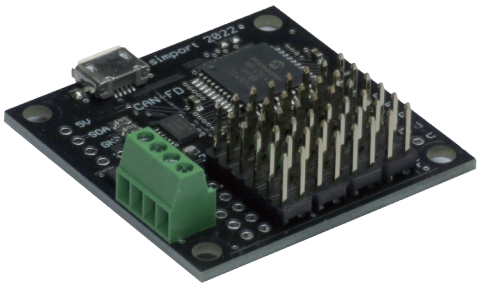
CiS

CAN in Simulation
Digital Module



Rev. 1.2

Digital Module

The digital module serves as an output device designed to control LEDs, relays, actuators, or various other digital components.

Supporting a maximum of 24 output lines, each with the capacity to source or sink up to 50 mA, the system ensures compliance as long as the total current does not surpass 350 mA or the total power dissipation remains below 200 mW.

The output drivers offer flexibility and can be configured as either push-pull or open-drain, providing adaptability to different circuit requirements.

CAN ID	node ID	data type	service code	message code	data byte 0	data byte 1	data byte 2	data byte 3
730h	node	0Bh	item	num	data	0	0	0

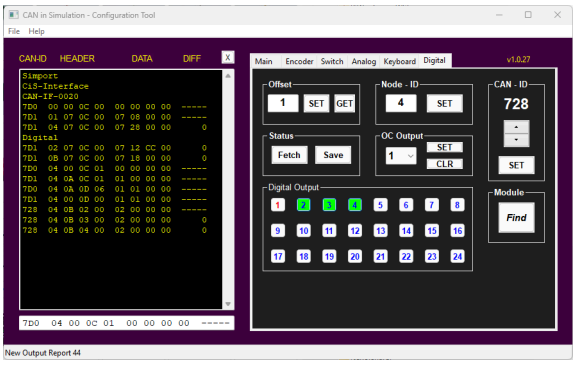
CAN Message

The *data* byte consists of binary values (0 or 1) and determines whether the output is driven *low* (Bit 1 = 1, data value = 2) or *high* (Bit 0 = 1, data value = 1).

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
–	–	–	–	–	–	low	high

Data Byte

This is how the Configuration Tool views a Digital Module (more on page 10):



The CAN-ID is displayed as 728 (hexadecimal), and the node ID is identified as 4.

You can change these values using the spin buttons for the CAN-ID or by editing the node ID. Clicking the SET button afterward will overwrite the corresponding value in the module.

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

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

Available Parameters

Offset

The Digital module is capable of handling up to 24 output lines, each of which is assigned a unique ID.

Starting with the *offset* value, the 24 lines 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 output lines can be distinguished under a given Node-ID.

Driver Mode

The *driver mode* can be set to push-pull or open drain. The push-pull output is capable of driving two output levels. It can either pull current to ground (sink current from the load) or push current from the power supply voltage (source current to the load).

In an open-drain configuration, the output is limited to driving to ground, with the alternative state being high impedance. This feature allows a connection of outputs to form a logical OR network.

Safe States

The *save states* command allows the durable preservation of current port states in EEPROM. Upon the subsequent power-up event, this stored configuration is retrieved, reinstalling the port states to their previous settings.

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`	`x`	`y`	`0`	`0`

node ID:	CAN node ID (*node*)
data type:	UCHAR (*0Ah, 10d*)
service code:	MCS (`0D`)
message code:	Parameter index (*pid*)
message data:	Parameter value (x, *data byte 0*) Affected port pin (y, *data byte 1*) or `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 and data byte 1 specifies the affected output port.

Parameters IDs

index	parameter	value(s)
1	offset	1 ... 255
6	driver mode	`0` = push-pull, `1` = open drain
7	save states	none, initiates a save command

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 Digital board messages is **`728h..72Fh`** (decimal **`1832..1839`**).

To change the CAN-ID of the Digital board, the CAN Identifier Setting Service (CSS) can be used. The message code (parameter ID) 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:	SHORT (*`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 Digital 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.

State Transmission

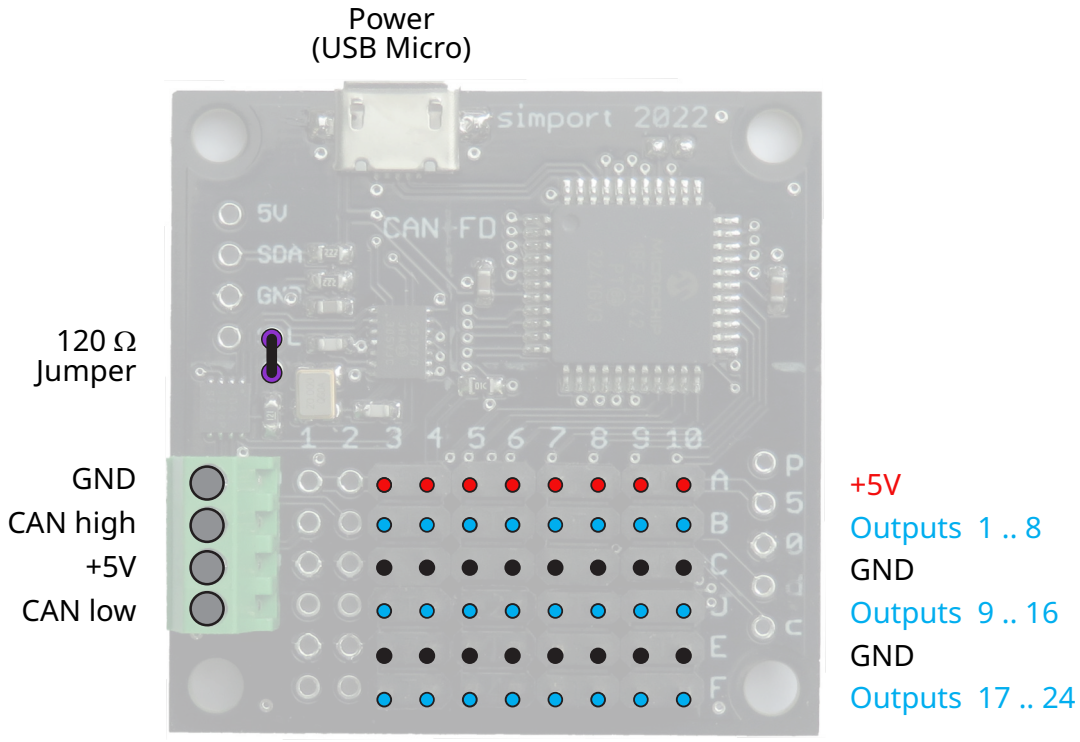
The status of a digital board‘s parameters can be obtained through the State Transmission Service (STS). With 24 distinct output lines, the individual driver modes or the output states are efficiently packed into 3 bytes (24 bits) for ease of transmission.

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	07h	0	0	0	0	0

node ID:	CAN node ID (*node*)
data type:	NODATA (*00h, 0d*)
service code:	STS (*07h, 7d*)
message code:	0
message data:	0

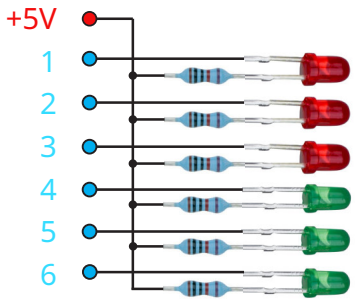
Upon completion of the State Transmission request, the response message will return the queried status bitwise packed across data bytes 0 to 3 with the highest bit first (most significant bit in data byte 0).

Board Layout

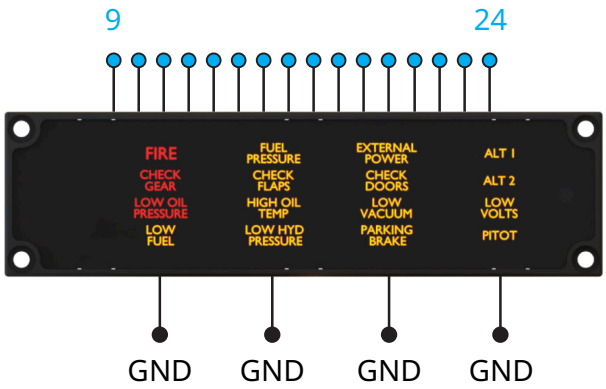


The 120 Ω jumper places a termination resistor between the CAN high and CAN low line.

Wiring Examples



Set output *low* to illuminate LED



Set output *high* to illuminate LED

Axis And Ohs (AAO) Scripting Example

A practical script can be set up as either a „Global Automated Script“ or an „Aircraft Automated Script“ in AAO, with a 200ms delay.

This example uses three red and three green LEDs as indicators for the landing gear status. The red LEDs illuminate while the gear is in motion, and the green LEDs light up when the gear is in the fully extended position. Output ports 1 to 3 control the red LEDs, while ports 4 to 6 are used for the green LEDs.

Assuming the digital output board has a node ID of 5 and responds to CAN-ID 0x728, there are two CAN output commands for controlling the outputs. These commands drive the output either low (to turn the LED on) or high (to turn the LED off):

LED on: (CANMSG:1|0728|050B0x0002000000)

LED off: (CANMSG:1|0728|050B0x0001000000)

Here, x represents the corresponding output port number.

The LEDs are connected as illustrated in the wiring example above.

script	*stack*
Fetch the variable for the nose gear position onto the stack: `(A:GEAR·LEFT·POSITION,·Percent·Over·100)`	- n_1
Duplicate it twice, d d, so that we have three numbers on the stack: `(A:GEAR·LEFT·POSITION,·Percent·Over·100)·d·d`	n_1 - n_1 n_1 n_1
Compare the top of the stack with 0 (the gear UP position), leaving a boolean: `(A:GEAR·LEFT·POSITION,·Percent·Over·100)·d·d·0·!=`	n_1 n_1 - n_1 n_2
Fetch the second-to-last item and compare it to 1 (the gear DOWN position) leaving another boolean: `(A:GEAR·LEFT·POSITION,·Percent·Over·100)·d·d·0·!=·r·1·!=`	n_1 n_2 - n_1 n_2 n_3
Perform a logical AND operation and push the result onto the stack, the result will be true if the gear position is neither UP nor DOWN. `(A:GEAR·LEFT·POSITION,·Percent·Over·100)·d·d·0·!=·r·1·!=·and`	n_1 n_2 n_3 - n_1 n_4

Axis And Ohs (AAO) Scripting Example (continued)

A local variable (L:center) is used to keep track of the state (moving or stable end position)

A value of 1 indicates that the gear is in motion, while a value of 0 means the gear is in a stable position (either UP or DOWN).

The ***if*** clause (gear is moving):

script	*stack*
Check if we are transitioning from a stable end position (local variabel is 0), then switch red LED on: **if{(L:center)·0·==·if{(CANMSG:1|0728|050B020001000000)**	n_1 n_4 -
Set local variable to 1 (gear is moving) and pop unused item from stack: if{(L:center)·0·==·if{(CANMSG:1|0728|050B020001000000) **·1·(>L:left)·p**	n_1 -
Switch the green LED off: if{(L:center)·0·==·if{(CANMSG:1|0728|050B020001000000) ·1·(>L:center)·p·**(CANMSG:1|0728|050B050002000000)**}	-
However, if the local variable has already been set to 1, then just discard the unused item from the stack: if{(L:center)·0·==·if{(CANMSG:1|0728|050B020001000000) ·1·(>L:center)·p·(CANMSG:1|0728|050B050002000000)}·**els{p}**}	-

Axis And Ohs (AAO) Scripting Example (continued)

The ***else*** clause (gear is UP or DOWN):

script	*stack*
Check if we are transitioning from moving to a stable position (local variabel is 1), then switch red LED off: **els{(L:center)·0·>·if{(CANMSG:1|0728|050B020002000000)**	n_1 -
Set local variable to 0 (gear is stable now): els{(L:center)·0·>·if{(CANMSG:1|0728|050B020002000000) **·0·(>L:center)**	n_1 -
If gear is now in DOWN position, then switch the green LED on. If the position is already stable, remove the unused value from the stack.: if{(L:center)·0·==·if{(CANMSG:1|0728|050B020001000000) ·0·(>L:center)**·1·==·if{(CANMSG:1|0728|050B050001000000)}}** **els{p}**}	n_1 -

Thus, a complete script for the landing gear is as follows:

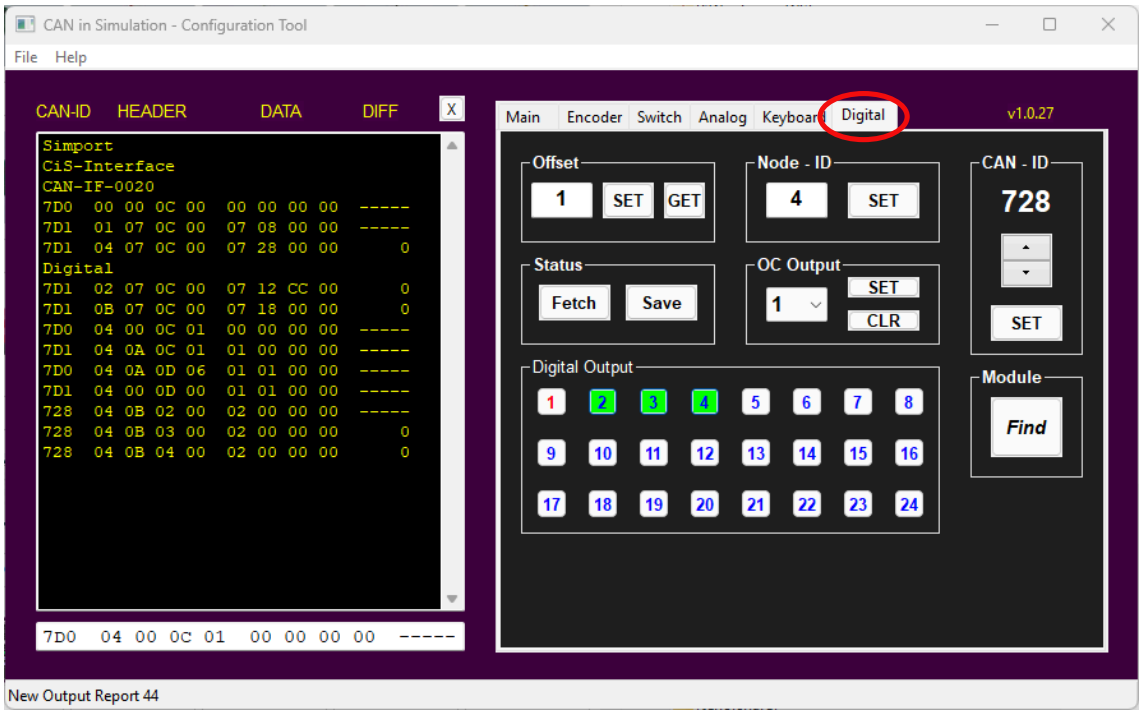
```
(A:GEAR·CENTER·POSITION,·Percent·Over·100)·d·d·0·!=·r·1·!=·and
if{(L:center)·0·==·if{(CANMSG:1|0728|050B020001000000)·
1·(>L:center)·p·(CANMSG:1|0728|050B050002000000)}·els{p}}
els{(L:center)·0·>·if{(CANMSG:1|0728|050B020002000000)·
0·(>L:center)·1·==·if{(CANMSG:1|0728|050B050001000000)}}}

(A:GEAR·LEFT·POSITION,·Percent·Over·100)·d·d·0·!=·r·1·!=·and
if{(L:left)·0·==·if{(CANMSG:1|0728|050B010001000000)·
1·(>L:left)·p·(CANMSG:1|0728|050B040002000000)}·els{p}}
els{(L:left)·0·>·if{(CANMSG:1|0728|050B010002000000)·
0·(>L:left)·1·==·if{(CANMSG:1|0728|050B040001000000)}}}

(A:GEAR·RIGHT·POSITION,·Percent·Over·100)·d·d·0·!=·r·1·!=·and
if{(L:right)·0·==·if{(CANMSG:1|0728|050B030001000000)·
1·(>L:right)·p·(CANMSG:1|0728|050B060002000000)}·els{p}}
els{(L:right)·0·>·if{(CANMSG:1|0728|050B030002000000)·
0·(>L:right)·1·==·if{(CANMSG:1|0728|050B060001000000)}}}
```

The Configuration Tool

The configuration tool features a "Digital" panel on the right side, which appears as follows:



When you open this panel for the first time, it automatically searches for a Digital Output 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.

The *Node-ID* and *Offset* parameters can be adjusted by editing the numbers in their respective fields. Pressing the **SET** button will immediately update these parameters in the module. The *CAN-ID* can be adjusted using the spin buttons.

In the ‚Status' field, the **Fetch** button retrieves the current output states from the board and displays them in the ‚Digital Output' field. A blue number indicates an output configured as push-pull, whereas a red number signifies an open-drain output. A white background indicates that the output is in a "passive" state, meaning it is at a high level or in high impedance. A green background signals that the output is active or at a low level.

You can toggle the output between high and low by clicking the corresponding button. This action sends a CAN message to the board to update the real output state. The states can be permanently saved to the board by using the **Save** button in the 'Status' field.

To change the output's driver mode, select the desired output in the ‚OC Output' field using the spin control. Then press **SET** to configure it as an open-drain output or **CLR** to switch it to push-pull mode.

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

Board Dimensions [mm]