GOcontroll®

Modular Embedded Electronics

Moduline embedded controllers

Software manual

Content

1. Introduction

This software manual was developed to ease the use and implementation of a Moduline series embedded controller. There are a few related documents which are also described in this chapter. Information listed in this document is subjected to change. Therefor a changelog is added to this first chapter.

1.1. Related Documents

Alongside this software manual, there are two related documents to ease the use and implementation of a Moduline series embedded controller.

1.1.1. Moduline Embedded Controllers - Quick Start Guide

This document, as the name suggests, offers a quick start guide to power on the controller and connect to the controller.

1.1.2. Moduline Embedded Controllers - Hardware Manual

This document, as the name suggests, offers a comprehensive hardware manual with all technical details of the controllers, pinout diagrams and more.

1.2. Changelog

This changelog keeps track of changes made to this document.

Version	Date	Changes
V1.00	06-12-2023	Initial version

2. Operating system

The Moduline III, Moduline IV, Moduline Mini I and Moduline Display are equipped with NXP i.MX8 series application processors. This provides the ability to run Debian Linux to provide the maximum flexibility, security and maintainability.

2.1. Introduction

The controllers use a Debian 11 root file system which is mounted on a 5.10.72 Linux kernel. Both the root file system and the kernel were developed in house for the best possible cohesion between software and the GOcontroll hardware. GOcontroll is actively maintaining the software to further improve stability, functionality, security and ease of use.

2.2. System and service management

System and service management are vital to ensure customer satisfaction and service delivery. Systemctl is a command-line tool that allows for the management and monitoring of the systemd system and service manager.

2.2.1. System management

In this context, system management refers to the OS configuration. For systemctl, this includes shutdown and startup options.

Directly reboot the controller

```
systemctl reboot
```

Directly poweroff the controller

```
systemctl poweroff
```

2.2.2. Service management

While systemctl can help manage the system itself, it's even more useful for managing services. Admins can display active services, start and stop services, and control whether services start when the system boots. The systemctl command also has troubleshooting options.

Directly start a service with the command listed below.

```
systemctl start <service>
```

Directly stop a service with the command listed below.

```
systemctl stop <service>
```

Start a service during boot with the command listed below.

```
systemctl enable <service>
```

Disable starting a service during boot with the command listed below.

```
systemctl disable <service>
```

Look at the status of a service

```
systemctl status <service>
```

Check whether a service is active or not

```
systemctl is-active <service>
```

2.3. Generic services

2.3.1. ModemManager

ModemManager is the default mobile broadband management service in most standard GNU/Linux distributions. This software manual describes a few functions of ModemManager, for a comprehensive overview of all the functionalities of ModemManager, please visit this link.

- List available modems with modem indices with the command listed below. If the controller is equipped with the optional LTE/GSM module, the modem should be listed as index 0, as there are no more modems attached.

```
mmcli -L
```

- List specific modem information based on modem index with the command listed below. This command will show, among other things, hardware information of the modem attached, system device information, modem status and the service provider.

```
mmcli -m <index>
```

- Monitor specific modem based on modem index with the command listed below. This command will give a

```
mmcli -m <index> -w
```

2.3.2. NetworkManager

NetworkManager is the standard Linux network configuration tool suite. It supports a large range of networking setups for a wide range of devices.

- Show list of connections and their statuses with the command listed below.

```
nmcli con
```

- See detailed information about a specific connection with the command listed below.

```
nmcli con show <connection>
```

- Show available Wi-Fi networks with the command listed below.

```
nmcli dev wifi
```

- Connect to a wifi network with the given name and password with the command listed below.

```
nmcli dev wifi connect <ssid> password <password>
```

- Set network priority of a specific connection with the command listed below.The lowest number has the highest priority, but default is 0, which is the auto setting.

```
nmcli connection modify <connection-name> ipv4.route-metric <priority>
```

- To set a static IP address, there are a few steps required.

  - Firstly, turn off the auto connect property of the automatic connection with the command listed below.

  ```
  nmcli con mod “Wired connection auto” connection.autoconnect no
  ```

  - Secondly, turn on the auto connect property of the static connection and set the desired IP address with the command listed below. As an example the IP address 192.168.19.85 is used.

  ```
  nmcli con mod “Wired connection static” connection.autoconnect yes 192.168.19.85/16
  ```

  - Thirdly, turn off the automatic connection with the command listed below.

  ```
  nmcli con down “Wired connection auto”
  ```

  - Lastly, turn on the static connection with the command listed below.

  ```
  nmcli con up “Wired connection static”
  ```

2.3.3. Node-RED

The Linux based Moduline are pre-programmed with Node-RED. Node-RED is a well maintained, open source graphical programming environment which enables the user to program their control algorithms with the help of nodes. An example of enabling the service for automatic start after the next reboot is listed below.

```
systemctl enable nodered
```

2.4. GOcontroll services

GOcontroll has developed a few custom services to control controller specific functionalities. These services are actively maintained, to further improve reliability and ease of use.

2.4.1. Auto shutdown

The auto shutdown service turns off the controller when all enables are inactive and there is no Simulink model running through the go-simulink service. If the go-simulink service is running it is expected to implement the shutdown procedure in there. An example of enabling the service for automatic start after the next reboot is listed below.

```
systemctl enable go-auto-shutdown
```

2.4.2. GOcontroll Bluetooth server

This service starts the GOcontroll Bluetooth server that is used with the GOcontroll configuration app available for Android devices. If this service is disabled, the app can no longer be used as the controller will no longer broadcast a Bluetooth signal. Example of enabling the service for automatic start after the next reboot:

```
systemctl enable go-bluetooth
```

2.4.3. Connectivity guard

This service provides a guard for WWAN, LAN or WiFi connection for stationary applications which can safely be reset during networking issues.
Example of enabling the service for automatic start after the next reboot:

```
systemctl enable go-connectivity-guard
```

2.4.4. MATLAB Simulink

This service provides the ability to run compiled MATLAB Simulink models.
Example of enabling the service for automatic start after the next reboot:

```
systemctl enable go-simulink
```

2.4.5. Upload server

This service provides an upload server for new or updated MATLAB Simulink models.
Example of enabling the service for automatic start after the next reboot:

```
systemctl enable go-upload-server
```

2.4.6. WWAN

This service provides the WWAN and GNSS functionality of the controller.
Example of enabling the service for automatic start after the next reboot:

```
systemctl enable go-wwan
```

2.5. GOcontroll commands

2.5.1. Identify controller

This command gives information about the controller and makes the enclosure LED's flash, add -v for more detailed information about the modules.

```
identify
```

2.5.2. CAN test

This command checks the functionality of the onboard CAN busses. If a can test plug is attached to the controller this can be called to test if the can busses are functioning correctly.

```
go-test-can
```

2.5.3. LED test

This command will control the enclosure LEDs to check functionality. Red, green and blue are sequentially cycled with this command to check all LEDs function properly.

```
go-test-leds
```

2.5.4. Parse a2l file

This command parses a /usr/simulink/gocontroll.a2l to a json string, this can then be used by other applications. Normally this gets called automatically by upload-server.js, but can also be called manually in case the server was not used to transfer the file.

```
go-parse-a2l
```

2.5.5. Update controller software

This command updates the controller from the command line and gives the option for a development or stable update.

```
go-manual-update
```

2.5.6. Scan installed modules

This command scans the module slots and lists the installed modules and the firmware that is installed.

```
go-scan-modules
```

2.5.7. Update module firmware

This command scans the module slots and lists the installed modules and the firmware that is installed. After this is completed it will automatically update the modules that have outdated firmware.

```
go-update-modules
```

2.5.8. Overwrite module firmware

In case a module firmware needs to be downgraded, the command needs to be extended to force the firmware update, otherwise the program will reject overwrite commands that don’t provide a newer firmware than is available on the module.

```
go-overwrite-module “slot” “firmware.srec” “1”
```

3. Simulink

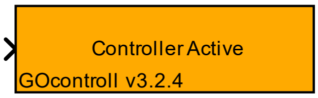
3.1. Introduction

All Moduline controllers can be programmed with MATLAB Simulink. MATLAB Simulink is a powerful, widely known graphical programming environment which enables the user to program their control algorithms with the help of blocks. MATLAB Simulink comes with a wide variety of standard blocks for most common interfaces and functions. GOcontroll developed a custom blockset to interface the optional plugin modules and the on-board features. Without the knowledge of code based programming, users are able to build, customize, monitor and maintain their control algorithms.

3.2. Blockset

GOcontroll created a custom blockset to use directly inside the MATLAB Simulink development environment. These blocks interface the modules plugged in to the controller, and the on-board features. When dragging and dropping nodes into the environment, the user is able to configure a specific feature according user demands by just selecting options.

3.2.1. Controller Active block

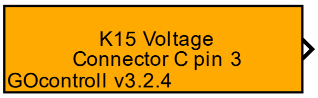


- Function

This block will cause a hard shutdown, this can potentially cause some file corruption if you are actively writing files in the background. To get a slower but gentle shutdown look at the Execute shell command block (3.2.30) to run the “shutdown now” command. If the controller is switched off by software, a restart is only possible when the device is activated by one of the K15 inputs.

- Input
    - Controller active
        - On (1)
        - Off (0)

3.2.2. K15 voltage block

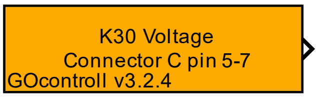


- Function

The K15 voltage provides the voltage on the K15 input pins. A positive voltage on one of these pins will start-up the controller. These function blocks can be used to read out which K15 input pin triggered the startup.

- Parameters
    - Sample time
    - K15 input pin
        - K15a
        - K15b
        - K15c
- Output
    - K15x input voltage in mV

3.2.3. K30 voltage block

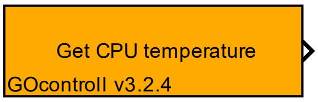


- Function

The K30 voltage provides the power supply to the controller. This function blocks can be used to read out the K30 supply voltage.

- Parameters
    - Sample time
- Output
    - K30 input voltage in mV

3.2.4. CPU Temperature block

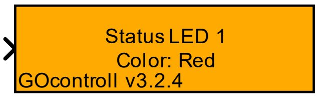


- Function

CPU temperature measurement from i.MX8M Mini's core. Value in °C.

- Parameters
  - Sample time

3.2.5. Enclosure LED block



- Function

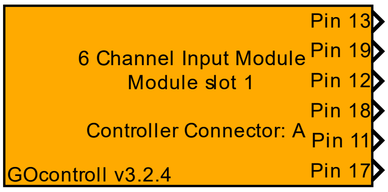
Control the four multicolor RGB LEDs independently. Brightness value 0-255, for Red, Green and Blue. 0 being minimum brightness, 255 being maximum brightness.

- Parameters
  - LED
    - LED 1
    - LED 2
    - LED 3
    - LED 4
  - Color
    - Red
    - Blue
    - Green
- Input
  - Brightness
    - 0-255

3.2.6. Input Module 6ch block

- Function

This block is used to read in a wide range of input signals. During initialization, the settings for each input channel are send to the module. The signal will be converted inside the module according the given settings. Configure module location, sample time and the module input channel configuration consisting of input type and the configuration of pull up and/or pull down resistors.



- Parameters
  - Controller type
    - Moduline IV
    - Moduline Mini I
    - Moduline Display
  - Sample time
  - Module location

  This parameter dictates the location of the installed module
    - Module slot 1
    - Module slot 2
    - Module slot 3
    - Module slot 4
    - Module slot 5
    - Module slot 6
    - Module slot 7
    - Module slot 8
  - 5V sensor supplies

  The built in 5V sensor supplies can be individually turned on and off.
    - Sensor supply 1
    - Sensor supply 2
    - Sensor supply 3
  - Function selection

  Select the desired input signal configuration.
    - Analog input - decimal (12 bit resolution)
    - Analog input - mV (1mV resolution)
    - Digital input - status (high or low)
    - Digital input - frequency (1Hz to 10 kHz)
    - Digital input - duty cycle low time (0.1% resolution)
    - Digital input - dutycycle high time (0.1% resolution)
    - Digital input - rotation speed (rpm)
    - Digital input - pulse counter (in combination with channel 2)
  - Input voltage range

  Select the expected input range of the signal.
    - 0-5 VDC
    - 0-10 VDC
    - 0-24V DC
  - Analog filter samples

  The number of samples that are passed into a moving average filter. (only active in analog input configuration)
    - 0-1000
  - Pulses per rotation

  The number of pulses for one rotation. This parameter is needed in case the module needs to calculate the rotation speed from i.e. an axle.
    - 1-200
  - Pull up resistors

  If required, a pull-up resistance can be added to the input signal.
    - 3.2 kΩ pull up resistor
    - 4.7 kΩ pull up resistor
    - 10 kΩ pull up resistor
  - Pull down resistors

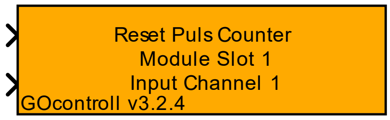
  If required, a pull-down resistance can be added to the input signal.
    - 3.2 kΩ pull down resistor
    - 4.7 kΩ pull down resistor
    - 10 kΩ pull down resistor

- Outputs
  - Input signal channel 1
  - Input signal channel 2
  - Input signal channel 3
  - Input signal channel 4
  - Input signal channel 5
  - Input signal channel 6

3.2.7. Input Module 6ch reset block

- Function

This block can be used to reset a pulse counter in the input module. If an input channel from the input module is configured as a pulse counter (to read in an encoder signal e.g.), the counter value can be re-assigned. In some cases during encoder calibration it is necessary to 'zero' this counter value. Configure module location and the input channel on which the pulse signal is applied.



- Parameters
    - Controller type
        - Moduline IV
        - Moduline Mini I
        - Moduline Display
    - Sample time
    - Module location

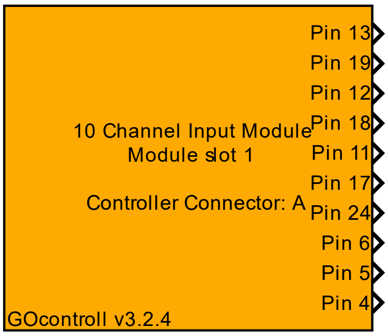
      This parameter dictates the location of the installed module
        - Module slot 1
        - Module slot 2
        - Module slot 3
        - Module slot 4
        - Module slot 5
        - Module slot 6
        - Module slot 7
        - Module slot 8
    - Input channel
        - Input channel 1
        - Input channel 2
        - Input channel 3
        - Input channel 4
        - Input channel 5
        - Input channel 6

- Inputs
    - Trigger
    - Pulse counter value

3.2.8. Input Module 10ch block

- Function

This block is used to read in a wide range of input signals. During initialization, the settings for each input channel are send to the module. The signal will be converted inside the module according the given settings. Configure module location, sample time and the module input channel configuration consisting of input type and the configuration of pull up and/or pull down resistors.



- Parameters
  - Controller type
    - Moduline IV
    - Moduline Mini I
    - Moduline Display
  - Sample time
  - Module location

    This parameter dictates the location of the installed module
    - Module slot 1
    - Module slot 2
    - Module slot 3
    - Module slot 4
    - Module slot 5
    - Module slot 6
    - Module slot 7
    - Module slot 8
  - 5V sensor supplies

    The built in 5V sensor supplies can be individually turned on and off.
    - Sensor supply 1
    - Sensor supply 2
  - Function selection

    Select the desired input signal configuration.
    - Analog input - decimal (12 bit resolution)
    - Analog input - mV (1mV resolution)
    - Digital input - status (high or low)
    - Digital input - frequency (1Hz to 10 kHz)
    - Digital input - duty cycle low time (0.1% resolution)
    - Digital input - dutycycle high time (0.1% resolution)
    - Digital input - rotation speed (rpm)
    - Digital input - pulse counter (in combination with channel 2)
  - Analog filter samples

    The number of samples that are passed into a moving average filter. (only active in analog input configuration)
    - 0-1000
  - Pulses per rotation

    The number of pulses for one rotation. This parameter is needed in case the module needs to calculate the rotation speed from i.e. an axle.
    - 1-200
  - Pull up resistor

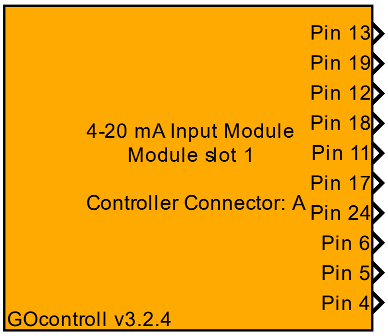
    If required, a pull-up resistance can be added to the input signal.
    - 3.2 kΩ pull up resistor
  - Pull down resistor

    If required, a pull-down resistance can be added to the input signal.
    - 3.2 kΩ pull down resistor
- Outputs
  - Input signal channel 1
  - Input signal channel 2
  - Input signal channel 3
  - Input signal channel 4
  - Input signal channel 5
  - Input signal channel 6
  - Input signal channel 7
  - Input signal channel 8
  - Input signal channel 9
  - Input signal channel 10

3.2.9. Input Module 4-20mA block

- Function

This block is used to read in 4-20mA input signals.



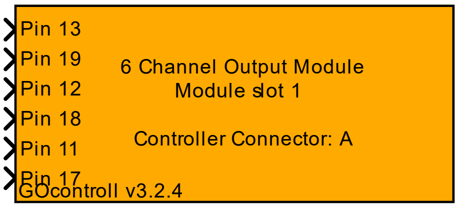
- Parameters
  - Controller type
    - Moduline IV
    - Moduline Mini I
    - Moduline Display
  - Sample time
  - Module location

    This parameter dictates the location of the installed module
    - Module slot 1
    - Module slot 2
    - Module slot 3
    - Module slot 4
    - Module slot 5
    - Module slot 6
    - Module slot 7
    - Module slot 8
- Outputs
  - Input signal channel 1
  - Input signal channel 2
  - Input signal channel 3
  - Input signal channel 4
  - Input signal channel 5
  - Input signal channel 6
  - Input signal channel 7
  - Input signal channel 8
  - Input signal channel 9
  - Input signal channel 10

3.2.10. Output Module 6ch block

- Function

This block is used to control the multi configurable output channels.



- Parameters
  - Controller type
    - Moduline IV
    - Moduline Mini I
    - Moduline Display
  - Sample time

    Execution frequency from function block. For proper functioning of the output module, us a sample time of 100 ms or lower.
  - Module location

    This parameter dictates the location of the installed module
    - Module slot 1
    - Module slot 2
    - Module slot 3
    - Module slot 4
    - Module slot 5
    - Module slot 6
    - Module slot 7
    - Module slot 8
  - Frequency

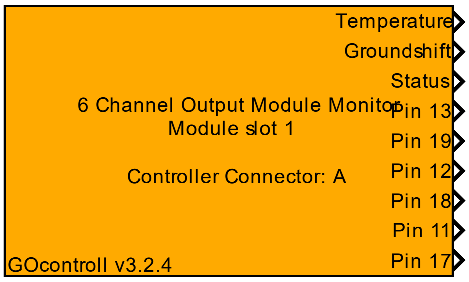
    In case a duty Cycle is used, select the frequency.
    - 100 Hz
    - 200 Hz
    - 500 Hz
    - 1 kHz
    - 2 kHz
    - 5 kHz
    - 10 kHz
  - Function selection

    Select the desired output configuration for each channel
    - Disable (output disabled)
    - Half Bridge dutycycle controlled (0 - 1000)
    - Low side switch dutycycle controlled (0 - 1000)
    - High side switch dutycycle controlled (0 - 1000)
    - Low side switch On - Off (0 - 1)
    - High side switch On - Off (0 - 1)
    - Peak and hold current mode (Half Bridge))
    - Frequency ouput 0 - 500 Hz (0 - 500)
  - Maximum channel current (mA)
    - 0-4000
  - Peak and hold parameters

    Select the desired output configuration for each channel
    - Peak current (mA)
    - Peak time (ms)

- Inputs
  - Output signal channel 1
  - Output signal channel 2
  - Output signal channel 3
  - Output signal channel 4
  - Output signal channel 5
  - Output signal channel 6

3.2.11. Output Module 6ch monitor block



- Function

This block is used to monitor the multi configurable output channels.

- Parameters
  - Controller type
    - Moduline IV
    - Moduline Mini I
    - Moduline Display
  - Sample time

    Execution frequency from function block. For proper functioning of the output module, us a sample time of 100 ms or lower.
  - Module location

    This parameter dictates the location of the installed module
    - Module slot 1
    - Module slot 2
    - Module slot 3
    - Module slot 4
    - Module slot 5
    - Module slot 6
    - Module slot 7
    - Module slot 8

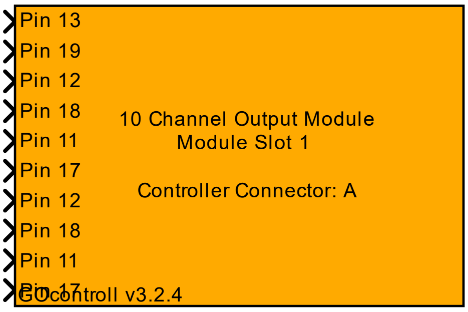
- Outputs

The current from each channel is measured and given in mA. If the ouput is configured as high side, the result will be positive. If the ouput is configured as low side, the value will be negative

  - Current measurement channel 1
  - Current measurement channel 2
  - Current measurement channel 3
  - Current measurement channel 4
  - Current measurement channel 5
  - Current measurement channel 6
  - Module temperature
  - Module ground-shift
  - Module status

Bit	Description	Hexidecimal value
0	Over current channel 1	0x1
1	Short circuit channel 1	0x2
2-3	Reserved	-
4	Over current channel 2	0x10
5	Short circuit channel 2	0x20
6-7	Reserved	-
8	Over current channel 3	0x100
9	Short circuit channel 3	0x200
10-11	Reserved	-
12	Over current channel 4	0x1000
13	Short circuit channel 4	0x2000
14-15	Reserved	-
16	Over current channel 5	0x10000
17	Short circuit channel 5	0x20000
18-19	Reserved	-
20	Over current channel 6	0x100000
21	Short circuit channel 6	0x200000
22-23	Reserved	-
24	Module ground shift	0x1000000
25	Module over temperature	0x2000000
26	Module total current exceeded	0x4000000
27	Module communication timeout	0x8000000
28	Module initialization failure	0x10000000
29	Module not communicating	0x20000000
30	Module checksum errors	0x40000000
31	Reserved	-

3.2.12. Output Module 10ch block



- Function

This block is used to control the multi configurable output channels.

- Parameters
  - Controller type
    - Moduline IV
    - Moduline Mini I
    - Moduline Display
  - Sample time

    Execution frequency from function block. For proper functioning of the output module, us a sampletime of 100 ms or lower.
    - Sample time
  - Module location

    This parameter dictates the location of the installed module
    - Module slot 1
    - Module slot 2
    - Module slot 3
    - Module slot 4
    - Module slot 5
    - Module slot 6
    - Module slot 7
    - Module slot 8
  - Frequency

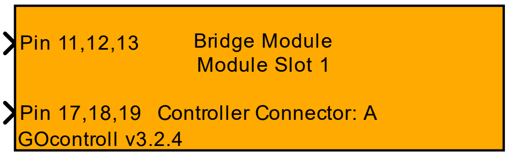
    In case a duty Cycle is used, select the frequency.
    - 100 Hz
    - 200 Hz
    - 500 Hz
    - 1 kHz
  - Function selection

    Select the desired output configuration for each channel
    - Disable (output disabled)
    - High side switch dutycycle controlled (0 - 1000)
    - High side switch On - Off (0 - 1)
    - High side peak and hold (0 - 1000)
    - Frequency ouput 0 - 500 Hz (0 - 500)
  - Peak and hold parameters

    Select the desired output configuration for each channel
    - Peak duty cycle (0-1000)
    - Peak time (ms)
- Input
  - Output signal channel 1
  - Output signal channel 2
  - Output signal channel 3
  - Output signal channel 4
  - Output signal channel 5
  - Output signal channel 6
  - Output signal channel 7
  - Output signal channel 8
  - Output signal channel 9
  - Output signal channel 10

3.2.13. Bridge module 2ch block

- Function



With this function block, you can configure each output of the bridge module (202001xx). During initialization of the controller, the corresponding outputs of the module are set according the settings. During runtime, the value pushed into the block will control the outputs.

- Parameters
  - Controller Type
    - Moduline IV
    - Moduline Mini I
    - Moduline Display
  - Sample time

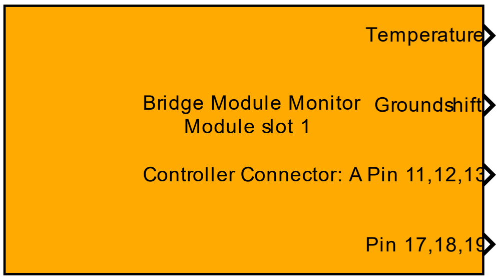
  The sample time is -1 for inherited since this block is a triggered system
  - Module location

  This parameter dictates the location of the installed module
    - Module slot 1
    - Module slot 2
    - Module slot 3
    - Module slot 4
    - Module slot 5
    - Module slot 6
    - Module slot 7
    - Module slot 8
  - Function selection

  Select the desired output configuration.
    - Disable (output disabled)
    - Half Bridge dutycycle controlled 0% - 100% duty cycle (0 - 1000)
    - Low side switch dutycycle controlled 0% - 100% duty cycle (0 - 1000)
    - High side switch dutycycle controlled 0% - 100% duty cycle (0 - 1000)
    - Low side switch On - Off (1-0)
    - High side switch On - Off (1-0)
  - Frequency

  In case a duty Cycle is used, select the frequency.
    - 100 Hz
    - 200 Hz
    - 500 Hz
    - 1 kHz
    - 2 kHz
    - 5 kHz
    - 10 kHz
- Input
  - Output signal channel 1
  - Output signal channel 2

3.2.14. Bridge module 2ch monitor block



- Function

With this function block, you can read parameters from the specified bridge module (202001xx). The block will provide some information about the module such as temperature and channel current.

- Parameters
  - Controller type
    - Moduline IV
    - Moduline Mini I
    - Moduline Display
  - Sample time

    The sample time is -1 for inherited since this block is a triggered system
  - Module slot

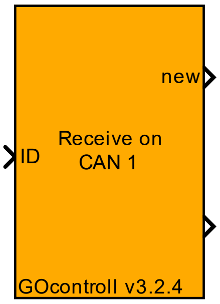
    Location of the installed module
    - Module slot 1
    - Module slot 2
    - Module slot 3
    - Module slot 4
    - Module slot 5
    - Module slot 6
    - Module slot 7
    - Module slot 8
- Output
  - Module temperature

    The temperature in degrees from the output module.
  - Module ground shift

    In case the ground connection of the module is poor, the power ground is shifted. The value gives the ground difference in mV and schould be beneath 50 mV
  - Channel current

    Current related output for each output channel in mA.

3.2.15. CAN Receive block



- Function

Receive the newest message from the specified CAN channel.

- Parameters
  - CAN channel

    The CAN channel to send the message on.
    - CAN1
    - CAN2
    - CAN3
    - CAN4
  - Output ports

    The amount of output ports (1 – 8)
    - 1 (default)
  - Output datatype

    The datatype of the output ports
    - int8
    - uint8
    - int16
    - uint16
    - int32
    - uint32
    - boolean
  - Byte order
    - MSB first
    - LSB first
  - Sample time

    The sample time is -1 or divisible by base sample time [seconds]
- Input
  - CAN ID

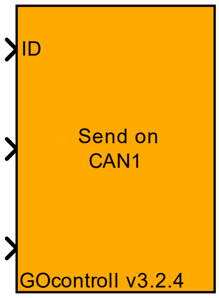
    (uint32) in hexadecimal ie: hex2dec
- Output
  - New

    Checks for new message(int8)
  - Message data
    - Byte 0
    - Byte 1
    - Byte 2
    - Byte 3
    - Byte 4
    - Byte 5
    - Byte 6
    - Byte 7

3.2.16. CAN Send block

- Function

Send a message on a specified CAN channel.



- Parameters
  - CAN channel

    The CAN channel to send the message on.
    - CAN1
    - CAN2
    - CAN3
    - CAN4
  - Identifier type
    - Normal ID
    - Extended ID
  - Input ports

    The amount of input ports (1-8)
    - 1 (default)
  - Input datatype

    The datatype of the input ports
    - int8
    - uint8
    - int16
    - uint16
    - int32
    - uint32
    - boolean
  - Byte order
    - MSB first
    - LSB first
  - Remote Transmission Request
    - On
    - Off
  - Sample time

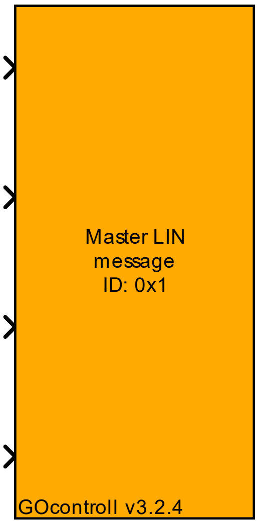
    The sample time is -1 or divisible by base sample time [seconds]
- Input
  - CAN ID

    (uint32) in hexadecimal ie: hex2dec
  - Message data
    - Byte 0
    - Byte 1
    - Byte 2
    - Byte 3
    - Byte 4
    - Byte 5
    - Byte 6
    - Byte 7

3.2.17. LIN Master block

- Function

This block implements the LIN master functionality for communication with LIN slaves. The block complies with LIN 1.3 an LIN 2.0. When sending data to a slave device, this block will add the data to the LIN bus. When retrieving data from slaves, The block will wait for incoming data over LIN.



- Parameters
  - Identifier

    Selects the identifier to communicate with.
    - 0x01 to 0x3B
    - 0x3C (diagnostic)
    - 0x3D (diagnostic)
    - 0x3E (reserved)
    - 0x3F (reserved)
  - Direction

    Request data from a slave device or send date to a slave device.
    - Request data
    - Send data
  - Datalength

    Defines the number of bytes that need to be send to- or requested from the slave.
    - 2 bytes
    - 4 bytes
    - 8 bytes
  - Checksum

    Classic for checksum based on databytes. Enhanced for databytes and ID byte.
    - Classic (LIN 1.3)
    - Enhanced (LIN 2.0)
  - Sample time

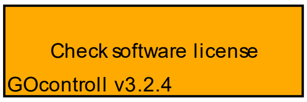
    Defines the execution interval of the block.
    - The sample time is -1 or divisible by base sample time [seconds]
- Input
  - Message data

    LIN Data bytes to be send to a slave device.
    - Byte 0
    - Byte 1
    - Byte 2
    - Byte 3
    - Byte 4
    - Byte 5
    - Byte 6
    - Byte 7
- Output
  - Message data

    LIN Data bytes received from a slave device.
    - Byte 0
    - Byte 1
    - Byte 2
    - Byte 3
    - Byte 4
    - Byte 5
    - Byte 6
    - Byte 7

3.2.18. License key block

- Function



Function block to check the controller for a license key upon executing the software. The block will decrypt the “license file” and the compare the result to the “check file”, if these match, the software will continue executing, otherwise it will exit. Because of the way this works it is a good idea to encrypt a system file that is unique to the system, for example the ethernet mac address in /sys/class/net/eth0/address.

encryption key can be 128 bits (16bytes) 192 bits (24 bytes) or 256 bits (32 bytes). It uses the OpenAES library, the tool can be used to generate a key for your software. from an earlier experience the order of the bytes of the key might have to be shifted when entered into the block so byte 1 of the key becomes byte 2 in the simulink block and byte 2 of the key becomes byte 1 of the simulink block, byte 3 becomes 4 and byte 4 becomes 3 etc:
key : 0x12 0x34 0x56 0x78...
simulink: {0x34, 0x12, 0x78, 0x56...}

- Parameters
    - Sample time

      Defines the execution interval of the block.
        - 1 (default)
    - Encryption key

      Example: An array of 16 bytes: {0x01, 0x23, 0x45, 0x67, 0x89, 0xAB, 0xCD, 0xEF, 0x01, 0x23, 0x45, 0x67, 0x89, 0xAB, 0xCD, 0xEF} Bytes can be written in any form accepted by the c compiler: 0x00/0 to 0xFf/255..
        - Key
    - Initialisation vector

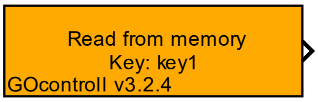
      A string of characters (max 16 bytes) example: 123kdgh34ndfg84.
        - Key
    - License file

      A string containing the path to the license file, example: /etc/simulink/license.
        - File path
    - Check file

      A string containing the path to the license file, example: /etc/simulink/check_file.
        - File path

3.2.19. Memory read block

- Function



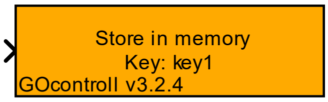
This function block reads a value from memory by the specified key. To store a value to memory, use the store to memory function block. Data type: single.

- Parameters
    - Memory read
        - Read once during startup
        - Read periodic based on sample time
    - Memory type
        - Persistent Memory (eMMC) (stored in /usr/mem-sim/)
        - Non-persistant Memory (RAM) (stored in /dev/shm/)
    - Sample time
    - Key
    - Initialization value
- Output
    - Value from memory

3.2.20. Memory write block

- Function

This function block stores a value to the memory with a specified key. To get the stored value from memory, use the read from memory function block and retrieve the value by key. Data type: single.

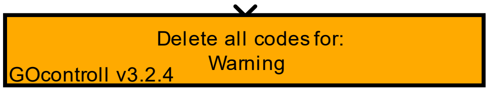


- Parameters
    - Memory type
        - Persistent Memory (eMMC) (stored in /usr/mem-sim/)
        - Non-persistant Memory (RAM) (stored in /dev/shm/)
    - Sample time
    - Key
- Input
    - Data with set key

3.2.21. Diagnostic delete all codes from one group block

- Function

This function block deletes stored codes under the specified diagnostic code type.



- Parameters
    - Sample time

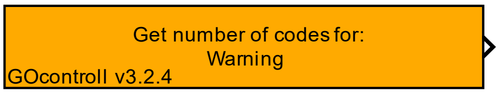
    The sample time is -1 for inherited since this block is a triggered system
    - Diagnostic code type

    The type defines the severity of the diagnostic code category.
        - Warning
        - Warning with action
        - Error
        - Error with action
        - Fatal error
- Input
    - Trigger input

3.2.22. Number of stored diagnostic codes block

- Function

This function block reads the number of stored codes under the specified diagnostic code type.



- Parameters
    - Sample time

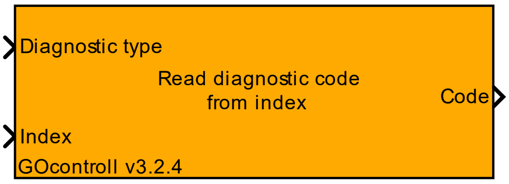
    Execution period for the function block
        - Sample time
    - Diagnostic code type

    The type defines the severity of the diagnostic code category
        - Warning
        - Warning with action
        - Error
        - Error with action
        - Fatal error
- Output
    - Number of stored diagnostic codes

3.2.23. Read diagnostic code block

- Function

This function block reads a stored diagnostic code from a specified diagnostic type. The code that is read from memory is specified by the index number on input port 2. The stored code goes out of the function block as a uint32 value.



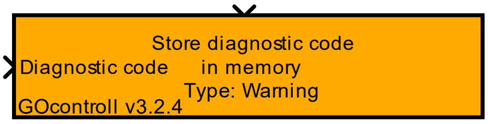
- Parameters
  - Sample time

    Execution period for the function block.
    - 0.5 (default)
- Input
  - Diagnostic type
  - Index
- Output
  - Code

3.2.24. Store diagnostic code block

- Function

This function block stores a diagnostic code to memory including maximal 10 freezed frame parameters. Once a rising edge on the trigger input is detected. The diagnostic code on input 1 and the freezed frame parameters on input 2 to 11 are stored to memory.

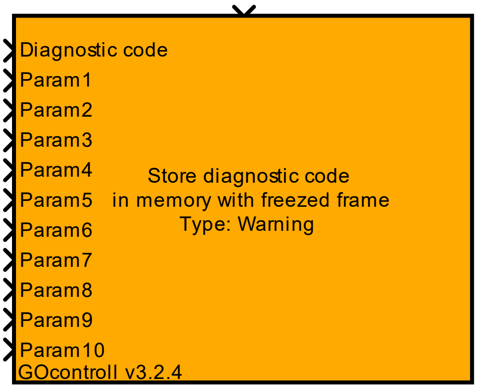


- Parameters
  - Sample time

    The sample time is -1 here for inherited since this function block is a triggered system.
  - Diagnostic code type

    The type defines the severity of the diagnostic code.
    - Warning
    - Warning with action
    - Error
    - Error with action
    - Fatal error
- Input
  - Trigger input
  - Diagnostic code

3.2.25. Store diagnostic code with freeze frame block



- Function

This function block stores a diagnostic code to memory including maximal 10 freezed frame parameters. Once a rising edge on the trigger input is detected. The diagnostic code on input 1 and the freezed frame parameters on input 2 to 11 are stored to memory.

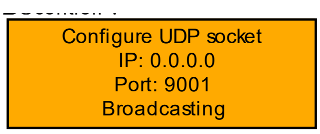
- Parameters
  - Sample time

    The sample time is -1 here for inherited since this function block is a triggered system.
  - Diagnostic code type

    The type defines the severity of the diagnostic code.
    - Warning
    - Warning with action
    - Error
    - Error with action
    - Fatal error
- Input
  - Diagnostic code
  - Freezed frame parameters.

    This describes the signal on the coresponding input. This description is stored along with the signal in the freezed frame.
    - Parameter 1
    - Parameter 2
    - Parameter 3
    - Parameter 4
    - Parameter 5
    - Parameter 6
    - Parameter 7
    - Parameter 8
    - Parameter 9
    - Parameter 10

3.2.26. Configure UDP socket block



- Function

Block to configure a UDP connection.

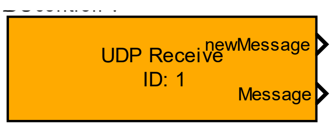
- Parameters
  - Port

    The port on which to listen
    - Ex. 9001
  - IP address

    The IP address on which to listen
    - Ex. 255.255.255.255
  - Broadcast

    Allow broadcasting of send messages
    - No
    - Yes
  - Sample time
    - Ex. 0.01

3.2.27. UDP receive block



- Function

Block to receive UDP messages.

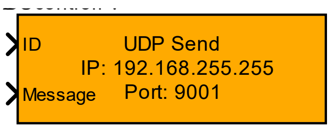
- Parameters
  - Message ID

    ID byte of the message The number of buffers allocated is the highest ID in the model + 1 so build it up from 0.
  - Buffer size
    - 1 (default)
  - Sample time
    - 0.01 (default)
- Output
  - New message flag

    Flag to tell if a new message arrived
  - Message data

    The message data in the form of [uint8;buffsize] buffer size is set in the configure UDP socket block.

3.2.28. UDP send block



- Function

Block to send UDP messages.

- Parameters
  - Port

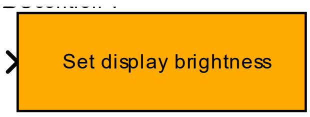
    The port to send the message to.
    - Ex. 9001
  - IP Address

    The IP address to send the message to.
    - Ex. 255.255.255.255
  - Sample time
    - Ex. 0.01
- Input
  - Message ID

    ID byte of the message The number of buffers allocated is the highest ID in the model + 1 so build it up from 0.
  - Message

    A [uint8] to send with the UDP protocol.

3.2.29. Moduline Display LCD brightness block



- Function

Set the brightness of a display (only works with the max25014 driver currently). Keeps the original brightness until the first non-zero value is received

- Parameters
  - Sample time
    - 0 (default)
- Input
  - Brightness value
    - 0-100

3.2.30. Execute shell command block

Execute command:
command
GOcontroll v3.2.4

- Function

Execute a Linux Shell command when the input is true.

- Parameters
  - Command
  - Sample time
    - 1 (default)
- Input
  - Trigger input

3.3. Compiler

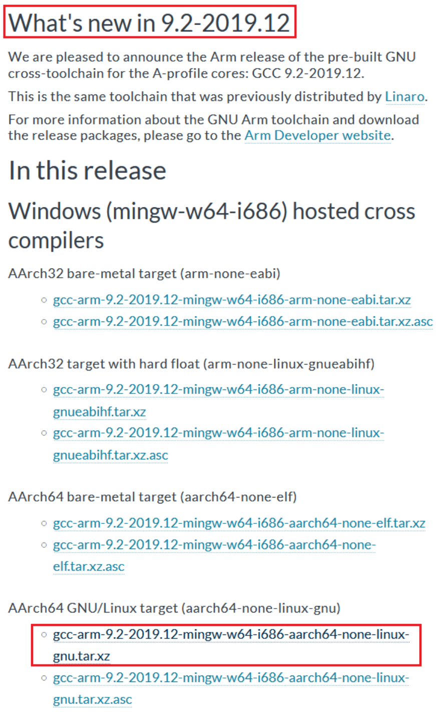
To generate C code which can be ran on the Moduline controllers, a specific compiler is required. An instruction for both Windows and Linux is provided below.

3.3.1. Windows

Start by downloading the right compiler from the developer website of ARM through the following link: https://developer.arm.com/downloads/-/gnu-a

The version that is required is **9.2-2019.12** From this version select the following file under the AArch64 GNU/Linux target (aarch64-none-linux-gnu) header:

- gcc-arm-9.2-2019.12-mingw-w64-i686-aarch64-none-linux-gnu.tar.xz



What's new in 9.2-2019.12

We are pleased to announce the Arm release of the pre-built GNU cross-toolchain for the A-profile cores: GCC 9.2-2019.12.

This is the same toolchain that was previously distributed by Linaro.

For more information about the GNU Arm toolchain and download the release packages, please go to the Arm Developer website.

In this release

Windows (mingw-w64-i686) hosted cross compilers

AArch32 bare-metal target (arm-none-eabi)

- gcc-arm-9.2-2019.12-mingw-w64-i686-arm-none-eabi.tar.xz
- gcc-arm-9.2-2019.12-mingw-w64-i686-arm-none-eabi.tar.xz.asc

AArch32 target with hard float (arm-none-linux-gnueabihf)

- gcc-arm-9.2-2019.12-mingw-w64-i686-arm-none-linux-gnueabihf.tar.xz
- gcc-arm-9.2-2019.12-mingw-w64-i686-arm-none-linux-gnueabihf.tar.xz.asc

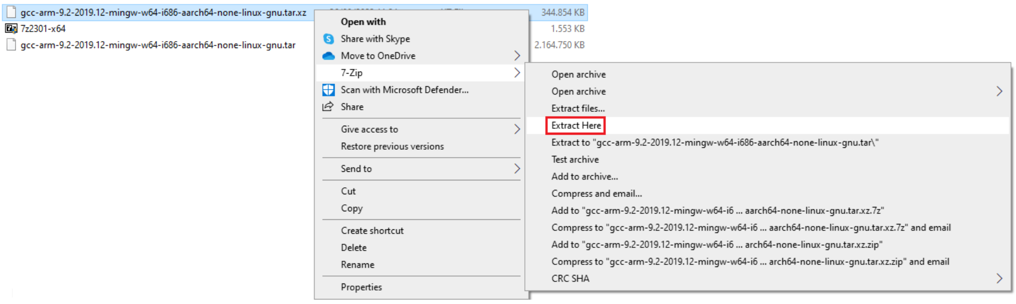
AArch64 bare-metal target (aarch64-none-elf)

- gcc-arm-9.2-2019.12-mingw-w64-i686-aarch64-none-elf.tar.xz
- gcc-arm-9.2-2019.12-mingw-w64-i686-aarch64-none-elf.tar.xz.asc

AArch64 GNU/Linux target (aarch64-none-linux-gnu)

- gcc-arm-9.2-2019.12-mingw-w64-i686-aarch64-none-linux-gnu.tar.xz
- gcc-arm-9.2-2019.12-mingw-w64-i686-aarch64-none-linux-gnu.tar.xz.asc

Windows cannot extract tar archives by default. GOcontroll advises to download 7-zip for this. This tool can be downloaded from the following website: https://7-zip.org After installing 7-zip you can right click the downloaded .tar.xz file go to the 7-zip options and select “extract here”. Then do it again for the .tar file that was generated.

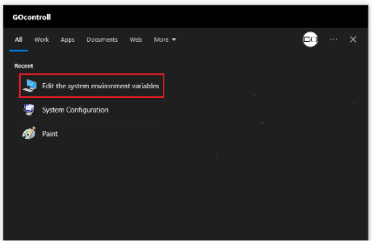


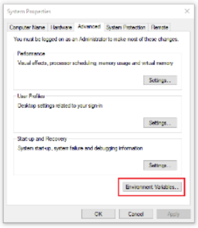
gcc-arm-9.2-2019.12-mingw-w64-i686-aarch64-none-linux-gnu.tar.xz 344.854 KB
7z2301-x64 1.553 KB
gcc-arm-9.2-2019.12-mingw-w64-i686-aarch64-none-linux-gnu.tar 2.164.750 KB

Open with
Share with Skype
Move to OneDrive
7-Zip
Scan with Microsoft Defender...
Share
Give access to
Restore previous versions
Send to
Cut
Copy
Create shortcut
Delete
Rename
Properties
Open archive
Open archive
Extract files...
Extract Here
Extract to "gcc-arm-9.2-2019.12-mingw-w64-i686-aarch64-none-linux-gnu.tar\"
Test archive
Add to archive...
Compress and email...
Add to "gcc-arm-9.2-2019.12-mingw-w64-i6 ... aarch64-none-linux-gnu.tar.xz.7z"
Compress to "gcc-arm-9.2-2019.12-mingw-w64-i6 ... aarch64-none-linux-gnu.tar.xz.7z" and email
Add to "gcc-arm-9.2-2019.12-mingw-w64-i6 ... arch64-none-linux-gnu.tar.xz.zip"
Compress to "gcc-arm-9.2-2019.12-mingw-w64-i6 ... arch64-none-linux-gnu.tar.xz.zip" and email
CRC SHA

As a quick sanity check, go into the extracted folder and then into the “bin” folder. Check if all the files have a size larger than 0. Sometimes things like virus scanners can wipe the files clean. If this is the case move the folder to its permanent location of your liking. In the case of this example it will be moved to “C:\Program Files (x86)\”

At this point we need to inform the operating system where the compiler is located. We do this by editing a system environment variable. Search system environment variables in the start menu search bar and click the option highlighted in the picture.
This will open the menu in the middle, it should already be on the “Advanced” tab but if you can’t find the “Environment Variables” button check if the right tab of the window is active.
This will open up the window on the right. We need to edit the System “Path” variable, and not the User variable. So click the highlighted Path field and then click “Edit”.
Here we take a quick detour back to the folder where you have put the compiler, enter the compilers bin folder like we did earlier. Copy the path to this folder from the navigation bar.



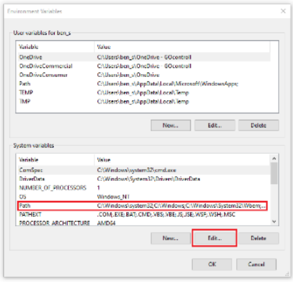

System Properties

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage and virtual memory

User Profiles
Desktop settings related to your sign-in

Start-up and Recovery
System start-up, system failure and debugging information


Environment Variables

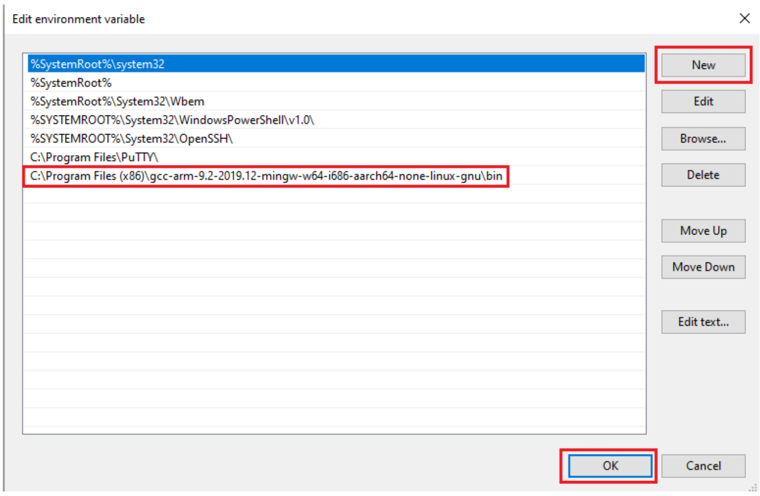
User variables for ben_s

Variable	Value
OneDrive	C:\Users\ben_s\OneDrive - GOcontroll
OneDriveCommercial	C:\Users\ben_s\OneDrive - GOcontroll
OneDriveConsumer	C:\Users\ben_s\OneDrive
Path	C:\Users\ben_s\AppData\Local\Microsoft\WindowsApps;
TEMP	C:\Users\ben_s\AppData\Local\Temp
TMP	C:\Users\ben_s\AppData\Local\Temp

System variables

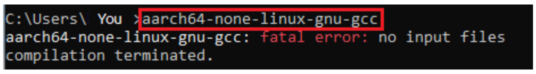
Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	1
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

Back in the “Edit environment variable” window click the “New” button and paste the path we just copied into the edit field that gets highlighted. Press enter, and now it should look somewhat like in the picture, except with your path.
Now reboot your computer to make the system load this change.


Edit environment variable

%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\PuTTY\
C:\Program Files (x86)\gcc-arm-9.2-2019.12-mingw-w64-i686-aarch64-none-linux-gnu\bin

Check if it was successful by opening Windows Command Prompt, also known as cmd, and entering the highlighted command, it should give the error in the picture as a response. If the error is anything else it was not successful.



```
C:\Users\ You >aarch64-none-linux-gnu-gcc
aarch64-none-linux-gnu-gcc: fatal error: no input files
compilation terminated.
```

If help is required in this process, please contact support@gocontroll.com

3.3.2. Linux

The only distribution validated by GOcontroll is Ubuntu 20.04. On this distribution the compiler can be set up by running the command listed below.

```
sudo apt install gcc-aarch64-linux-gnu
```

This should install version 9.3.0 of the compiler which contains the correct glibc version for the controller. The controller uses version 2.31. **Note: any newer compilers will contain a newer glibc version which the controller will not be able to handle.**

On ubuntu 22.04 for example defaults to compiler version 11.2.0 which is linked to glibc 2.35 which is too new and won't work. Even if you install gcc-9-aarch64-linux-gnu, this is 9.5.0 and contains glibc 2.34 which is also too new.

As a final resort the compiler can be manually downloaded from the following link: https://developer.arm.com/downloads/-/gnu-a Make sure to download 9.2-2019.12 aarch64-none-linux-gnu and add it to your $PATH env variable.

If help is required in this process, please contact support@gocontroll.com

3.4. Upload software model

The compiled MATLAB Simulink model can be uploaded to the controller in three ways. First of all, models can be directly uploaded during the build process from within MATLAB Simulink. Node-RED also provides the ability to upload compiled software models with the GOcontroll settings node. Lastly the model can be directly placed into the right folder with a SFTP client like FileZilla.

3.4.1. MATLAB Simulink

To generate C code from the Simulink model simply press CTRL-B on the keyboard when the model is opened or click the incremental built button in the Simulink toolbar. The model will 'grey' out during the build procedure. Once the model indicates ready in the lower left corner, the software program is built.
The compiled model can be uploaded to an network attached controller directly after this building process if needed. This can be configured in the model configuration parameters with the 'Auto upload' option. The IP address of the network attached controller can be added here too.

MATLAB Simulink also allows for manual upload, when the model doesn't need to be rebuild. This can be done by running the command listed below in the MATLAB console window.

```
ManualUpload "IP address"
```

This command can also be used if there's the need to program a controller through a different port, like when using a direct VPN connection with a controller. In this case, the port needs to be added to the command.

```
ManualUpload "IP address / url" "port"
```

3.4.2. Node-RED

The compiled C code, generated in MATLAB Simulink can also be uploaded through Node-RED. The GOcontroll Settings node provides an user friendly interface to upload the compiled c code to the controller. Simply click the upload button and drag and drop your .elf file into the upload window. To use the 'read Simulink' and 'write Simulink' nodes the matching a2l file also needs to be uploaded, this is done in the same way as uploading the model.

3.4.3. SFTP Client

Lastly, the compiled C code, generated in MATLAB Simulink can be uploaded through a SFTP client like FileZilla. In this case the model needs to be manually restarted with the "systemctl restart go-simulink" command, it is also recommended to run "go-parse-a2l" if your Node-RED flow includes simulink read/write nodes. The model should go in /usr/simulink/ and should be the only file there with a .elf extension.

4. Node-RED

4.1. Introduction

The Linux based GOcontroll Moduline embedded controllers are equipped with a preinstalled version of Node-RED. Node-RED is a well maintained, open source, graphical programming environment, allowing users to program hardware without a single line of code. By providing a web-based flow editor with an extensive library of nodes, Node-RED does not require any software to be installed on your computer or laptop.

The open source community has contributed enormously to the functionality and availability of nodes, offering well over 4000 nodes to suit all kinds of applications. To interface with the hardware of the Moduline embedded controllers, GOcontroll has developed a custom library with a broad spectrum of nodes. Without any programming knowledge or experience, users are able to build, customize, visualize and maintain their application specific control algorithms.

4.2. Basic information

Node-RED can be accessed as a web-service using a browser. A connection with the controller is mandatory and can be established using the on-board WLAN, configured as access point, or using the wired ethernet connection. Pointing to the web-service from within the browser will provide access to the development environment. The environment provides a clear, intuitive platform with on the left a node library (palette) in the center the control algorithms, build by the user, (flow) and on the right side an information and configuration overview.

The web-based service can be reached through port 1880, an example of a valid IP address is showed below.

```
192.168.1.15:1880
```

For more information about Node-RED and it’s functionalities and capabilities, have a look at the Node-RED website.

4.3. GOcontroll nodes

GOcontroll created several nodes to use directly inside the Node-RED development environment. These nodes interface the modules plugged in to the controller, and the on-board features. When dragging and dropping nodes into the environment, the user is able to configure a specific feature according user demands by just selecting options.

4.3.1. GOcontroll settings node

- Function

The settings node allows for some MATLAB Simulink model options and VPN connection settings.

  - Simulink model upload:
    Compiled Simulink models can be uploaded within the Node-RED environment. Simply click the upload button and drag and drop your .elf file into the upload window. To use the ‘read Simulink’ and ‘write Simulink’ nodes the matching a2l file needs to be uploaded, this is done in the same way as uploading the model.
  - VPN connection settings:
    The settings node can be used to upload VPN certificates to start a so called secure tunnel connection. Start: Choose when the VPN connection needs to be established.

4.3.2. Controller Active node

- Function

The GOcontroll Moduline controllers are able to stay on when the enable signal is removed. As long as the K30 power is applied and the input of the Controller Active node is high, the controller will stay on even if the K15 pins are low. If the controller is switched off by software, a restart is only possible when the device is activated by one of the K15 inputs. To keep the controller switched on, inject a 1. To shut down the controller, inject a 0.

- Input
  - msg: {“controllerActive” : value}

4.3.3. CPU Temperature node

- Function

CPU temperature measurement from i.MX8M Mini’s core. Value in °C. The temperature from the CPU is pushed out every 5 seconds as a temperature property with the temperature in degrees.

4.3.4. Status LED node

- Function

The status LED node can be used to give feedback to the end user by controlling the four LED's on top of the GOcontroll Moduline Enclosure. The LED's can be switched on and off in three different colors, red, green and blue. Select the LED you want to control. To control a LED, simply inject the desired color with the brightness. 0 to 255 corresponds to 0 to 100%.

- Parameters
  - LED
    - LED 1
    - LED 2
    - LED 3
    - LED 4
- Input
  - msg: {“color”: brightness)

4.3.5. GPS node

- Function

The onboard GPS receiver provides several GPS related values. These values are pushed out of the node in JSON object format.

NOTE: To use GPS, be sure to ENABLE the WWAN interface by command -systemctl enable go-wwan- A SIM card with data plan is not required. The optional WWAN module is required.

- Output
  - msg: {
        “latitude” : value,
        “longitude” : value,
        “altitude” : value,
        “speed” : value
    }

4.3.6. Read from memory node

- Function

The memory blocks can be used to exchange data between a running Simulink model and Node-RED. The read block needs to be configured with minimal one key value. If more key's need to be retrieved from memory, separate each key with a comma (,). When the block finds the provided key(s) in memory, the corresponding value for each key are pushed out the node in JSON object format with an interval provided in the interval drop down list.

- Parameters
  - Key(s)
  - Memory
    - Persistent memory (eMMC)
    - Non-persistent memory (RAM)
  - Type
    - Output as object with key-value pair
    - Output as numeric payload
  - Interval
    - Only on startup
    - 50 ms
    - 100 ms
    - 500 ms
    - 1 s
- Output
  - msg: {“key” : value}

4.3.7. Write to memory node

- Function

The memory blocks can be used to exchange data between a running Simulink model and Node-RED. The write block needs to be configured with a key value. When JSON objects are pushed into the node, it will search for the provided key. If the key is available in the JSON object, the corresponding value will be stored in memory.

- Parameters
  - Key
  - Memory
    - Persistent memory (eMMC)
    - Non-persistent memory (RAM)
  - Type
    - Input as object with key-value pair
    - Input as numeric payload
  - Decimals
    - 0 decimals
    - 1 decimal
    - 2 decimals
    - 3 decimals
    - 4 decimals
- Input
  - msg: {“key” : value}

4.3.8. Input Module node

- Function

This node is used to read in a wide range of input signals. During initialization, the settings for each input channel are send to the module. The signal will be converted inside the module according the given settings. Configure module location, sample time and the module input channel configuration consisting of input type and the configuration of pull up and/or pull down resistors.

- Parameters
  - Module type
    - 6 Channel input module
    - 10 Channel input module
  - Sample time
    - 50 ms
    - 100 ms
    - 200 ms
    - 1000 ms
  - Module location

    This parameter dictates the location of the installed module
    - Module slot 1
    - Module slot 2
    - Module slot 3
    - Module slot 4
    - Module slot 5
    - Module slot 6
    - Module slot 7
    - Module slot 8
  - 5V sensor supplies

    The built in 5V sensor supplies can be individually turned on and off.
    - Sensor supply 1
    - Sensor supply 2
    - Sensor supply 3
  - Function selection

    Select the desired input signal configuration.
    - Analog input - decimal (12 bit resolution)
    - Analog input - mV (1mV resolution)
    - Digital input - status (high or low)
    - Digital input - frequency (1Hz to 10 kHz)
    - Digital input - duty cycle low time (0.1% resolution)
    - Digital input - dutycycle high time (0.1% resolution)
    - Digital input - rotation speed (rpm)
    - Digital input - pulse counter (in combination with channel 2)
  - Input voltage range

    Select the expected input range of the signal.
    - 0-5 VDC
    - 0-10 VDC
    - 0-24V DC
  - Pull up resistors

    If required, a pull-up resistance can be added to the input signal.
    - 3.2 kΩ pull up resistor
    - 4.7 kΩ pull up resistor
    - 10 kΩ pull up resistor
  - Pull down resistors

    If required, a pull-down resistance can be added to the input signal.
    - 3.2 kΩ pull down resistor
    - 4.7 kΩ pull down resistor
    - 10 kΩ pull down resistor

- Outputs
  - msg: {

    ```
    "channel1key": value,
    "channel2key": value,
    "channel3key": value,
    "channel4key": value,
    "channel5key": value,
    "channel6key": value,
    }
    ```

4.3.9. Input Module reset node

- Function

This node can be used to reset a pulse counter in the input module. If an input channel from the input module is configured as a pulse counter (to read in an encoder signal e.g.), the counter value can be re-assigned. In some cases during encoder calibration it is necessary to 'zero' this counter value. Configure module location and the input channel on which the pulse signal is applied.

- Parameters
    - Module type
        - 6 Channel input module
        - 10 Channel input module
    - Module location

      This parameter dictates the location of the installed module
        - Module slot 1
        - Module slot 2
        - Module slot 3
        - Module slot 4
        - Module slot 5
        - Module slot 6
        - Module slot 7
        - Module slot 8
    - Input channel
        - Input channel 1 & 2
        - Input channel 3 & 4
        - Input channel 5 & 6
        - Input channel 7 & 8
        - Input channel 9 & 10
- Inputs
    - msg: {"pulscounterValue" : value }

4.3.10. Output Module node

- Function

The output module can be used to control resistive and inductive loads with a current up to 5 Ampère for each channel. The node will also output data with actual information from the module such as the current flow for each channel (only in high side configuration), the temperature of the module and the ground shift of the module (explained later on)

- Parameters
  - Module type
    - 6 Channel output module
    - 10 Channel output module
  - Sample time
    - 50 ms
    - 100 ms
    - 200 ms
  - Module location

    This parameter dictates the location of the installed module
    - Module slot 1
    - Module slot 2
    - Module slot 3
    - Module slot 4
    - Module slot 5
    - Module slot 6
    - Module slot 7
    - Module slot 8
  - Frequency

    In case a duty Cycle is used, select the frequency.
    - 100 Hz
    - 200 Hz
    - 500 Hz
    - 1 kHz
    - 2 kHz
    - 5 kHz
    - 10 kHz
  - Configuration

    Select the desired output configuration for each channel
    - Disable (output disabled)
    - Half Bridge dutycycle controlled (0 - 1000)
    - Low side switch dutycycle controlled (0 - 1000)
    - High side switch dutycycle controlled (0 - 1000)
    - Low side switch On - Off (0 - 1)
    - High side switch On - Off (0 - 1)
    - Peak and hold current mode (Half Bridge))
    - Frequency ouput 0 - 500 Hz (0 - 500)
  - Maximum channel current (mA)
    - 0-4000
  - Peak and hold parameters

    Select the desired output configuration for each channel
    - Peak current (mA)
    - Peak time (ms)

- Inputs
  - msg: {

    ```
        “channel1key” : value,
        “channel2key” : value,
        “channel3key” : value,
        “channel4key” : value,
        “channel5key” : value,
        “channel6key” : value,
    }
    ```

- Outputs
  - msg: {

    “channel1keyCurrent” : value,
    “channel2keyCurrent” : value,
    “channel3keyCurrent” : value,
    “channel4keyCurrent” : value,
    “channel5keyCurrent” : value,
    “channel6keyCurrent” : value,
    “moduleTemperature” : value,
    “moduleGroundshift” : value,
    “moduleStatus” : value,
    “moduleVoltage” : value,

    }

Note: the parameter moduleVoltage is only available on the 10 channel input module and the 6 channel input module version 7 and onwards.

4.3.11. Bridge Module node

- Function

  The bridge module can be used to control resistive and inductive loads with a current up to 7,5 Ampère for each channel. The node will also output data with actual information from the module such as temperature and groundshift.

- Parameters
  - Sample time
    - 50 ms
    - 100 ms
    - 200 ms
  - Module location

    This parameter dictates the location of the installed module
    - Module slot 1
    - Module slot 2
    - Module slot 3
    - Module slot 4
    - Module slot 5
    - Module slot 6
    - Module slot 7
    - Module slot 8
  - Frequency

    In case a duty Cycle is used, select the frequency.
    - 100 Hz
    - 200 Hz
    - 500 Hz
    - 1 kHz
    - 2 kHz
    - 5 kHz
    - 10 kHz
  - Configuration

    Select the desired output configuration for each channel
    - Disable (output disabled)
    - Half Bridge dutycycle controlled (0 - 1000)
    - Low side switch dutycycle controlled (0 - 1000)
    - High side switch dutycycle controlled (0 - 1000)
    - Low side switch On - Off (0 - 1)
    - High side switch On - Off (0 - 1)

- Inputs

  -
    ```
    msg: {
        “channel1key” : value,
        “channel2key” : value,
    }
    ```

- Outputs

  -
    ```
    msg: {
        “channel1keyCurrent” : value,
        “channel2keyCurrent” : value,
        “moduleTemperature” : value,
        “moduleGroundshift” : value,
        “moduleStatus” : value,
    }
    ```

4.3.12. CAN Receive node

- Function
  The CAN receive node receives a CAN message from one of the onboard CAN busses. The node will convert the messages to a JSON string with corresponding key's for the value.
- Parameters
  - CAN channel
    The CAN channel to send the message on.
    - CAN1
    - CAN2
    - CAN3
    - CAN4
  - CAN ID
    - (uint32) in hexadecimal
  - Number of signal keys
    The amount of signal keys (1 – 8)
    - 8 (default)
  - CAN data alignment
    - Signal 1 key, start byte and end byte
    - Signal 2 key, start byte and end byte
    - Signal 3 key, start byte and end byte
    - Signal 4 key, start byte and end byte
    - Signal 5 key, start byte and end byte
    - Signal 6 key, start byte and end byte
    - Signal 7 key, start byte and end byte
    - Signal 9 key, start byte and end byte
- Output
  - msg: {“signalName” : value}

4.3.13. CAN Send node

- Function
  The CAN receive node receives a CAN message from one of the onboard CAN busses. The node will convert the messages to a JSON string with corresponding key's for the value.
- Parameters
  - CAN channel
    The CAN channel to send the message on.
    - CAN1
    - CAN2
    - CAN3
    - CAN4
  - Identifier type
    - Standard identifier (11 bits)
    - Extended identifier (29 bits)
  - CAN ID
    - (uint32) in hexadecimal
  - Send trigger
    - Send on changing value
    - Send on interval
  - Number of signal keys
    The amount of signal keys (1 – 8)
    - 8 (default)
  - CAN data alignment
    - Signal 1 key, start byte and end byte
    - Signal 2 key, start byte and end byte
    - Signal 3 key, start byte and end byte
    - Signal 4 key, start byte and end byte
    - Signal 5 key, start byte and end byte
    - Signal 6 key, start byte and end byte
    - Signal 7 key, start byte and end byte
    - Signal 9 key, start byte and end byte
- Input
  - msg: {“signalName” : value}

4.3.14. Read from Simulink node

- Function

  The Read Simulink Signal block is used to read the value of globally exported signals in Simulink. Select the desired signals and sample time and it will output their values.

- Parameters
  - Sample time (ms)
  - Simulink signal
    - Signal1

      .

      .
    - Signaln
  - Validate the XCP ID of the Simulink model
    - Yes
    - No
- Output
  - msg: {

    “signalName” : value,

    “signalName” : value,

    }

4.3.15. Write to Simulink node

- Function

  The Write Simulink Parameter block is used to overwrite the value of globally exported parameter(s) in Simulink.

- Parameters
  - Output mode
    - Never
    - On input
    - Interval
  - Pre select a Simulink parameter from the list
    - Yes
    - No
  - Input key
    - Signal1

      .

      .
    - Signaln
  - Validate the XCP ID of the Simulink model
    - Yes
    - No
- Input
  - msg {“Signal” : value}

5. FileZilla

FileZilla Client is a fast and reliable cross-platform FTP, FTPS and SFTP client with lots of useful features and an intuitive graphical user interface. Besides a quick and easy connection with FTP, FTPS and SFTP servers, the tool allows for fast and reliable data transfer.

5.1. Connect to the controller

In order to exchange data, first the connection with the controller needs to be established.

5.1.1. Host

The host is the Moduline controller connected to the network. Enter the IP address of the controller in this field.

5.1.2. Username

The username is a set parameter in the controller, by default this is set to “root”. Enter the username parameter of the controller in this field.

5.1.3. Password

The password is a set parameter in the controller, by default this is set to “root”. Enter this password parameter of the controller in this field

5.1.4. Port

The port to be used is 22, which is reserved for SSH/SFTP connections.

5.2. Data exchange

Data exchange within Filezilla is extremely easy, as files can be easily accessed in a graphic folder structure.

6. HANTune

HANtune is a user friendly graphical tuning and calibration application developed by HAN Automotive. HANtune is using the XCP protocol, which allows for real time monitoring and tuning of parameters in the MATLAB Simulink environment. The GOcontroll Moduline series Embedded controllers are compatible with XCP over Ethernet.

When starting HANtune for the first time, a blank project is presented. The steps which need to be taken in order to get up-and-running are as follows:

- Start HANtune
- Load the .a2l file (automatically generated while building your application in MATLAB Simulink)
- Select the right communication medium (Ethernet)
- Drag and drop the desired parameters and signals into your free HANtune tab space, select desired editor/viewer to represent the parameters/signals respectively.
- Click Connect (or F5)
- Start tuning your control algorithm!

More information about HANTune can be found on the OpenMBD website.