspberry PI® 3, as its dimensions, host connector and mounting holes are designed to match the Raspberry PI® 3's. Using adapters or custom designs, the CANgineBerry can be fitted onto many other computerized platforms. Although use on such other platforms is possible, warranty is limited to operation with the Raspberry PI® 3.

3.1 Dimensions



CANGINEBERRY DIMENSIONS

3.2 Technical Data

Power supply	4.5V-5.5V
Power consumption	22 mA (typ.)
CAN transceiver	MCP2562FD
CAN connector	Sub-D 9 male
CAN bitrate	Configurable, default 125.000 bps
UART	Female jumper array 2,54 mm
UART baudrate	Configurable, default 921.600 bps
Display	LED RUN (green) and LED ERR (red)
Dimensions	56 x 28 x 28 mm³
Weight	< 15g
Temperature range	-40 to +85 °C

3.3 Pin assignment header socket

Pin	Signal
1	5V
2	GND
3	RX (High 1.85 to 3.3V)
4	TX (High 1.85 to 3.3V)

3.4 Pin assignment CAN connector

Pin	Signal	Pin	Signal
1	nc	6	GND
2	CAN_Low	7	CAN_High
3	GND	8	nc
4	nc	9	nc
5	nc		

3.5 Jumpers

The module has jumpers for advanced users:

Jumper	Function	Remarks
CAN_TERM	On = 125 Ohms CAN termination Off = no CAN termination	Leave off and use external termination if possible
RST	Reset	Connect to generate a hardware module reset
BOOT	Activate STM32 on-chip bootloader after reset	See warning below

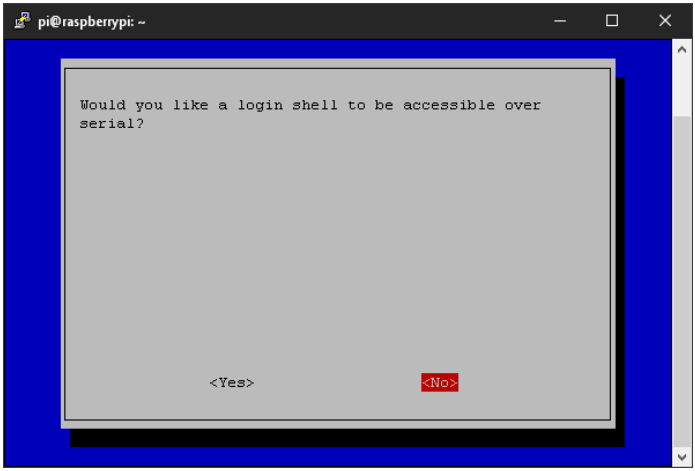
All warranties expire if the BOOT jumper is used,
or any soldering job is performed on the CANgineBerry!

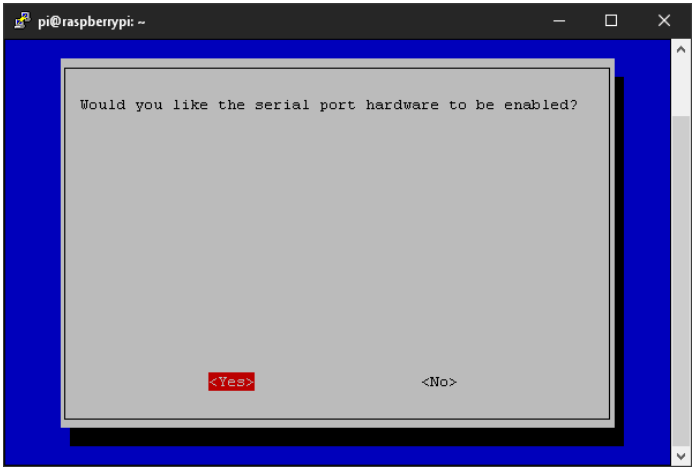
4 Using CANgineBerry with a Raspberry PI® 3

In the Raspberry PI® 3, the UART at GPIO14 (pin 8) and GPIO15 (pin 10) is not available by default. It has to be enabled and the login shell via UART disabled in the raspi-config tool for it to be used with the CANgineBerry:

```
sudo raspi-config
```

Go to 5 – Interfacing Options, P6 – Serial, then:





Confirm and select “Finish” to save, then reboot. Note that Bluetooth is now no longer available.

5 Loading an application Firmware

The CANgineBerry supports loading code and configuration files through the serial interface. This gives the host full configuration control, as it can activate the desired functionality and configuration.

The CANgineBerry must follow a bootloader activation sequence (to prohibit accidental misconfigurations). The provided *COIAUpdater* command line utility automatically handles the bootloader activation sequence.

The command line utility (see chapter 7 CANopenIA Command Line Utility) is available for Windows and the Raspberry Pi® running Linux.

5.1 Using the COIAUpdater utility

The *COIAUpdater* utility is used to load an application, or an update of an application, into the Flash memory of the CANgineBerry. If the application loaded is a CANopen application supporting the Binary Electronic Data Sheet (BEDS), then these configuration files can also be loaded using the *COIAUpdater*.

On a command line, type

```
COIAUpdater -h
```

to get a list of supported parameters. Output:

```
COIAUpdater version x.xx.xxxx
((c) EmSA 2018, All Rights Reserved, www.em-sa.com
Usage: COIAUpdater [options]

Options:
  -h, --help
      Shows this help

  -p, --port=<name or number>
      Serial port to use. May only be used once. MANDATORY

  -f, --flashfirmware=<firmwarefile>
      Programs the firmware

  -d, --flashdata=<binaryedsfile>
      Programs the data, e.g. containing a binary object dictionary defini-
tion. Needs to be supported by the firmware

  -n, --node-id=<node_id>
      Configures CANopen node ID. Must be between 1-127.

  -r, --bitrate=<can_bitrate>
      Configures CAN bitrate in kbps. Supported values: 125, 250, 500,
1000.
```

```
  -l, --lss=<lss_enabled>
      Configures enable or disable CANopen LSS support.

  -i, --info
      Shows information about the COIA device

  -c, --command-file=<filename>
      File containing command line options. This is processed after all
other options. May only be used once
```

A Windows or Linux host system supports multiple serial ports. These are numbered and when starting the application, the serial port number that has the CANgineBerry device connected needs to be specified (for Windows, check the Device Manager) using the -p parameter.

The -f parameter is used to select the application update file programmed to the CANgineBerry. CANgineBerry application files usable are provided for download on cangineberry.com. Example programming the COIA “BEDS” application:

```
> COIAUpdater.exe -p 2 -f CgB_COIA_BEDS1.0_sec.bin

COIAUpdater version 1.03.4226
(c) EmSA 2018, All Rights Reserved, www.em-sa.com
Connected to COM6 at 921600 bps
Firmware file to write: CgB_COIA_BEDS1.0_sec.bin
Waiting for bootloader to start...
Bootloader is running.
Bootloader version:     1.2
Hardware serial number: 0A801B000943364638363520
Stored configuration:
  Node ID:              9 (0x09)
  CAN bitrate:          125 kbps
  LSS:                  disabled
Application found:
  Type:                 0x01 (CANopen IA BEDS slave)
  Version:              1.0
Erasing flash...done.
Programming.........................done.
Update completed.
Disconnected from COM2
```

If a CANopen application loaded supports the BEDS configuration files, these can be loaded using the -d parameter. CANopen BEDS configuration files usable are provided for download on cangineberry.com. Custom BEDS configurations can be created with the optional CANopen Architect EDS Editor and Checker that can be downloaded for free at esacademy.com/en/products/free-software.html.

5.2 Custom applications?

The CANgineBerry does NOT support loading custom firmware versions!

Only use the provided command line utility to load a new software!

Using other code loading utilities might erase the bootloader. If the bootloader gets erased, the CANgineBerry can no longer load CANgineBerry applications!

6 CANgineBerry Applications

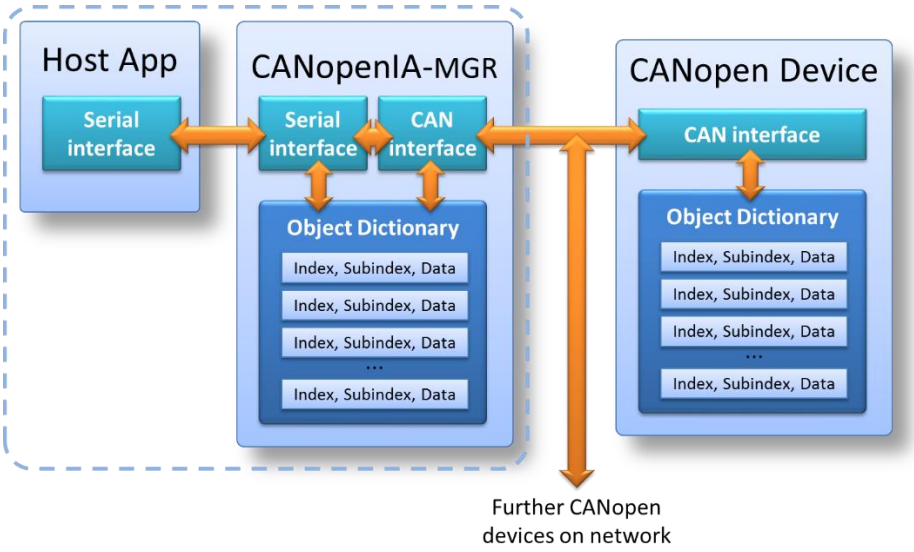
The default application loaded upon delivery is the CANopenIA-MGR minimal CANopen Manager that gives the host quick and easy access to a network of CANopen devices.

For a complete list of all available CANgineBerry applications, see the CANgineBerry.com web page. See the next chapter for instructions on how to load an application or its configuration into the CANgineBerry.

6.1 CANopenIA-MGR: Minimal CANopen Manager

This application firmware turns the CANgineBerry into a minimal CANopen Manager. It can handle up to 31 CANopen devices (with node IDs in range from 2 to 31).

The configuration requirements are minimal, as the Manager auto detects the devices. It then scans all vital communication parameters and configures itself to consume and produce all matching CAN messages (CANopen PDO communication).



CANgineBerry as Minimal CANopen Manager

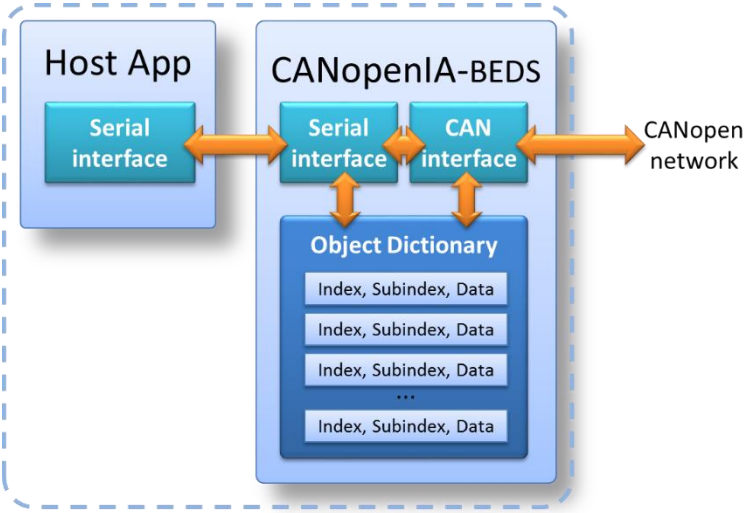
The Raspberry Pi receives simple serial indications with new data arriving (indicates from which node ID and which data object by index and subindex) and may use commands to send data (also based on node ID, index and subindex of data object) to the connected devices.

For a detailed description of the CANopenIA-MGR application, see the CANopenIA co-processor manual delivered with the CANopenIA-MGR firmware.

6.2 CANopenIA-BEDS: CANopen Device (Slave)

This application firmware implements a CANopen device (slave). The device profile supported depends on the binary electronic data sheet (BEDS) loaded. Several default BEDS files are provided along with the application. These provide the communication variables and parameters for a generic input/output device for digital and analog data, for an encoder or for a simple stepper motor.

Custom BEDS files can be created with EmSA's CANopen Architect, an editor for CANopen EDS files.



CANGINEBERRY AS CANOPEN DEVICE

For a detailed description, see the CANopenIA-RA-Pub document.

7 CANopenIA Command Line Utility

The simple command line utility “COIA” can be used with the CANopenIA-MGR and CANopenIA-BEDS firmware versions. The utility interacts with the firmware and can be used to trigger local as well as remote access (MGR application only).

7.1 Functionality

The following functionality is provided:

- Scan network, show a list of nodes found along with node ID, vendor ID and vendor name.
- View previous scan results
- Scanning is limited to 32 nodes, if there are more than 32 nodes then the user will be told there are others but no details.
- Set heartbeat times of all nodes on the network
- Send NMT commands to network
- Read/write access to the local Object Dictionary of the CANopenIA device
- Read/write access to remote Object Dictionaries of any node on the network (MGR only)
- Read commands from a file
- Generic CAN message transmit and receive
- Displays serial number of STM32F09.

7.2 Command Line Interface usage

The command line interface has the following format:

`COIA [options]`
where `[options]` is a list of optional options. Standard Linux format is used for options, for example:

```
-f
-g=3
--foo
--bar=”yes”
```

Running the command line utility without any options will simply print the utility name and version number.

7.3 Command Line Parameters

This section describes all supported command line options.

On a command line, type

```
COIA -h
```

To get a list of supported parameters. Output of the initial release:

```
COIA version 1.70.xxxx
(c) EmSA 2018, All Rights Reserved, www.em-sa.com
Usage: COIA [parameters]

Basic parameters:

  -h, --help
      Shows this help text

  -p, --port=<name or number>
      Serial port to use. May only be used once. MANDATORY

  --reset
      Resets the COIA device

  -i, --info
      Displays information about the COIA device

  -m, --monitor
      Monitors changes to the local object dictionary (data received).
      Continues until Ctrl-C is pressed

  --limited-monitor=<milliseconds>
      Monitors changes to the local object dictionary (data received).
      Continues for number of milliseconds specified

  -s, --node-status
      Displays a summary of the status of all seen nodes

  --set-configuration=<nodeid>|default,<bitrate_kbps>|default,temp|store
      Sets the basic COIA configuration

Local Object Dictionary access parameters:

  -w, --write=<index>,<subindex>,<bytesize>,<value>
      Write to the local object dictionary of the COIA device

  -r, --read=<index>,<subindex>
      Read from the local object dictionary of the COIA device

Remote access parameters (Manager only):

  -n, --nmt=<nodeid>|all,<command>|op|preop|stop|reset|resetcom
      COIA device sends a network management master message

  --node-write=<nodeid>,<index>,<subindex>,<bytesize>,<value>
      Write to the object dictionary of a specific node

  --node-read=<nodeid>,<index>,<subindex>
      Read from the object dictionary of a specific node
```

```
  --net-info
      Displays information about the network

  --node-info=<nodeid>
      Displays information about a node

  --rpdos=<nodeid>
      Displays the receive PDOs of a node

  --tpdos=<nodeid>
      Displays the transmit PDOs of a node

  --rescan=<nodeid>
      Rescans a node

Manager configuration parameters:

  --keep-nodes-op=yes|no
      Enables/disables keep nodes operational

  --hb-producer-time=<milliseconds>
      Sets the default heartbeat producer time

  --hb-consumer-time=<milliseconds>
      Sets the default heartbeat consumer time

  --hb-monitoring=yes|no
      Enables/disables heartbeat monitoring

  --pdo-update-time=<milliseconds>
      Sets the PDO update time

  --pdo-event-time=<milliseconds>
      Sets the PDO transmission event time

  --pdo-inhibit-time=<multiple 100us>
      Sets the PDO transmission inhibit time

  --tpdo-handling=yes|no
      Enables/disables PDOs received by devices (their TPDOs)

  --rpdo-handling=yes|no
      Enables/disables PDOs sent to devices (their RPDOs)

  --use-cache=yes|no
      Enables/disables local caching

Flow control and scripting:

  -d, --delay=<milliseconds>
      Delays processing of next command for a period of time

  -c, --command-file=<filename>
      Process a file containing command line options.
      This is processed after all other options.
      May only be used once
```

The most important parameters are explained below:

-p, --port=<name or number>

A Windows or Linux host system supports multiple serial ports. Under Windows these are numbered COM1 and up and when starting the application, the serial port number that has the CANgineBerry device connected needs to be specified (for Windows, check the Device Manager, section “Ports”) using the -p parameter. On a Raspberry Pi® 3 under Raspbian, the first UART the CANgineBerry is connected to is typically /dev/ttyS0 and the full name should be used.

-m, --monitor
--limited-monitor=<milliseconds>

Monitoring modes simply display any changes made to the local object dictionary. Every time data is written to the local object dictionary (from the CANopen side), an output is generated with {Node ID from which data is received, Index, Subindex, Data}.

-w, --write=<index>,<subindex>,<bytesize>,<value>
-r, --read=<index>,<subindex>

These parameters provide access to the local object dictionary (of the COIA device). Data can be written or read.

--node-write=<nodeid>,<index>,<subindex>,<bytesize>,<value>
--node-read=<nodeid>,<index>,<subindex>

With the CANopenIA Manager, these parameters allow accessing the object dictionaries of all accessible nodes. Although most accesses are made by CANopen SDO services, PDO services might be automatically used where supported.