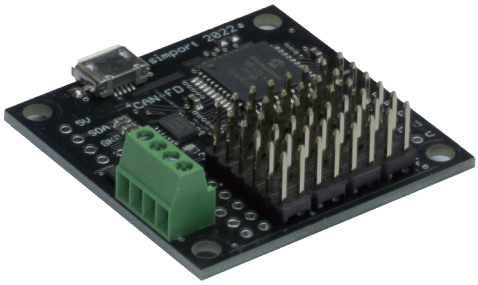
CAN in Simulation

Encoder Module



Rev. 1.2

Encoder Module

The Encoder module is designed to handle up to eight encoders, each of which can be optionally equipped with a pushbutton. The module is capable of decoding the direction of rotation, whether it‘s clockwise or counterclockwise, as well as detecting the speed of rotation, whether it‘s fast or slow. When an event occurs, the corresponding CAN message will include a data byte that uses five bits to indicate the specific event.

CAN ID	node ID	data type	service code	message code	data byte 0	data byte 1	data byte 2	data byte 3
708h	node	0Bh	item	num	data	0	0	0

CAN Message

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
fast	-	-	-	off	on	ccw	cw

Data Byte

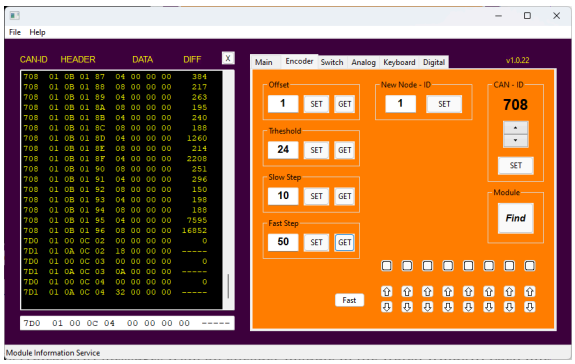
The encoder module uses two bits to signify either clockwise (cw) or counterclockwise (ccw) rotation, indicating an *increment* or *decrement* event based on the encoder‘s wiring.

The on/off bits are set when the pushbutton is either *pressed* or *released*.

The fast bit is used to indicate *rapid movement*, which is determined by a configurable time period between two consecutive encoder pulses.

Additionally, the input for the pushbutton on an encoder can be repurposed for another switch if needed.

This is how the Configuration Tool views an Encoder Module (more on page 7):



The CAN-ID is displayed as **708** (hexadecimal), and the node ID is identified as **1**.

The Find button searches for any attached digital modules, which is useful when the modules are changed.

There are four parameters available that can be altered upon request, which are described in the next paragraph.

Available Parameters

Offset

The Encoder module is capable of handling up to 8 incremental encoders, each of which is assigned a unique ID.

Starting with the *offset* value, the 8 encoders are given consecutive ID values, which will be included in the CAN message sent by the board.

Since the ID values are 1 byte wide, up to 256 different encoders can be distinguished under a given Node-ID.

Threshold

The *threshold* is a time value, measured in milliseconds, that serves as a threshold for distinguishing between a fast pulse train and a slow one.

The speed of the encoder turning is determined by the time between two consecutive encoder events (clicks). If the time between these events is below the threshold value, the *fast* flag in the CAN data byte is set, indicating that the encoder is turning quickly. Conversely, if the time between events is above the threshold, the *fast* flag is cleared.

Parameter Setting

To modify the parameters of a module, the Module Configuration Service (MCS) is utilized. The MCS is assigned a unique CAN-ID of `7D0h` (equivalent to decimal value `2000`):

CAN ID	node ID	data type	service code	message code	data byte 0	data byte 1	data byte 2	data byte 3
`7D0h`	`node`	`0Ah`	`0Dh`	`pid`	`data`	`0`	`0`	`0`

node ID:	CAN node ID (*`node`*)
data type:	UCHAR (*`0Ah, 10d`*)
service code:	MCS (*`0Dh`*)
message code:	Parameter index (*`pid`*)
message data:	Parameter value (*`data byte 0`*)

The parameter ID (pid) is used to identify which specific parameter needs to be modified. Data byte 0 contains the value of the parameter.

Parameters IDs

index	parameter	value(s)
1	offset	1 ... 255
2	threshold	1 ... 255 [ms]

Upon completion of the parameter modification request, the response message will have a message code of **`0`** (zero) if the operation was successful. However, if the requested parameter is out of the valid range or the parameter ID is invalid, the response message will contain a message code of **`-6`**.

CAN-ID Setting

The CAN-ID range for Encoder board messages is **`708h..70Fh`** (decimal **`1800..1807`**).

To change the CAN-ID of the Encoder board, the CAN Identifier Setting Service (CSS) can be used. The message code should be set to 0.

CAN ID	node ID	data type	service code	message code	data byte 0	data byte 1	data byte 2	data byte 3
`7D0h`	`node`	`0Ch`	`0Eh`	`0`	`0`	`0`	`xh`	`xl`

node ID:	CAN node ID (*node*)
data type:	SHORT2 (*`0Ch`, `12d`*)
service code:	CSS (*`0Eh`, `14d`*)
message code:	`0`
message data:	New CAN ID high byte (`xh`, data byte 2) New CAN ID low byte (`xl`, data byte 3)

Upon completion of the CAN Identifier Setting request, the response message will have a message code of **`0`** (zero) if the operation was successful, or **`-6`** if the ID is out of the valid range.

Node-ID Setting

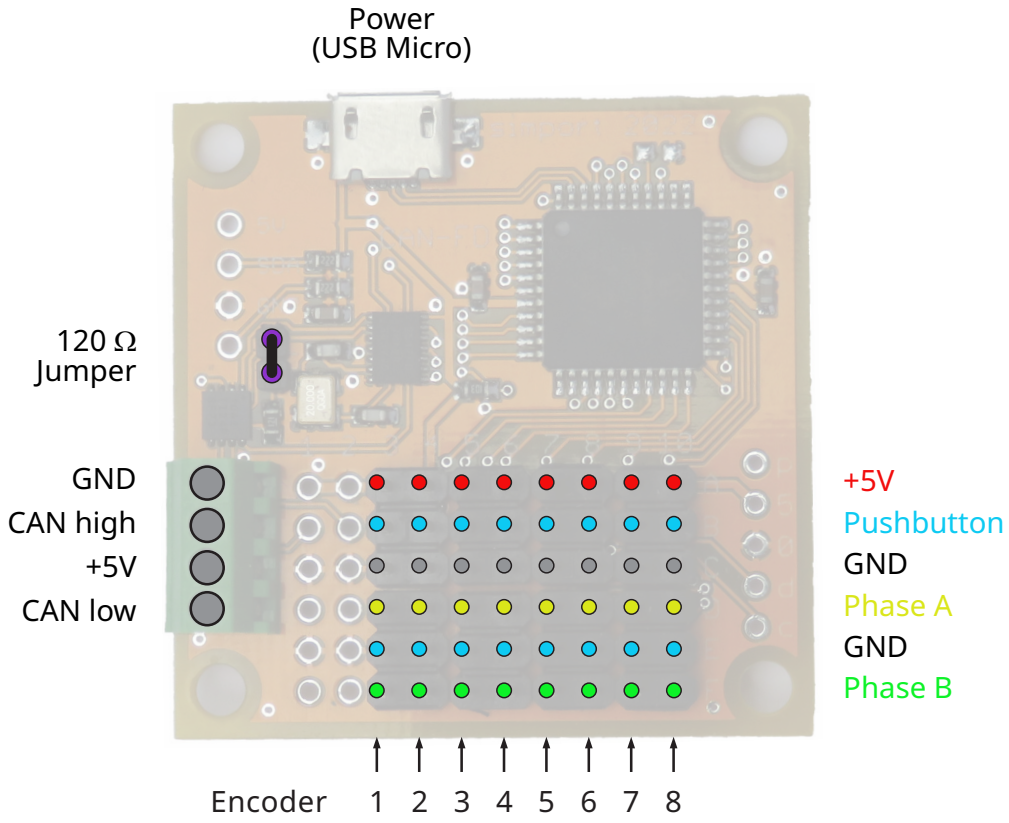
To change the Node-ID of the Encoder board, the Node ID Setting Service (NIS) can be used. Node-ID values are in the range of 1 to 255.

CAN ID	node ID	data type	service code	message code	data byte 0	data byte 1	data byte 2	data byte 3
7D0h	node	0	0Bh	x	0	0	0	0

node ID:	CAN node ID (*node*)
data type:	NODATA (*00h, 0d*)
service code:	NIS (*0Bh, 11d*)
message code:	New node ID (*1 ≤ X ≤ 255*)
message data:	0

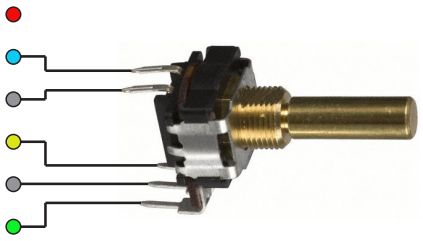
Upon completion of the Node Identifier Setting request, the response message will have a message code of **0** (zero) if the operation was successful.

Board Layout

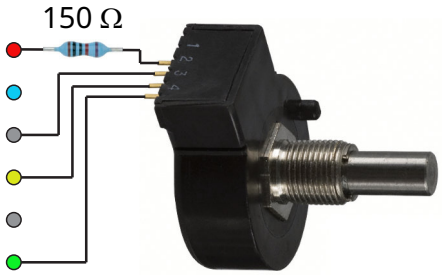


The 120 Ω jumper places a termination resistor between the CAN high and CAN low line.

Wiring Examples



Mechanical Encoder



Optical Encoder

Axis And Ohs (AAO) Scripting Example

This example uses a rotary incremental encoder with an attached wheel or knob to serve as a trim input for the elevator. The encoder events are linked to RPN scripts. Based on the encoder's rotation speed, there are four events to consider, each with its own script:

	clockwise	*counterclockwise*
slow	TrimElevatorUpSlow	TrimElevatorDownSlow
fast	TrimElevatorUpFast	TrimElevatorDownFast

Script (TrimElevatorUpSlow)	*stack*
Retrieve the current position of the trim foil and push it onto the stack: `(A:ELEVATOR TRIM POSITION,·Radians)·`	- n_1
Add a small value (0.001 rad), then duplicate the result on the stack: `(A:ELEVATOR·TRIM·POSITION,·Radians)·0.001·+·d·`	n_1 - n_2 n_2

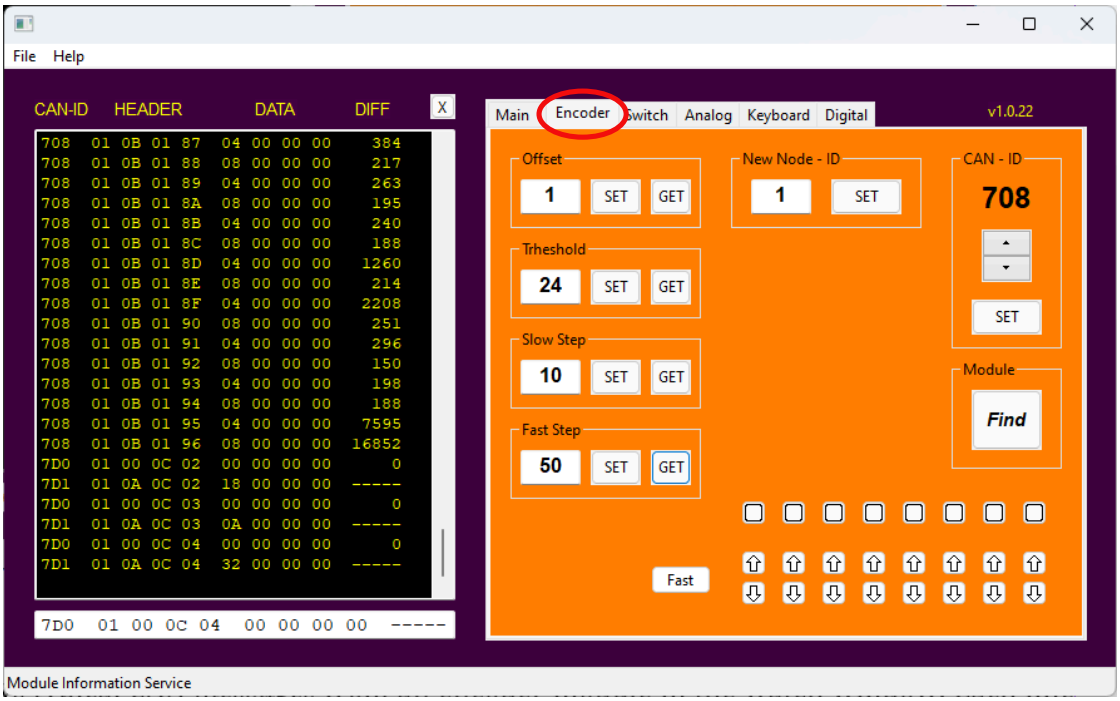
Fetch the max deflection and convert to radians:
`(A:ELEVATOR·TRIM·POSITION,·Radians)·0.001·+·d·`
`(A:ELEVATOR·TRIM·UP·LIMIT,·Degrees)·dgrd`
	n_2 n_2
	-
	n_2 n_2 n_3

Compare the new value to the maximum deflection; if it is less, update the position; otherwise, remove the value from the stack:
`(A:ELEVATOR·TRIM·POSITION,·Radians)·0.001·+·d·`
`(A:ELEVATOR·TRIM·UP·LIMIT,·Degrees)·dgrd·`
`<=·if{·(>A:ELEVATOR·TRIM·POSITION,·Radians)·}·els{·p·}`
	n_2 n_2 n_3
	-

The remaining scripts function similarly. For Trim DOWN events, a decrement is subtracted from the current position and compared to the minimum deflection. Fast movements use a larger increment or decrement (e.g., 0.001 becomes 0.01).

The Configuration Tool

The configuration tool displays an 'Encoder' panel on the right side, featuring the following layout:



When you open this panel for the first time, it automatically searches for an Encoder Board on the CAN bus. If a board is found, its *Node-ID*, *CAN-ID*, and *Offset* parameter are displayed. Subsequently, clicking the **Find** button initiates a new search.

Use the **GET** buttons to retrieve the *Threshold* parameter from the board.

All parameters, except for *CAN-ID*, can be adjusted by editing the values in their respective fields. To update these parameters in the module, press the **SET** button. For *CAN-ID*, use the spin buttons to adjust the value, then press the **SET** button to apply the changes.

The activity of the board's eight encoders is indicated by small up/down arrows ⇧⇩ that light up when an encoder is turned. The **Fast** button flashes when an encoder is turned quickly, and the squares light up green 🟩 when an encoder's pushbutton is pressed.

To simulate this activity and send the corresponding messages on the CAN bus, simply click on any of the arrow or square buttons. Toggling the **Fast** button will toggle the *fast* bit in the message.

You can monitor all CAN bus activity in the left window.

Board Dimensions [mm]