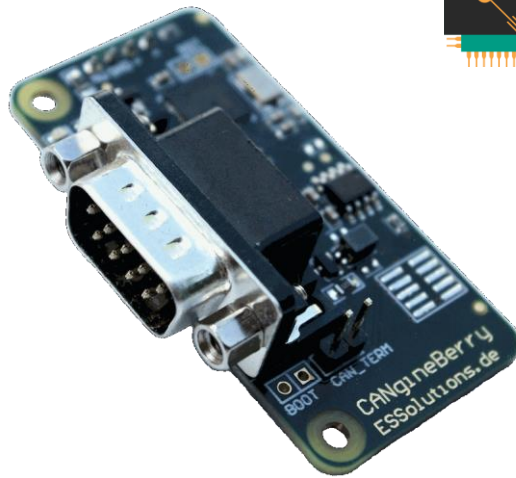


CANgineBerry

Active CAN and CANopen interface for embedded computers

for hardware versions 1.3, 1.5 and 1.6



Rev. 1.11 of 17th January 2024

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

COPYRIGHT 2017-2024 BY EMBEDDED SYSTEMS ACADEMY GMBH

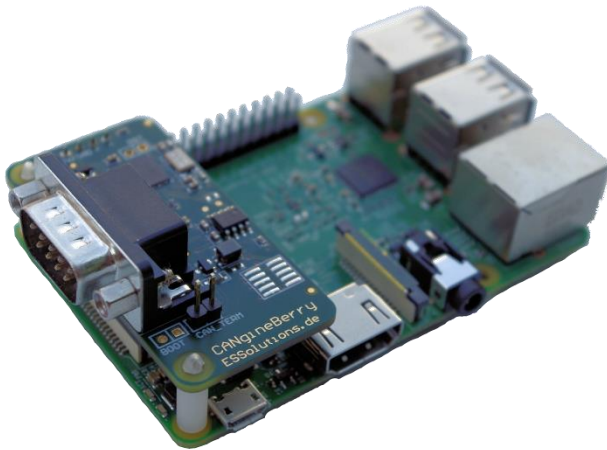
1 Contents

| | | |
|-----|--|----|
| 2 | Module Overview..... | 3 |
| 3 | Hardware..... | 6 |
| 3.1 | Dimensions..... | 6 |
| 3.2 | Hardware Versions..... | 6 |
| 3.3 | Technical Data | 7 |
| 3.4 | Pin assignment header socket | 7 |
| 3.5 | Pin assignment CAN connector..... | 7 |
| 3.6 | Jumpers..... | 8 |
| 4 | Using CAngeBerry with a Raspberry PI® 3 and up | 9 |
| 5 | Loading an application Firmware | 11 |
| 5.1 | Using the COIAUpdater utility | 11 |
| | Selecting UART port and bitrate | 12 |
| | Selecting CANopen node ID and bitrate | 12 |
| | Programming a firmware | 12 |
| | Loading a configuration..... | 13 |
| 5.2 | Custom applications?..... | 14 |
| 6 | CAngeBerry Applications..... | 15 |
| 6.1 | CANopenIA-MGR: Minimal CANopen Manager..... | 15 |
| 6.2 | CANopenIA-BEDS: CANopen Device..... | 16 |
| | CAN232..... | 16 |
| 7 | CANopenIA Command Line Utility | 17 |
| 7.1 | Functionality..... | 17 |
| 7.2 | Command Line Interface usage | 17 |
| 7.3 | Command Line Parameters | 18 |

2 Module Overview

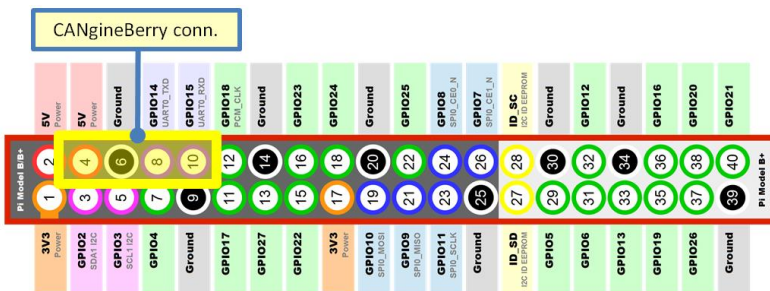
CANgineBerry is an active CAN module intended for the Raspberry Pi®. It has a CAN (DB9) connector and uses only four pins as connection to the Raspberry Pi: +5V, GND and the 3.3V Rx/Tx signals of the serial channel. Applications use a green and a red LED to indicate system states and errors.

The form factor of the module is designed to fit directly on top of the Raspberry Pi® (version 2 or higher), connecting directly to the UART in the GPIO header row.



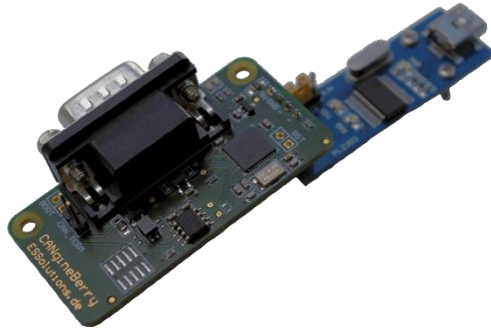
CANGINEBERRY ON RASPBERRY PI® 3

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.

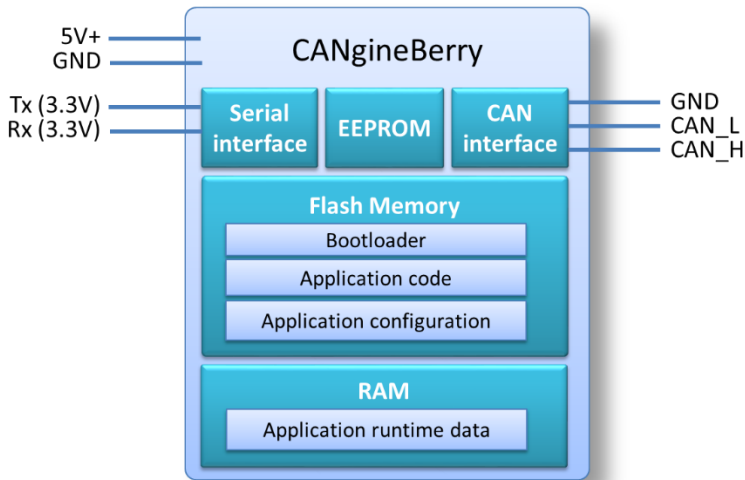


CAnGINEBERRY WITH A SERIAL (TTL UART) TO USB CONVERSION MODULE

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.



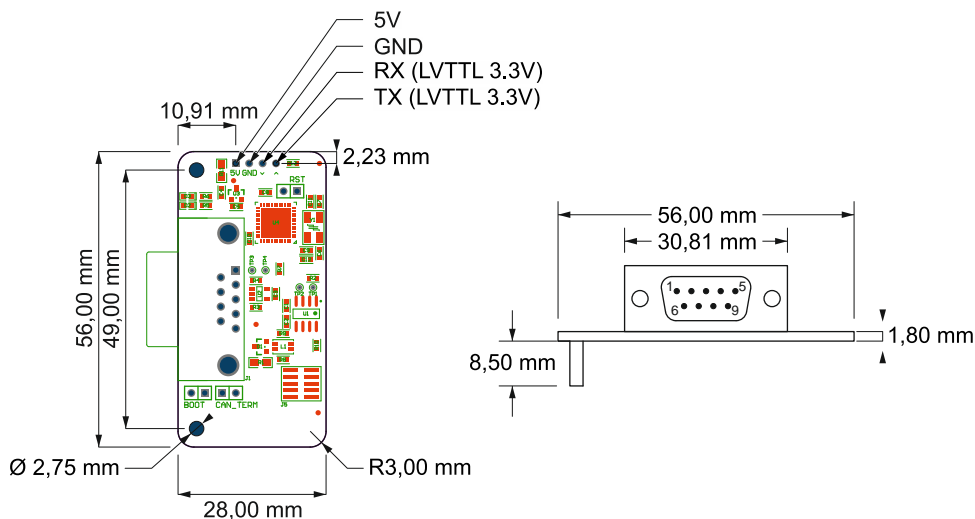
CANINEBERRY OVERVIEW

CANineBerry has an internal bootloader to load one of the provided CANineBerry 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® version 2 or higher, as its dimensions, host connector and mounting holes are designed to match the Raspberry PI®'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®.

3.1 Dimensions



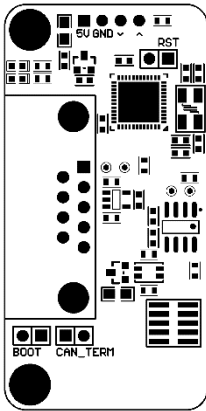
CANGINEBERRY DIMENSIONS

3.2 Hardware Versions

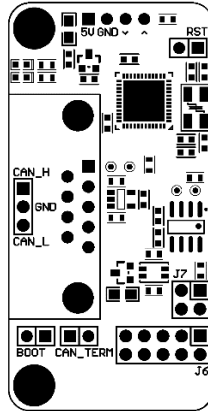
There are three different hardware versions as shown below. Some modifications were made over time to allow more options for microcontroller and CAN transceiver. Starting with version 1.5 the jumper J7 was added allowing the configuration to enforce boot-loader activation and enforcing the safe default UART speed of 115200 bps.

The latest hardware version can also be ordered as “Maker Edition”. In that version the DB9 CAN connector and all headers are NOT populated, making this version more suitable for integration into tight spaces.

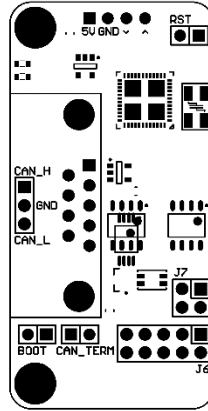
Version 1.3



Version 1.5



Version 1.6



3.3 Technical Data

| | |
|-------------------|--|
| Power supply | 4.5V-5.5 V |
| Power consumption | 22 mA (typ.) |
| CAN transceiver | MCP2562FD or similar |
| CAN connector | Sub-D 9 male |
| CAN bitrate | Configurable, default 125,000 bps |
| UART | Female jumper array 2.54 mm |
| UART baudrate | Default 921,600 bps Safe default 115,200 bps (activate on J7) |
| Display | LED RUN (green) and LED ERR (red) |
| Dimensions | 56 x 28 x 28 mm ³ |
| Weight | < 15 g |
| Temperature range | -40 to +85 °C |

3.4 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) |

3.5 Pin assignment CAN connector

The regular CANineBerry uses a male DB9 connector for CAN. Starting with hardware version 1.5, a “Maker Edition” of the CANineBerry is available without the DB9 con-

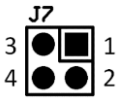
necter where new connection pads become visible, with GND, CAN_Lo and CAN_Hi clearly marked.

Pin assignment of the DB9 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 | | |

3.6 Jumpers

The module has single jumpers for advanced users.

| Jumper | Function | Remarks |
|----------|---|---|
| CAN_TERM | Closed = 125 Ohms CAN termination Open = no CAN termination | Leave off and use external termination if possible |
| RST | Reset | Close to generate a hardware module reset |
| BOOT | Activate STM32 on-chip bootloader on reset | See warning below |
| J7 | Close 1-2 to activate the safe default UART speed of 115200 bps Close 3-4 to enforce the activation of the CAnGINEBerry bootloader |  |
| J6 | Optional I/O for future use | Unused by current firmware |

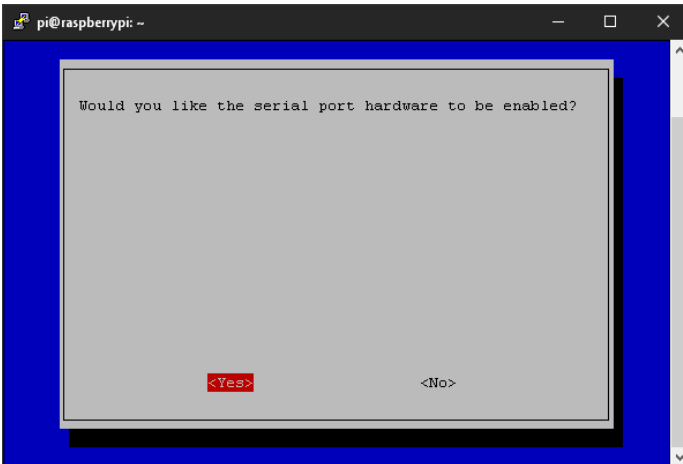
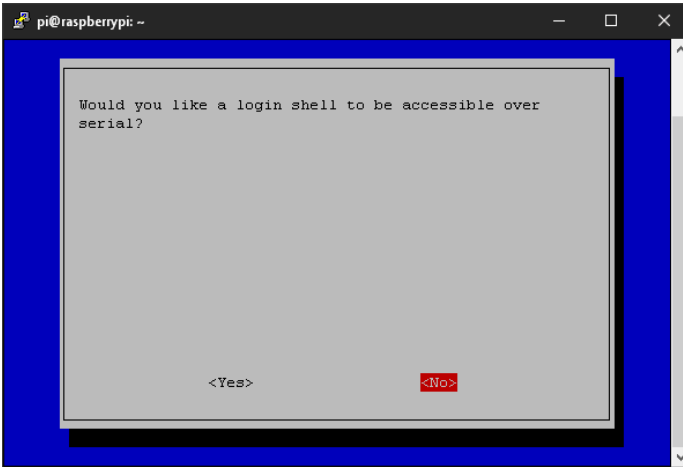
**All warranties expire if the BOOT jumper is used,
or any soldering job is performed on the CAnGINEBerry!**

4 Using CANgineBerry with a Raspberry PI® 3 and up

In the Raspberry PI® 3 and up, the UART at GPIO14 (pin 8) and GPIO15 (pin 10) is not available for user applications 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 Interface Options – Serial Port, then:



Confirm and select “Finish” to save, then reboot. Note that the serial port is accessible as (typically) /dev/ttyAMA0 or /dev/ttyS0 and Bluetooth may now be no longer available depending on the hardware version.

For performance and reliability reasons, don't try to use the mini UART in the Raspberry Pi® 3/4 with CAngeBerry!

For details on the Raspberry Pi® UART configuration options, refer to

<https://www.raspberrypi.com/documentation/computers/configuration.html#configuring-uart>

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
```

under Windows, or:

```
coiaupdater -h
```

under Linux to get a list of supported parameters. Output:

```
COIAUpdater version 1.11.XXXX  
(c) EmSA 2022, 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
- u, --usesafebaudrate
Use safe lower baudrate 115200 Baud on serial port instead of default 921600 Baud
- f, --flashfirmware <firmwarefile>
Programs the firmware
- d, --flashdata <binaryedsfile>
Programs the data, e.g. containing a binary object dictionary definition. 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. Possible values: 20, 50, 125, 250,
    500, 1000.

-l, --lss <0|1>
    Configures disable (0) or enable (1) 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

```

Selecting UART port and bitrate

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.

If the UART connection from the host system does not support the default 921600 baud, set the -u parameter to switch to the safe default of 115200 baud.

NOTE: this requires that you set a jumper on Pins 1-2 of Jumper 7!

Selecting CANopen node ID and bitrate

The -n and -r parameters can be used to select a default CANopen node ID and bitrate that the CAnGINEBerry should use. This information is stored in non-volatile memory and is available after power cycles.

When Layer Setting Services (LSS) are enabled, then the CAnGINEBerry does NOT have a node ID. Instead, it waits for the LSS request and replies to those accordingly. A CANopen Manager or other device capable of sending LSS requests must use these LSS services to assign a node ID to the CAnGINEBerry.

Note that some firmware versions may overwrite these or enforce their own values here.

Programming a firmware

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 -p 2 -f CgB_COIA_BEDS1.10_sec.bin
```

under Windows, or:

```
coiaupdater -p /dev/ttyAMA0 -f CgB_COIA_BEDS1.10_sec.bin
```

under Linux.

Output:

```
COIAUpdater version x.xx.xxxx
(c) EmSA 2022, All Rights Reserved, www.em-sa.com
Connected to COM6 at 921600 bps
Firmware file to write: CgB_COIA_BEDS1.8_sec.bin
Waiting for bootloader to start...
Bootloader is running.
Bootloader version:      1.6
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:                1.10
Erasing flash...done.
Programming.....done.
Update completed.
Disconnected from /dev/ttyAMA0
```

Loading a configuration

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

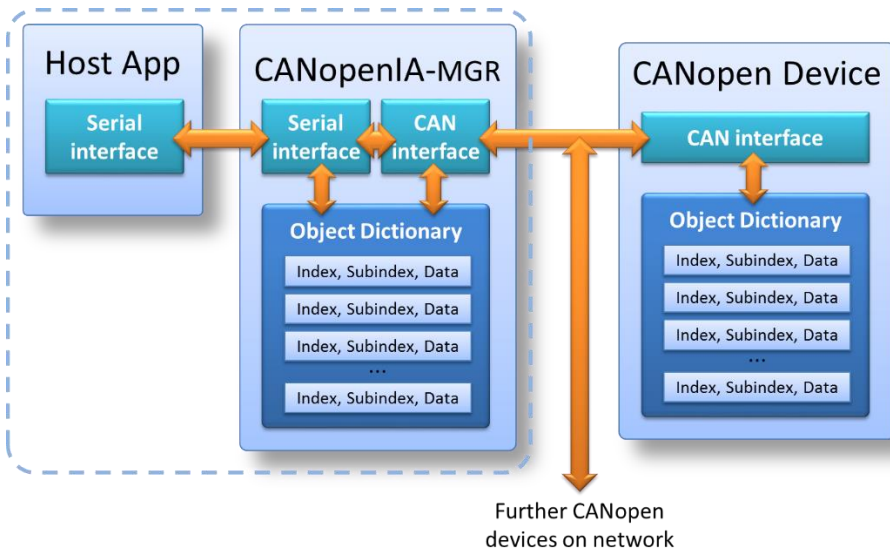
The default application loaded upon delivery is the CANopenIA-MGR minimal CANopen Manager that gives the host quick and easy access to a network of CANopen devices.

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

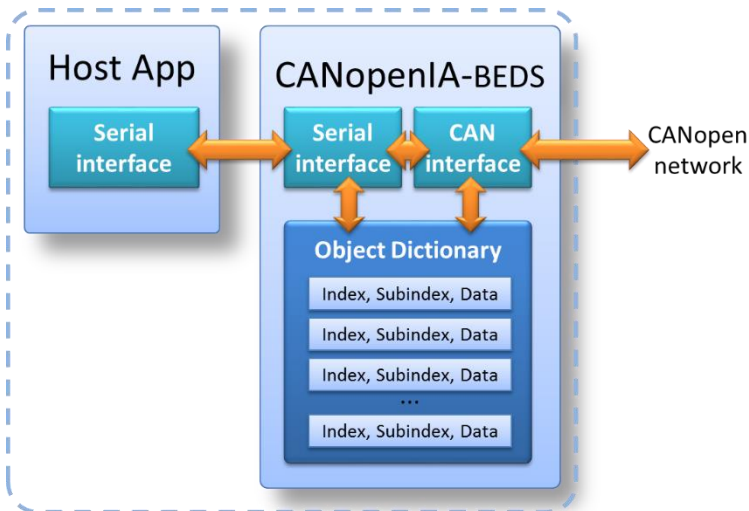
The Raspberry Pi receives simple serial indications with new data arriving (indicates from which node ID and which data object by index and subindex) and may use commands to send data (also based on node ID, index and subindex of data object) to the connected devices.

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

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-CANgineBerry document.

CAN232

The CANopenIA-BEDS application also supports selected commands from the CAN232 protocol by Lawicel.

For a detailed description, see the CANopenIA-RA-CANgineBerry 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
- Display serial number of the embedded microcontroller

7.2 Command Line Interface usage

The command line interface has the following format:

COIA [options]

under Windows, or:

coia [options]

under Linux.

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
```

under Windows, or:

```
coia -h
```

under Linux to get a list of supported parameters. Output of the initial release:

```
COIA version x.xx.xxxx
(c) EmSA 2022, All Rights Reserved, www.em-sa.com
Usage: COIA [options]
```

Options:

Basic Parameters:

```
-h, --help
    Shows this help

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

-u, --usesafebaudrate
    Use safe lower baudrate 115200 Baud on serial port instead of default
    921600 Baud

--reset
    Resets COIA device

-i, --info
    Displays information about COIA device

-m, --monitor
    Monitors changes to process data. Continues until Ctrl-C is pressed

--limited-monitor <milliseconds>
    Monitors changes to process data for a limited time

-s, --node-status
    Displays status of seen nodes

--set-configuration <nodeid|defaults>[,<bitrate_kbps>,enable|disable]
    Sets the basic COIA configuration
```

Local Object Dictionary Access Parameters:

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

-r, --read <index>,<subindex>
    Read from object dictionary of COIA device
```

Remote Access Parameters (Manager Only):

```

-n, --nmt <nodeid>|all,<command>|op|preop|stop|reset|resetcom
    Sends a network management message

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

--node-read <nodeid>,<index>,<subindex>
    Read from object dictionary of node

--net-info
    Displays information about the network

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

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

--tpdos <nodeid>
    Displays 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 for a period of time

-c, --command-file <filename>
    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 and up under Raspberry Pi OS, the first UART the CANgineBerry is connected to is typically /dev/ttyAMA0 or /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.