CANopenIA-M0

Embedded CANopen module for embedded computers

for version 2.2 or higher

Rev. 2.01 of 18th August 2021

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

Original hardware development by
Embedded Systems Solutions GmbH
www.essolutions.de

Contents

1. Module Overview

The CANopenIA-M0 is a CANopen chip solution providing instant access from process data both analog and digital to CANopen networks. The combination of a lightweight ARM Cortex microcontroller with a dedicated CANopen implementation results in a high efficiency implementation with shortest processing times. A total of seven I/O ports with each four bits are available for digital I/O. Three out of these seven ports can be configured as analog I/O to a maximum of four analog input channels with 10bit resolution, four analog input channels with 12bit resolution and four analog output channels with 12bit resolution.

One port can alternatively be configured to support the Remote Access protocol using a UART (TTL level). In this mode, additional, custom Object Dictionary entries can be handled by a host system.

Configuration of the CANopenIA-M0 is achieved by a dedicated, provided setup utility. Advanced configuration is supported using the CANopen Architect line of EDS editors. A PEAK System PCAN USB interface is required to load the configuration.

1.1 Available Inputs and Outputs

The seven ports can be configured as follows:

- Ports 0 to 3: 4bit digital in or 4bit digital out
  Port 0 can be configured as Remote Access (UART/TTL) port.
- Port 4: 4bit digital in or 4bit digital out or 4 analog input channels of 10bit
- Port 5: 4bit digital in or 4bit digital out or 4 analog output channels of 12bit
- Port 6: 4bit digital in or 4bit digital out or 4 analog input channels of 12bit

1.2 Performance

The CANopenIA-M0 features short processing times. A digital input signal gets converted to the corresponding Transmit PDO message within 15 microseconds when the bus is available and a change-of-state detection without timers is used. A Receive PDO message received is processed immediately and the corresponding digital outputs get switched within 20 microseconds.

2. Hardware

The CANopenIA-M0 module allows integration of the CANopenIA-M0 chip functions in user's hardware without taking care about clock generation, EEPROM hardware and the status and error indication LEDs. The module can easily be implemented in user's hardware with a two row 1,27 mm grid connector. As components can be placed under the module on the main PCB, the required PCB space is less than 3 sq cm or 0.5 sq inch using SMD mounted connectors.

The CANopenIA-M0 Module contains

- the CANopenIA-M0 CANopen protocol chip
- the 12 MHz oscillator,
- the serial EEPROM to store the configuration data
- two LEDs for ERR and RUN display

The optimized CANopen protocol implementation is CiA301 and CiA401 compliant and located in the flash memory of the chip. The configuration can be modified and loaded through CANopen with the provided CANopenIA setup utility.

The CANopenIA-M0 module is also contained in the CANopenIA-M0 starter kit, which allows easy and quick start with CANopen and CANopenIA-M0.

2.1 Electrical characteristics

Symbol	Parameter	Min	Typ	Max	Unit
V_{DD}	Supply voltage for internal regulator	1,8		3,6	V
V_{CC}	Supply voltage for CAN transceiver	4,5		5,5	V
V_{DD_CAN}	Supply voltage for CAN I/O level	2,8		5,5	V
I_{DD}	Supply current for internal regulator		17[1]	203	mA
I_{CC}	Supply current for CAN transceiver		5[2]	70	mA
I_{DD_CAN}	Supply current for CAN I/O level		0,05[1]	0,5	mA

[1] @ 3,3 V, no I/O connected

[2] @ 5 V, CAN connected

TABLE 1 – ELECTRICAL CHARACTERISTICS

For details of the electrical characteristics of all chip pins see the CANopenIA-M0 data sheet and the data sheet of the NXP LPC11C24 micro controller.

2.2 Physical Parameters

Parameter	Value	Unit
Temperature	-40 .. +85	°C
Size (w x l x h)	19,5 x 31,5 x 13	mm
Weight	4	g

TABLE 2 – PHYSICAL PARAMETERS

2.3 Pin configuration

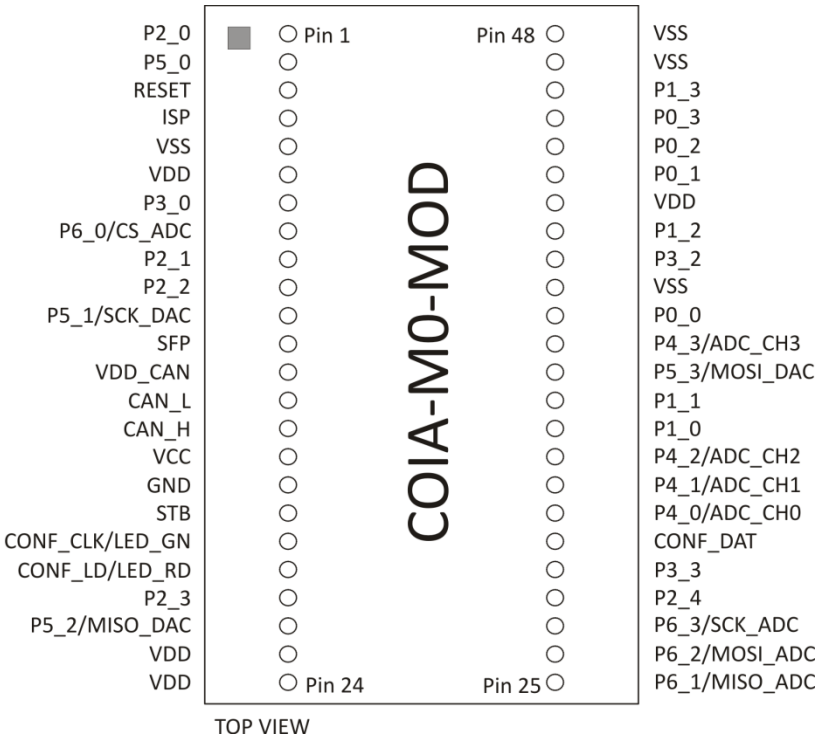


FIG. 1 – CANOPENIA MODULE PINOUT

COIA-M0	LPC11Cx4	Pin	Remarks
P0_0	PIO1_4	40	
P0_1	PIO1_5	45	
P0_2/RA_RX	PIO1_6	46	When used: remote access rx, TTL level
P0_3/RA_TX	PIO1_7	47	When used: remote access tx, TTL level
P1_0	PIO3_0	36	
P1_1	PIO3_1	37	
P1_2	PIO3_2	43	
P1_3	PIO3_3	48	
P2_0	PIO2_6	1	
P2_1	PIO2_7	11	
P2_2	PIO2_8	12	
P2_3	PIO2_10	25	
P3_0	PIO1_8	9	
P3_1	PIO1_10	30	
P3_2	PIO1_11	42	
P3_3	PIO0_10/SWCLK	29	
P4_0/ADC_CH0	PIO1_0	33	
P4_1/ADC_CH1	PIO1_1	34	
P4_2/ADC_CH2	PIO1_2	35	
P4_3/ADC_CH3	PIO1_3/SWDIO	39	
P5_0/CS_DAC	PIO2_0/SSEL1	2	
P5_1/SCK_DAC	PIO2_1/SCK1	13	
P5_2/MISO_DAC	PIO2_2/MISO1	26	
P5_3/MOSI_DAC	PIO2_3/MOSI1	38	
P6_0/CS_ADC	PIO0_2/SSEL0	10	
P6_1/MISO_ADC	PIO0_8/MISO0	27	
P6_2/MOSI_ADC	PIO0_9/MOSI0	28	
P6_3/SCK_ADC	PIO2_11/SCK0	31	
CONF_CLK/LED_GN	PIO0_6	23	
CONF_LD/LED_RD	PIO0_7	24	
CONF_DAT	PIO0_11	32	
SCL	SCL	15	
SDA	SDA	16	
ISP	PIO0_1	4	External pull-up resistor to VDD required
SFP	PIO0_3	14	Special function pin
CAN_L	CAN_L	18	
CAN_H	CAN_H	19	
STB	STB	22	Silent mode control input for CAN transceiver
VDD_CAN	VDD_CAN	17	Supply voltage for I/O level of CAN transceiver
VCC	VCC	20	Supply voltage for CAN transceiver
GND	GND	21	Ground for CAN transceiver
RESET	RESET/PIO0_0	3	
VDD	VDD	8	Supply voltage to the internal regulator, and internal ADC reference voltage
VDD	VDD	44	
XTALIN	XTALIN	6	
XTALOUT	XTALOUT	7	
VSS	VSS	5	Ground
VSS	VSS	41	

TABLE 3 - CANOPENIA-M0 VS. LPC11C24 PIN COMPARISON

2.4 Mechanical data

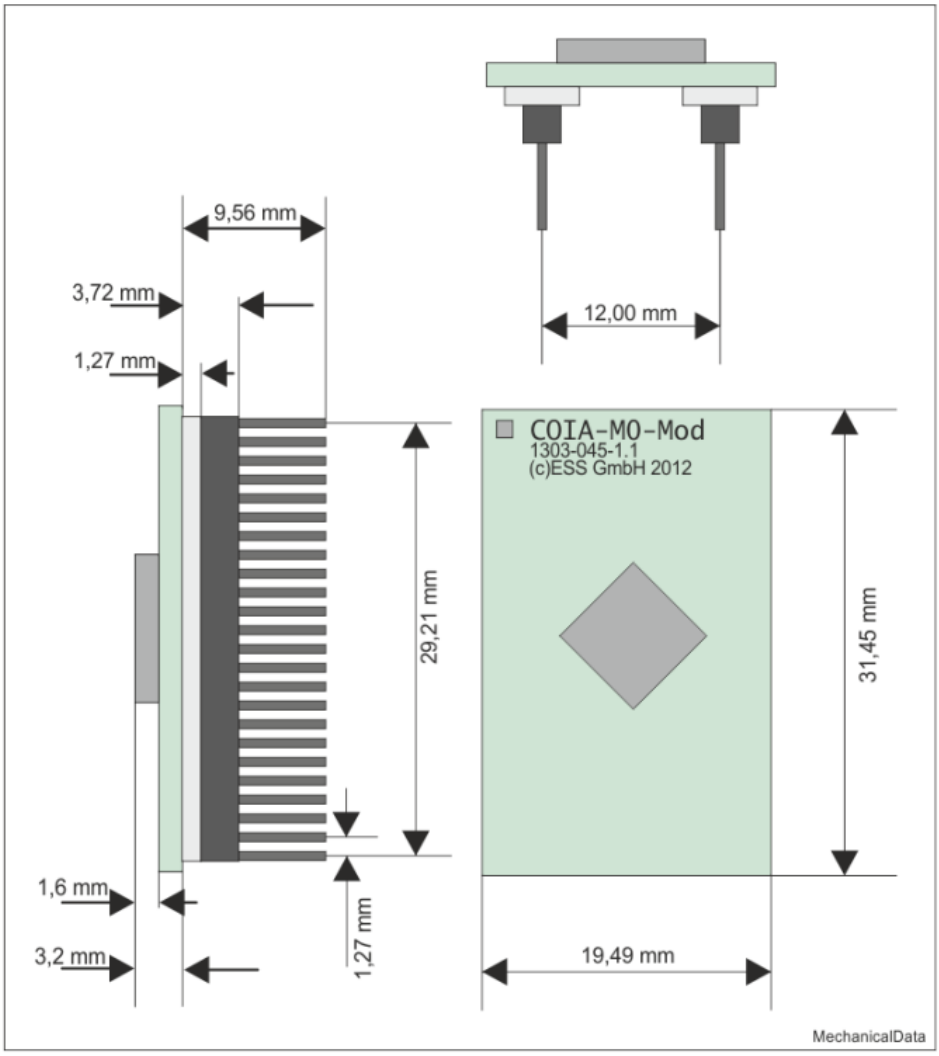


FIG. 2 – MECHANICAL DRAWING

For the mechanical characteristics of the pin headers refer to the data sheet FTR-124-01-G-S from Samtec. Preferred montage on the socket connectors RSM-124-02-L-S.

2.5 Optional Components

Shift Register and status LEDs

An external 10-bit shift register can be used to override CAN bit rate and node ID configuration stored in the CANopenIA configuration from the EEPROM. The value of the shift register is read on each chip start-up, after that signals CONF_CLK/LED_GN and CONF_LD/LED_RD are used as status LEDs. If no shift register is detected or if the node id value is read to 0 (illegal value) the CAN bit rate and node ID configuration is taken from the CANopenIA configuration stored in EEPROM. A combination of node ID of 0x7F and bit rate setting of 1Mbps (all shift register bits high) is reserved and invokes a recovery mode.

bit	15 .. 10	9 .. 3	2..0
value	0	Node id	Baud

TABLE 4 - SHIFT REGISTER BIT DESCRIPTION

Baud Value	Bit Rate Used
0 (all three switches ON)	10 kbps
1	20 kbps
2	50 kbps
3	125 kbps
4	250 kbps
5	500 kbps
6	800 kbps
7 (all three switches OFF)	1 Mbps

TABLE 5 – BAUD RATE SELECTION

Shift register signals:

- CONF_CLK/LED_GN is used as clock output with max. frequency in range 800 – 900 kHz and is LOW-to-HIGH edge-triggered
- CONF_LD/LED_RD as asynchronous parallel load output (active LOW)
- CONF_DAT Serial data input

Recommended connection of two 8-bit shift registers 74HC165C is in Fig. .

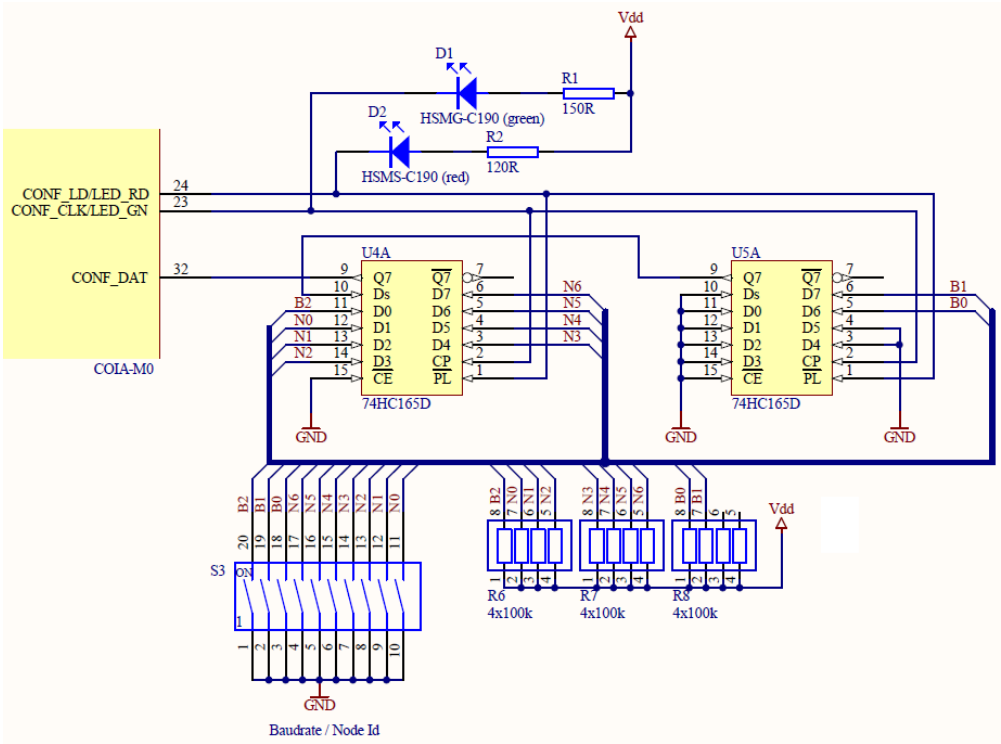


FIG. 3 - RECOMMENDED CONNECTION OF SHIFT REGISTER AND STATUS LEDS

If no shift register is connected, then the pins CONF_CLK/LED_GN, CONF_LD/LED_RD and CONF_DAT should be left floating.

External ADC

Optional external analog-to-digital converters can be connected to CANopenIA-M0 using the SPI interface on port 6. CANopenIA-M0 takes master role within SPI communication. Supported ADCs are:

- Analog devices AD7923
- Texas instruments ADS7841
- Microchip MCP3204

Used signals (from side of COIA-M0):

- P6_0/CS_ADC - SPI chip select

- P6_1/MISO_ADC - SPI data input
- P6_2/MOSI_ADC - SPI data output
- P6_3/SCK_ADC - SPI serial clock

Please refer to selected converter's manufacturer documentation for proper electrical interfacing.

Which specific ADC part is used is stored in the CANopenIA-M0 configuration in the EEPROM. The CANopenIA setup software allows selecting one of the three devices.

External DAC

Optional external digital-to-analog converters can be connected to CANopenIA-M0 using the SPI interface on port 5. CANopenIA-M0 takes master role within SPI communication. Supported DACs:

- Analog devices AD5624B
- Texas instruments DAC124S085
- Maxim MAX5500

Used signals (from side of COIA-M0):

- P5_0/CS_DAC - SPI chip select
- P5_1/SCK_DAC - SPI serial clock
- P5_2/MISO_DAC - SPI data input, not used (and can be left floating) for parts AD5624B and DAC124S085
- P5_3/MOSI_DAC - SPI data output,

Please refer to the selected converter's manufacturer documentation for proper electrical interfacing.

Which specific ADC part is used is stored in the CANopenIA-M0 configuration in the EEPROM. The CANopenIA setup software allows selecting one of the three devices.

3. CANopen Features Supported

The CANopenIA-M0 implementation is based on the CANopen Standards CiA301 V4.2 “CANopen Application layer and communication profile” and the LSS Fastscan function of CiA305 V2.2.14 “Layer setting services (LSS) and protocols” and CiA401 V3.0 “Device profile for generic I/O modules”. For details on the features below please refer to the original documents or a CANopen book like “Embedded Networking with CAN and CANopen”.

The CANopen functionality provided includes:

- Node ID assignment by LSS (or pre-configured, or hardware configured)
- Heartbeat production instead of the obsolete node guarding
- Emergency production with error history
- Configurable identification entries for manufacturer name and hardware and software version strings as well as CANopen Vendor ID
- Hard coded Object ID Revision and Serial number based on the chip’s serial number, al-so used for LSS services
- Configurable SYNC consumer with support of SYNC counter
- Up to four Receive PDOs, also supporting SYNC.
- Up to four Transmit PDOs supporting change-of-state (COS) detection, inhibit time, event time and SYNC.
- Configurable polarity for all digital ports
- Configurable default error values for all outputs
- Many configurable parameters can be set through the CANopen network (for example by a CANopen manager)
- Configuration through Binary EDS file (created by CANopen Architect EDS Software) or CANopenIA-M0 setup tool

3.1 Setup of bitrate and node ID

In a CANopen system, setting up the bitrate and node ID are essential. In order to support a variety of applications, CANopenIA-M0 supports several methods:

1.) HW configuration (switches or dials)

Bit rate and node ID are read from pins via a shift register

2.) Setup with CANopen-IA setup utility

Bit rate and node ID are read from a setup stored in non volatile memory. The setup file can be written to the chip via CANopen.

3.) CANopen LSS - Layer Setting Services, Fastscan

The node ID is assigned using CANopen LSS Fastscan service. A CANopen LSS manager supporting Fastscan can identify or detect the device and assign it a node ID.

3.2 CANopenIA-M0 bitrate and node ID selection

The following diagram shows how the bitrate, node ID and object dictionary are selected on re-set.

Note that when a DIP switch is set to ON the corresponding shift register bit value is zero, there-fore all DIP switches set to ON is the same as leaving the CONF_DAT pin floating or connected to ground, as an internal weak pull-down is enabled on the CONF_DAT pin. Setting all DIP switches to OFF is the same as connecting CONF_DAT to Vdd.

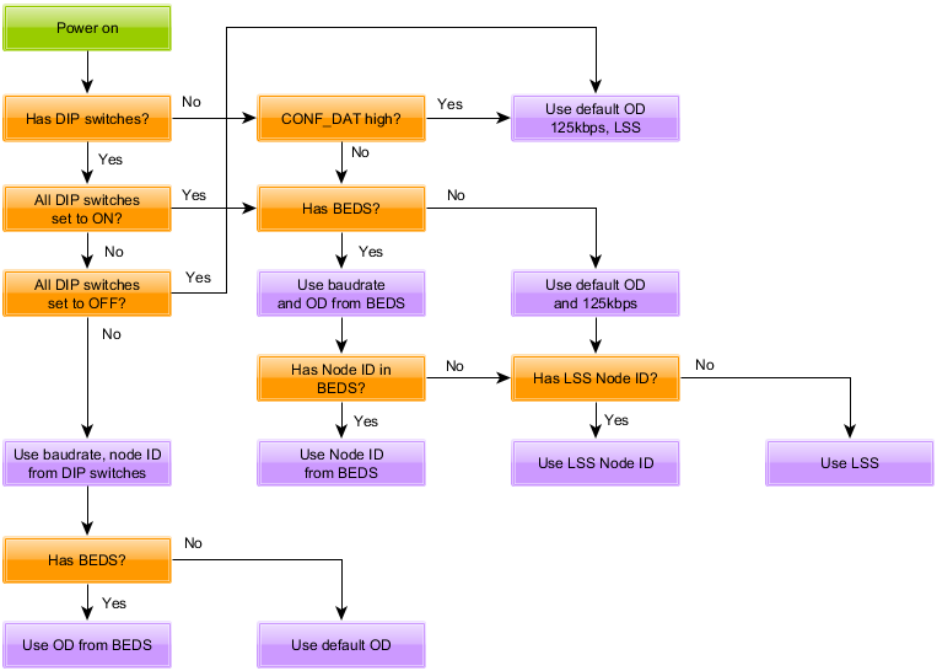


FIG. 4 – BOOTUP SEQUENCE FOR ASSIGNING OD, NODE ID AND BAUD RATE

In the diagram BEDS refers to the downloaded binary EDS configuration or CANopenIA setup file.

3.3 Setup of ports, PDOs and startup configuration

The CANopenIA-M0 Setup Utility is a PC program that allows setting up the CANopenIA-M0 settings and features to be applied after power up. The setup is transferred to the CANopenIA-M0 via CANopen communication and stored locally in non-volatile memory. The file format used is that of a binary EDS, as also supported by CANopen EDS Editors like CANopen Architect.

Assignment of Digital Ports to Object Dictionary entries

The data from the digital I/O ports is automatically assigned to the appropriate Index of the CANopen Object Dictionary. The Subindex and nibble (lower or upper 4bits) information is con-figurable.

Object Dictionary at Index 6000h, digital inputs

The four bits from each port can be configured to be located at any Subindex from range 1 to 8, low or high nibble. See Object Dictionary entries at Index 5F01h for details.

Object Dictionary at Index 6200h, digital outputs

The four bits from each port can be configured to be located at any Subindex from range 1 to 8, low or high nibble. See Object Dictionary entries at Index 5F01h for details.

Auto-Assignment of Analog Ports to Object Dictionary entries

The analog data from the I/O ports is automatically assigned to the CANopen Object Dictionaries.

Object Dictionary at Index 6401h, analog inputs

The four channels of the first port configured to analog input are assigned to OD entry [6401h,1-4].

The four channels of the second port configured to analog input are assigned to [6401h,5-8].

The analog input channels provide 10bit or 12bit of data which is copied into UNSIGNED16 data types. The default configuration is such, that on the CANopen side, all analog values use the full available data range from 0 to FFFFh. Other shifting options are configurable.

Object Dictionary at Index 6411h, analog outputs

The four channels of the first port configured to analog output are assigned to OD entry [6411h,1-4].

The analog output channels can handle 12bit of data. Data received via the CANopen network is of data type UNSIGNED16. The default configuration is such, that on the CAN-open side, all ana-log values use the full available data range from 0 to FFFFh. Other shifting options are configurable.

PDO mapping: CiA401 default

The CANopenIA-M0 uses the default CiA401 setting for the PDO mapping, which is generated by the CANopenIA setup utility. These defaults only have a minimum difference to the official CiA401 defaults: unused data is not mapped and unused PDOs are disabled. Please see [CiA401] for information about the default PDO mapping configuration.

More flexible PDO configurations are supported via binary EDS files generated by CANopen Architect EDS.

Start up and error Behavior

What exactly happens to the I/O ports of CANopenIA-M0 before start up or in an error condition is configurable.

CAN error passive

A CANopen emergency message may be transmitted, when the CAN controller detects that it goes into error passive mode.

CAN bus off reaction

When the CAN controller detects a bus off condition (CAN controller shut down), the CANo-penIA-M0 can either stop or after a time delay re-start with a reset.

Default outputs

For all output ports a default value may be specified. This value is applied within XXX ms after power up or reset of the CANopenIA-M0.

Error outputs

For all output ports an error mode and error value may be specified. If error output mode is ena-bled for a port, the error value will be applied to the output in case of an error or the NMT state changing to STOP.

3.4 Indicators: CANopen LEDs

CANopenIA-M0 supports both a red error LED and a green run LED as specified in the document [CiA303-3]. The pins used for these are shared with the shift register. Each LED

can be on, off, flickering [10Hz], blinking [2.5Hz], single flash [one blink with 1s break] or double flash [two blinks with 1s break].

The red error LED is switched on after reset and goes off after successful initialization, including transmission of the bootup message.

If the red error LED flickers in combination with the run LED, then the device is in LSS mode, it waits for an LSS Master to detect it and to assign it a node ID number.

If the red error LED goes on during operation, it means that a serious error occurred, typically the CAN controller went into bus off mode.

The green run LED blinks, when the device is in pre-operational state. It is on, when it is in operational state. It shows a single flash, when the device is in stopped state.

3.5 Revision Number and Serial Number

The default configuration uses a revision number and serial number that should be different on each CANopenIA-M0 device. Uniqueness is not guaranteed. These numbers are generated from the 128bit internal chip ID. The full 128bit ID is available at object [5F00,05].

If the user programs in a custom configuration the following rules apply.

When using advanced configuration with EDS editors: If the revision number and serial number ([1018,03] and [1018,04]) are set to access type “Constant” then the built-in values are used and the values in the configuration are not used.

If the revision number and serial number are both set to the values 0xFFFFFFFF in the configuration then the built-in values are used and the values in the configuration are not used.

If the revision number or serial number are set to any other value than 0xFFFFFFFF in the configuration then the values in the configuration are used.

4. CANopen Object Dictionary

The table bellows shows all object dictionary entries that the CANopenIA-M0 implements in pure I/O mode without Remote Access support. If a specific entry is provided or not may depend on the configuration. All Indexes and Subindexes are given in hexadecimal.

When Remote Access is enabled, additional Object Dictionary entries may be defined using the CANopen Architect software.

Index	Sub-index	Description
1000	00	Device Type
1001	00	Error Register
1003	00-04	Error Field
1005	00	COB ID SYNC
1008	00	Device Name
1009	00	Hardware Version
100A	00	Software Version
1014	00	COB ID EMCY
1015	00	Inhibit Time Emergency
1017	00	Heartbeat Producer
1018	00-04	Identifiction Object
1019	00	Synchronus Counter Overflow
1400	00-02	Receive PDO 1 Communication Parameter
1401	00-02	Receive PDO 2 Communication Parameter
1402	00-02	Receive PDO 3 Communication Parameter
1403	00-02	Receive PDO 4 Communication Parameter
1600	00-08	Receive PDO 1 Mapping Parameter
1601	00-08	Receive PDO 2 Mapping Parameter
1602	00-08	Receive PDO 3 Mapping Parameter

1603	00-08	Receive PDO 4 Mapping Parameter
1800	00-06	Transmit PDO 1 Communication Parameter
1801	00-06	Transmit PDO 2 Communication Parameter
1802	00-06	Transmit PDO 3 Communication Parameter
1803	00-06	Transmit PDO 4 Communication Parameter
1A00	00-08	Transmit PDO 1 Mapping Parameter
1A01	00-08	Transmit PDO 2 Mapping Parameter
1A02	00-08	Transmit PDO 3 Mapping Parameter
1A03	00-08	Transmit PDO 4 Mapping Parameter
1F80	00	NMT Startup (autostart bit only)
5F00	00-06	Extended status, used by remote access
5F01	00-07	CANopenIA Port Configuration
5F02	00-04	CANopenIA Hardware Settings
5F03	00-02	Remote access configuration
5FFF	00-02	CANopenIA Setup, load new configuration
6000	00-08	Read Digital Input
6002	00-08	Polarity Digital Input
6200	00-08	Write Digital Output
6202	00-08	Polarity Digital Output
6206	00-08	Error Mode Output
6207	00-08	Error Value Output
6401	00-08	Read Analog Input
6411	00-04	Write Analog Output
6443	00-04	Analog Output Error Mode
6444	00-04	Analog Output Error Value

TABLE 6 – OBJECT DICTIONARY ENTRIES

4.1 Index 1000h to 1F80h

These Object Dictionary entries are standard CiA301 entries and are used as specified in [CiA301] and [CiA401]. Where configurable, the configuration can be made with the CANopenIA setup utility, a binary EDS file download or via the CANopen network using a CANopen configuration utility.

The PDO mapping is automatically made depending on port configuration, the default [CiA401] PDO mapping is used, unused bytes are disabled / not mapped.

An individual mapping can be configured by loading an appropriate binary EDS configuration file into the CANopenIA-M0.

4.2 Index 5F01h to 5FFFh

These Object Dictionary entries are used for the CANopenIA-M0 setup. They are read-only and can only be changed using the CANopenIA-M0 setup utility or by loading a binary EDS into the CANopenIA-M0

Extended status 5F00h

These entries are all read-only and are primarily used by remote access (serial port) to provide system information to the host system.

Subindex 1, UNSIGNED8, own node id

Subindex 2, UNSIGNED8, own nmt state

Subindex 3, UNSIGNED8, own config state (0 OK, else config error)

Subindex 4, UNSIGNED32, own type/version (internal)

Subindex 5, UNSIGNED128, chip serial number

Subindex 6, UNSIGNED8, chip type (1 for NXP LPC11C24)

Port configuration 5F01h

This Object Dictionary entry located at Index 5F01h has seven Subindexes, one for each of the seven ports of the CANopenIA-M0. Each entry is of type UNSIGNED8. These entries must be present and must have valid values for the CANopenIA-M0 to be usable.

Bit 0-3: Port type of the 4-bit port

1: Digital input

2: Digital output

3: Analog input (internal ADC)

4: Analog output (external DAC)

5: Analog input (external ADC)

6: Remote Access port

Bit 4-6: Digital port Subindex

For digital ports (Index 6000h or 6200h) this specifies the Subindex value. A value from 0 to 7 is used to indicate Subindex 1 to 8.

Bit 7: Digital port nibble info

For digital ports this bit specifies if the 4 bits of the port are located in the lower (0) or the upper (4) nibble of the UNSIGNED8 object dict. entry.

Hardware settings 5F02h

The hardware settings allow the selection of external analog components, their usage and set-ting extended error behavior. Subindexes 1 to 5 are available.

Subindex 1, UNSIGNED8, External ADC type selector

The following values are supported:

0: Analog devices AD7923

1: Texas instruments ADS7841

2: Microchip MCP3204

Subindex 2, UNSIGNED8, External DAC type selector

The following values are supported:

0: Analog devices AD5624B

1: Texas instruments DAC124S085

2: Maxim MAX5500

Subindex 3, UNSIGNED8, Error condition

This entry is currently reserved, set to zero.

Subindex 4, UNSIGNED16, Error reset delay

Upon detection of a fatal failure the CANopenIA-M0 resets itself. The default delay for the reset is 100ms. Using this entry the error reset delay can be configured in the range from 100ms to 30,000ms.

Subindex 5, UNSIGNED8, Analog shifting

On the CANopen side the data representation for all analog data is UNSIGNED16 (0 to FFFFh) and the entire data range may be used. This configuration option allows the selction of various shifting methods:

0:	(default) full 16bit range usable, data is shifted such, that the least significant bits are ignored
1:	raw mode, data is passed on 1:1, the most significant bits are ignored
2:	backward compatible CANopenIA-X0 mode

Subindex 6, UNSIGNED16, Analog Transmit Delta

Setting this to a non-zero value modifies the change-of-state transmission behavior for analog input TPDOs. A raw analog input value must change by at least the delta value to trigger trans-mission of a TPDO. If multiple analog channels are mapped to a TPDO only a single channel needs to be at least the delta to trigger transmission of the TPDO.

Subindex 7, UNSIGNED16, SDO Response Time

Setting this to a non-zero values causes SDO responses to be delayed by the specified number of milliseconds. Set to 17 for compatibility with the previous generation CANopenIA device.

Subindex 8, UNSIGNED16, Compatibility Settings

Each bit controls an aspect of behavior in order to improve compatibility with the previous gen-eration CANopenIA device.

Bit 0:	When set an extra heartbeat is transmitted 50ms after bootup.
Bits 1-15:	Reserved. Set to zero

Remote access configuration 5F03h

These entries define additional parameters for the configuration of the remote access via UART.

Subindex 1, UNSIGNED8, UART bit rate (1 for 115200 bps)

Subindex 2, UNSIGNED8, extended remote access functionality for messages received

- 0: No additional message reporting functionality
- 1: Support of CAN232 sytle message reporting
- 2: Full format report using Object 5F0Bh
- 3: Condensed 11bit format report using object 5F0Ch

Configuration download 5FFFh

This index allows managing the configuration of the CANopenIA-M0 using a binary EDS setup file as supported by the EDS Editor CANopen Architect Standard or the CANopenIA-M0 setup utility. The device MUST be placed into the NMT state pre-operational before accessing these entries. The maximum file size currently supported is 4064 bytes.

Subindex 1, STRING4 (4 visible chars), Activate binary EDS access

This entry must be written with “INIT” to activate access to the binary EDS download. If the next write to this entry is “BACK”, then the EEPROM containing the binary EDS file is erased. On the next power or reset cycle the CANopenIA-M0 will start with its on chip default configuration.

Subindex 2, DOMAIN, Binary EDS

A binary EDS file with a valid CANopenIA-M0 configuration may be written to this entry. At the end of the write transfer, the CANopenIA-M0 generates emergency messages to indicate the progress of storing the file in EEPROM:

EMCY code FF00h, “WAIT ” in manufacturer specific bytes

Transfer of setup file into RAM completed, file contains a valid configuration, programming the EEPROM started.

EMCY code FF01h, “FAIL1” in manufacturer specific bytes

Checksum failure (CRC does not match), EEPROM is now erased.

EMCY code FF02h, “FAIL2” in manufacturer specific bytes

Illegal setup, file does not contain a usable setup, EEPROM is now erased.

EMCY code FF02h, “FAIL3” in manufacturer specific bytes

EEPROM programming failure, EEPROM is now erased.

EMCY code 0000h, “OVER ” in manufacturer specific bytes

Programming the EEPROM completed successfully, new setup will be used after next reset.

4.3 Index 6000h to 6444h

These Object Dictionary entries are used to implement the CiA401 device profile for generic I/O devices. Refer to [CiA401] for details.

Which subindexes are made available and which port they are connected to depend on the configuration.

5. CANopenIA Setup Utility

The provided software utility “CANopenIA Setup” can be used to set configurable parameters of the CANopenIA-M0. Once a configuration is made, it can be transferred to a CANopenIA-M0, where it gets stored in an external EEPROM.

Any www.peak-system.com PCAN PC interface is required for the transfer.

5.1 Node Configuration

Device Information

After program start, the CANopenIA Setup utility shows the device information stored in the device’s object dictionary. All fields can be edited to change content.

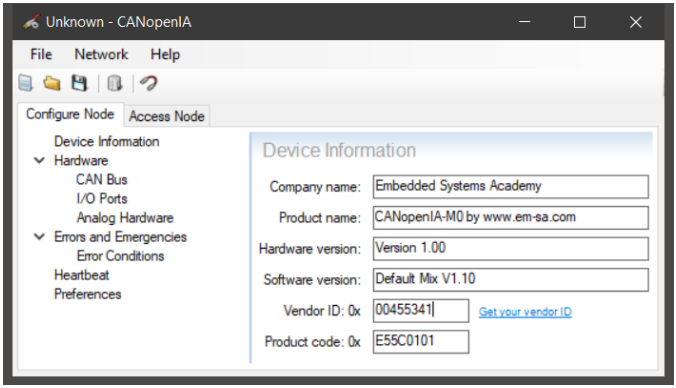


FIG. 5 – DEVICE INFORMATION PARAMETERS

Hardware – CAN Bus

In this section, you can select the CAN bitrate and the CANopen node ID used by the device. Selecting “Not set” for the node ID means that it is either set by hardware (shift register) or LSS (Layer Setting Services).

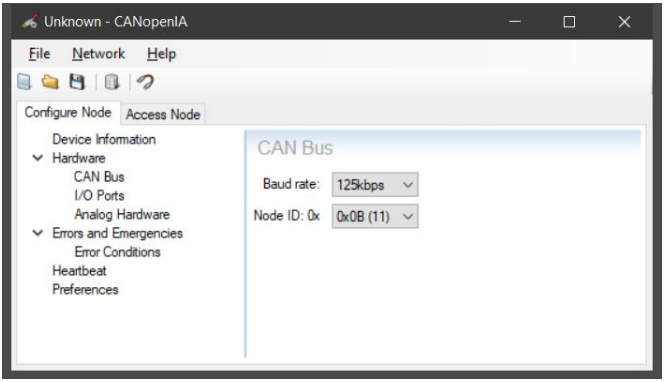


FIG. 6 – HARDWARE – CAN BUS

Hardware – I/O Ports

In this section, you can select how each of the seven ports is used. All ports can be configured to be digital input or output. Analog input ort output or remote access (UART/TTL) are only available on dedicated ports.

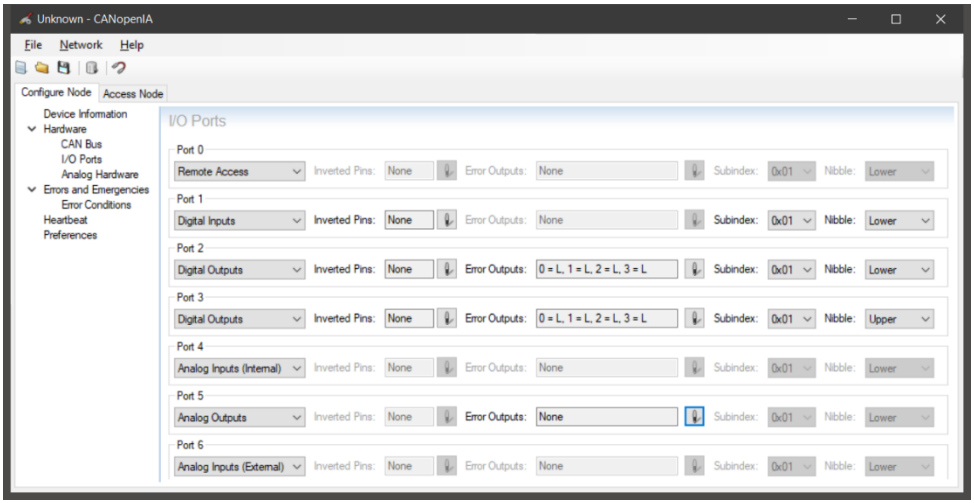


FIG. 7 – HARDWARE – I/O PORTS

Error outputs can only be selected for ports used as output. The error output values specified get activated when the CANopenIA-M0 device detects an error situation.

The Subindex and Nibble value specify where the port's data is placed into the Object Dictionary. The Index is automatically assigned in accordance with the device profile for

generic I/O [CiA 401]. The subindex can be manually selected. As digital ports only have 4 bits, the Nibble selection can be used to place the data into the lower or upper 4 bits of an 8bit Object Dictionary entry.

Hardware – Analog Hardware

In this section, you can select the external analog components used on port 5 and 6. If port 5 or 6 are activated to use analöog data, external ADC and DAC converters need to be connected. The supported external devices are listed in the pull-down menus.

For analog inputs, a delta can be specified, that is added to the values.

The Shifting parameter can be set to 16bit, if all analog values should be expanded to 16bit. When selected, a 10bit analog value is shifted to bits 6-15 of the 16bit CANopen value.

The setting raw does not implement any shifting, analog values are passed on without shgifting, a 10bit analog value uses bits 0-9 of the 16bit CANopen value.

Remaining settings

The remaining settings are self-explanatory. Some are to provide backwards compatibility with previous implementations by providing artificial delays.

Further parameters include setting the device's heartbeat producer time and to enable/disable autostart. When checked, the device goes into operational mode after power-up or reset without waiting for an NMT master message.

5.2 Access Node

The functions provided here can only be used, if any www.peak-system.com PCAN PC interface is connected and the appropriate drivers are installed. The CAN bitrate used is that defined in the previous section (Hardware – CAN Bus).

If a CANopenIA-M0 device does not have a node ID assigned and is in LSS mode (flickering of both LEDs), select a node ID to be used and press the "Assign Node ID" button. This starts the LSS cycle and assigns the node ID to the device.

Use the “Download Configuration” button to transfer the current configuration to the device.

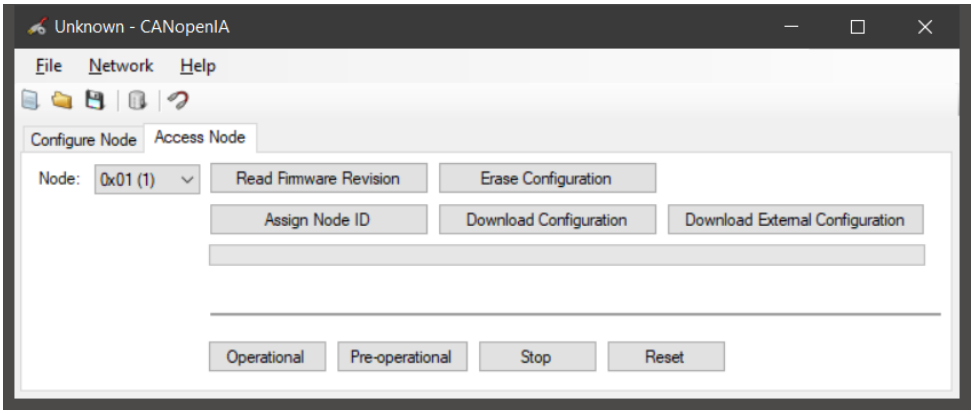


FIG. 8 – ACCESS NODE

The four buttons on the bottom produce the corresponsing NMT master messages. If a device has autostart enabled (goes operational after power-up), it might need to be set back to Pre-operational before a configuration can be loaded.

6. Advanced Configuration with CANopen Architect

The CANopen Architect Standard software by Embedded Systems Academy is a CANopen EDS (Electronic Data Sheet) Editor that allows exporting the binary EDS file format as supported by CANopenIA-M0. The steps to create and load a configuration are:

1. Use CANopen Architect to import an existing CANopenIA-M0 EDS file (as generated by setup utility, or example provided)
2. Edit configuration as required
3. From CANopen Architect export binary EDS (BEDS) hex file
4. Use the CANopenIA Setup Utility to load the BEDS to a CANopenIA

These steps are explained in more detail below.

6.1 Edit Configuration with CANopen Architect

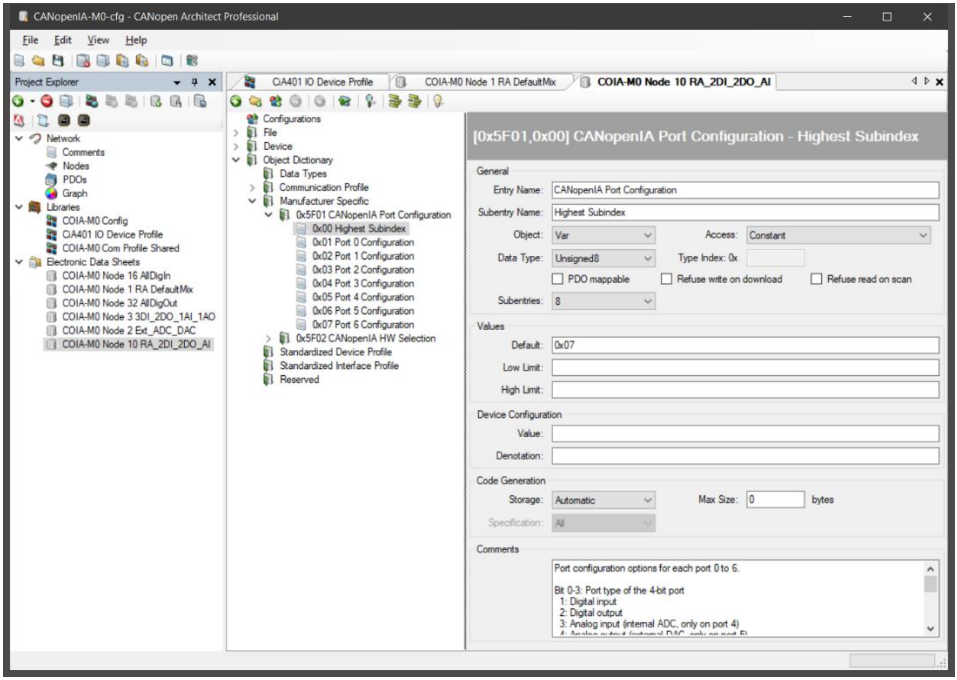


FIG. 9 – SETUP WITH CANOPEN ARCHITECT

The EDS editor CANopen Architect can be used to load and modify a configuration generated by the setup utility (.eds) or a provided example configuration (.cax). There is a free version CANopen Architect Light available from www.canopenarchitect.com.

Especially PDO parameters (14xxh, 16xxh, 18xxh and 1Axxh) and hardware configuration parameters (5F01h and 5F02h) can now be set individually.

When using the Remote Access mode, create any further OD entries as required by the host system.

6.2 Export Configuration as binary EDS

After modifications are completed and have been saved, it can be exported by clicking on the right mouse button over a project name in the Project Explorer window. Select the option “Export Binary EDS” to export a file supported by CANopenIA-M0. When requested to select a target, select CANopenIA-M0.

Load the configuration with CANopenIA Setup Utility

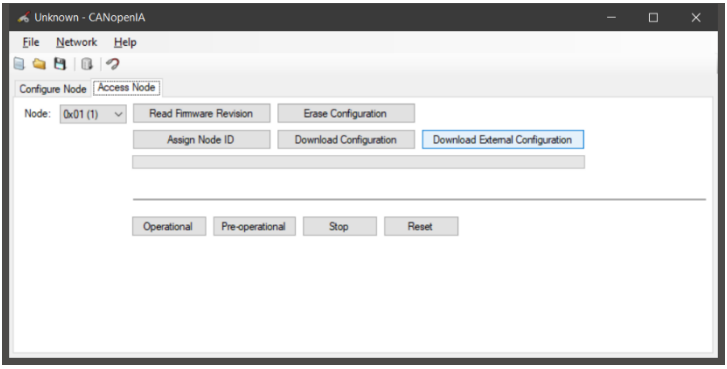


FIG. 10 – LOAD CONFIGURATION WITH THE CANOPENIA SETUP UTILITY

The CANopenIA setup utility offers an “Access Node” section. In there, first select the current node ID of the target CANopenIA-M0 device, then click the button “Download External Configuration”. Now select the binary EDS file to transfer to the module.

7. Remote Access to CANopen via UART

Using a serial channel, a host system can communicate with the CANopenIA-M0 coprocessor via a regular serial channel. The protocol used is ESAcademy's CANopen remote access protocol described in this chapter. communication.

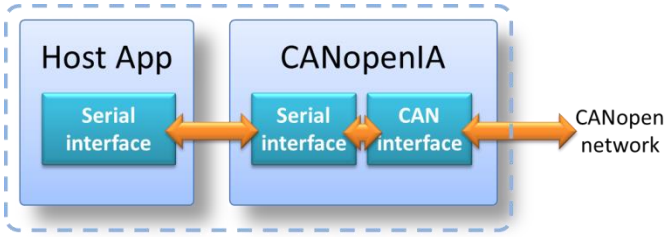


FIG. 11 – COPROCESSOR ITH HARDWIRED SERIAL INTERFACE

The CANopen Object Dictionary (OD) of the CANopenIA-M0 is available to the CANopen network as well as to the host system via the serial channel. Which OD entries are present in the CANopenIA-M0 depends on the configuration. Customized configuration files can be generated using the CANopen Architect EDS Editor and transferred into the flash memory of the CANopenIA-M0 using the CANopenIA setup utility.

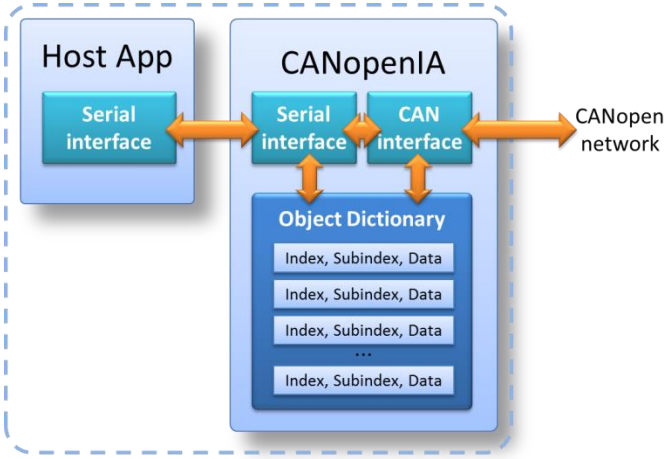


FIG. 12 – OBJECT DICTIONARY IS ACCESS FROM SERIAL AND CANOPEN SIDE

7.1 Definitions

Byte or UNSIGNED8:
8-bit, unsigned value

UNSIGNED16:
16-bit, unsigned value

UNSIGNED32:
32-bit, unsigned value

Host:
The processor or application controlling the CANopen CANopenIA via the interface specified in this document

Command:
Message from host to CANopenIA with a request to execute a command.

Response:
Message from CANopenIA to host in response to a command. Every command triggers a response. Some responses may take longer as CANopen communication might be involved. As a result one or multiple Indications might occur before receiving a response.

Indication:
Message from CANopenIA to host indicating the host that an event occurred.

Max data size:
In this version, the maximum user data size is 28 bytes. Including overhead, this results in a maximum serial packet size of 35 bytes.

Message Definition
Any message exchanged between Host and the CANopen node use the following structure (all Bytes):

<start character><length><command/response/indication><checksum>

Multi-Byte values are transmitted in little-endian format.

<start character> (Byte) default: 11h

1. Bits 0 to 3 indicate the network number, the value of zero is reserved, the default is one.

2. Bit 4 indicates if a checksum is used or not. If set, checksum is used, the default is one, using a checksum.

3. Bit 5 indicates if the length value has 8 or 16 bit. If set, 16 bits are used, the default is zero, using 8 bits for the length value.

4. Bits 6 to 7 are reserved.

<length> (Byte or UNSIGNED16, see Bit 5 of start character)

The total length of the command/response/indication in bytes.

<command/response/indication>

The data transferred in this packet can be a command, a response or an indication. For details see specifications below.

<checksum> (UNSIGNED16 or not used, see Bit 4 of start character)

A 16-bit CRC calculated with the Polynomial x16 + x15 + x2 + 1. The checksum calculation does not involve the start character.

7.2 Error Codes

Most of the responses contain an error code field. A value of zero means "no error". The bits in the error code field have the following meanings:

Bit	Meaning
0	Object Dictionary entry not found
1	Invalid command length
2	Invalid command
3	Reserved
4	No resources client)
5	Transmit buffer is full
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Unknown/miscellaneous error
11	Not supported
12	Non-volatile memory write failure
13	Reserved

TABLE 7 – REMOTE ACCESS ERROR CODES

7.3 Access Object Dictionary

The commands, responses and indications of this section are used to access the local object dictionary of the CANopenIA Coprocessor.

The syntax shown is split into the serial version “Serial” (on lowest level, treated as an array of bytes) and the “C” style function interface, if used in the library version.

Indication "D": New process data written to local Object Dictionary

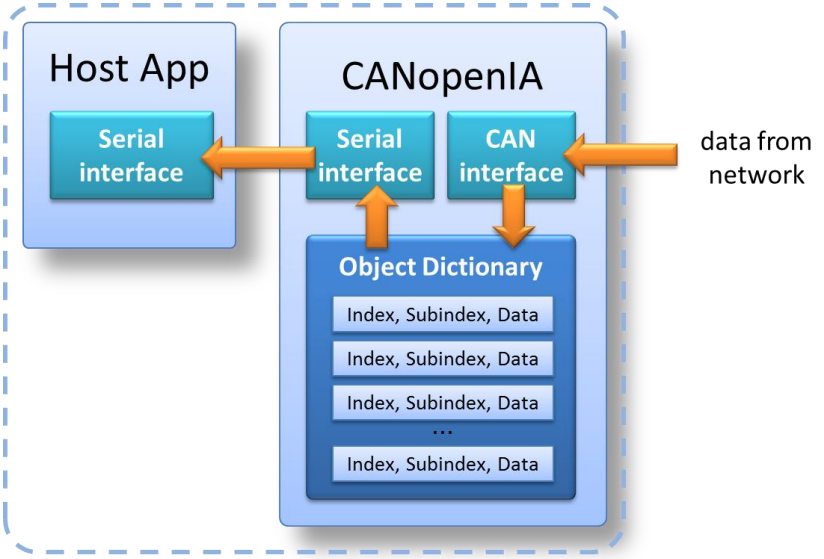


FIG. 13 – INFORMATION FLOW ON NEW DATA RECEIVED

New process data arrived from the CANopen network and was written to a local Object Dictionary entry. The node ID of the sender (if known), the Object Dictionary entry in question and the new data is part of this indication. This applies to both data received by SDO and PDO access. Data size is indicated via length field of lower communication layer or length parameter when use is a library.

In minimal manager and CiA 447 mode ALL CiA defined/known PDOs are received and cause an indication. Advanced versions allow setting of optional filters to ignore unwanted PDOs.

Serial: `D<nodeid><index><subindex><data>`

Language	Prototype
C	`void NewData(UNSIGNED8 nodeid, UNSIGNED16 index, UNSIGNED8 subindex, UNSIGNED16 length, UNSIGNED8 *data);`
C++	`static void NewData(UNSIGNED8 nodeid, UNSIGNED16 index, UNSIGNED8 subindex, UNSIGNED16 length, UNSIGNED8 *data);`
Java	`public static void NewData(byte NodeID, int Index, byte Subindex, int DataLength, Pointer Data, Pointer Param);`

Parameter	Description
nodeid	The ID of the node sending the data
index	Index of the object dictionary entry in the node
subindex	Subindex of the object dictionary entry in the node
length	Length of data data
data	The data

Command "W": Write to a local Object Dictionary entry

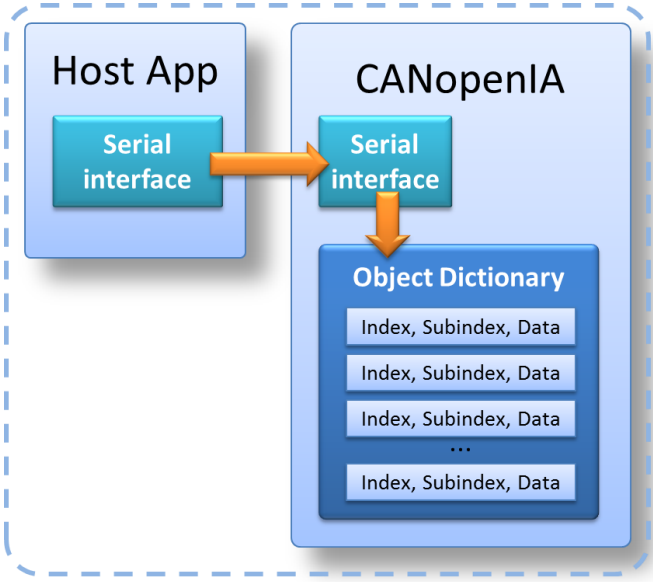


FIG. 14 – INFORMATION FLOW ON R/W COMMANDS

Writes data to one local Object Dictionary entry. Data size is indicated via length field of lower communication layer (see message definition).

Serial: `W<index><subindex><data>`

Language	Prototype
C	`UNSIGNED32 C_SerialProtocol_WriteLocalOD(UNSIGNED16 index, UNSIGNED8 subindex, UNSIGNED32 datalength, UNSIGNED8 *data)`
C++	`UNSIGNED32 SerialProtocol::WriteLocalOD(UNSIGNED16 index, UNSIGNED8 subindex, UNSIGNED32 datalength, UNSIGNED8 *data)`
Java	`long C_SerialProtocol_WriteLocalOD(short index, byte subindex, int datalength, Pointer data)`

Parameter	Description
index	The index of the object dictionary entry to write to
subindex	The subindex of the object dictionary entry to write to
datalength	Number of bytes to write
data	Data to write

Response "W": Write (local) response

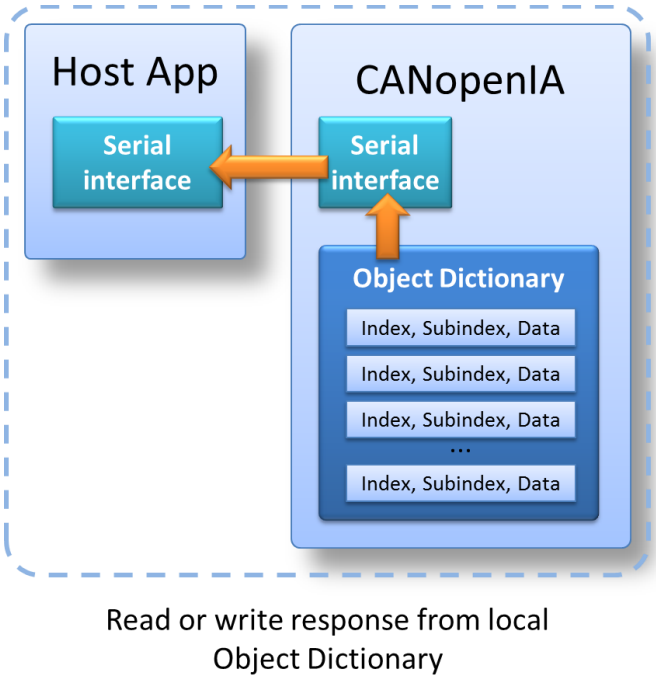


FIG. 15 – INFORMATION FLOW ON R/W RESPONSE

The following message is a response from the CANopen device to every “W” message processed.

Serial: `W<index><subindex><err>`

Not used in the programming interface *(WriteLocalOD is blocking and returns values)*

Command "R": Read from a local Object Dictionary entry

Request to read data from one Object Dictionary entry. Data size is indicated via length field of lower communication layer.

Serial: `R<index><subindex>`

Language	Prototype
C	`UNSIGNED32 C_SerialProtocol_ReadLocalOD(UNSIGNED16 index, UNSIGNED8 subindex, UNSIGNED32 *datalength, UNSIGNED8 *data)`
C++	`UNSIGNED32 SerialProtocol::ReadLocalOD(UNSIGNED16 index, UNSIGNED8 subindex, UNSIGNED32 *datalength, UNSIGNED8 *data)`
Java	`long C_SerialProtocol_ReadLocalOD(short index, byte subindex, Pointer datalength, Pointer data)`

Parameter	Description
index	The index of the object dictionary entry to read from
subindex	The subindex of the object dictionary entry to read from
datalength	When called set to the maximum number of bytes to read. On return holds the number of bytes read
data	Filled with read data

Response "R": Read (local) response

The following message is a response from the CANopen device to every “R” message processed. Data size is indicated via length field of lower communication layer (see message definition).

Serial: `R<index><subindex><err><data>`

Not used in the programming interface *(ReadLocalOD is blocking and returns values)*

The return value is ERROR_NOERROR or an error code, ERROR_xxx.

7.4 Remote Access Demo

As part of the delivery, a programming example is provided. It is named RA-App and uses the serial commands and responses to access the CANopen network.

The remote access app is provided as command line executable for Linux and Windows along with all source files. It can be used as basis for own developments.

Use the -h parameter to get a list of supported command line parameters supported.

RA_App -h

Once connected, the RA_App displays all data received as follows:

{node ID, Index, Subindex, Length, Data}

Data received is displayed in hexadecimal along with the node ID from which the data was received (zero if unknown), followed by the Index and Subindex indicators, followed by the data.

8. CANopenIA-M0 Starter Kit

The CANopenIA-M0 Starter Kit provides a quick and easy way to evaluate the features of the module. It is available in a compact form factor with convenient connectors and mounting holes.

8.1 Connectors

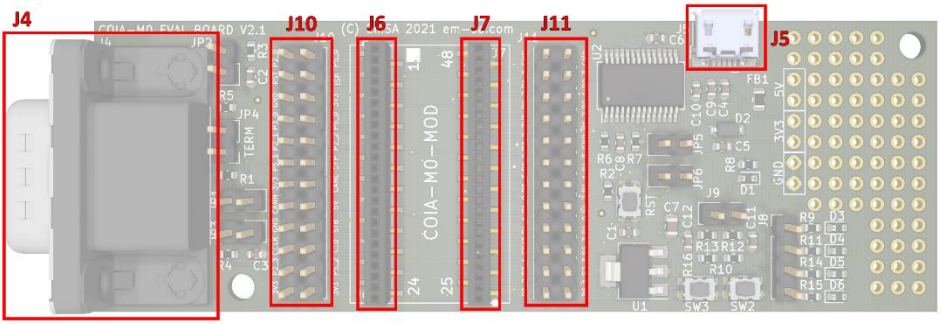


FIG. 16 – CONNECTORS OF THE STARTER KIT

The following connectors are available:

Connector	Description
J4	9-pin male D-type CAN connector using the standard pinout. CANL – 2, GND – 3, CANH – 7
J5	USB Micro B connector provides power to the board and serial communications to the module. When connected to a PC it will appear as a virtual COM port in Windows Device Manager. See jumper JP5/6 settings for connection of the serial port
J6/J7	Insert a CANopenIA-M0 module into the board using these connectors. Pay attention to the polarity markings
J10/J11	All signals on the module pins are brought to these headers. The silk-screen indicates the functionality of each pin

TABLE 8 – CONNECTORS OF THE STARTER KIT

Modules are marked with a square on the top of the board to indicate pin one. Ensure when inserting a module that pin one is aligned with pin one on the board, also marked with a square.

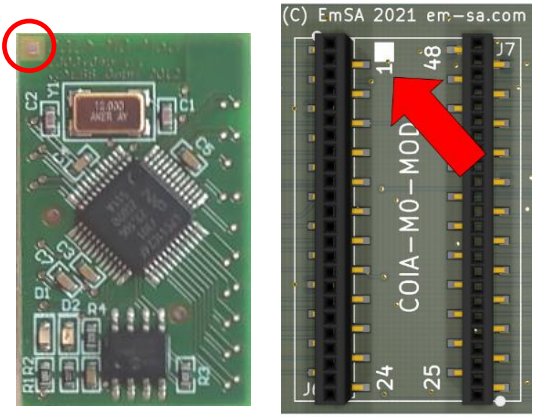


FIG. 17 – LOCATION OF PIN 1 ON THE MODULE AND THE BOARD

8.2 Jumpers

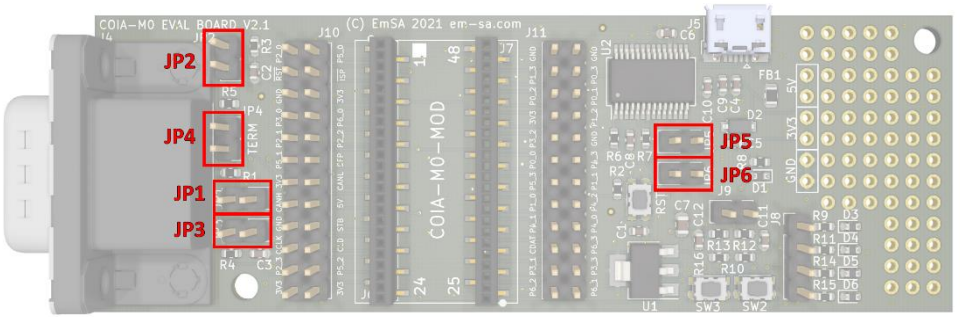


FIG. 18 – JUMPERS ON THE STARTER KIT

The following jumpers are available:

Jumper	Description
JP1	When closed the CAN transceiver will be in standby/silent mode. When open the CAN transceiver will be active.
JP2	On power-up and reset: When closed the module will boot from the NXP on-chip bootloader. When open the module will start the firmware.
JP3 / SFP	Special function pin, on power-up and reset: When closed AND JP2 is closed the module will boot from the NXP on-chip CAN bootloader. When closed AND JP2 is open the stored configuration is ignored, CAN bitrate is set to 125kbps. When open the module will start the firmware using stored configuration.
JP4	When closed 120 Ohms of termination resistance will be applied to the CAN bus. When open no termination resistance will be applied.
JP5/JP6	When closed the USB virtual COM port is connected to the module. When open the P0_2 and P0_3 pins on the module can be used for I/O

TABLE 9 – JUMPERS OF THE STARTER KIT

8.3 Inputs and Outputs

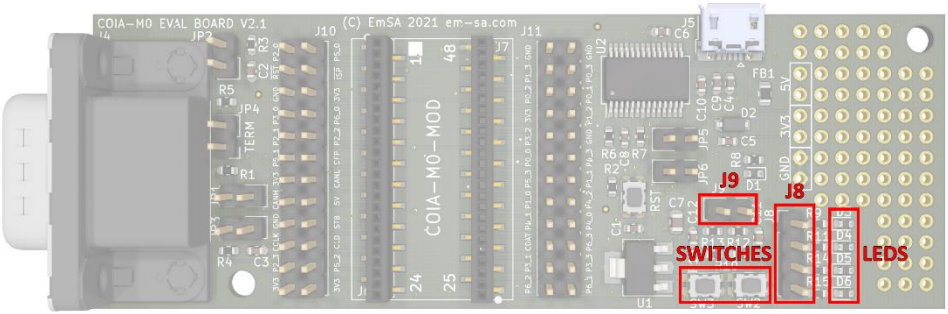


FIG. 19 – STARTER KIT I/O

The board features two switches and four LEDs.

The switches are connected to header J9. The left switch goes to the left pin and the right switch goes to the right pin. Pressing a switch will pull the pin low.

The LEDs are connected to header J8. The top pin goes to the top LED and the pins are in the same order as the LEDs. Pulling a pin low will turn the LED on.

Pins on the J8 and J9 headers can be connected to the breakout connectors J10 and J11 using jumper wires (aka Dupont wires).

8.4 Reset

The module can be reset by pressing the reset button.

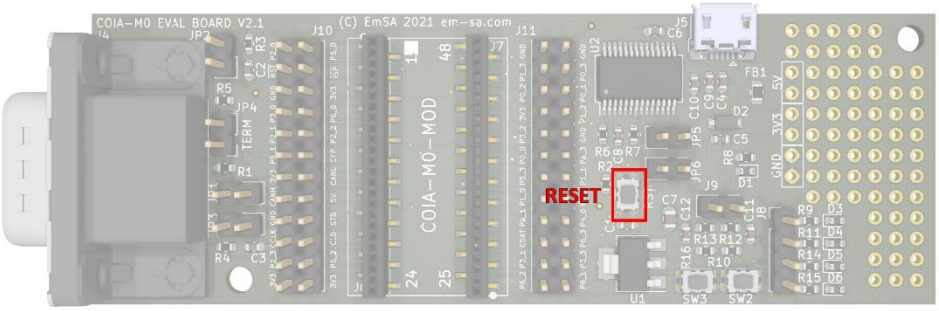


FIG. 20 – STARTER KIT RESET

8.5 Prototyping Area

A 2.54mm grid of holes is available for prototyping, e.g. to test with a specific analog to digital converter (ADC).

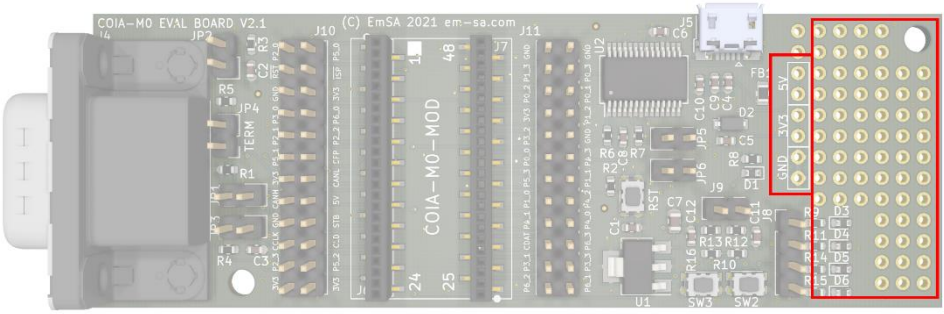


FIG. 21 – STARTER KIT PROTOTYPING

5V, 3.3V power and ground are provided on pairs of holes to the side.

8.6 Mounting Holes

Two 3mm (M3) mounting holes are provided on a 1mm grid. Do not use washers or cause any part of a bolt to touch exposed metal or components on the board. Do not overtighten bolts.

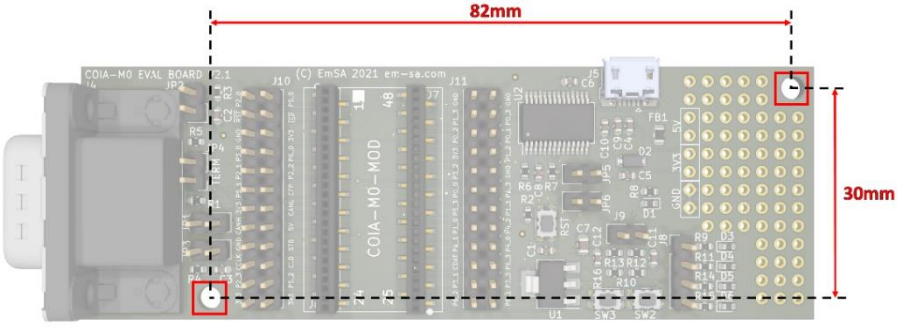


FIG. 22 – STARTER KIT MOUNTING

9. Diagnostics, Errors & Recovery

9.1 CANopen Emergency Messages

The CANopenIA-M0 devices produce CANopen EMCY messages. These use the CAN message ID 80h plus the node ID. The first 2 data bytes contain a 16-bit EMCY code. If that code is smaller or equal to 100h, then all previous emergencies are resolved (EMCY reset). EMCY codes used are listed in the table below.

EMCY	Description	Additional info (last 5 data bytes)
0000h	Error reset on system reset	Config active, ports ok, source of node ID, config use, 0 (*)
0000h	Config storage complete	"OVER0"
6100h	Fatal system error	16bit internal error code
6201h	Bootloader, download error	
6300h	Config storage error (memory error)	"MEMER"
6301h	Configuration error on system reset	Config active, ports ok, source of node ID, config use, 0 (*)
8130h	Heartbeat Lost	Node ID of device lost
8200h	Unknown NMT command received	
8210h	Receive PDO length mismatch	16bit PDO number, configured length, received length, 0
FF00h	Config storage in progress (do not power cycle)	"WAIT0"

TABLE 10 – EMCY CODES USED

(*) The "Config active" byte is 0 when the configuration was loaded and activated, 1 when none is available and 2 when there was an error loading it.

The "ports ok" byte is 0 when all ports where properly initialized. Otherwise bits are set for each port that has a configuration error (bits 0 to 6 used for the seven ports).

The “source of node ID” byte uses bits 0 to 6 to indicate:
bit 0 set: node ID taken from shift register
bit 1 set: shift register error
bit 2 set: node ID taken from configuration storage
bit 3 set: configuration storage failure
bit 4 set: node ID range error
bit 5 set: node ID taken from LSS
bit 6 set: defaults enforced by SFP or shift register

9.2 CANopen Indicator LEDs

The behavior of the green RUN LED and the red ERR LED is defined by CiA 303-3 (Device and network design, Part 3: CANopen indicators). Each LED can show one of the following patterns:

- Flickering (50ms blink)
- Blinking (200ms blink)
- Single Flash (one 200ms blink in 1s)
- Double Flash (two 200ms blink in 1s)
- Triple Flash (three 200ms blink in 1s)

The tables below list the situations where which LED shows which pattern.

Patterns used by the green RUN LED

Pattern	Description
Off	CANopen not active (initialization, bootloader, error)
Flickering	LSS active (waiting for node ID assignment)
Blinking	NMT state pre-operational
Single Flash	NMT state stopped
Double Flash	Unused
Triple Flash	Configuration or software download in progress
On	NMT state operational

TABLE 11 – RUN LED PATTERNS

Patterns used by the red ERR LED

Pattern	Description
Off	No error
Flickering	LSS active (waiting for node ID assignment)
Blinking	Invalid configuration
Single Flash	CAN error counter warning limit reached
Double Flash	Heartbeat consumer lost a heartbeat
Triple Flash	Unused
On	CAN bus off or other fatal error

TABLE12 –ERR LED PATTERNS

9.3 Ignore Configuration Stored

Once the bit rates for CAN and UART are configured, communicating with the device will only work using the configured bit rates.

For debugging, test and recovery of CANopenIA-M0 devices, the default configuration can be activated by:

a) Activating (pull-up) the SFP on power-up / reset
b) Setting all bits of the shift register to 1 (pull-down) on power-up / reset
c) If the shift register is not used, pull-down CONF_DAT on power-up / reset

The CANopenIA-M0 then uses

- node ID 40h (64d),
- a CAN bit rate of 125 kbps and
- a UART bit rate of 115200 bps.

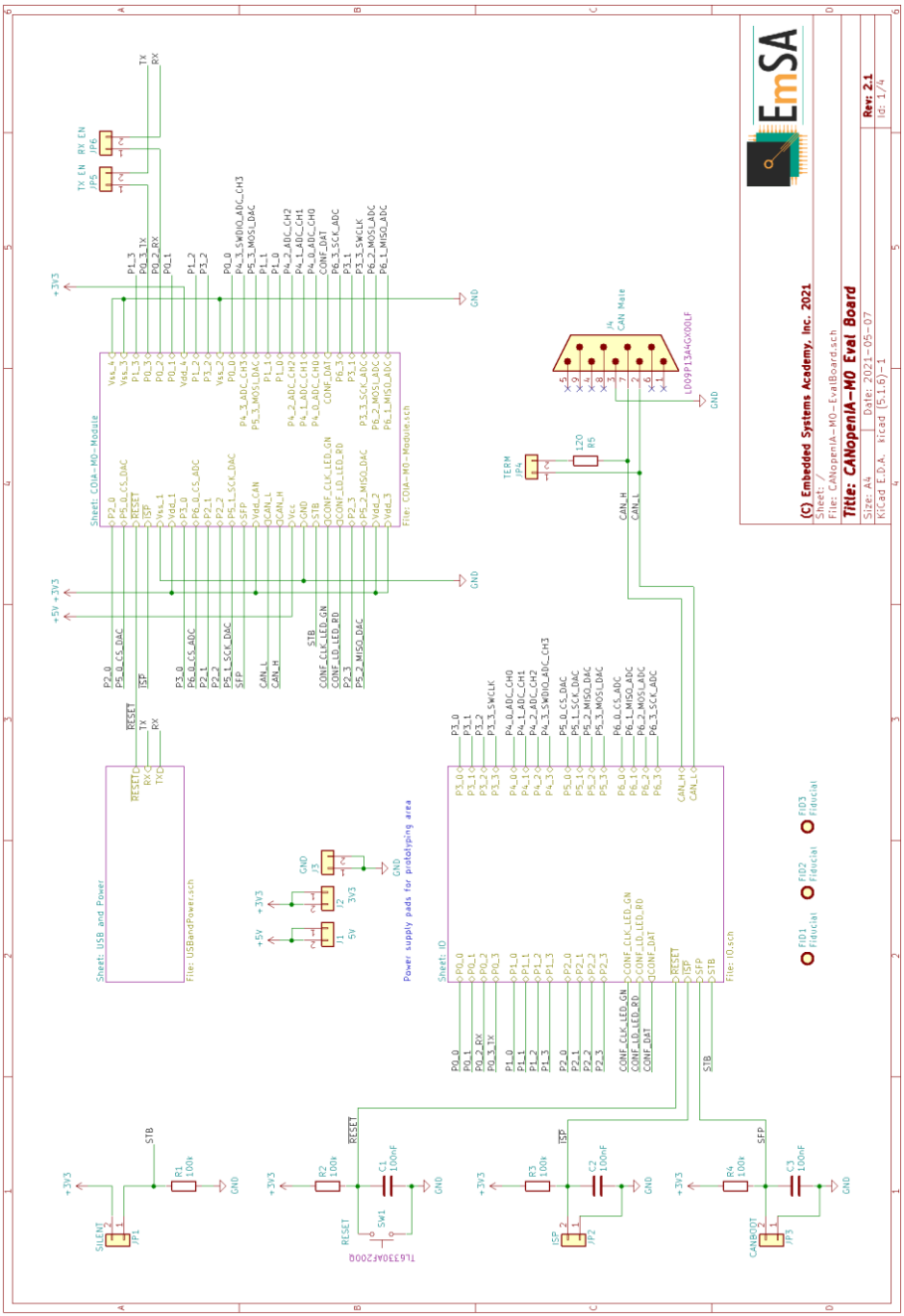
10. Disclaimer

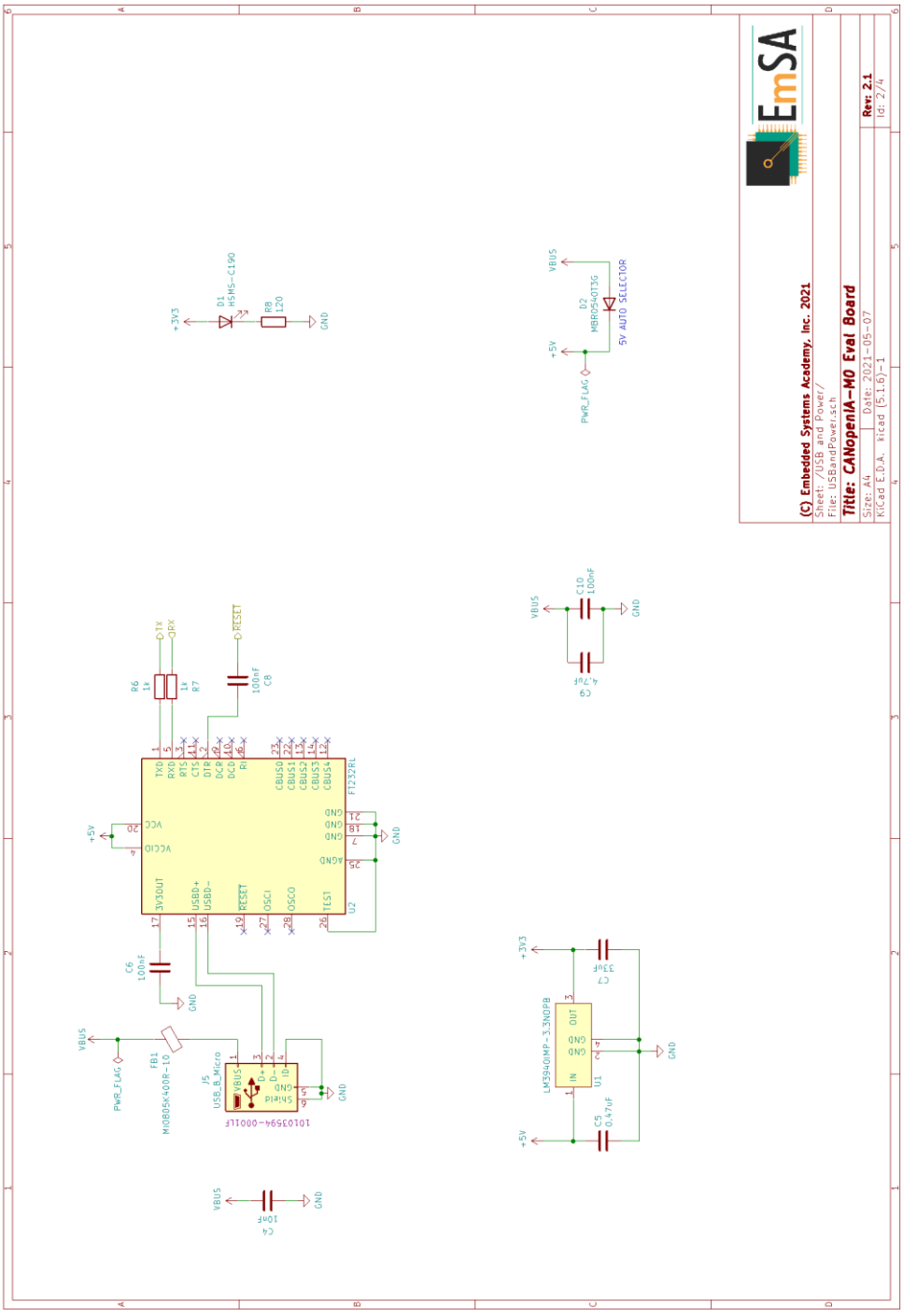
The products described in this manual are not designed for use in life support appliances, devices or systems where malfunction of these products can reasonably be expected to result in personal injury. EmSA Embedded Systems Academy GmbH customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify EmSA Embedded Systems Academy GmbH for any damages resulting from such application.

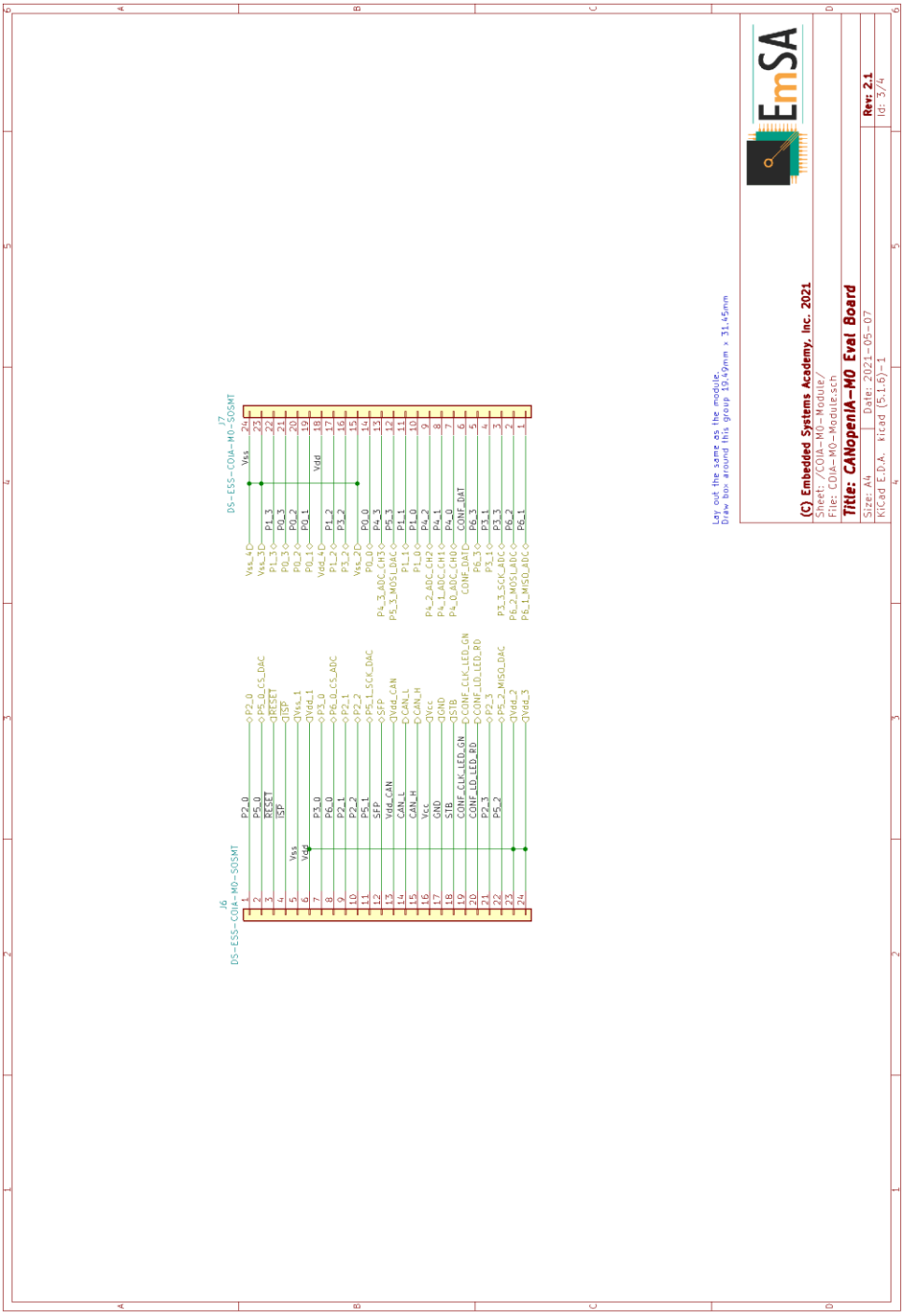
11. Right to make changes

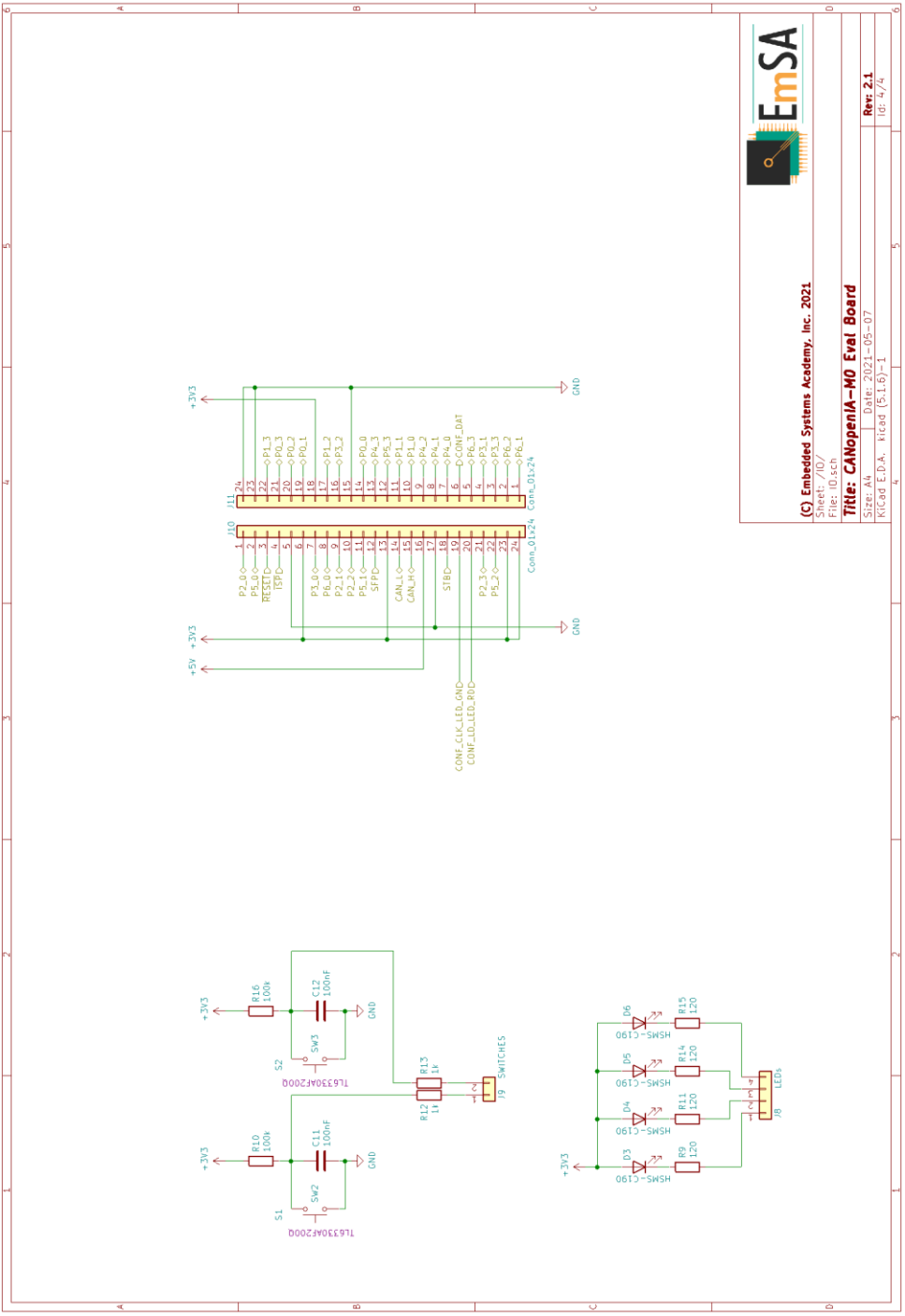
EmSA Embedded Systems Academy GmbH reserves the right to make changes, without notice, in the products, and/or software, described or contained herein in order to improve design and/or performance. EmSA Embedded Systems Academy GmbH assumes no responsibility or liability for use of any of these products, conveys no license or title under any patent, copyright, or mask work to right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

Appendix: Starter Kit Schematics




(C) Embedded Systems Academy, Inc. 2021
Sheet: /USB and Power/
File: USBandPower.sch
Title: CANopenIA-M0 Eval Board
Size: A4
Date: 2021-05-07
KiCad E.D.A. kicad (5.1.6)-1
Rev: 2.1
Id: 2/4
EmSA
5V AUTO SELECTOR
FT232RL
USB_B_Micro
LM3940IMP-3.3NOPB
MBR0540T3G
HSMS-C190
PWR_FLAG
VBUS
+5V
+3V3
GND


J6
DS-ESS-COIA-M0-SOSMT
J7
DS-ESS-COIA-M0-SOSMT
P2_0
P5_0
RESET
ISP
Vss
Vdd
P3_0
P6_0
P2_1
P2_2
P6_1
SPP
Vdd_CAN
CAN_L
CAN_H
Vcc
GND
STB
CONF_CLK_LED_GN
CONF_LD_LED_RD
P2_3
P5_2
P4
Vss_4D
Vss_3D
P1_3
P0_3
P0_2
P0_1
Vdd_4D
P1_2
P3_2
Vss_2D
P0_0
P0_4
P5_3
P1_1
P1_0
P4_2
P4_1
P4_0
CONF_DAT
P6_3
P3_1
P3_3
P6_2
P6_1
P2_0_CS_DAC
P6_0_CS_ADC
P2_1
P2_2
P5_1_SCK_DAC
SPP
Vdd_CAN
CAN_L
CAN_H
Vcc
GND
STB
CONF_CLK_LED_GN
CONF_LD_LED_RD
P2_3
P5_2_MISO_DAC
Vdd_2
Vdd_3
Vss
Vdd_1
P3_0
P6_0_CS_ADC
P2_1
P2_2
P5_1_SCK_DAC
SPP
Vdd_CAN
CAN_L
CAN_H
Vcc
GND
STB
CONF_CLK_LED_GN
CONF_LD_LED_RD
P2_3
P5_2_MISO_DAC
Vdd_2
Vdd_3
P4_3_ADC_CH3
P5_3_MOSI_DAC
P1_1
P1_0
P4_2_ADC_CH2
P4_1_ADC_CH1
P4_0_ADC_CH0
CONF_DAT
P6_3
P3_1
P3_3_SCK_ADC
P6_2_MOSI_ADC
P6_1_MISO_ADC
Lay out the same as the module.
Draw box around this group 19.49mm x 31.45mm
EmSA
(C) Embedded Systems Academy, Inc. 2021
Sheet: /COIA-M0-Module/
File: COIA-M0-Module.sch
Title: CANopenIA-M0 Eval Board
Size: A4
Date: 2021-05-07
Rev: 2.1
KiCad E.D.A. kicad (5.1.6)-1
Id: 3/4


(C) Embedded Systems Academy, Inc. 2021
Sheet: /IO/
File: IO.sch
Title: CANopenIA-M0 Eval Board
Size: A4
Date: 2021-05-07
Rev: 2.1
KiCad E.D.A. KiCad (5.1.6)-1
Id: 4/4
EmSA
J11
J10
Conn_01x24
+3V3
+5V
GND
CONF_CLK_LED_GND
CONF_LD_LED_RED
R16 100k
R10 100k
C12 100nF
C11 100nF
SW3
SW2
S2
S1
TL6330AF200Q
R13 1k
R12 1k
J9 SWITCHES
D6
D5
D4
D3
HSMS-C190
R15 120
R14 120
R11 120
R9 120
J8 LEDs