

# Gertbot GUI.

Rev 2.3, 27-October-2014

## Introduction

This document describes the Gertbot Gui. For details of the Gertbot board itself see the Gertbot manual. In several places this document refers to that manual.

The Gertbot Gui is designed for debug end development. It is not means as standard GUI to control motors or as a DCC controller. However as all the code is released under GPLv3 thus you are welcome to take the code apart and use it to build your own stand-alone GUI.

There are two version of the Gertbot GUI. One for the Raspberry-Pi and one for a windows PC. (The latter requires an interface cable to be soldered). The only visible difference is that the Windows version allows the user to select a COM port interface.

The Gertbot GUI for both can be downloaded from [www.gertbot.com](http://www.gertbot.com).

## Notes

### Note 1:

The Gertbot GUI is still under development. It has been tested but there may still be corner cases where it does not work as expected. However as the source code is in the public domain you are welcome to fix any issues you find yourself.

### Note 2:

The GUI is a one way interface. It can send commands and for some commands it can receive replies. But the GUI is NOT automatically updated when the Gertbot board state changes. For example: if an end-stop is hit the board will stop the motor. However the GUI is not informed of this and thus will display as if the motor is still running.

### Note 3:

It is possible to run the GUI at the same time as your application. (At least on the Raspberry-Pi, this is not possible under Windows.) This is extremely useful but also dangerous as there is no protection on the UART to make the commands exclusive. I have not seen any problems after using it for six months but then I am also very careful to use the Gertbot Gui in a 'mutual exclusive' manner: I don't press any Gui buttons when application is sending or receiving data.

## Contents

Introduction.....	1
Notes .....	1
Starting .....	3
Motor controls.....	4
Brushed motor controls.....	6
Stepper motor controls.....	8
J3 connector .....	10
Input/Output.....	11
ADC .....	11
DAC .....	11
Board controls.....	12
DCC controls .....	12
Slider .....	14
File save load .....	15

## Starting.

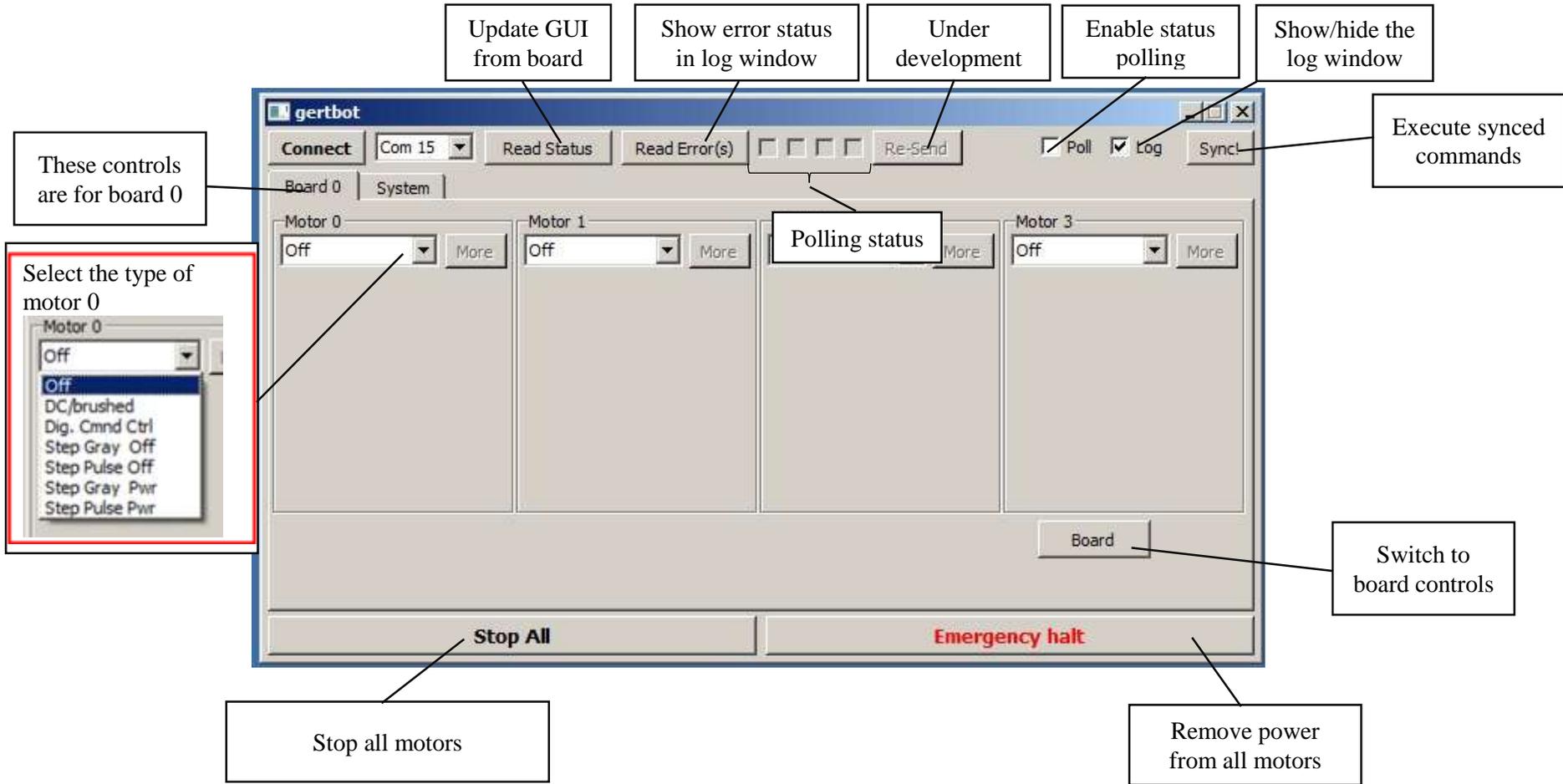
After starting the program looks like this:



When you press the 'connect' button the program will look for any connected boards and change its appearance accordingly.

### Motor controls

This is how it looks if a board with ID 0 is connected:



## Motor controls

Here is a short description of each control. For details of how the motor controls board works please read the Gertbot manual.

Select the type of motor 0	This is the control you will use first. You have to tell the board what type of motor is connected to each of its four outputs. If you select a Stepper motor the controls for the next motor in the GUI get disabled.
Update GUI from board	This control reads the status of all boards and all motors and updates the GUI with what. Use this to find the status of a board which already has received commands.
Show error status in log window	This control reads the error status of all boards and displays any error messages in the log window. (Which means you must have the log window enabled for it to be useful)
Re-send all status to board	This control is not yet implemented (Reads all controls and send the commands to the boards)
Enable status polling	This starts a timer which sends a 'status poll' command to each board in turn. The timer speed is set in the 'system' tab. The poll command updates the poll status squares.
 Polling status	These four 'squares' show the result of the polling of board 0..3. Green: OK Red: Halted Yellow: Error message in queue Orange: bridge has error NOW. (rarely seen)
Show/hide the log window	Enables or disables showing the log window.
Execute synced commands	Sends out a SYNC-command. Only useful if one or more boards operate in sync mode and there are outstanding commands. See the Gertbot manual for synchronised mode.
Switch to board controls	Switch to the 'board control' view. There you can control the features which have to do with the board. Thus includes the operation of the J3 connector.
Stop all motors	Send a stop command to all motors. The brushed motors will use ramp-halt. The stepper motors will stop (with or without power, depending on the stepper mode)
Remove power from all motors	Stops all motors on all boards NOW. The power is removed from every mootr. (This is a command equivalent of pulling the HALT line low.

### Brushed motor controls

Selecting the 'DC/Brushed' mode of operation brings up the following controls:

This screenshot shows the initial control panel for Motor 0. It features a dropdown menu set to 'DC/brushed' and a 'More' button. Below this, there are input fields for 'Frequency' (2000) and 'Duty Cycle' (50.0). At the bottom, there are three buttons: a left arrow for 'Move in direction A', a stop icon for 'Stop', and a right arrow for 'Move in direction B (Currently active)'. A dashed arrow points from the 'More' button to the next screenshot.

Motor 0 is a DC or Brushed motor

PWM freq.is 2 KHz

Duty cycle (DC) is 50%

Move in direction A

Stop

Move in direction B (Currently active)

Show more controls

Brushed motor controls 1 of 2

This screenshot shows the expanded control panel for Motor 0. It includes a 'Back' button, a 'Ramp' section with 'Up' (1.00) and 'Down' (0.50) settings, and a 'Short/Hot' section with a dropdown set to 'Stop 0' and checkboxes for 'End-stop A' and 'End-stop B'. A 'High' checkbox is also present. A dashed arrow points from the 'Back' button to the first screenshot.

Back to movement controls

Start ramp-up time (1 second)

Stop ramp-down time (1/2 second)

End-stop or "stop all", ramp-down speed (0 seconds)

If shorted or hot stop this motor only

End-stop A and B are enabled

Stop if end-stop A/B is or goes high

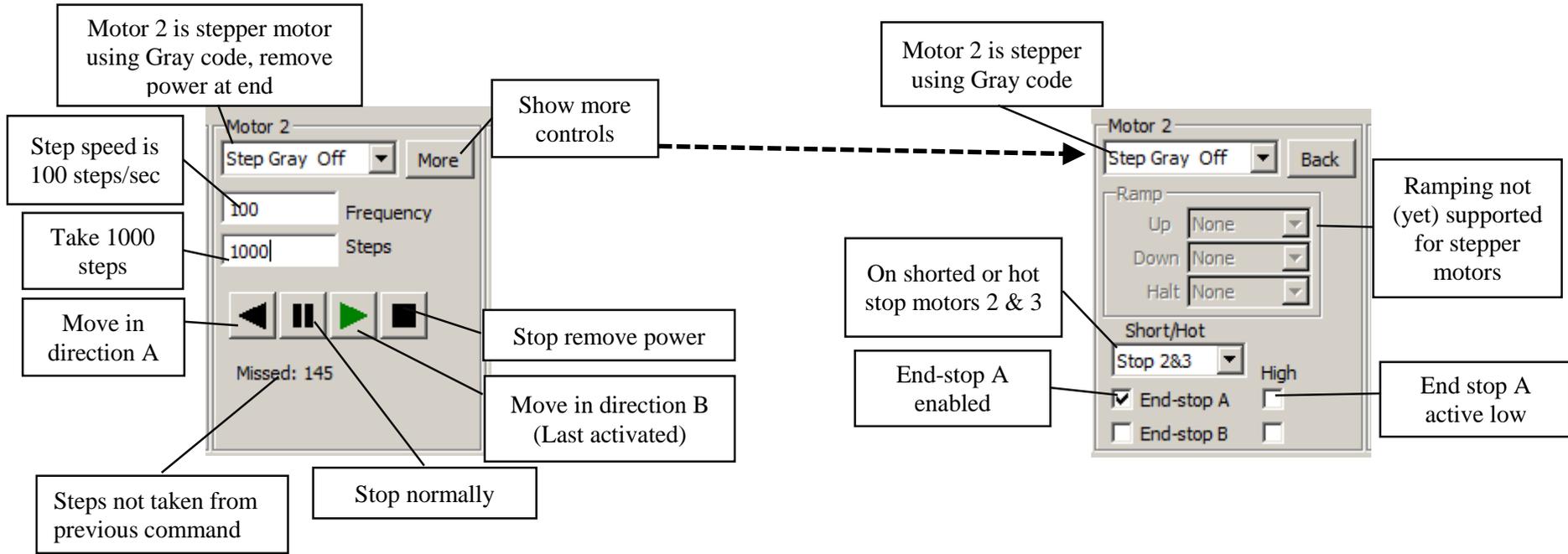
Brushed motor controls 2 of 2

## Brushed motor controls

Motor 0 is a DC or Brushed motor	Shows the type of motor selected which should be a DC/Brushed motor (Otherwise the other controls show in this picture are not relevant)
PWM frequency is 2 KHz	Controls the PWM frequency. A higher frequency makes the motor run more smoothly. For details see the chapter about PWM in the Gertbot manual.
Duty cycle (DC) is 50%	Controls the Duty cycle of the PWM. 100% is fully on. For details see the chapter about PWM in the Gertbot manual. You can use the up/down buttons or manually edit the value.
Move in direction A	These buttons start or stop the motor. The 'down' position shows the last command given. These button are not updated when a motor is stopped by an error condition.
Stop	
Move in direction B (Currently active)	
Show more controls	Pressing the button shows the more motor control choices.
Back to movement controls	Pressing the button gets you back to the motor controls.
Start ramp-up time (1 second)	This controls the ramp-up time (0 to 100% duty cycle) when a move-start command is given. Here the time is set to 1 second.
Start ramp-up time (1/2 second)	This controls the ramp-down time (100% to 0% duty cycle) when a move stop command is given or when the direction is reversed. Here the time is set to 1/2 second.
End-stop or "stop all", ramp-down speed (0 seconds)	This controls the ramp-down time (100% to 0% duty cycle) when a stop-all command is given or when an end-stop is activated. Here the time is set 'None' which means 0 seconds: the motors are stopped immediately.
End-stop A and B are enabled	These check boxes enable or disable the end-stops for that motor.
Stop if end-stop A/B is or goes high	Sets the polarity of the end-stop. If checked the end-stop is active high. (Stop when high). If not-checked the end-stop is active low (Stop when low).

For more details see the chapter about end-stops in the Gertbot manual.

### Stepper motor controls



Stepper motor controls 1 of 2

Stepper motor controls 2 of 2

## Stepper motor controls

Motor 2 is stepper using Gray code..	Shows the type of motor selected which should be a stepper motor (Otherwise the other controls show in this picture are not relevant). This stepper motor is controlled using Gray code and when finished taking steps the power is removed. (The anchor is not held)
Step speed is 100 steps/sec	Controls how fast steps are generated. Big stepper motors have a lower maximum step frequency. Check the specification of your motor or try it out.
Take 1000 steps	When the 'run' button is pressed take the specified number of steps. The maximum amount in one go is 8.388.608 steps. (8.3 million steps).
Move in direction A	Start the stepper motor (positive steps). The GUI does not check in any way if the motor has finished stepping.
Stop, Normally	Stop the stepper motor. It depends on the stepper motor operating mode what happens. For modes "Gray Off" and "Pulse Off" the power is removed from the motor, leaving the anchor free. For modes "Gray Pwr" and "Pulse Pwr" the power is supplied to the motor, leaving the anchor rigid. Note in that the latter mode the system can consume a large amount of current and your motor can get hot.
Move in direction B	Start the stepper motor (negative steps). The GUI does not check in any way if the motor has finished stepping. The green colour shows that this is the button last pressed.
Stop, remove power	Stop the stepper motor and always remove power from stator. The power is also removed for the "Gray Pwr" and "Pulse Pwr" modes
Steps not taken from previous command	Normally you want the stepper motor to run the given amount of steps. If for some reason the stepper motor gets a new command before that, the Gertbot keeps track of the amount of steps not taken.

Currently there is no provision for ramping up or down stepper motors.

End-stop A enabled	Enables or disables end-stops A and B. For details see the chapter about end-stops in the Gertbot manual. Here end-stop A is enabled.
End stop A active low	Sets the polarity of the end-stop. If checked the end-stop is active high. (Stop when high). If not-checked the end-stop is active low (Stop when low). For details see the chapter about end-stops in the Gertbot manual.

### J3 connector

The Gertbot GUI gives the user control over the J3 connector. Every pin can function as input or output. Besides that some pins have special functions. The connector has 4 major sections:

Connector J3										
Pin:	19	17	15	13	11	9	7	5	3	1
Mode:	Gnd	3V3	ADC3	ADC1	Gnd	Input	Input	Input	Input	Input
Read			---V	---V		<input type="checkbox"/> x				
Mode:	Input	Input	ADC2	ADC0	Halt	Input	Input	Input	Input	Input
Pin:	20	18	16	14	12	10	8	6	4	2
Disp	DAC0 & 1		ADC0-3			Spare	Extra / End-Stop			

I/O or DAC (Digital to Analogue converter)

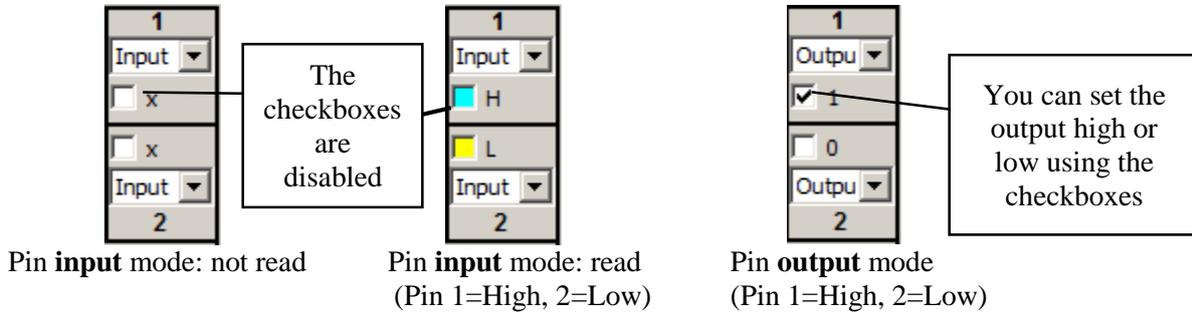
I/O or ADC (Analogue to Digital converter)

I/O only

I/O or End stop

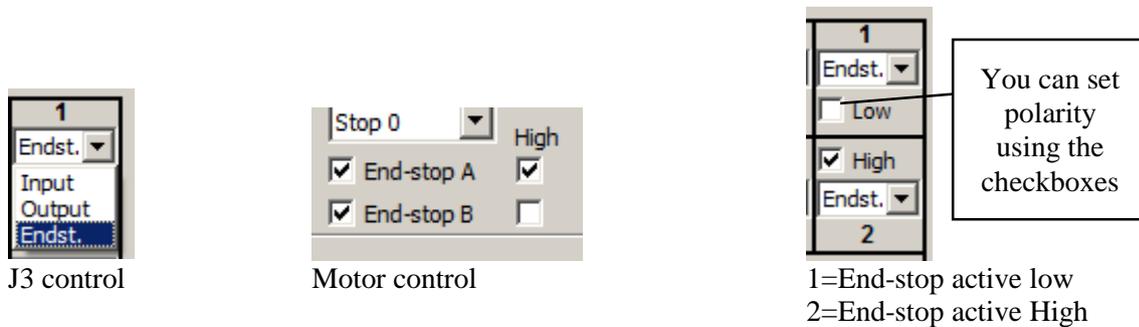
### Input/Output

As mentioned every pin can be a digital input or output.  
 The input status is update when the 'Read' button is pressed.  
 You cannot change the state of the input (the checkboxes are disabled)  
 If a pin is set to output mode you can change the state.



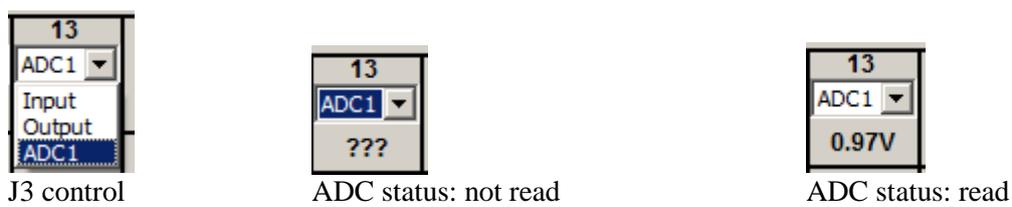
### End-stop.

By operating the pull-down menu you can set any of the pins 1-8 into end-stop mode.  
 There is a second way: if you change the end-stop checkboxes associated with the motors.



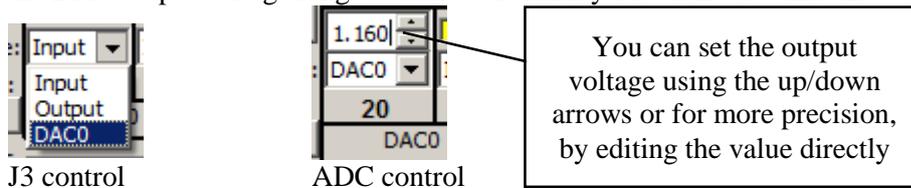
### ADC

By operating the pull-down menu you can set any of the pins 13-16 into ADC mode. This is the default mode in which the boards starts-up. You can read the ADC values by pressing the 'Read' button.

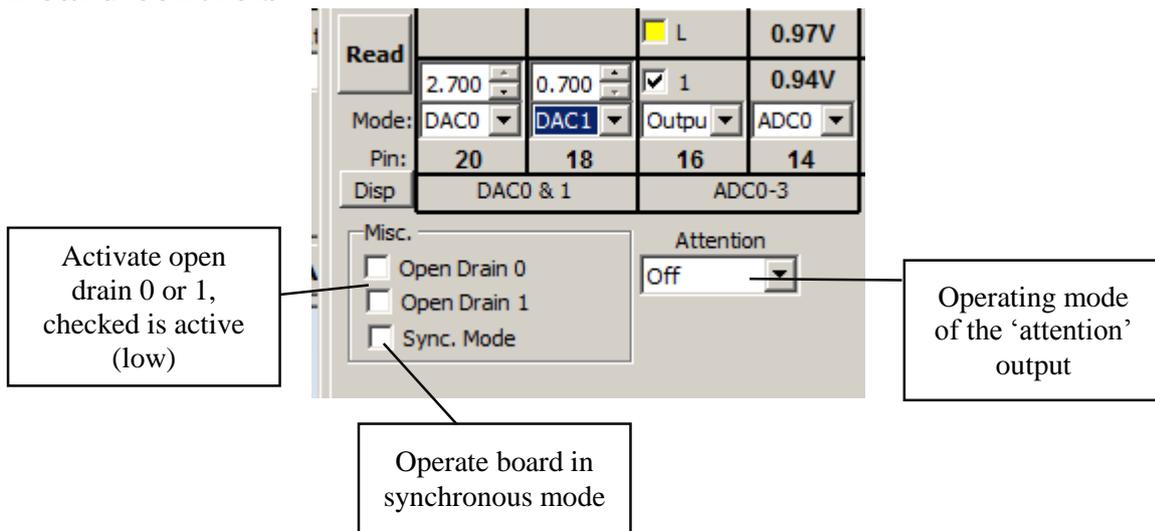


### DAC

By operating the pull-down menu you can set any of the pins 18 or 20 into DAC mode. Beware that the DAC output voltage range is limited. It can lay between ~0.7 and ~2.7 volts.



## Board controls



## Board controls

Activate open drain 0 or 1, checked is active (low)

A board supports two open drain outputs (max 30v max 3A). These check boxes control the open drain output. If a box is checked the open drain output is activated. This has the effect as if closing a switch thus the voltage at the open drain output goes LOW (not high as with an output pin is activated).

Operate board in synchronous mode

In order to start and stop multiple motors at exactly the same time the Gertbot board supports a synchronous operating mode. If this box is checked the synchronous operating mode is active. The board will remember the last 'movement' command and execute them if 'sync' command is received (See *Sends out a SYNC-command*)Sends out a SYNC-command

Switch to motor controls

Pressing this button will get you back to the motor control window.

## DCC controls

The Gertbot supports a DCC mode. DCC is used to control model trains. The Gertbot system lets you send arbitrary commands to the tracks.

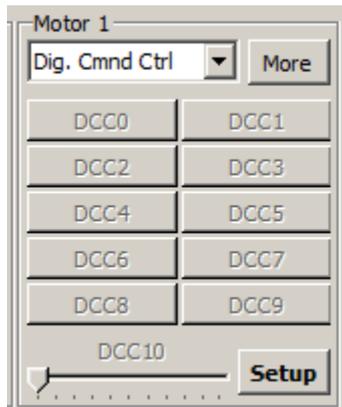
### How not what.

The following is a description of **how** to send commands using the Gertbot Gui. You have to find out **what** commands are required from the manufacturer of the receiver (called a decoder). Some commands described here are taken from the standard and should be accepted by all decoders.

### Where.

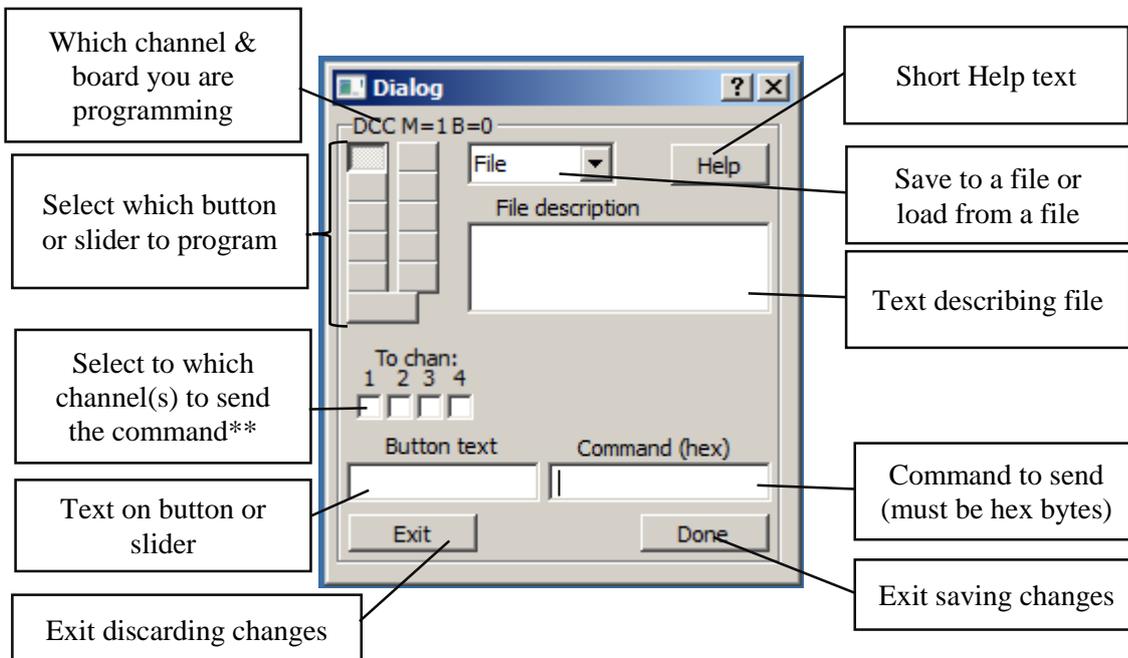
The DCC commands are an exception in that a command is send to a board from there the command can be replicated to *any or all channels*. This means that although the controls are shown for one channel, you can still send commands to any of the other channels as well. The system is safe in that the Gertbot will refuse to send data to channels which are not set up as DCC.

When selecting the DCC operating mode the following controls appear:



That is: you are offered ten buttons and a slider. Pressing a button or moving the slider will send a DCC message. However before it can do so you have to configure the system to tell it what command to send. Thus you must first Set-up the DCC system.

Pressing the Setup button gives you the DCC setup menu:



Here is an example programming button DCC1:

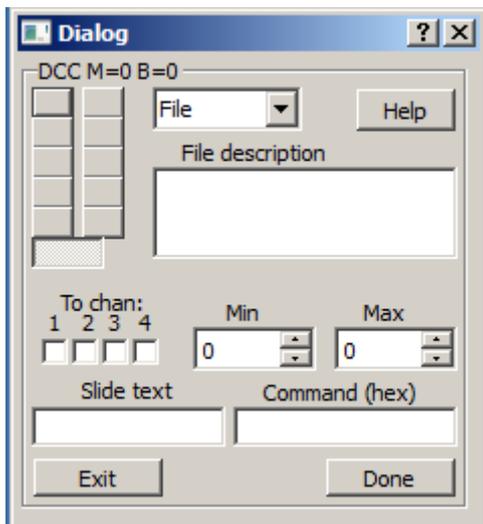


Thus when button 2 is pressed (the button with text “Stop 2”) the command sequence 0x02 0x40 is send to the Gertbot. There it will be replicated to go out on channels 1 *and* 2. The Gertbot will also add the checksum byte (which will be 0x42).

## Slider

Programming the slider is slightly different.

First the slider has two extra controls which specify the slider range.



Second the “Command text” has a special syntax.

In the Command there should be an ‘S’ an ‘>’ or a ‘<’ character.

The position of the ‘<’ or the ‘>’ is replaced with the slider byte value.

You can place two of the same to generate 2 byte commands.

<< gives two bytes LS first

>> gives two byte MS first

20 > 40

Example 1:

Min = -127, Max=127 command is “20 > 40”

When moving the slider the commands generated will range from

0x20 0x81 0x40 (slider at the left hand side)

to

0x20 0x7F 0x40 (slider at the right hand side)

**Example 2:**

Min = 0, Max=4095 command is "20 >> 40"

When moving the slider the commands generated will range from  
0x20 0x00 0x00 0x40 (slider at the left hand side)

to

0x20 0x0F 0xFF 0x40 (slider at the right hand side)

'S' is used to indicate a motor speed. The GUI has two internal tables which hold the motor speed in either 14 or 28 steps as specified by the "S9 packet format" standard: '02DCSSS'.

**14-speed steps.**

The 14-speed step table is selected by:

- Using the 'S' in the command text.
- Setting the min to -14 and the max to 14

The 14-step motor speed table has the C bits always zero.

**24-speed steps.**

The 24-speed step table is selected by:

- Using the 'S' in the command text.
- Setting the min to -28 and the max to 28

The 28-step motor speed table uses the C as LS speed bit.

**Reverse slider**

For *all* of the above the slider operation can be reversed by swapping the max and min value.

**One slider only?**

Yes the Gertbot Gui has only one slider per channel. If you want two sliders you can set another channel to DCC mode and use the control from that to send commands.

**File save load**

You can save or load the DCC setting for one channel or for all boards and all channels. The convention is to use the extension .dcc for a single channel and .dca for all boards and channels.

If you choose to do a 'load all' from a file the data is stored into the system directly. Which means if the 'exit' button (to exit without changes) is no longer applicable.