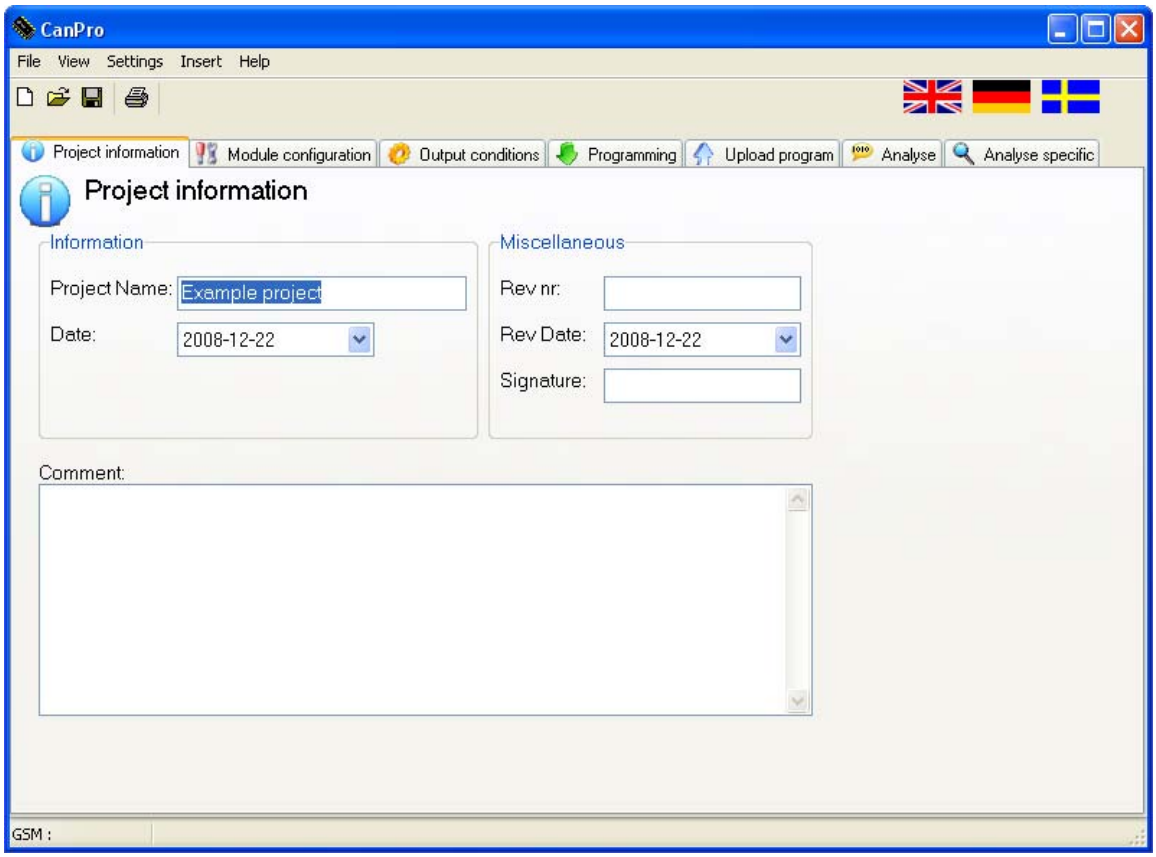
Index

Project Information



Project name: Name of the project

Date: When the project was created.

Rev no: Revision no

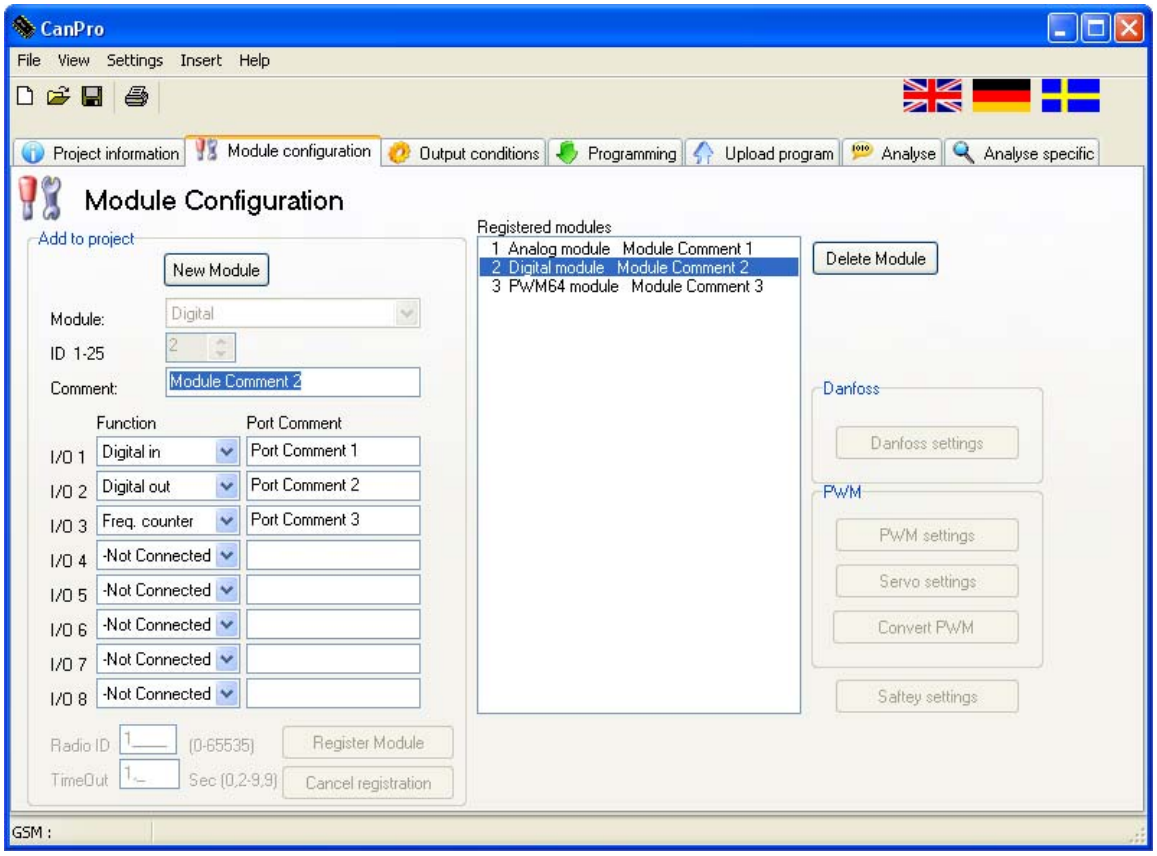
Rev. Date: Date of the last revision.

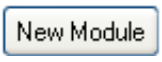
Signature: Signature of the person who last revised the project.

Comment: Possibility to save some comments about the project.

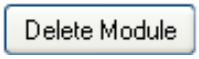
Flags: Choose the current language to use.

1 Module Configuration

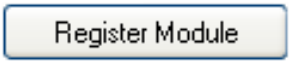




Add a new module (hardware representation) to the current project. Each module must have a ID to identify it on the bus. The lowest free ID will automatically be assigned, if not changed.



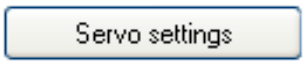
Delete the currently selected module from the project. If the module is used in the programming logic, a warning will appear before the module is deleted.



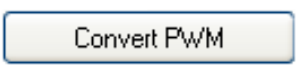
When a new module has been added and the different ports have been configured, this button will add the module to the project. The registered module will now appear in the “registered modules” list.

Show settings window for PWM-ports on the selected module.

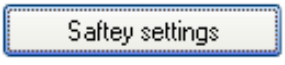
Show settings window for Danfoss-ports on the selected module.



Shows setting window for the “servo loops” on the selected module.



Convert between different PWM-hardware. (used for legacy hardware)



Show settings window for safety features on the safety module.

Module: Choose what type of hardware you want to add to the project

Digital module	Analogue module	PWM-64 module
Radio module	PWM-module	Text module
Servo module		

ID: 

Sets the ID to be used on the hardware module. (1-25)

Comment: A short naming of the module.

Function: The selected module has a number of ports that has to be enabled.

The different options follow:

Digital module:
Digital in
Digital out
Frequency counter
- Not connected -

Analogue module:
Analogue in
Analogue out
- Not connected -

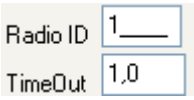
Radio module:
Digital in
Analogue in
- Not connected -

PWM-module:
PWM-out,
- not connected -

Text-module:
Text out
- Not connected -

Servo module:
Servo out
- Not connected –

Port comment: Each port on a module has a short description field.
E.g. Button1 or LED1



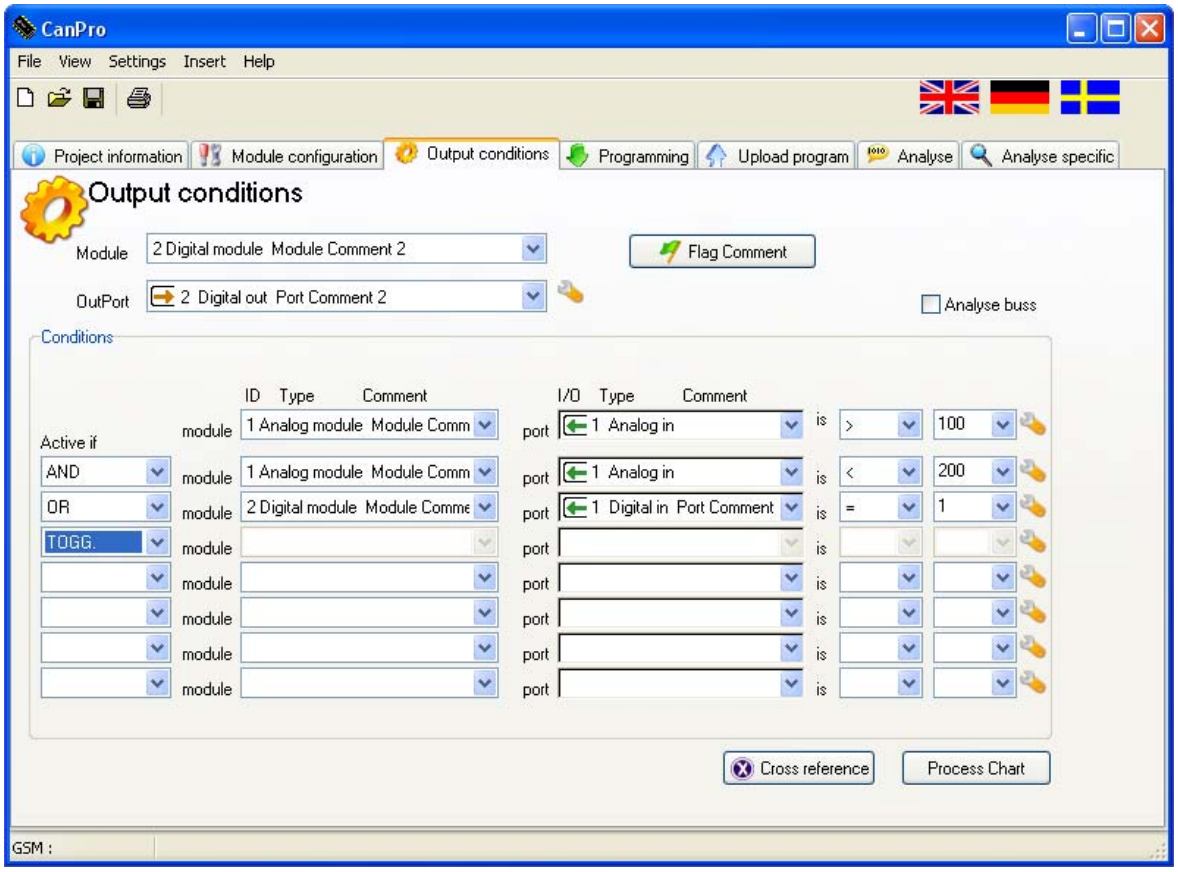
If a radio module is used, then you can enter a specific radio code to be used (1-65535). This same code must be used on both the receiver and the sender.

The radio timeout configures for how long a radio signal can be lost without the receiver to enter a TimeOut state. When the receiver enters the time out state, it will no longer send its last known value from the sender, but instead set all values to 0. This can than be detected by in the programming logic.

Registered modules:

Registered module list shows all modules in the project. The modules are ordered by ID. To edit the settings of a module, the select the module from the list and the corresponding values will show.

2 Output conditions



Module: Select what module you want to edit

Out port: Select on witch port on the selected module you want to program.

Conditions: The eight rows are used to set the programming logic for the selected port. The programming logic can be built up by using I/Os, timers or by using sub-routines called Flags.

2.1 Logical Operators

AND

Logical AND operator.

OR

Logical OR operator.
If a list of conditions shall be used with an OR operator, then write the conditions in a FLAG and then use OR on the flag.

SET

Logical SET operator.
Sets a value to the port (0-255).

FOLLOW

Logical FOLLOW operator.
The current port will follow another port.

LIMIT

Logical LIMIT operator. The current port will follow another port, but with the boundary checks. The operators <, > *is used to limit the ports values to a constant value.*
To be able to limit the ports out value to a variable limit (e.g. if the limit value is set by a analogue input) use <I/O, >I/O.

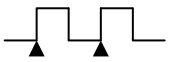
OR FOLLOW

Logical ORFOLLOW operator.
An extension of the operator FOLLOW. This operator allows for multiple ports to follow, but only one at the time. The ORFOLLOW that has a value that differs the most from the normal center value is the one to be used. (The modules PWM, PWM64 and Danfoss has a center value/default value that can be changed. All other modules have a centre value/default value = 0)

TIMER

Timer 1 has a 0.1 sec resolution and can range from 0.0 - 25.5 seconds. Timer 2 has a 1 sec resolution and can range from 0 - 255 seconds. Puls timer has a range from 0.2-25.5 seconds and a constant duty cycle of 50%.

TOGG.



Logical TOGG. Operator.
Toggle/changes between True and False, each time the value is true.
Default value is False.

FOLLOW+

Logical FOLLOW+ operator.
The current port will be assigned the value from another port, and added a constant value.
Some module support FOLLO + I/O (see datasheet for support)

FOLLOW-

Logical FOLLOW- operator.
The current port will be assigned the value from another port, and subtracted a constant value.
Some module support FOLLO - I/O (see datasheet for support)

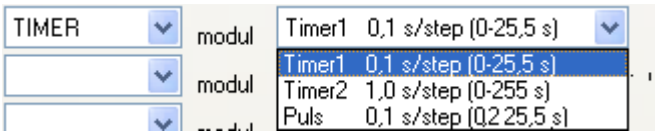
FOLLOW.I

Logical FOLLOW INVERTED operator.
Same as FOLLOW, but inverted. (255 – value to follow)

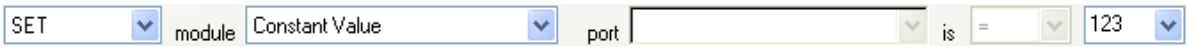
Logical LIMIT INVERTED operator.
Same as the limit function, but.(255 - value to follow/limit)

In column two in the condition group box you specify what module that should be in the condition.

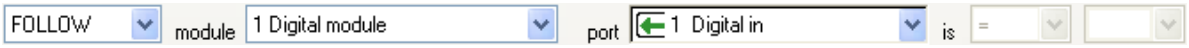
If the operator TIMER is used you can choose what type of timer to be used.



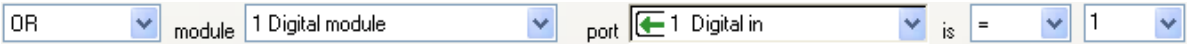
If the operator SET is used some of the following dropdown boxes will be disabled.



If operator is FOLLOW the two last boxes in this row will be disabled.



OP 1 MODULE / TIMER PORT / SUBROUTINES OP2 OP3



Port Select what port to be used in the current condition.
Both in-ports and out-ports can be used. It's is also possible to use subroutines in the condition.

Operator2 (OP2)

= > < <> Comparison operators, where the later operand (OP3) is a constant.

=I/O >I/O <I/O <>I/O. Comparison operators, where the later operand (OP3) is an I/O.

Operand3 (OP3)

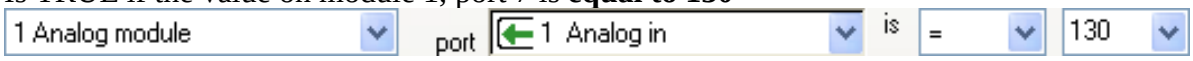
Define the constant value or the I/O to be used in the logical condition line.
If a constant value is used, the value can range from 0 to 255. If operator 2(OP2) Uses a I/O comparison, then the the I/O is defined as follows in the box (OP3).

Example 1 OP3: **25** corresponds to ID **2** port **5**
Example 2 OP3: **236** corresponds to ID **23** port **6**

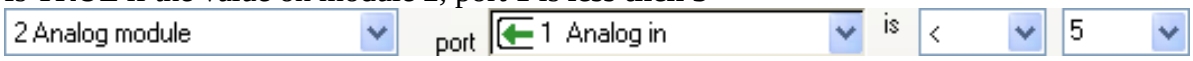
Operator 2 (OP2)

The following examples are TRUE if:

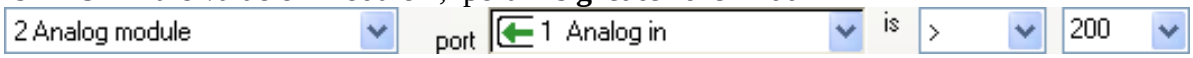
Is TRUE if the value on module 1, port 7 is **equal to 130**



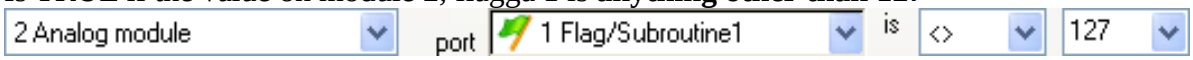
Is TRUE if the value on module 2, port 1 is **less then 5**



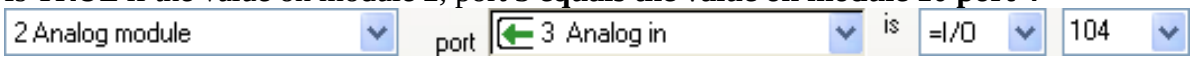
Is TRUE if the value on module 2, port 1 is **greater then 200**



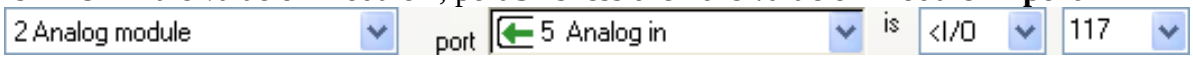
Is TRUE if the value on module 2, flagga 1 is **anything other than 127**



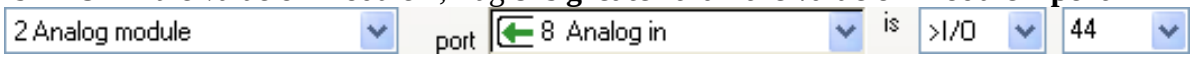
Is TRUE if the value on module 2, port 3 **equals the value on module 10 port 4**



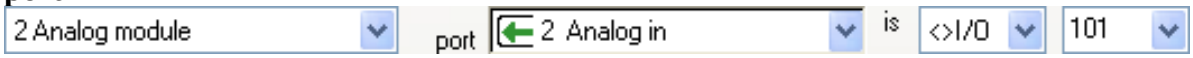
Is TRUE if the value on module 2, port 5 **is less then the value on module 11 port 7**



Is TRUE if the value on module 2, flag 8 **is greater than the value on module 4 port 4**



Is TRUE if the value on module 2, port 2 **is anything other than the value on module 10 port 1**



2.2 Editing condition list

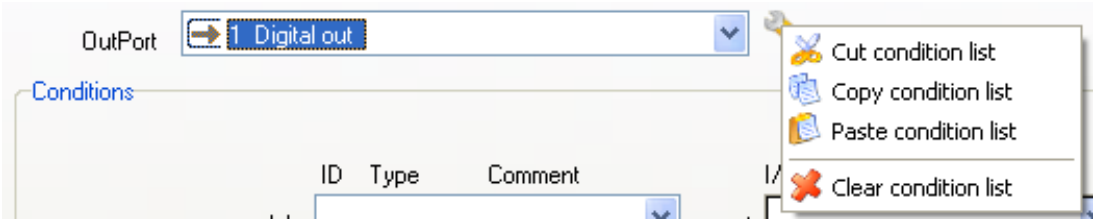
By right clicking on the spanner icon at the right of the “Out port box”, a menu will show you the following:

Cut condition list
This will cut out all the conditions shown, and place it in the memory.

Copy condition list
Copy all conditions shown, and place them in the memory.

Paste condition list
Paste the condition list currently saved in the memory.
Be aware that if a condition uses flags, the flag on one module do not correspond to flags on other modules. All flags are local to the module and you will get a warning.

Clear condition list
Clear all conditions on the selected port.



2.3 Editing specific conditions in the list

By right clicking on the spanner icon at the right of the condition row, a menu will show you the following:

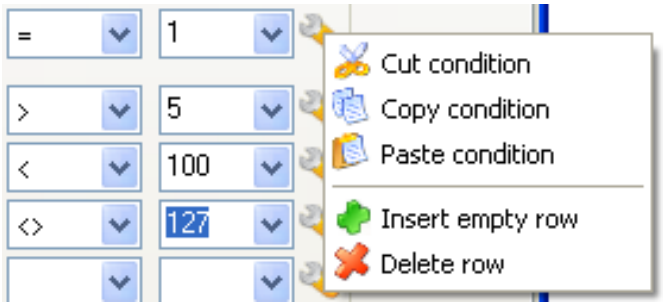
Cut condition
This will cut out all the conditions shown, and place it in the memory.

Copy condition
Copy all conditions shown, and place them in the memory.

Paste condition
Pastes the conditions list currently saved in the memory.
Be aware that if a condition uses flags, the flag on one module do not correspond to flags on other modules. All flags are local to the module and you will get a warning.

Insert empty row
This will insert an empty row into the condition list.

Delete row
Clear all conditions on the selected port.



3 Diagram

When programming in *CanPro* you can use a process chart over the program logic. This can make it easier to get an overview of the functions. It's also possible to print the process chart to a printer.
Here is an example of a program and the diagram connected to this condition list.

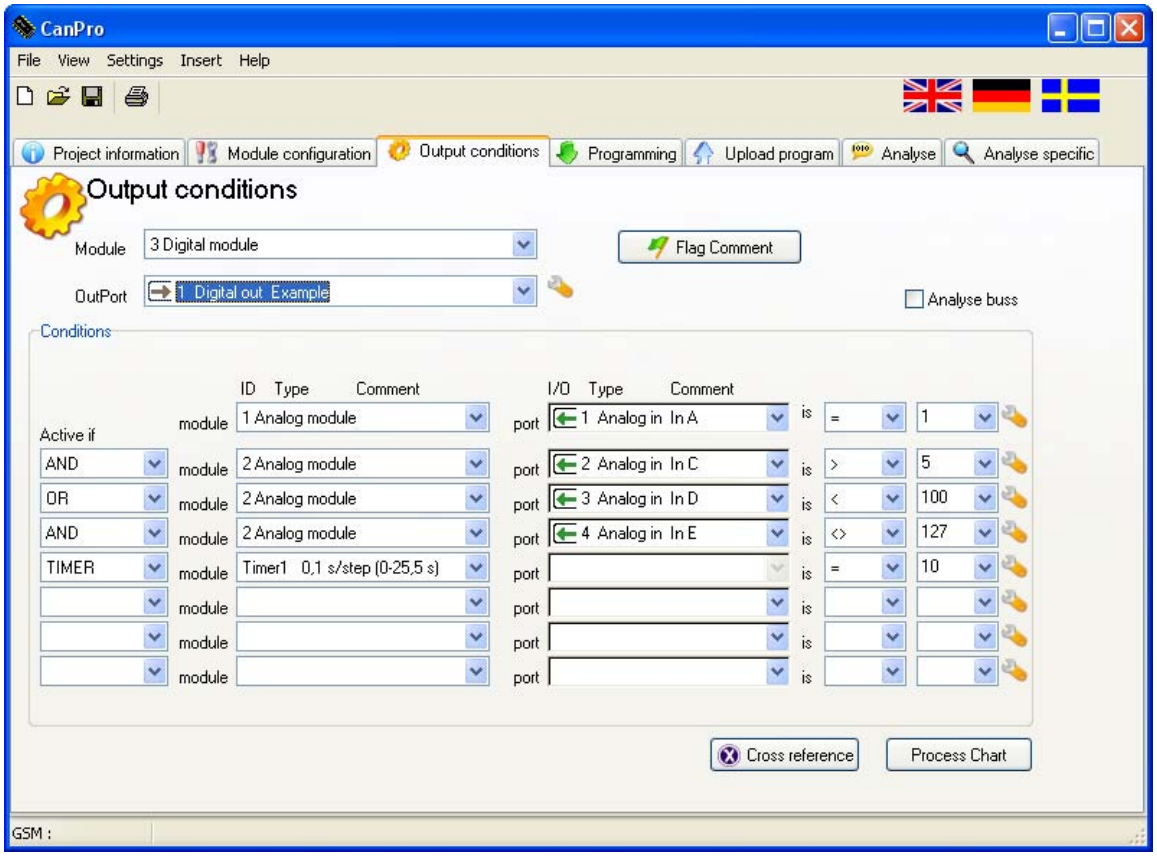


Diagram without the checkbox *show comments* disabled.

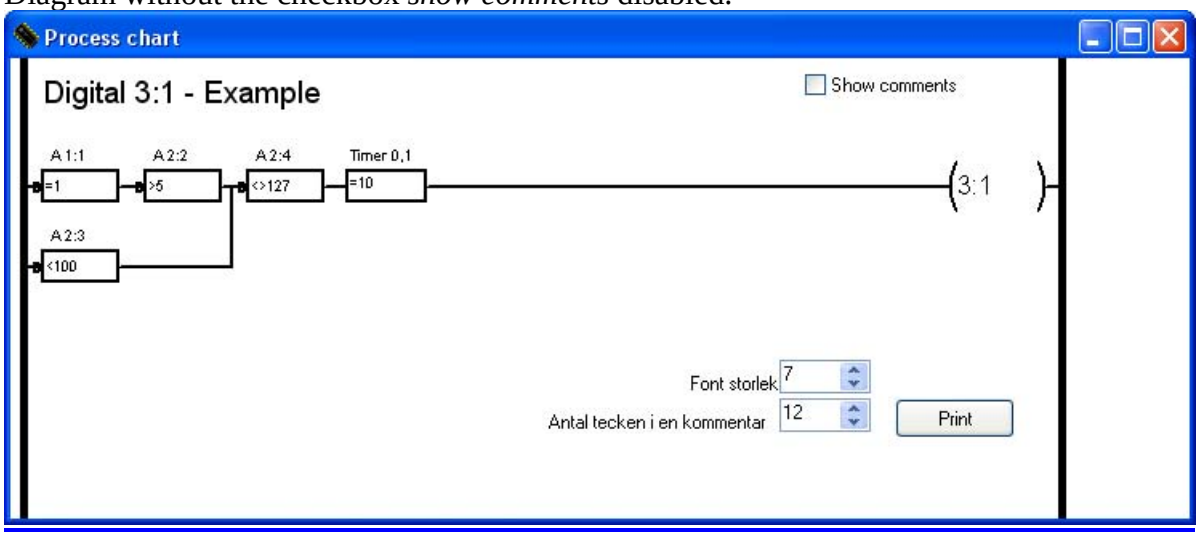
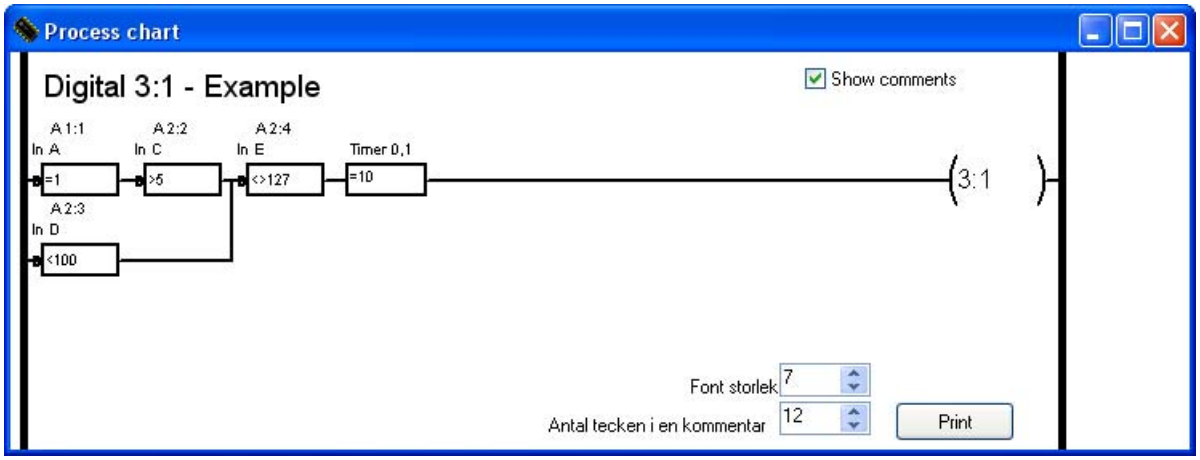


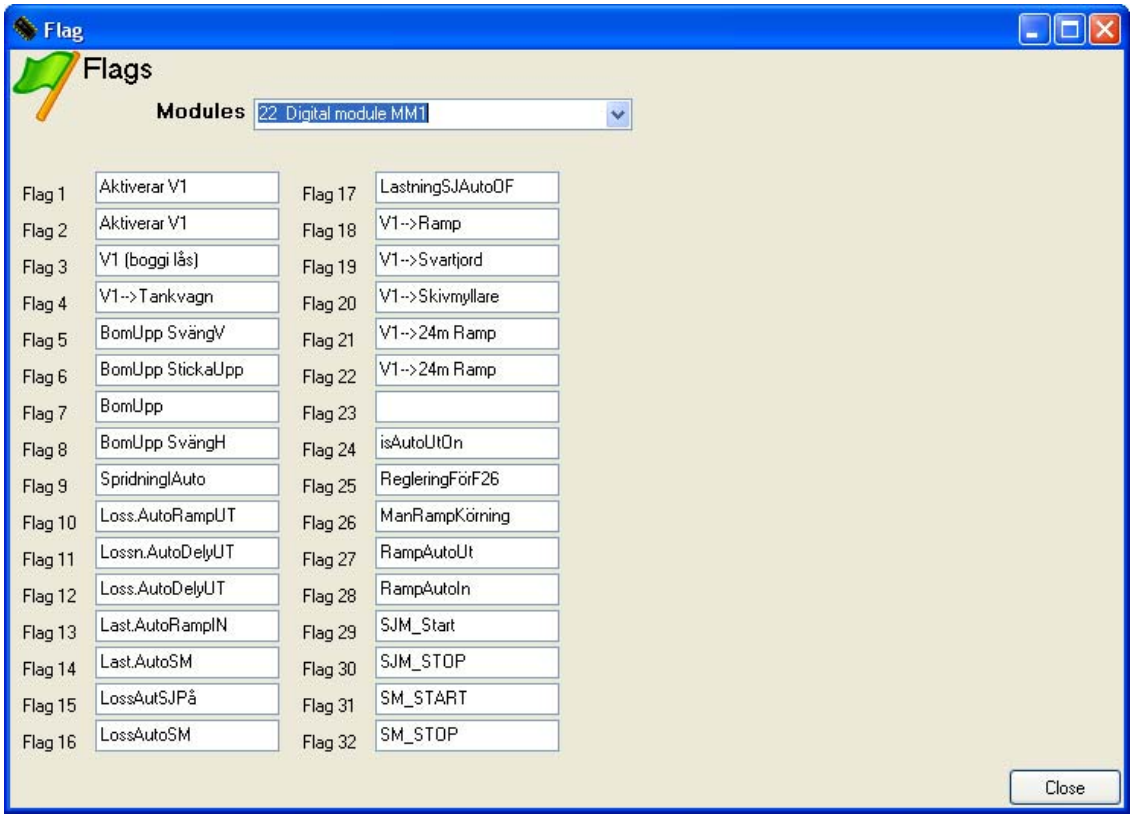
Diagram with the checkbox *show* comments enabled.
The fonts and text length can be manipulated with the two boxes, *font size* and *max no of char in a comment*.



4 Flags / Subroutines

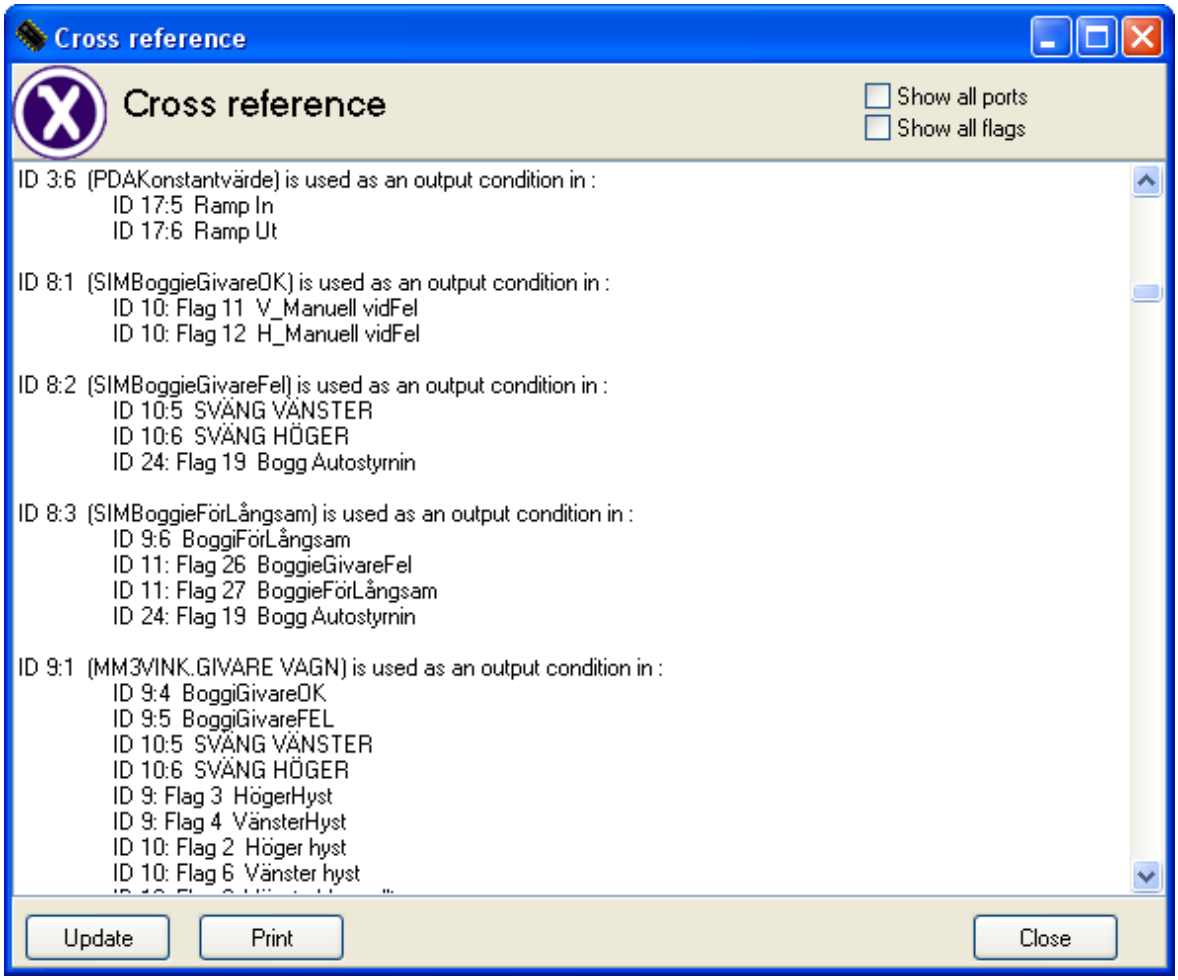
By first selecting a module, you can get an overview of all flags/subroutines in a module. You can then edit the name of the different flags/subroutines.

All flags/subroutines are local to the module, and the information in a flag is not shared over the CAN-bus to other modules.



5 Cross reference

The cross reference window is a good tool, when you want to know where in the program a specific I/O is used.



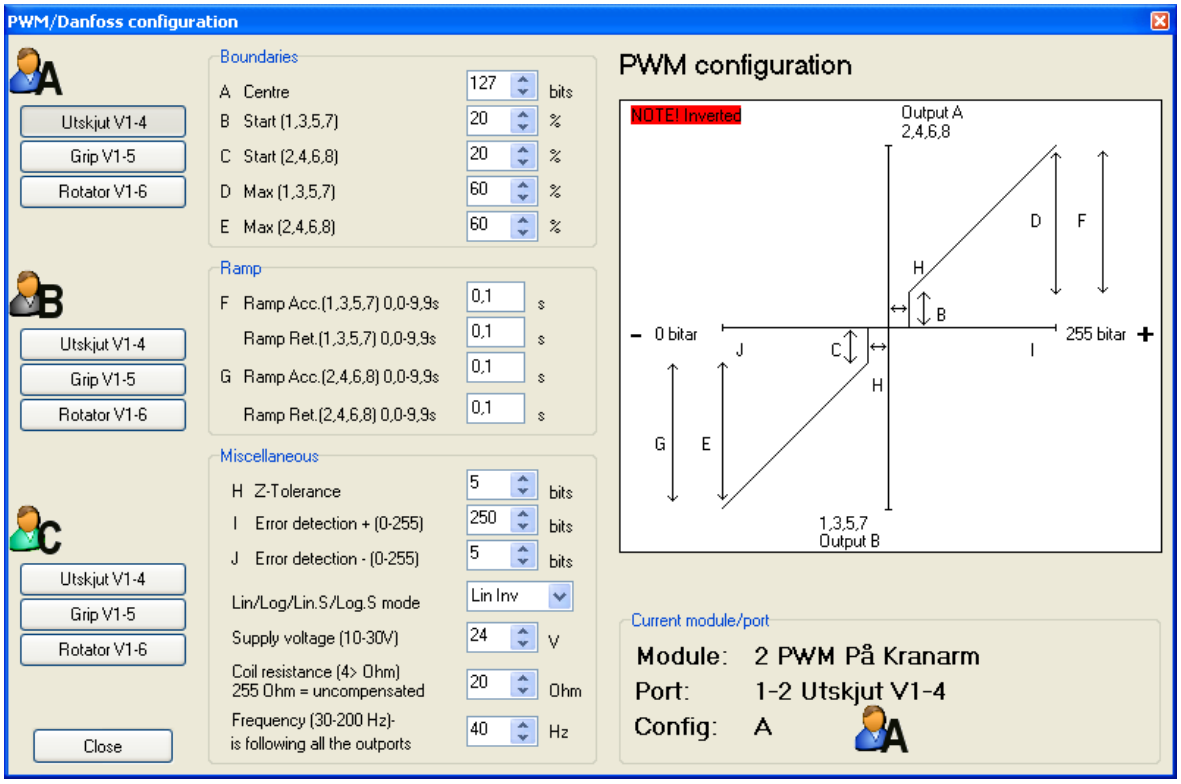
By checking the box *Show all ports*, all ports will show in the list. Not only referenced ports but also ports that are not used in other ports/flags will show.

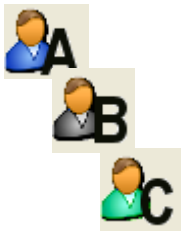
By checking the box *Show all flags*, all flags will show in the list. Not only referenced flags but also flags that are not used in other ports/flags will show.

Update — This button will update the list. This could be useful if the window is open at the same time as the program is being written.

Print — Print the information listed in the window.

6 PWM/Danfoss Configuration

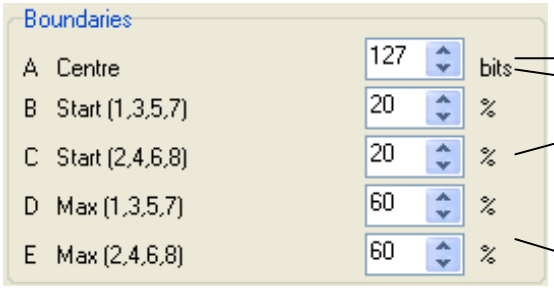




3 different user settings can be programmed. For example if different users wants different characteristics on the machine. It can also be used for different moods different modes, where you have a fast, medium and slow mode.

If flag 30 is true on the current module, then setting A is used.
If flag 31 is true on the current module, then setting B is used.
If flag 32 is true on the current module, then setting C is used.

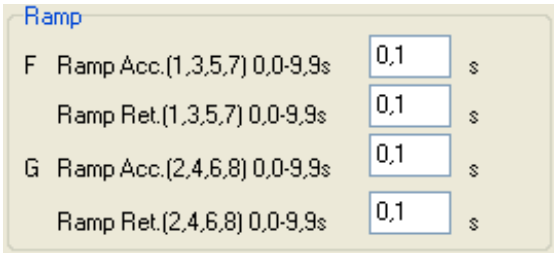
With the buttons to the left, you can choose what port to configure.



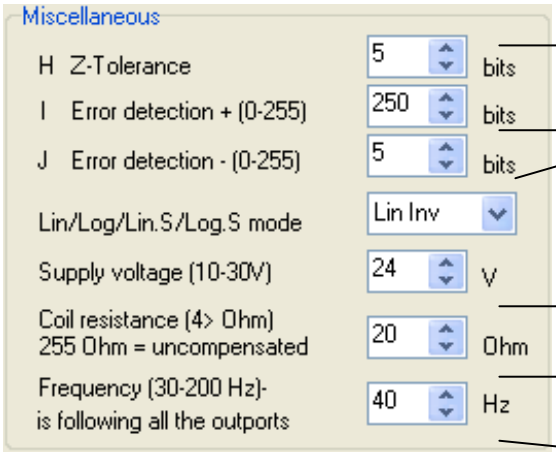
In signal centre value (e.g. Joystick center)

Start current for proportional valves (min flow)

Max current for proportional valves (max flow)



Separate acceleration and “negative acceleration” settings in both directions. Range 0.0 - 9.9 seconds.



Dead band on e.g. a joystick.

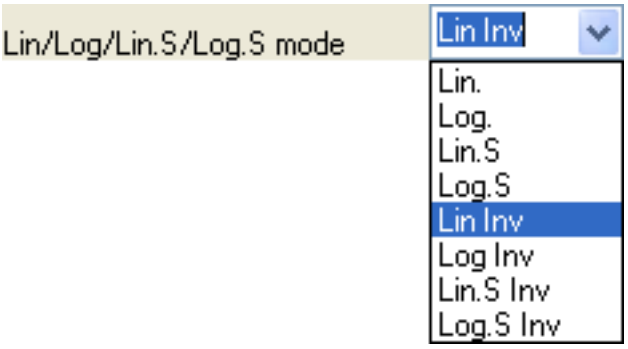
Defines what is a valid value. This is used to detect cable faults.

Characteristics of the output signal.

The systems nominal supply voltage.

Nominal resistance in the proportional valve

Ripple frequency (Used on all ports on the module)



Lin / Log / Lin.s / Log.s / Lin Inv / Log Inv / Lin.s Inv / Log.s Inv
Logarithmic or linier scale, dual or single proportional, inverted signal.
The image to the right in the program will illustrate the different signal characteristics.

Lin	Linier output. Two valves for one section.
Log	Logarithmic output. Two valves for one section. More precise precision at lower speed, and faster response at higher speed.
Lin.s	Linear output. One valve with a directional valve.
Log.s	Logarithmic output. One valve with a directional valve. More precise precision at lower speed, and faster response at higher speed.

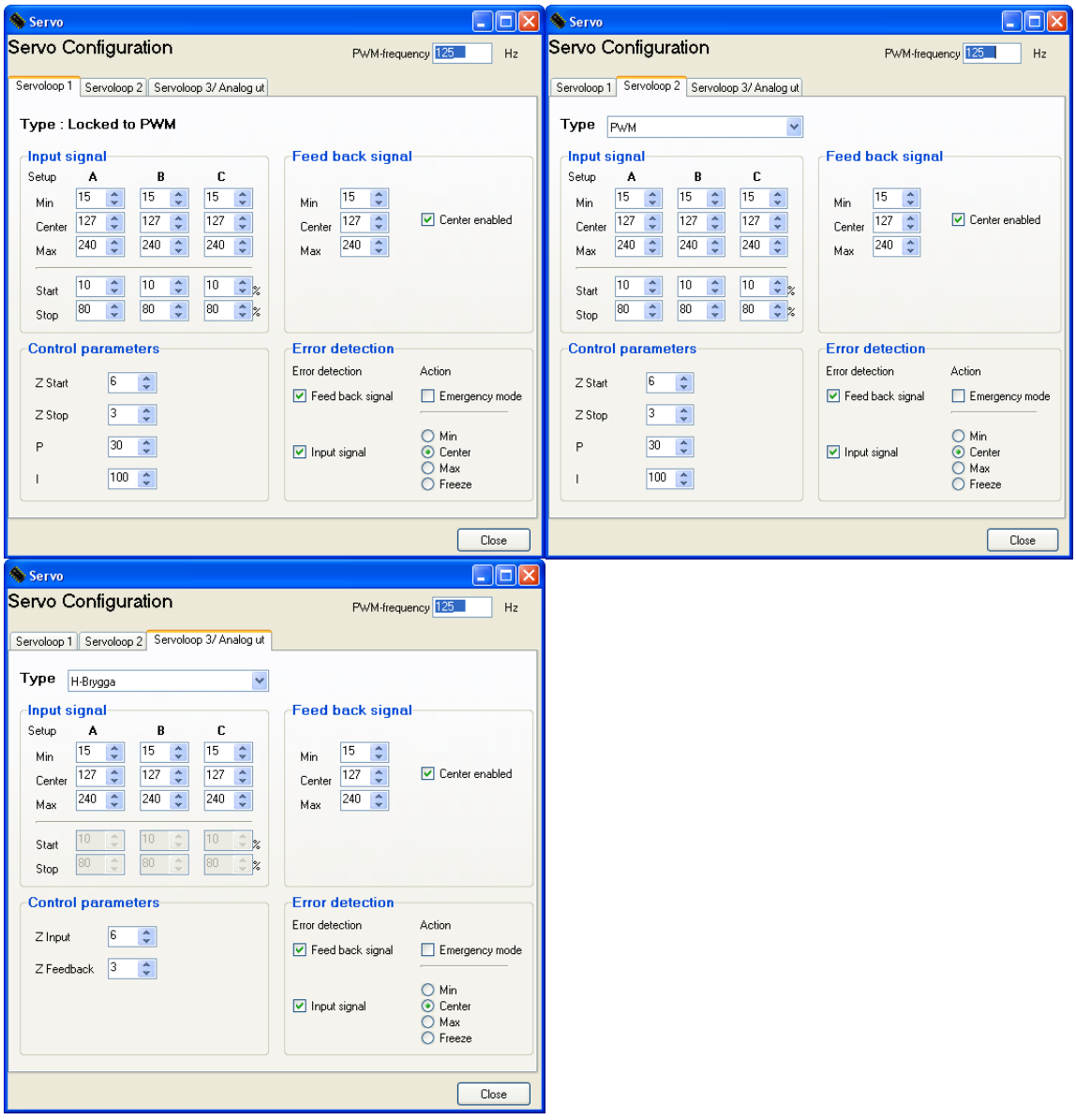
Same characteristics as above, but the input signal is inverted. This is useful for example if for example right and left should change when the driver turns his seat around in the vehicle. The invert function mirrors the in signal so therefore it is very important that the centre value is nearly 127bit.

Lin Inv	Linier output. Two valves for one section. Inverted signal.
Log Inv	Logarithmic output. Two valves for one section. More precise precision at lower speed, and faster response at higher speed. Inverted signal.
Lin.s Inv	Linear output. One valve with a directional valve. Inverted signal.
Log.s Inv	Logarithmic output. One valve with a directional valve. More precise precision at lower speed, and faster response at higher speed. Inverted signal.

7 Servo settings:

A servo module consists of two feedback loops, with an additional out signal.
The first servo loop uses PWM, whiles the other loop can choose between PWM, H-Bridge(actuator).
The third out port can be used in one of the following:

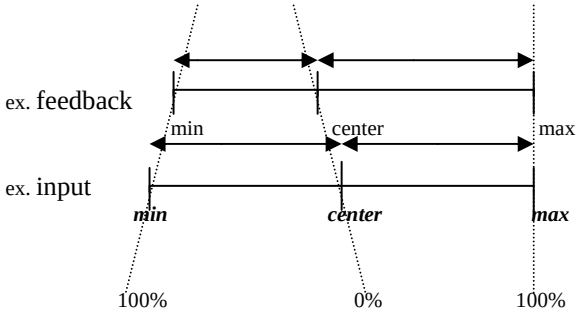
- PWM: 0-100%
- Voltage: 0-5V
- Current: 4-20mA
- PWM@533Hz: 8-92% duty cycle



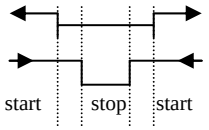
7.1 Settings – Servoloop1 and Servoloop2

Start, Max
The PWM output signal is 0-100% when not reduced.
(To limit an actuators working span/length, use *feed back signal min*, and *feedback signal max*)

Min, Center, Max for both input signals and feedback signals
Configure the working ranged for the input signal and feedback signal using *max* and *min*. If a center position exits, enable the checkbox *center enable*. For example, a steering wheel has a center, but a accelerator pedal does not have a center. If *min* is larger than *max* the function will be inverted.



Z, Start, Stopp
Z-Start sets how big the difference between the input signal and the feedback signal is when a movement starts. If the difference is smaller then Z-start, nothing will happen.
Z-Stop set how big the difference between the input signal and the feedback signal should be before stopping the current movement.
If the system seams nervous, and never find a idle position, the *Z-start*/*Z-stop* parameters can be changed to get the system to stop within a correct value. The max current for the actuator/valve have also relevance to these settings.
The *Z-Start* should be greater or equal to *Z-Stop*.



P,I
These parameters is only used with PWM.
P is the amplification of the error offset.
I is how fast the movement should accelerate if the error/offset remains constant.
As long as the error/offset is present, I will be added to the movement, each 25 ms. The longer a error/offset is present, the actuator/valve will becom stronger/faster to correct the error/offset.
The resolution of the PWM signal is 460800/PWM-frequency
At 125 Hz the resolution equals 3686 steps.

Example **P**. At 20 bits offset the correction should be at full power. 3686/20 = **184**.
Example **I**. The acceleration should go from 0 to 100% in 1 second. I= (3686*0,025)/1 = **92**

Error detection input:
Error detection can be individually activated for both the input signal and the feedback signal. An error is detected when a value goes outside a given boundary. This boundary is defined as 10 bits under *min*, and 10 bits over *max*.
If the *min* value is smaller then 10 the erroneous value is set to 0.
If the *max* value is set to 245 or larger, the erroneous value is set to 255.

If an error is detected on the input signal, it's possible to choose how the output signal will behave.
The different modes are *min*, *max*, *center*, or *freeze*. This action will also start if the CAN-communication goes down.

Error detection feedback:
When an error is detected on the feedback signal (H-bridge or PWM) it's possible to enter an emergency mode. In emergency mode the 30 bits closest to max and the 30 bits closest to min will activate the output in that direction, for at most 20 seconds at a time.

Example: The feedback signal is missing on a boat with a steering wheel controlling a rudder. To steer right in emergency mode, the steering wheel must be turned to the 30 last bits at the right direction. This will activate the rudder to steer right as long as you keep the steering wheel in this position, but as most for 20 seconds. When the steering wheel is turned against the center the rudder will stop turning and remain at its current position.

PWM-Frequency
Shared PWM frequency on both servoloop1 and servoloop2.

7.2 Settings – Servoloop 3 / Analog ut

The following output input signals can be used on

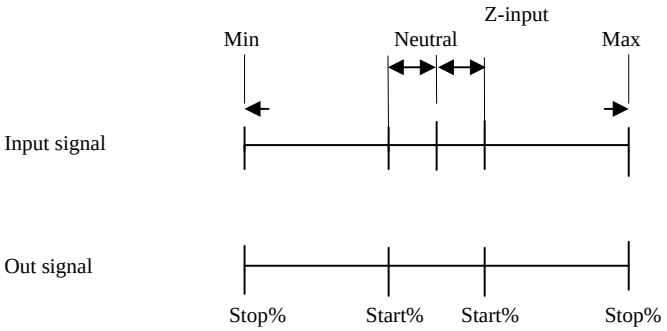
- H-bridge
- Voltage: 0-5V
- Current: 4-20mA
- PWM@533Hz: 8-92% duty cycle

Min, Center, Max for both input signals and feedback signals
Configure the working ranged for the input signal. If H-bridge is used the feed back signal is configured with min and max. If a center position exits, enable the checkbox *center enable*.

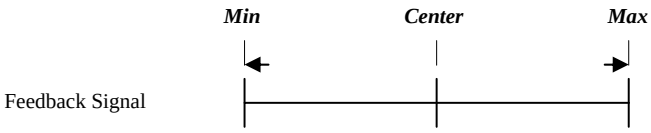
For example, a steering wheel has a center, but a accelerator pedal does not have a center. If *min* is larger than *max* the function will be inverted.

Z input is the deviation in bits from the neutral bit value before more throttle is given than the idle throttle.

Z-Feedback configures how sensitive the regulator is (H-bridge).
(To low value can give a nervous system, to high can give a late start on the regulator)



When (0-5V), (4-20mA), (PWM@533Hz) is selected:
The lowest value present on the port is adjustable with start%.
The highest value present on the port is adjustable with stop%.



When H-bridge is used the feedback signals bounders is configured under feedback signal.

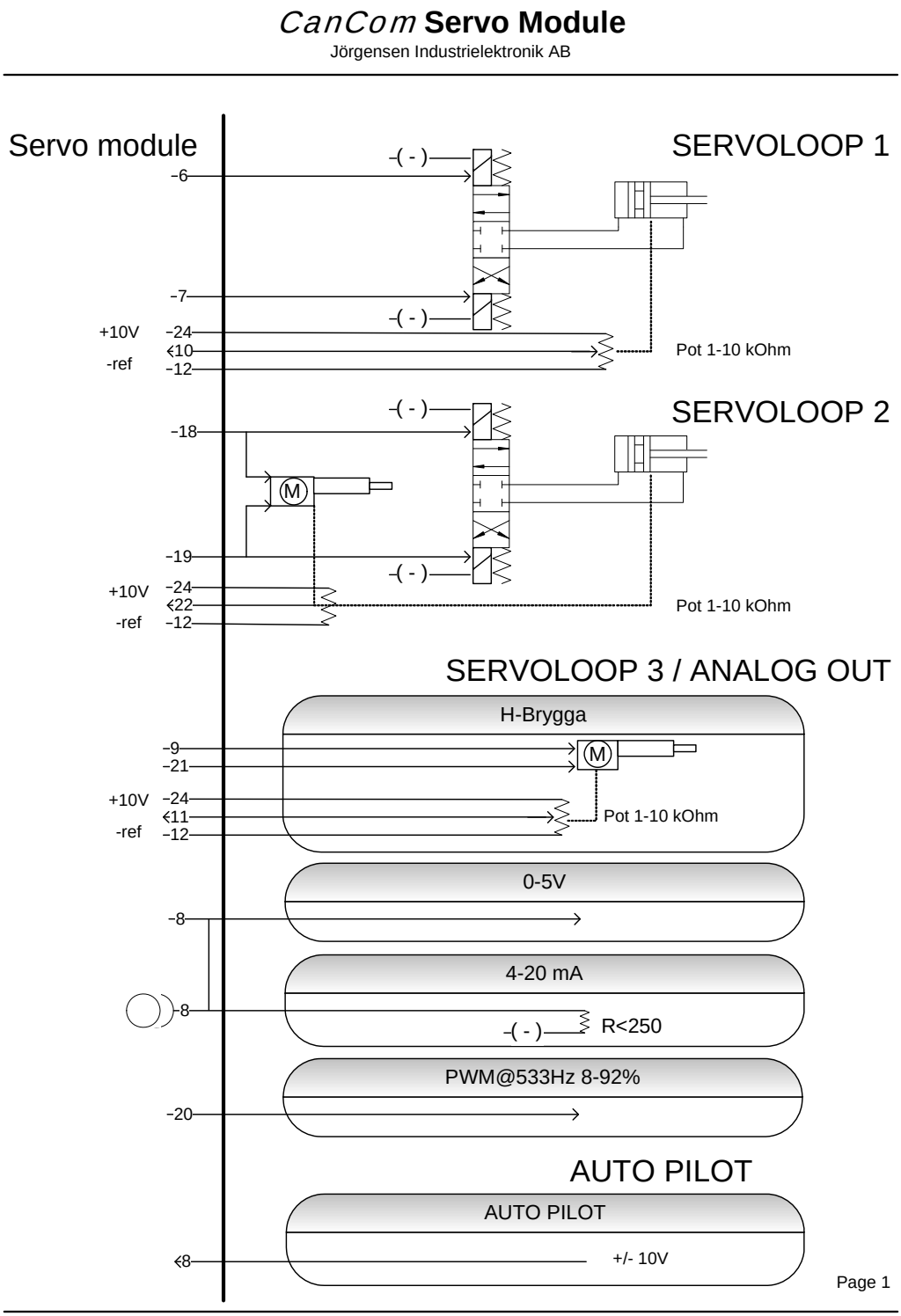
Error detection input:
Error detection can be individually activated for both the input signal and the feedback signal. An error is detected when a value goes outside a given boundary. This boundary is defined as 10 bits under *min*, and 10 bits over *max*.
If the *min* value is smaller then 10 the erroneous value is set to 0.
If the *max* value is set to 245 or larger, the erroneous value is set to 255.

If an error is detected on the input signal, it's possible to choose how the output signal will behave.
The different modes are *min*, *max*, *center*, or *freeze*. This action will also start if the CAN-communication goes down.

Error detection feedback:
When an error is detected on the feedback signal (H-bridge or PWM) it's possible to enter an emergency mode. In emergency mode the 30 bits closest to max and the30 bits closest to min for the input signal will activate the output in that direction as on/off, for at most 20 seconds at the time.

Example: The feedback signal is missing on a boat with a steering wheel controlling a rudder. To steer right in emergency mode, the steering wheel must be turned to the 30 last bits at the right direction. This will activate the rudder to steer right as long as you keep the steering wheel in this position, but as most for 20 seconds. When the steering wheel is turned against the center the rudder will stop turning and remain at its current position.

7.3 Connections

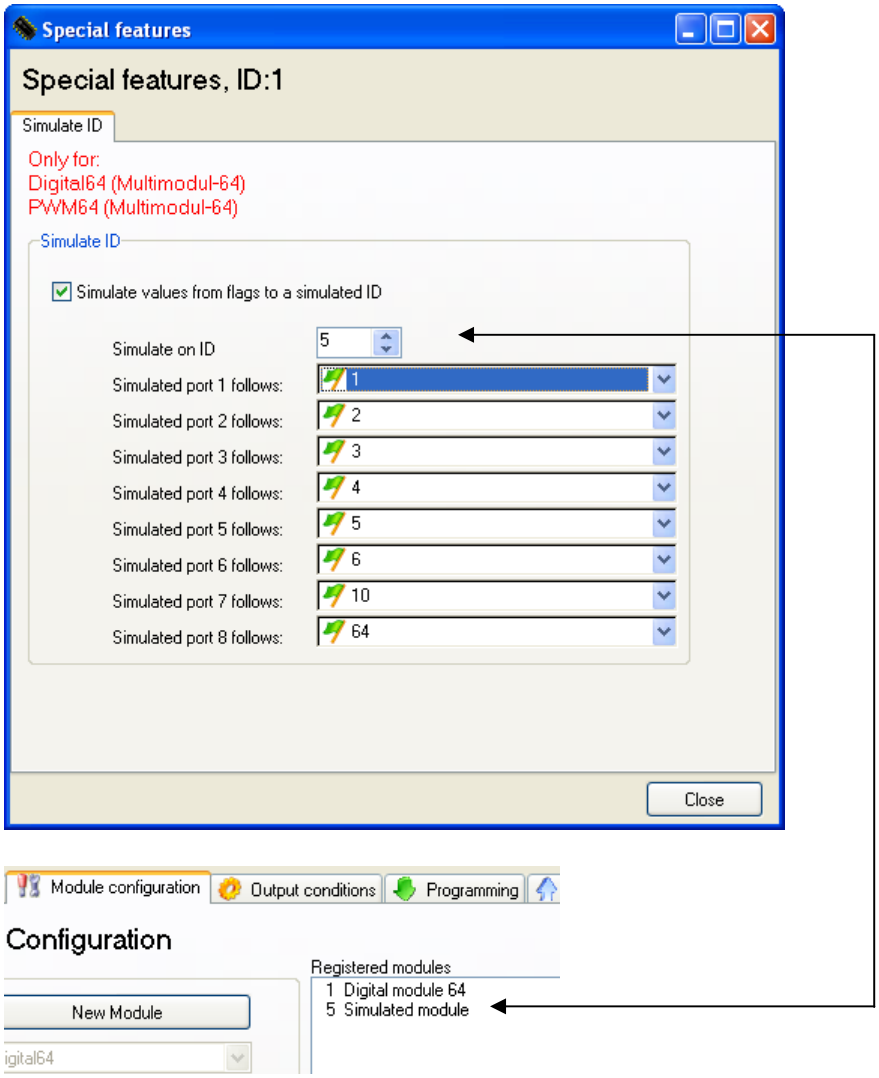


8 Special Features

When a module is selected that have special features, the button special features will be enabled. When the button is clicked a new window will show the configuration possibilities.

8.1 Simulate ID

Below is an example of the function “Simulate ID”. To simulate an ID on the CAN bus you first need to enable the function and assign an ID number to the simulated module. After this has been done, the eight ports of the simulated module must be connected to a value. Possible values to connect is the internal flags (sub routines) that the current module have.



In this example a new simulated module is created on ID 5. The values of the simulated module is fetched from ID 1:s flags (sub routines).

This is a good way to send internal flag values out on the bus so that other modules can use this value.

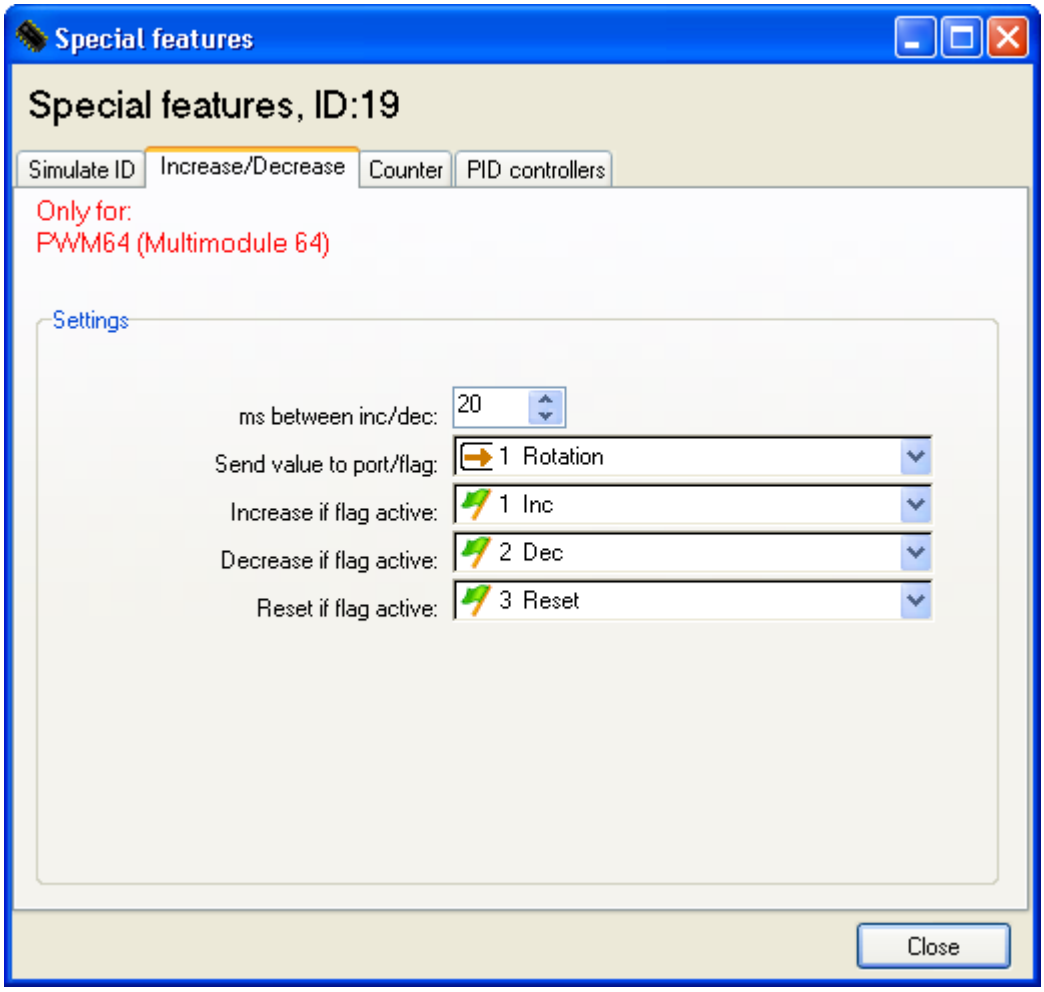
8.2 Increase/Decrease

The function Increase/Decrease can be used to change a value up and down, by just having push buttons. The module will remember this value until it is restarted.

Example:
A concrete mixer want to rotate the concrete at a certain speed. They have one button to increase the speed and one button to decrease the speed of rotation. The third button is used to stop the rotation.

Statements for the flags:
Flag1: Button 1= 1 (inc)
Flag2: Button 2= 1 (dec)
Flag3: Button 3= 1 (stop)

Flags can be chosen freely between 1 and 64

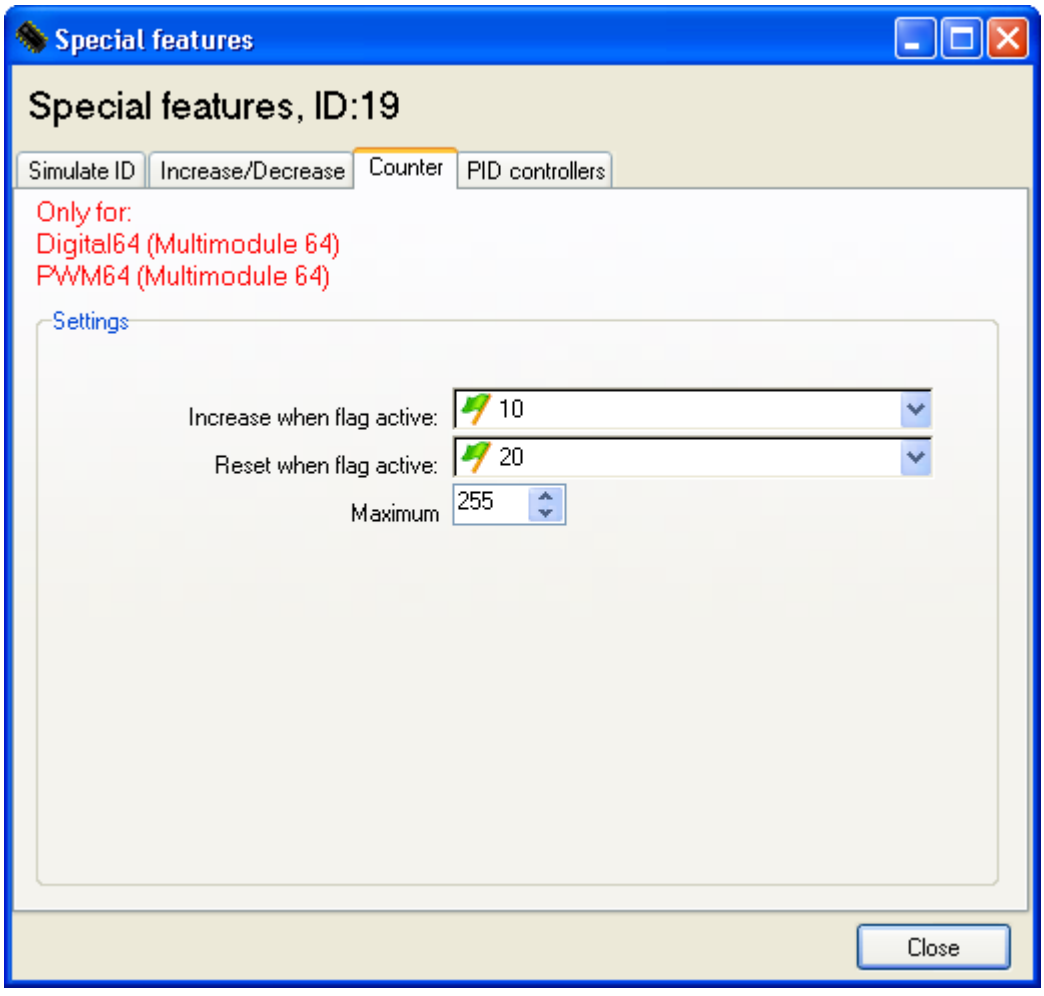


8.3 Counter

The function Counter can be used to count how many times a signal has been activated. First you select what flag that will trigger the counter to increase. This flag will also get the value of the counter. Then you select a flag that will reset the counter. Last you select how high the counter can count. When the maximum value is reached the counter will stay on the maximum value.

Example:
In the example below, the counter will increase each time flag 10 is active.
The counter will reset as soon as flag 20 is active.
The value of the counter is saved in flag 10.
The counter can be useful in sequence programs or to count.

Flags can be chosen freely between 1 and 64.



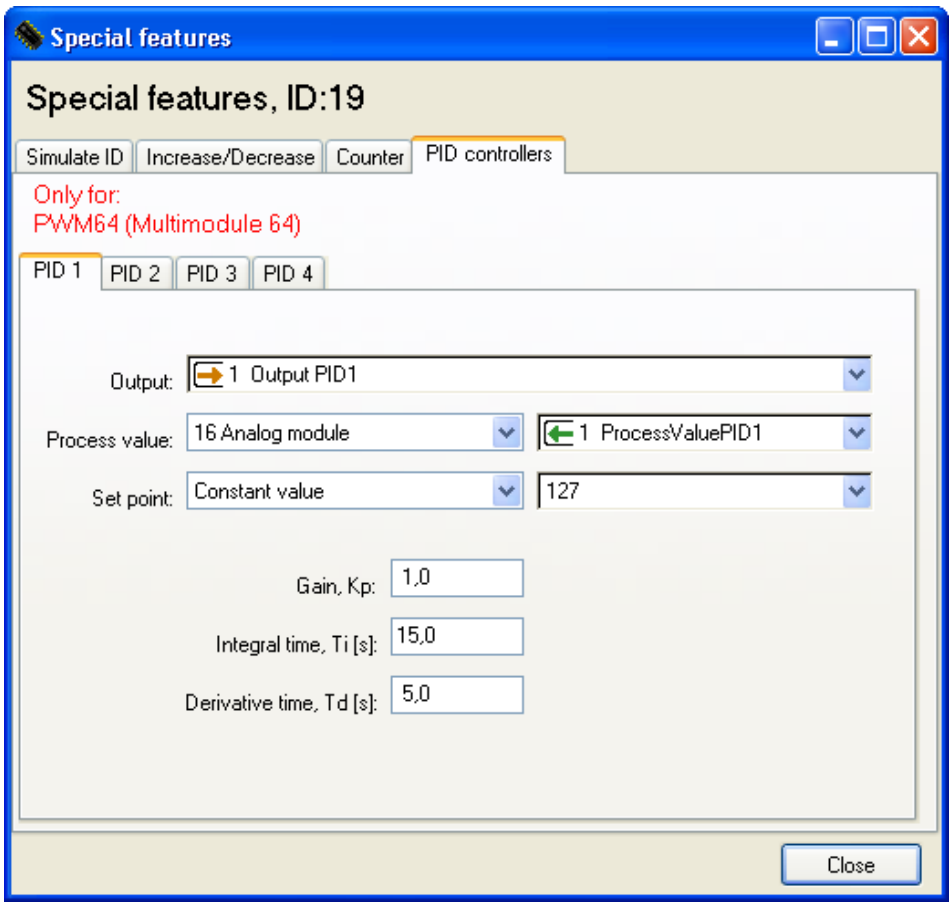
8.4 PID controllers

The PID controller can be used to keep e.g. a speed, a position or a temperature at a desired value. It measures the actual value, process value, and compares it with the desired value, setpoint. The difference between process value and setpoint is called control error. The output is calculated and controls e.g. a motor, an actuator or a heater, in order to minimize the control error.

The PID controller output is the sum of three parts:

- The proportional part is the control error, it becomes greater the more process value and setpoint differ. Often, the proportional part alone can not completely remove the control error.
- The integral part is the sum of all previous control errors, the output signal will increase or decrease as long as there is a control error. The integral part makes any remaining control error disappear.
- The derivative part senses the rate of change of the process value. It tries to predict changes in the process value, so that the controller can compensate for them before they become too big. The derivative part has a dampening effect and can make the control both faster and more stable.

The behavior of the controller can be changed by adjusting the three parameters:
Gain (Kp), Integral time (Ti) and Derivative time (Td).



Output
Selects where the controller output is sent, or if the controller should be turned off. The output can be sent to any port or flag in the PWM64 module, and takes precedence over any conditions in the selected port or flag. The controller output has its center value at 127.

Process value
The actual, measured value for the system to be controlled. Can be obtained from any port or flag in any module in the project, or be set to a constant value.

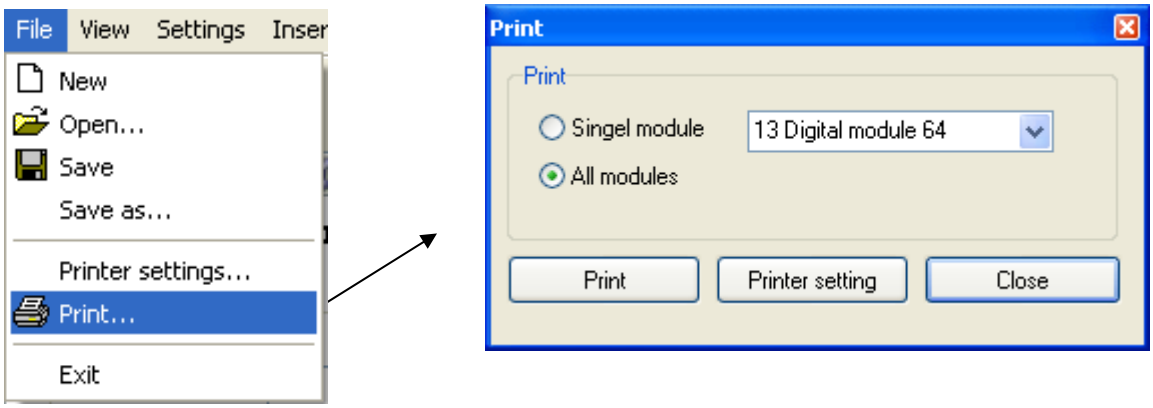
Setpoint
The desired value for the system to be controlled. Can be obtained from any port or flag in any module in the project, or be set to a constant value.

Gain, Kp
Gain of the controller, range 0.0-25.5 times.
A gain of 1.0 means that a change of the process value gives an equally large change of the output value (disregarding integral and derivative part). A gain of 2.0 gives a twice as large change of the output value.

Integral time, Ti
Integral time of the controller, range 0.0-25.5 seconds.
Specifies the time it will take before the integral part has contributed as much to the output as the proportional part has, at a constant control error. A larger value gives less integral action, while a low value gives more. The exception is if 0.0 is specified, then the integral part is turned off completely.

Derivative time, Td
Derivative time of the controller, range 0.0-25.5 seconds.
Specifies how far ahead in time the derivative part predicts changes. A larger value gives more derivative action, while the value 0.0 turns off the derivative part completely.

9 Print:



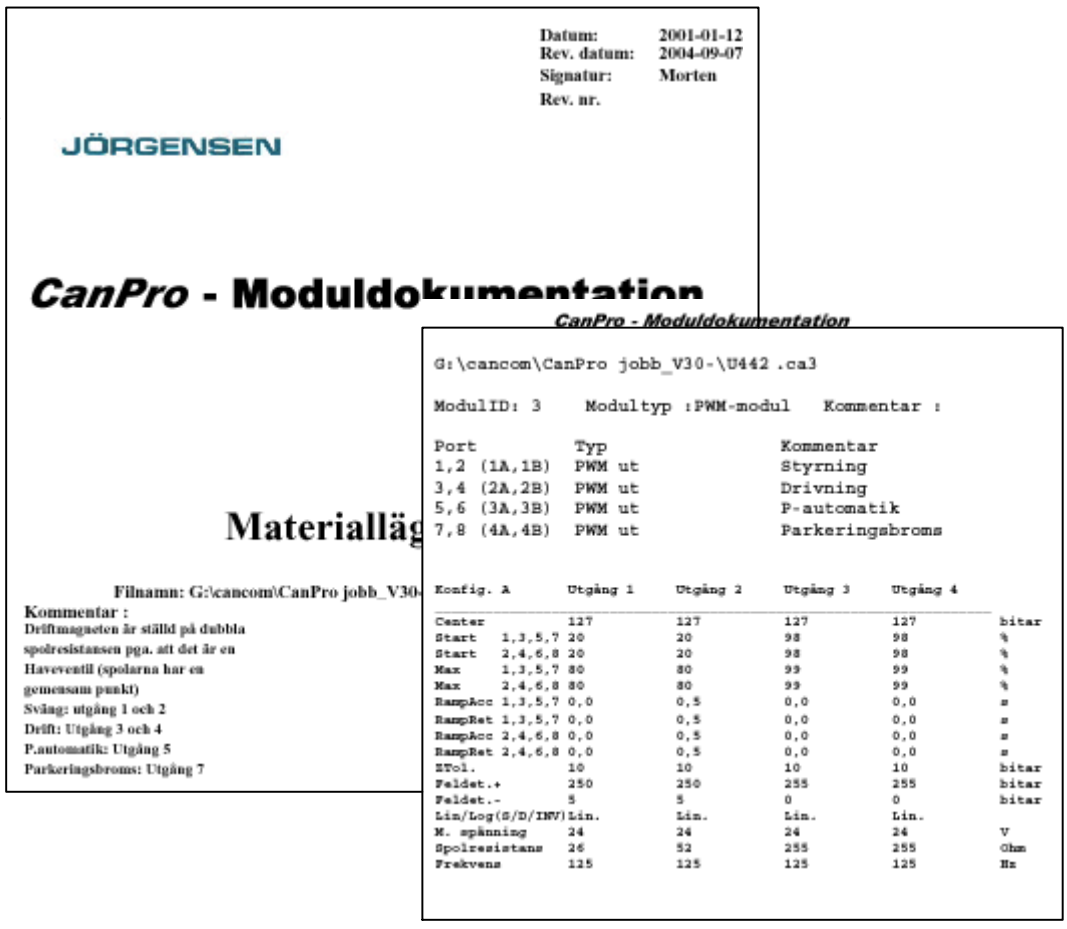
Single module: Print a single module with the ID nr entered in the box.

All modules: Print all modules in the project.

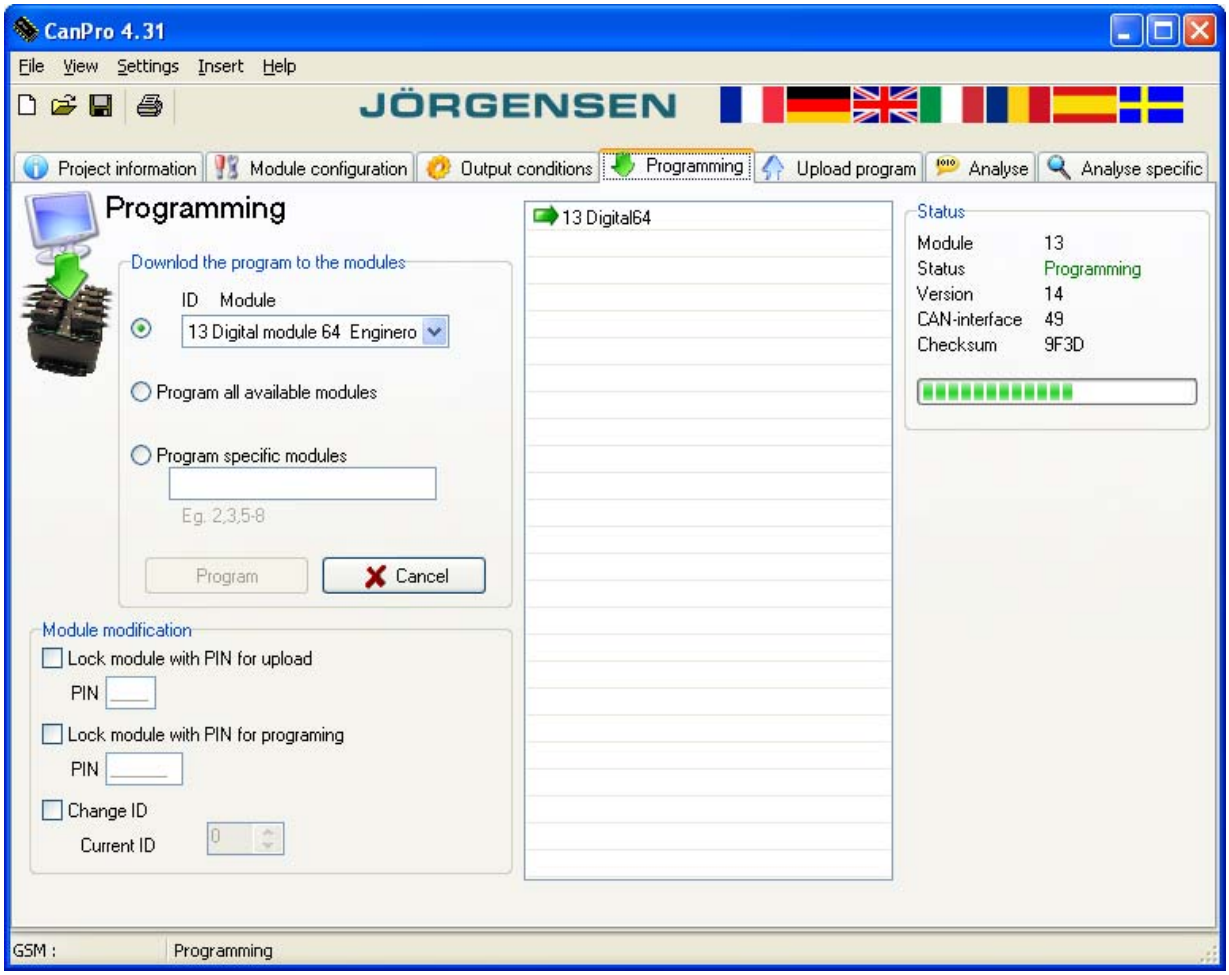
Printer settings: Shows a printer dialog, where you can select witch printer to use.

Print: Starts the print.

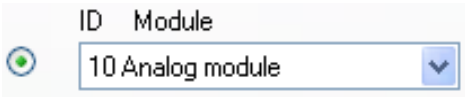
Close: Closes the printer window.



10 Programming module:

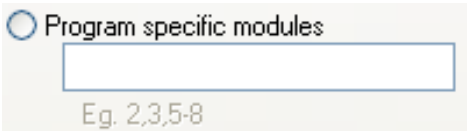


Program: There is 3 different ways to program the system.



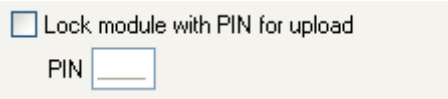
This selection can be used when programming a single module. Select the module to program and press the program button.

Program all available modules is used when you want to program all modules in the project. Press the program button, and a list of modules will appear to the right.



This is a flexible way to program a set of modules from the project.
It's very much like how you enter what pages you want to print.
Example: 3,5,7,8,9,10 (can also be written as: 3,5,7-10).

If CanPro is in the process of sending its program to the hardware module, it's possible to cancel this action.



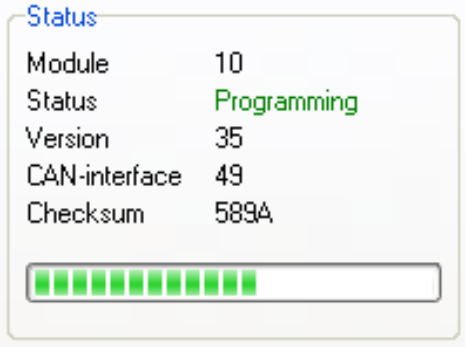
You can lock the program in the module by entering a PIN number, and then program this to the hardware. It's now not possible to upload the program from the hardware back to CanPro without the right PIN.

By locking the ID with a 6-digit PIN code, programming the module by unauthorized personel is avoided. The modules can be programmed with different PIN codes.

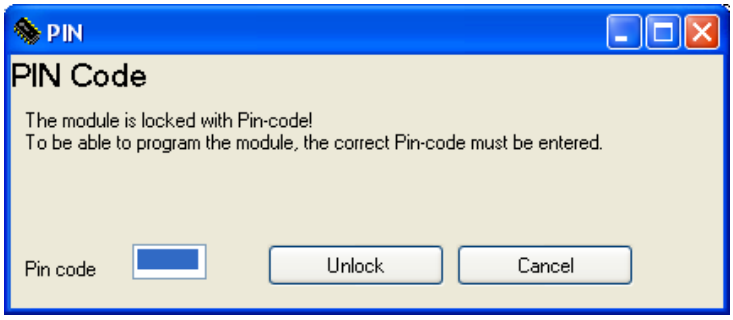
If a module in the project has an ID different than the hardware you want to send the program to it's possible to change the ID of the hardware.

Check the box and enter the current ID of the hardware. After you have programmed the module, it will have changed its ID to the ID of the module in the project.

Example: In my program I have a module assigned to ID 5. My hardware already has an ID assigned to it, ID 10. I select ID 5, check the box *change ID*, and set the value to 10. Then I program. Now the hardware has the ID 5.

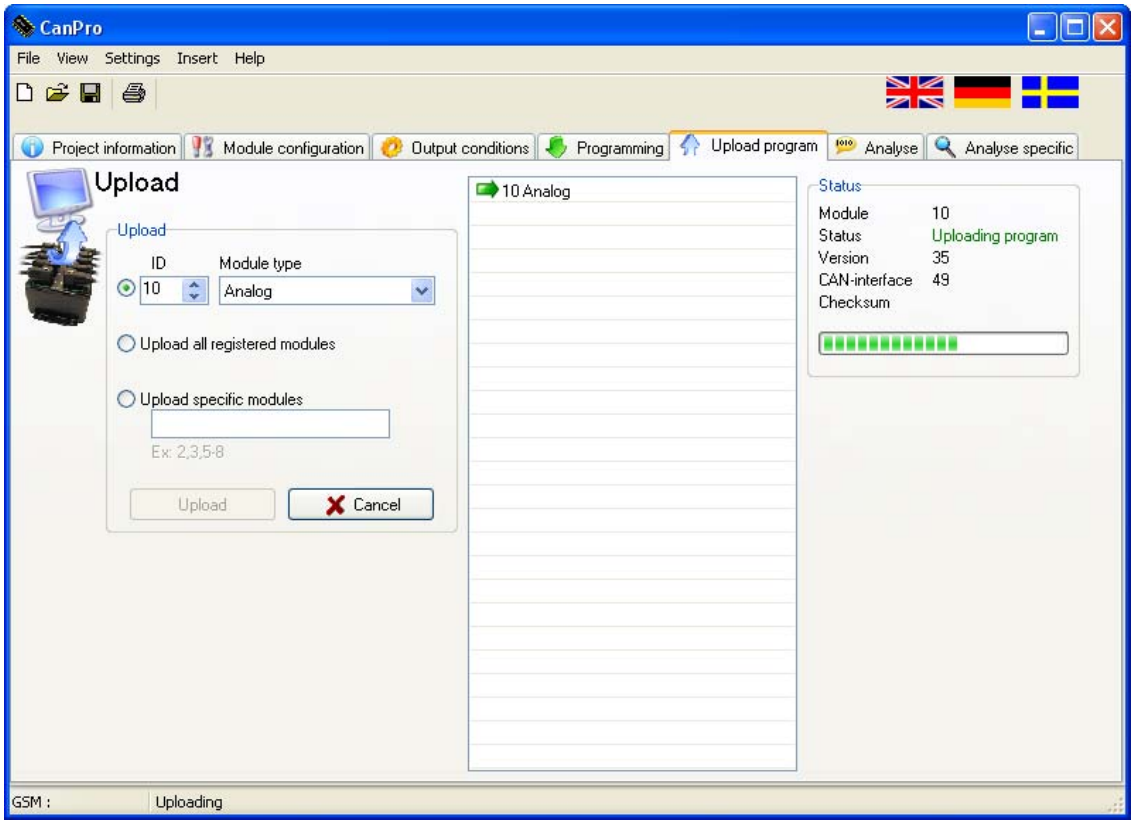


Module: Shows witch ID currently being programmed.
Status: Show the status of the current action.
Version: Version of the module being programmed.
***CAN*-interface:** Version of the CAN interface.
Checksum: Checksum of the program being sent.

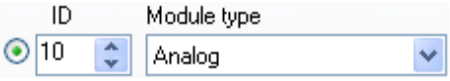


In those cases when the CanCom module has been programmed with the programming-PIN, will this dialog show. PIN code has to be entered to program the CanCom module again.

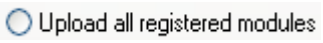
11 Upload Program:



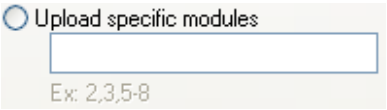
Uploading of program:



This selection can be used when uploading a single module.
Select the module to upload and press the upload button.



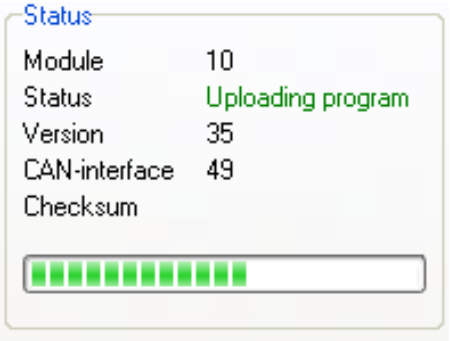
Upload all registered modules is used when all modules on the CAN bus shall be uploaded to CanPro.



This is a flexible way to upload a set of modules from the CAN bus to the project. A requirement is that the project already has the ID:s that you want to upload registered in the project.
After the upload the modules in the project will have the exact code of the modules on the bus.

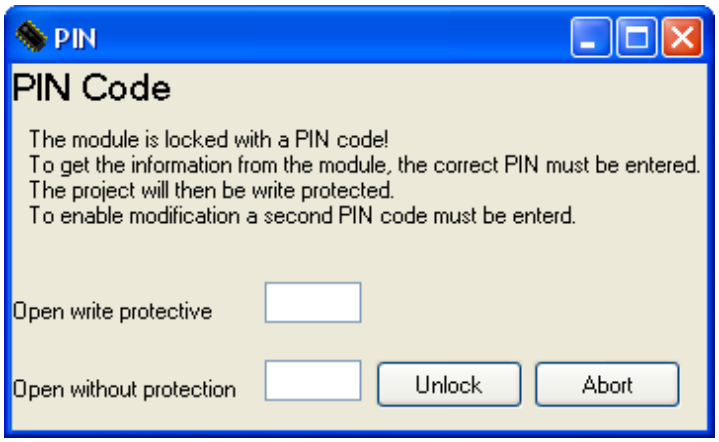
To enter witch modules to upload I similar to how you enter what pages you want to print
Example: 3,5,7,8,9,10 (can also be written as: 3,5,7-10).

If CanPro is in the process of uploading a program from the hardware module, it's possible to cancel this action.



Module: Shows witch ID currently being rogrammed.
Status: Show the status of the current action.
Version: Version of the module being programmed.
***CAN*-interface:** Version of the CAN interface.
Checksum: Checksum of the program being sent.

Note:
If you have an empty project and register a set of modules, it's possible to upload only one of the registered modules. However if the uploaded module interacts with any other module, this will not show in the condition list, since the information from the other modules is missing. In this case the correct thing would be to upload all modules registered in the project.



If the hardware module has been programmed with a PIN code, an upload will result in a dialog window opening. Here it's possible to upload the module to the project as Read-Only or upload the module to the project as normal, with no limitations.
If you want to open up a locket module with no limitations, you have to contact the supplier of the hardware to get the master PIN.

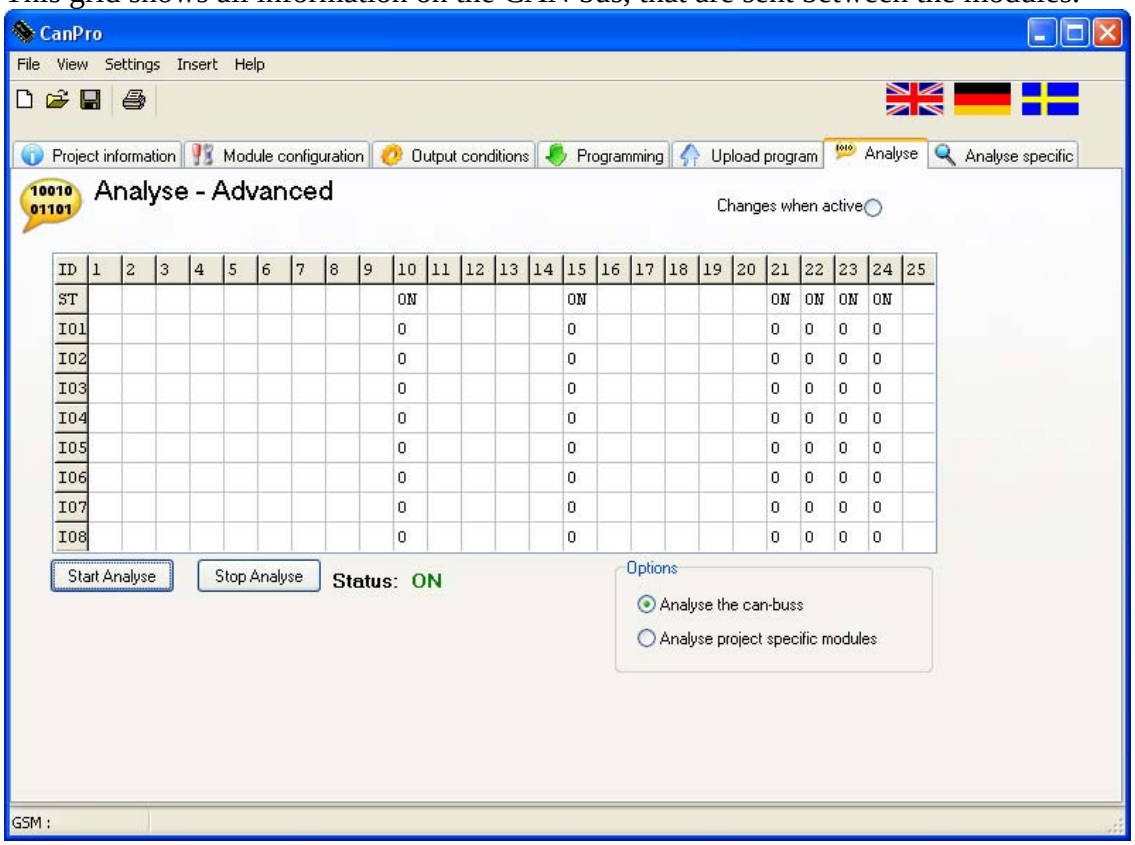
If you upload a module with read-only permissions, you can still change the PWM-settings. This allows you to calibrate the different settings, but will not allow you to change any program logic.

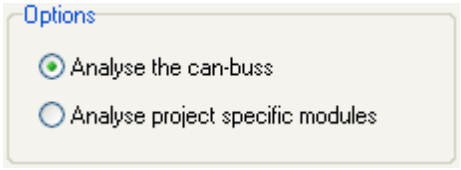
12 Analyse the CAN bus:

This chapter describes how it's possible to analyse the data on the CAN bus in CanPro. This enables you to easily detect errors and get information about inputs and outputs.

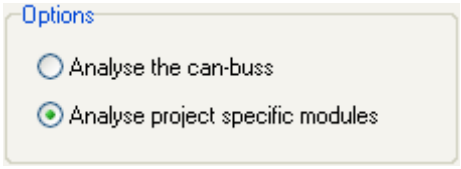
12.1 Analyse - Advanced

This grid shows all information on the CAN bus, that are sent between the modules.

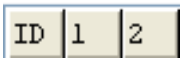




In the *options* box you can choose to analyse all modules on the bus. This option does not require a any registered modules.

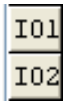


This selection is to analyse only ID:s that are registered in the current project.



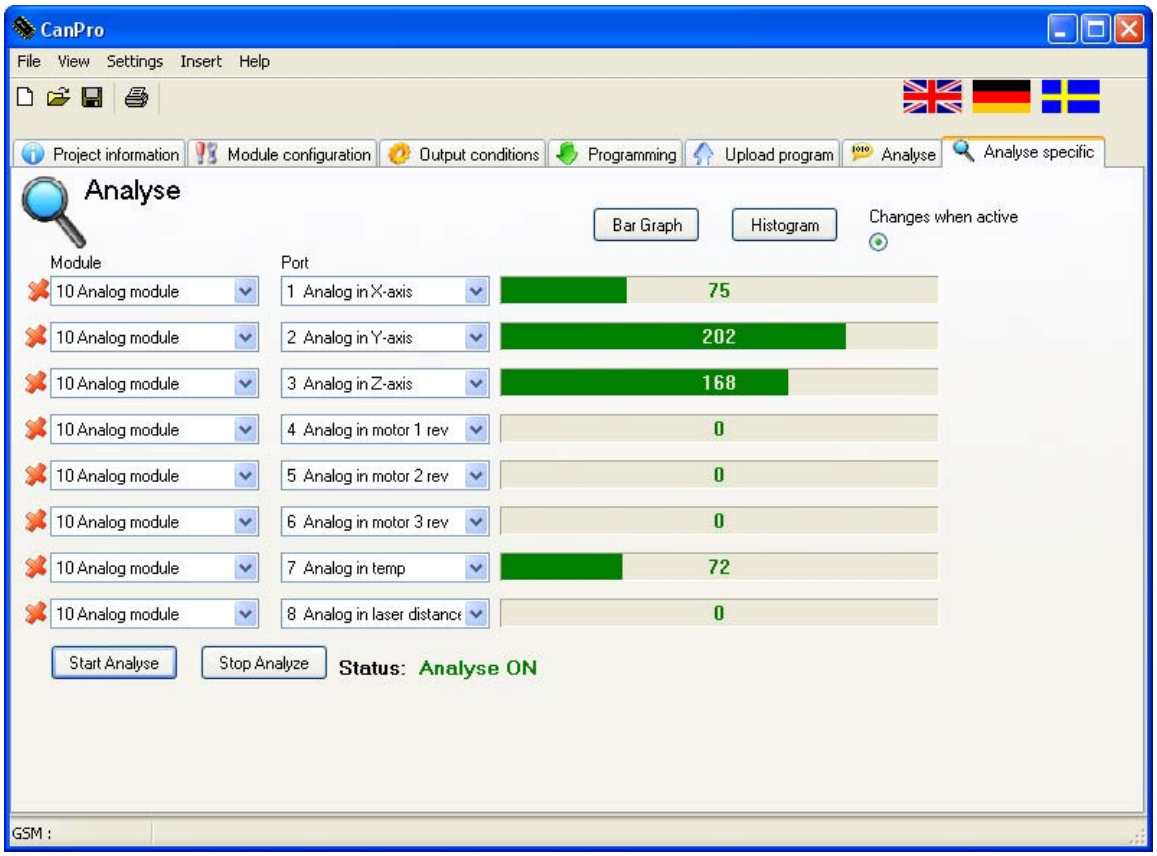
← Columns represent different ID:s (ID1 – ID25)

← ST shows the status of the ID. (ON / OFF)



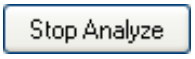
← IO1..8 represents the ports on each module. (0-255 bits)

12.2 Analyse Specific – Bar graph



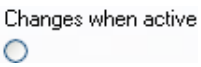
Select *Bar graph* to show the selected ports in real time with green bars. Its possible to show up to eight bars at the same time. To select a port, drop down the boxes at the left side, and select the port from your project.

Starts the analyser.



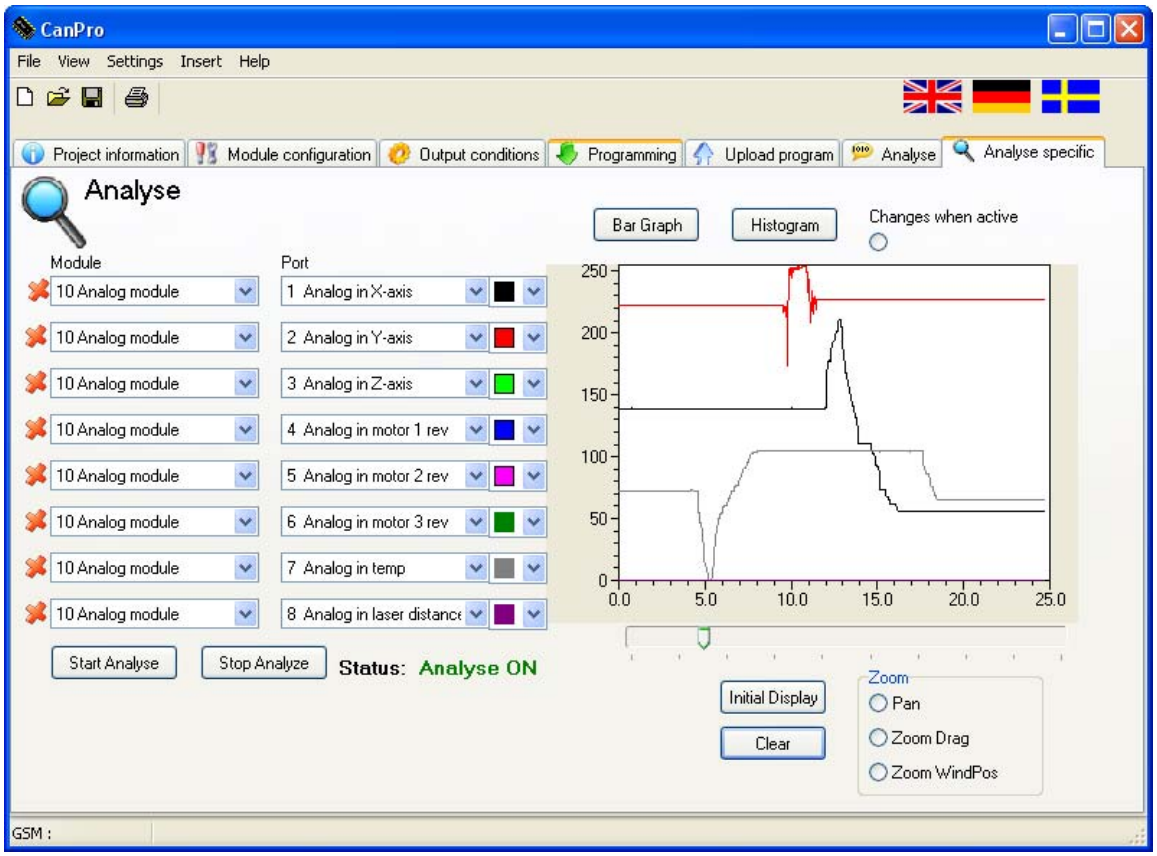
Stops the analyser. If you leave the Analyse specific tab, the analyse will automatically shut off.

Clear the current row from a selected IO.



This radio button will shift(on/off) if data is coming in.

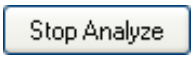
12.3 Analyse Specific – Histogram:



Click on the *Histogram button* to enable the histogram graph. It's possible to record up to eight different I/Os at the same time. As with the bar-graph, the IDs ports are selected from the boxes at the left of the window. Each port has also a color assigned to it, showed in the graph.

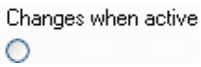
← Color to show in the histogram.

Starts the analyser.

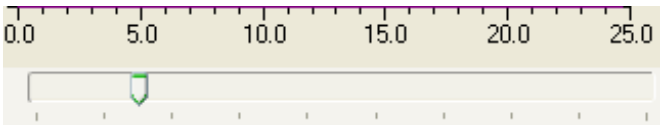


Stops the analyser. If you leave the Analyse specific tab, the analyse will automatically shut off.

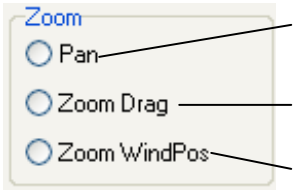
Clear the current row from a selected IO.



This radio button will shift(on/off) if data is coming in.



Timebase in seconds.
Can be changed before under and after analysis.

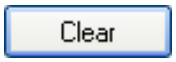


Pan the histogram.

Stretch the window in both X and Y axis

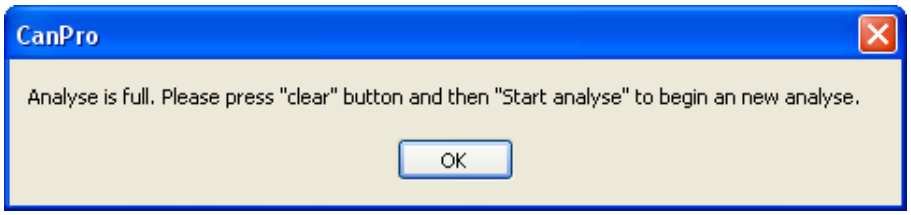
Zoom a selected area.

Reset the histogram to its initial window settings.



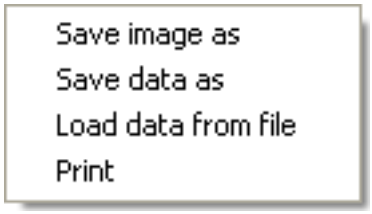
Clear the histogram from previous log information.

Clear the current row from a selected IO.



A log file can at most hold 19.000 log points. The logs are saved ever 20 ms, for as many as 8 channels at the time. In other words, more channels equals lesser time before the log is full.

By right clicking on the histogram the following popup menu will show:



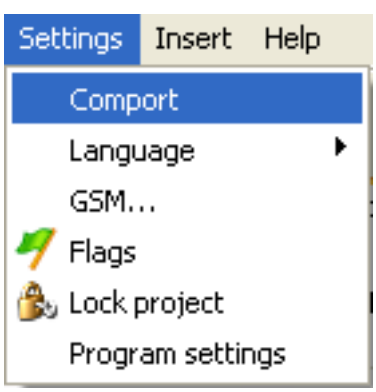
Save image as: Save the histogram as a bmp image.

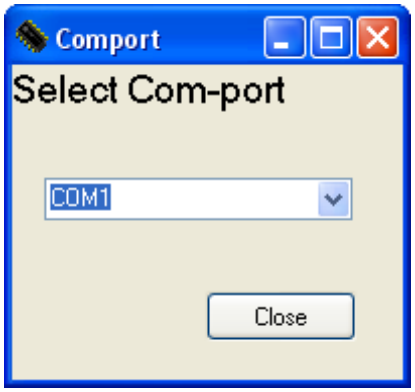
Save data as: Save the data in a log file. This file can be opened in CanPro at a later time.

Load data from file: Load in data from a log file.

Print: The current histogram is send to a printer with information about what ports has been analysed and their colour code. If a log file is loaded into CanPro information about colour and what ports

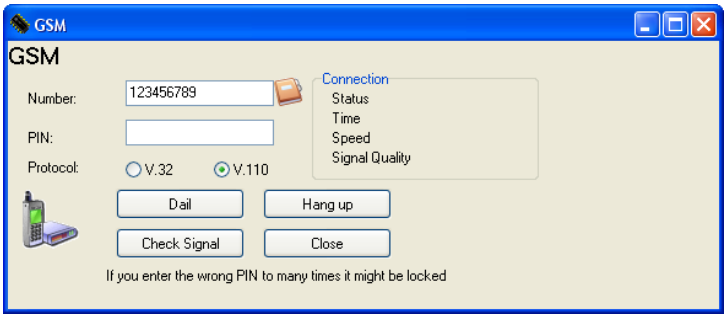
13 Com-port





Select the serial port on which the CAN-interface is connected.

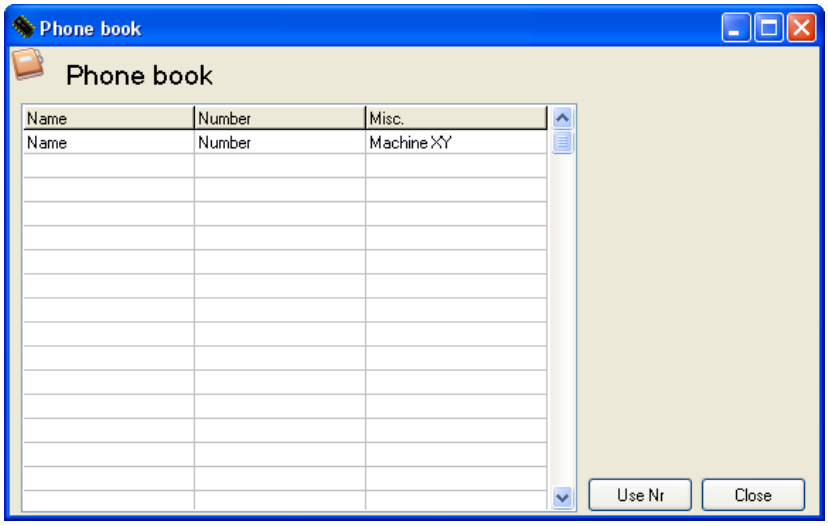
14 GSM



When a CanCom GSM modem is connected to a CanCom control system a remote PC can:

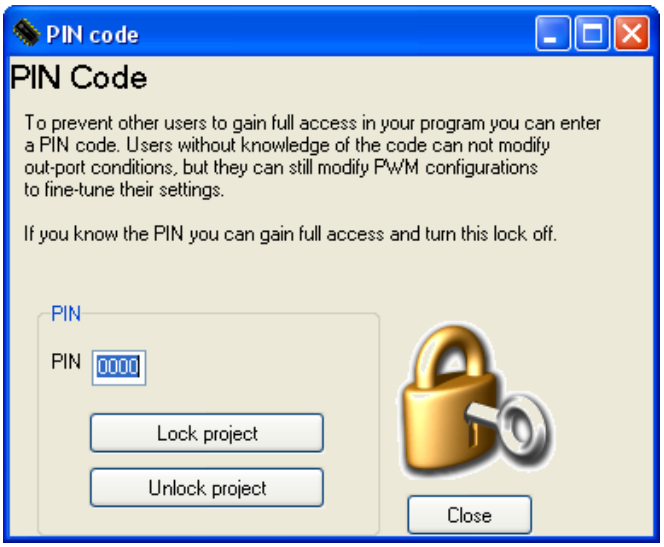
- Remotely update a program
- Remotely upload the program from the hardware
- Remotely analyse information on the bus to detect errors.

Depending on the telecom operator and country different protocols apply. You have the possibility to change from V.110 to V.32.



To open the phone book click on the phonebook icon. Here you have the possibility to store up to 100 numbers.

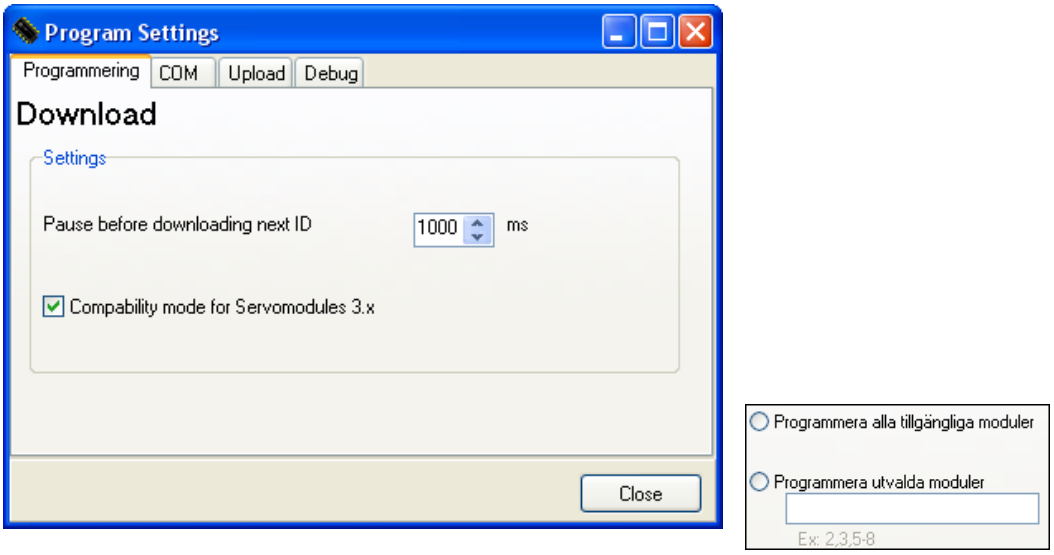
15 Lock Project



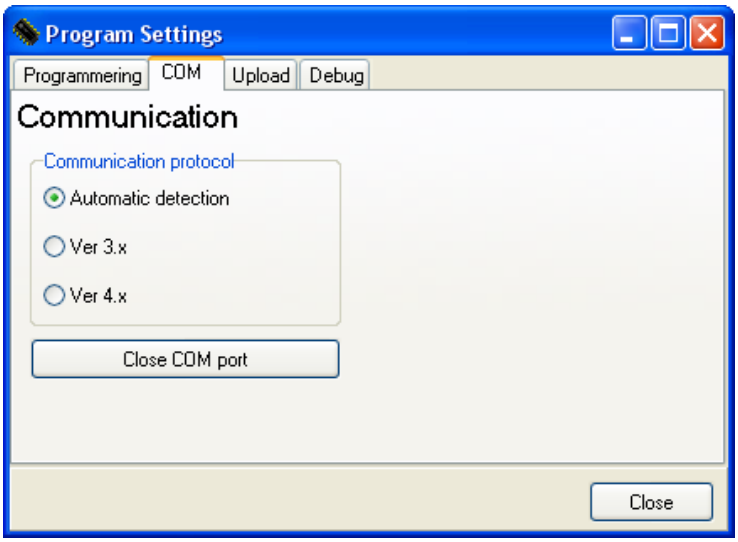
The CanPro project can be protected by a 4 digit code. This will protect the projects from other users to change settings/code in the project. How ever, the project can be opened with a “read only” protection. This enables you to release the project to otherwise less trusted customers. They will then have the possibility to program their machine without you having to worry about them changing any settings or program logic.

Also read about locking program in the hardware modules.

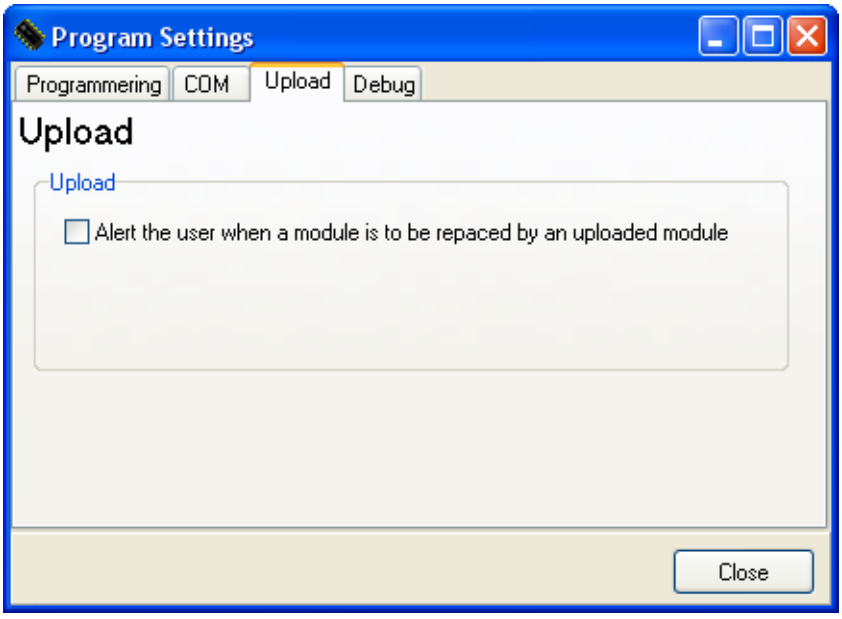
16 Program Settings



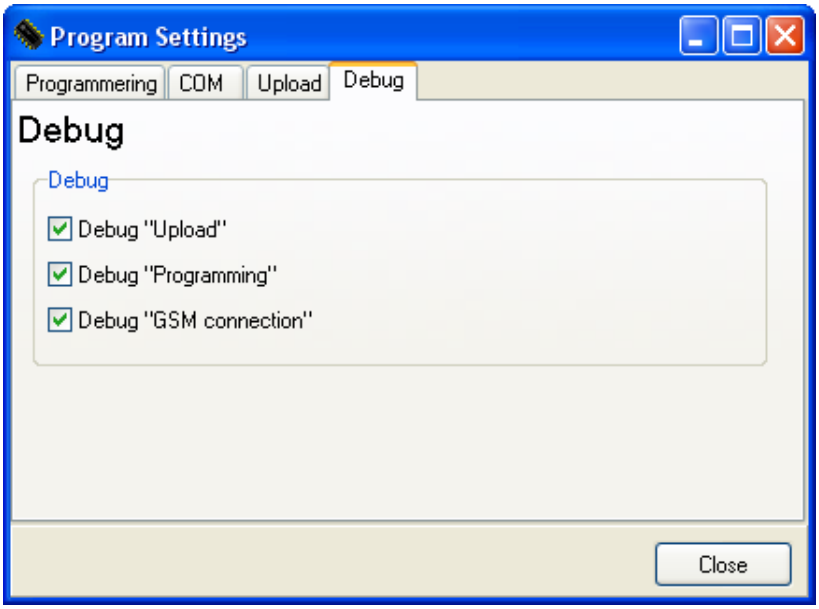
When programming more than one ID at the time, there is a pause time between the different ID:s. It's possible to change this pause time to optimize programming, or to raise the time if problems occur when programming.



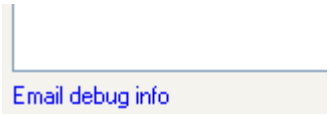
CanPro can be set in three different communication moods. This is due to backward capability with old hardware. If the radio button is set to automatic, the hardware version is detected and the appropriate algorithm is used. The two later radio buttons can lock the communication into ver3 or ver4 communication protocol. This could be used if problems occur when programming with unknown hardware.



When uploading a module from hardware into the CanPro project, and there already exists a registered module in the project with the same ID, a warning will be showed. This gives the user the chose to either continue with the upload and overwrite the previously registered module with the uploaded module, or to abort.
If many modules shall be uploaded you can disable this warning and the modules will be overwritten.

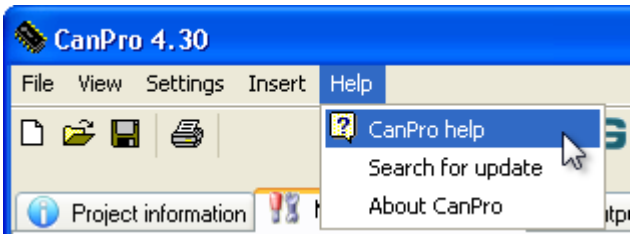


If having problem with the CanPro software, there is a debug functionality built in.
To enable this functionality, pleas check the boxes as showed above.
Collected debug information can then be emailed to our software engineers for further investigation.



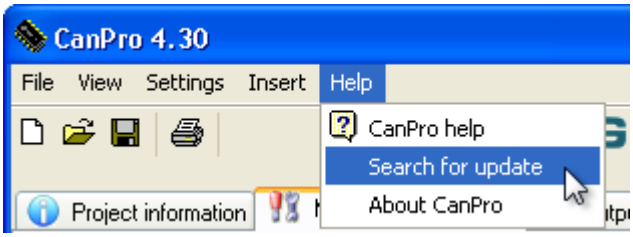
17 Help

17.1 Manual



Opens this manual. This requires a PDF reader to be installed on the computer.
If CanPro uses swedish language, the swedish version of this manual will be opened.
Otherwise the english version vill be opened.

17.2 Updates



If connected to the internet, you can search for newer versions of CanPro.
You can also visit www.cancom.se to download the software.