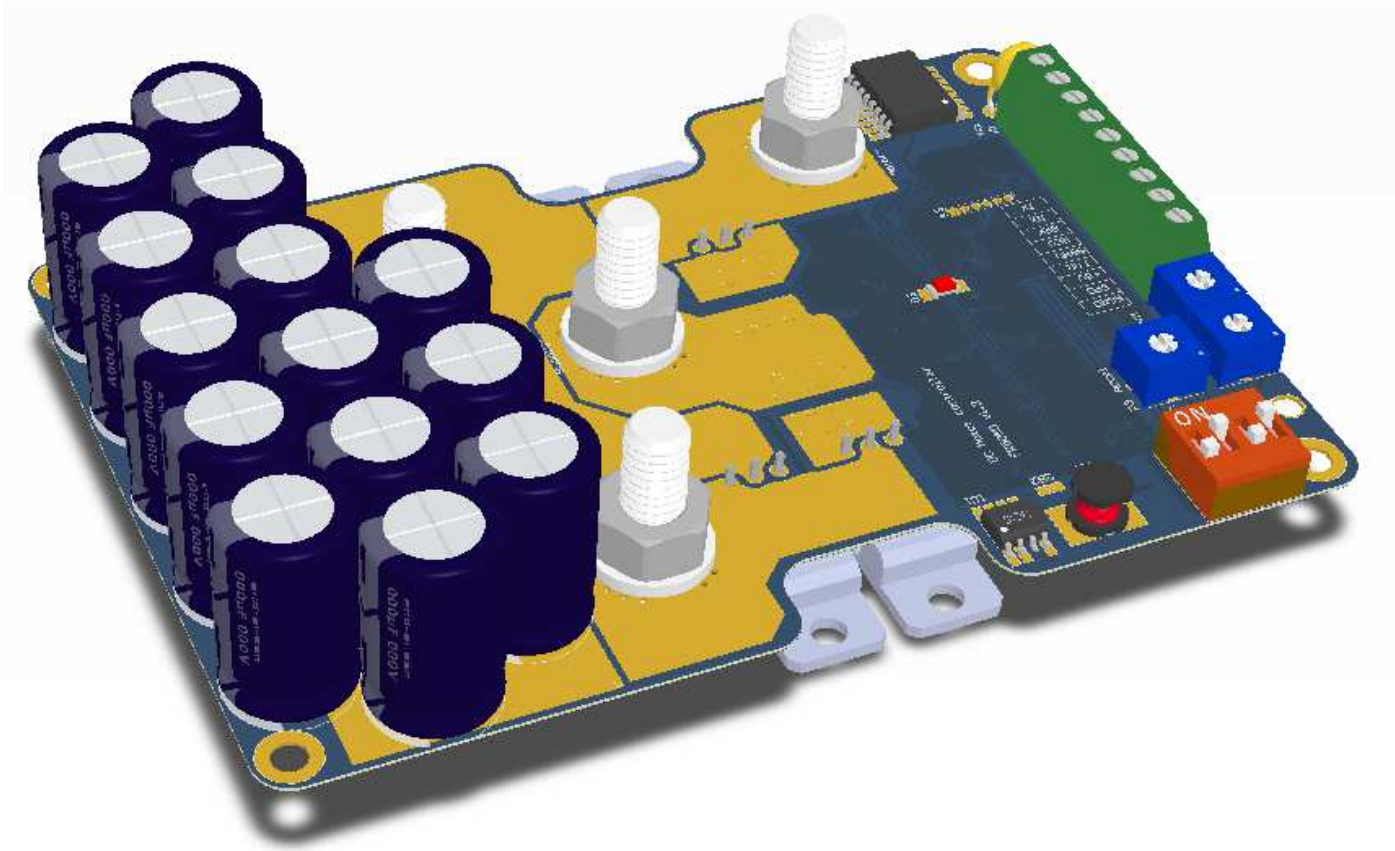
# 50A DC brushed motor PWM speed controller

DATASHEET FOR: Motor PWM FR10050 1.2 (100V version)

**TECHNICAL SPECIFICATIONS:**

| | |
|---|---|
| Supply voltage: | 18-100V |
| Output current: | 70A peak 50A continuous at 25°C **PCB version needs additional heatsink!** |
| Minimum duty cycle: | 0% |
| Maximum duty cycle: | 100% |
| Protections: | Over current protection |
| | Output short protection with minimal load inductance |
| | Over heating protection |
| Switching frequency: | 18kHz |
| Power connection: | M6 |
| Dimensions: | 112 x 75 x 30 mm |



**PIN DESCRIPTION:**

**Mode switch settings with red DIP switch:**

*These settings are only monitored when the controller is turning on. These settings can not be changed when the controller is already running.*

1OFF-2OFF: **Hall throttle or potentiometer mode.**

*Use 0-5V signal on the Speed input to change the output duty cycle. It can be a potentiometer, hall throttle or other analog 0-5V signal. Use the direction switch on Forward Reverse inputs to switch directions.*

1OFF-2ON: **Joystick mode.**

*Use potentiometer or 0-5V analoge signal on the Speed input and change both the output duty cycle and output directions. Center potentiometer generates zero ouput. Turn left or right to increase the output in forward or reverse.*

1ON-2OFF: **0-100% PWM input mode**

*Use 3.3-10V PWM signal on PWM input to change the output duty cycle and use the Forward Reverse inputs to switch directions. If needed optocoupler isolators can be used between the motor controller board and the PWM signal source.*

1ON-2ON: **RC-receiver PWM mode**

*Use 3.3-10V PWM signal on PWM input to change both the output duty cycle and output directions. If needed optocoupler isolator can be used between the motor controller board and the PWM signal source.*

3OFF: **Peak current turned off**

*Current limit settings works as normal reaching the level set by current limit potentiometer. Output current can not be higher than the current limit.*

3ON: **Peak current on**

*Output current can be doubled for a short (5-20sec) period of time the controller allows tipically +30-100% more output current depending on the temperature and the lenght of the peak load.This can allow higher extra acceleration in the motor.*

4ON: **RS232, UART access with simple commands**

*(more infomation in the Simple Mode serial comunication sheet)*

4OFF: **RS232, USART access with advanced commands**

*(more information in the Advanced Mode serial comunication sheet)*

**Motor+:** The motor negative connection.
**Motor-:** The motor positive connection.
**Suppy+:** The positive power supply connection.
**Suppy-:** The negative supply connection.
**Speed GND:** Ground for the speed potentiometer
**Speed:**
Speed potentiometer input. Possible to use 1-10 kOhm linear potentiometer or 5V throttle or 0-5V controll voltage. It behaves differently depending which operation mode is active.

- Potentiometer mode: 0.8V or less = 0% output duty cycle
  4.2V or higher = 100% output duty cycle
- Joystick mode: 0-2.4V = 0-100% output duty and reverse direction
  2.5V = 0% output
  2.6-5V = 0-100% output duty and forward direction
- *USART mode: Optional Analog input measuring external signal.

**+5V:** +5V reference voltage for the speed potentiometer and for small external circuits. Max load current 50mA.

**Forward/Reverse inputs:** Inputs for the forward and reverse signal or switch. Active when it is connected to S-GND input. When both inputs are open the controller's output is disabled.

**Brake/PWM input:**

- Potentiometer or Joystick mode: It behaves as brake switch input. It brakes the output to 0V when Brake input connected to S-GND.
- 0-100% PWM input mode: You can use your own PWM controll signal from your PLC or microprocessor. The minimum controll PWM frequency is 100Hz. The step resolution 1us. The acceptable voltage level: 3.3-10V. 0% PWM (low level) generates 100% output. 100% PWM (high level input) generates 0% output.
- RC-receiver PWM mode: PWM input for RC receiver or other PWM source
  1.5msec-2msec pulse: 0-100% output duty and reverse direction
  2msec pulse: 0% output duty cycle.
  2msec-2.5msec piulse: 0-100% output duty and forward direction
- Possible function to measure PWM, logical and pulse in USART mode

**RX/TX serial communication:**

*Serial communication is availale in all operation modes (potentiometet,joysick, PWM input and RC receiver mode). It is possible to read or set additional parameters in the controller while it is in any operation mode.Other possible option is to controll the the controller with only serial communication commands. The way of communicaton can be Simple mode or Advanced mode. (For information use theserial comunication sheets).*

**Controll potentiometers:**
P1: Curren limit potentiometer 0-50A from CCW to CW.
P2: Acceleration limit potentiometer 0-41sec
P3: Regenerative current limit 0-50A from CCW to CW.

**LED light functions:**

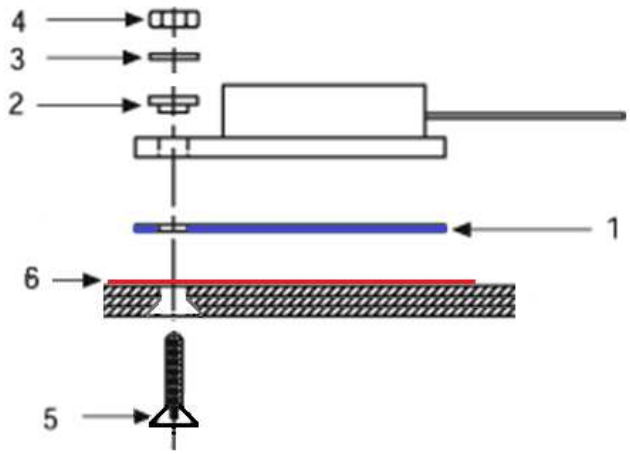
-2 blinks at starting up.
-At "potentiometer mode"operation when both Forward and Reverse inputs are open.
-Output current limitation is active.Output current reached the current limitation level.
-Over voltage protection. Input voltage higher than the allowed maximum level.
-Under  voltage protection. Input voltage lower than the minimal input voltage
-Over temperature protection is active. The temperature sensor reaching 65 °C temperature.
-While UART communication. A short blink at every sent or received serial communication command.

**Istructions for the heatsink:**
The PCB version of the controller requires an additional heatsink or aluminium case to cool the power transistors. The dimensions of the heatsink may change depending on the supply voltage output current ambient temperature and the air convection. Normally use a 100mm x 100mm x 25mm (4" x 4" x 1") aluminium heatsink or equivalent aluminium case.
**Heat sensor:** The controller also has a heat sensor with 15cm cable. Fix this sensor to the heatsink as well just like the power transistors. But it is not important to isolate this sensor.

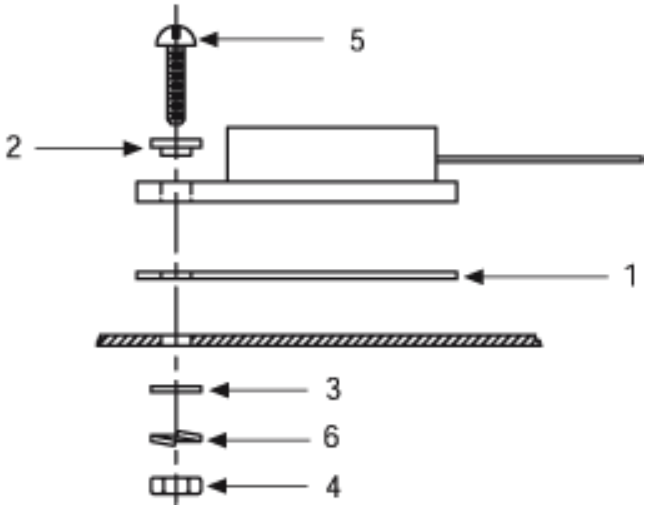
Step by step:

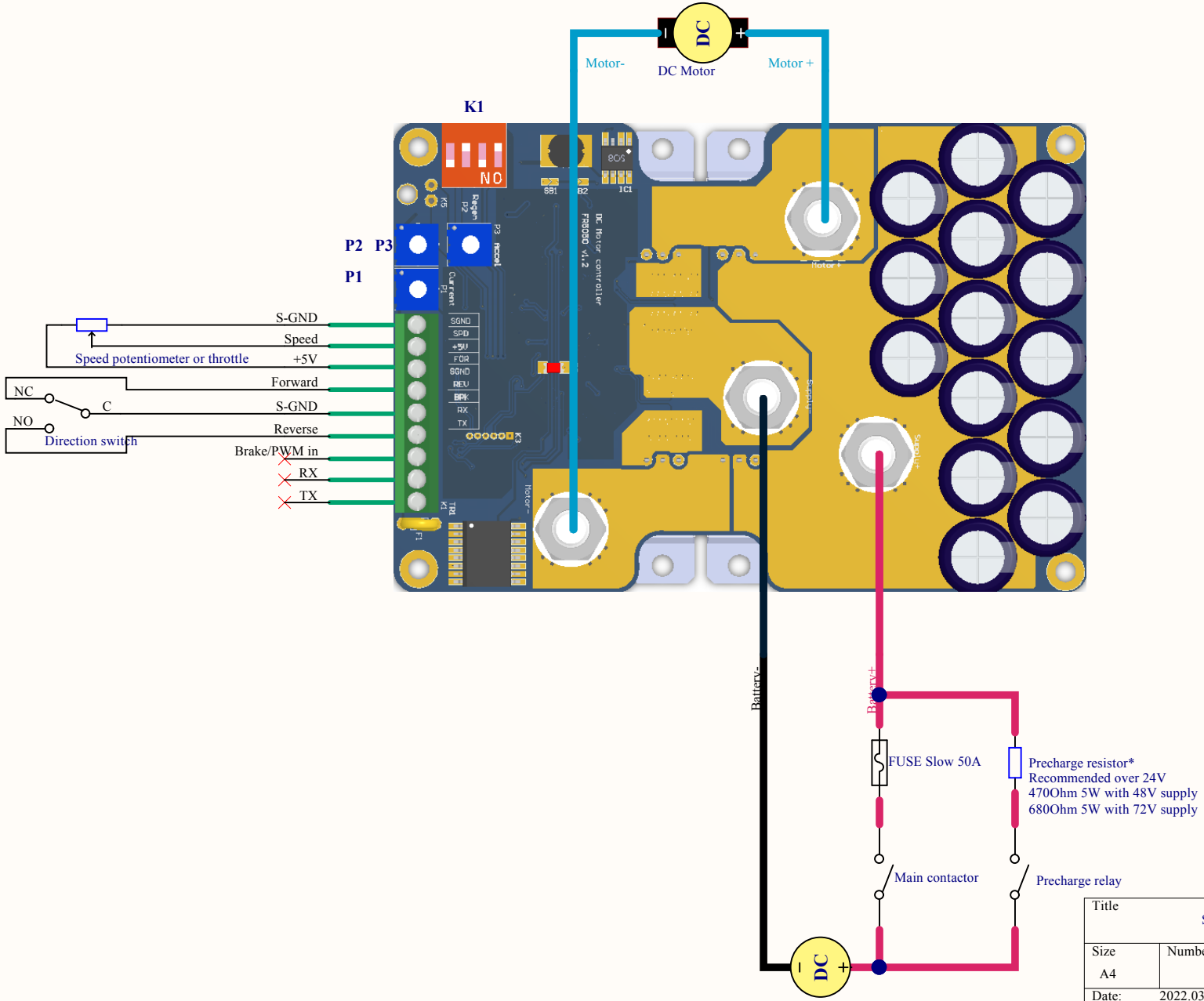
1. In the right positions drill holes for the power transistors and fixing the board corners. If you have self tapping screws drill 2,5mm holes. For metric screws with nuts use M3 holes.
2. The hole's edges needs top be clear and burr free (small burr under the transistors can harm and penetrate the isolators causing short between the transistors and the heatsink).
3. Place heat conductor paste (6) under the isolator.Quantity is not important it need to be only a thin layer.
4. Place the silicone isolators in place. (1)
5. Make sure all the transistors flags are parallel with the heatsink. Having gap under the transistors flags can cause bad thermal contact overheating and damage.
5. Place the board and the transistors on the heatsink and fix it with the screws (5). Use the small white plastic isolator rings (2) under the screws or nuts to avoid the short between the metal flags and the screws /heatsink.
6. Make sure all the screws are tight enough to have the best thermal contact.
7. Check the isolation between the transistors flags and the heatsink with digital ohm meter

<u>Fixing from down side:</u>

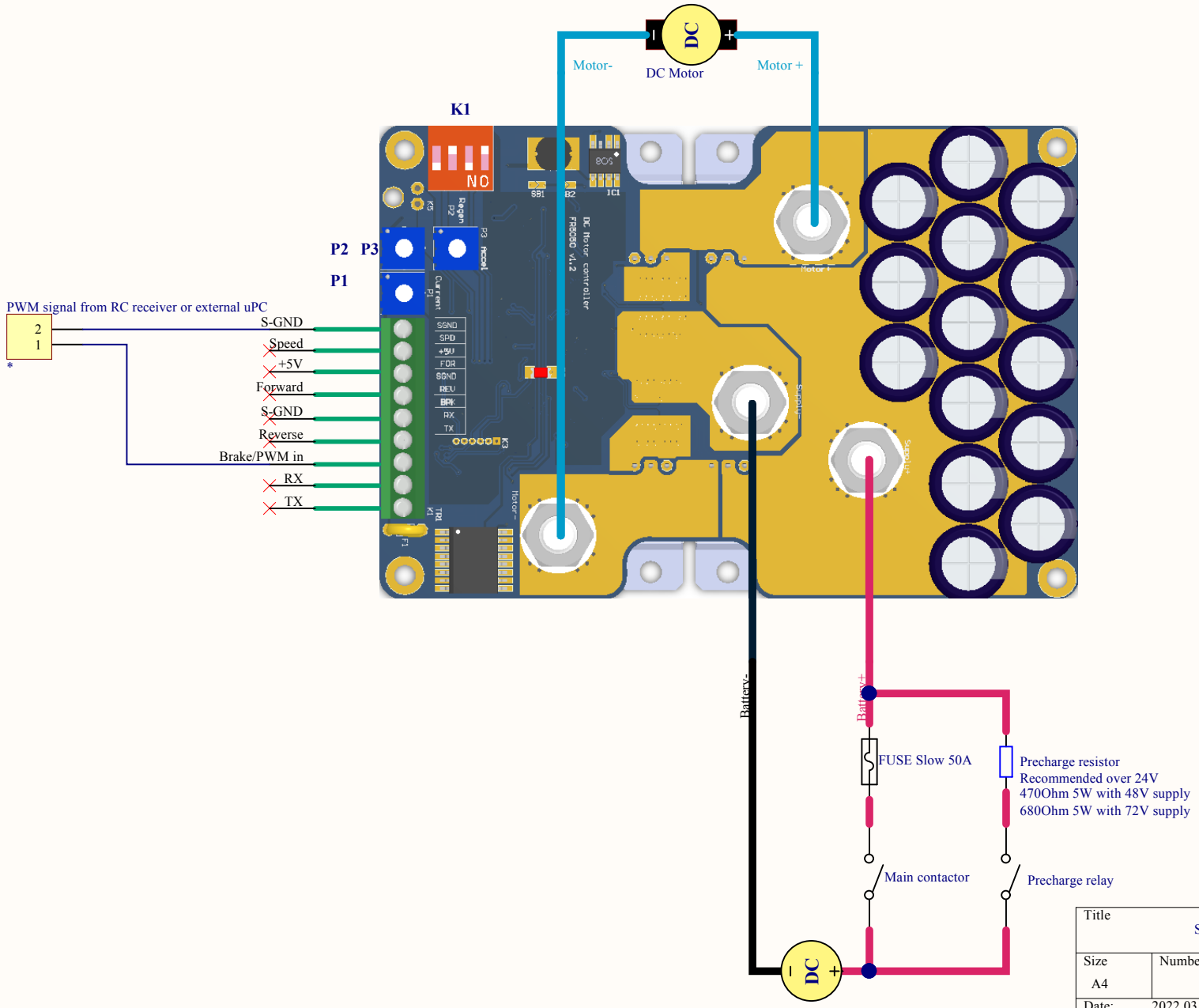



<u>Fixing from up side:</u>






| | | |
|---|---|---|
| Title | Standard potentiometer or throttle | |
| Size A4 | Number FR10050 v1.2 | Revision Rev1 |
| Date: 2022.03.24. | | Sheet of |
| File: E:\Projektek\..\Wir1.SchDoc | | Drawn By: |




| | | | |
|---|---|---|---|
| Title | Standard potentiometer or throttle | | |
| Size A4 | Number FR5050 v1.2 | Revision Rev1 | |
| Date: 2022.03.24. | | Sheet of | |
| File: E:\Projektek\..\Wir2.SchDoc | | Drawn By: | |

# Controll register map for simple UART communication:

## UART Data transmission:

*revision: 1.* *UART firmware v1.0*
Supported default baud rate: 9600 baud/sec
Supported bit rate: 1 start 8 data bit 1 stop and no parity bit.

Writing register you need to send first the comand byte than the second the data byte which contains the value of the wrtiten register. Reading a register send the request command and wait for the data from the controller.

### Sent bytes

| Command | Data | Write registers: | remark: |
|---|---|---|---|
| 0x80 | 0x00-7F | speed motor 1 | |
| 0x82 | 0x00-7F | current limit | 0,47A/lsb |
| 0x83 | 0x00-7F | regen limit | 0,47A/lsb |
| 0x84 | 0x00-7F | accel limit | 0,32ms/lsb |
| 0x85 | 0x00-7F | decel limit | 0,32ms/lsb |
| 0x86 | 0x00-7F | turn off low | 1,23V/lsb |
| 0x87 | 0x00-7F | turn on low | 1,23V/lsb |
| 0x88 | 0x00-7F | turn on high | 1,23V/lsb |
| 0x89 | 0x00-7F | turn off high | 1,23V/lsb |
| 0x8A | 0x00-03 | motor deirection | 0x00: stop, 0x01 forward, 0x02: reverse, 0x03 regen |
| 0xE0 | - | USART command mode | Autostop after 3 sec without speed update |
| 0xE1 | - | USART command mode | No Autostop |
| 0xE2 | - | cancel USART mode | Controll with onboard potentiometers |
| 0xA0 | 0x00-7F | saving_bits L | for saving use advanced mode |
| 0xA1 | 0x00-7F | saving_bits H | for saving use advanced mode |

| Command | | Read registers: | |
|---|---|---|---|
| 0xC0 | 00-7F | read motor 1 speed | |
| 0xC2 | 00-7F | read current limit | 0,47A/lsb |
| 0xC3 | 00-7F | read regen limit | 0,47A/lsb |
| 0xC4 | 00-7F | read accel limit | 0,32ms/lsb |
| 0xC5 | 00-7F | read decel limit | 0,32ms/lsb |
| 0xC6 | 00-7F | read turn off low | 1,23V/lsb |
| 0xC7 | 00-7F | read turn on low | 1,23V/lsb |
| 0xC8 | 00-7F | read turn on high | 1,23V/lsb |
| 0xC9 | 00-7F | read turn off high | 1,23V/lsb |
| 0xCA | 00-7F | motor 1 current | 0,94A/lsb |
| 0xCC | 00-7F | suppyl voltage | 1,23V/lsb |
| 0xCD | 00-7F | speed input | 37,7mV=lsb |
| 0xCE | | heatsink_temp | 1C |
| 0xCF | | mainboard_temp | 1C |
| 0xD0 | | status bits L | |

| | |
|---|---|
| 0xD1 | status bits H |
| 0xD2 | configurations_bits_1 L |
| 0xD3 | configurations_bits_1 H |
| 0xD4 | configurations_bits_2 L |
| 0xD5 | configurations_bits_2 H |
| | |
| 0xDE | software version |
| 0xDF | hardware version |

If you need more details about what the status register and confoguration_bits are doing please check the advanced communication.

## Examples:

1, Start the controller in uart controll mode:

**Send 0xE0 or 0xE1** (with E0 you need to send a new speed data in every 3 sec otherwise the controller will stop the motor)

2, Select direction forward:

**Send 0x8A 0x01**

3, Speed up the motor slowly:

Acceleration set: **Send 0x84 0x0F**
Speed set: **Send 0x80 0x1F**

4, Set lower 50% speed:

**Send 0x80 0x3F**

5, Read motor current:

**Send 0xCA**

6, Read supply voltage:

**Send 0xCC**

7, Select direction reverse:

**Send 0x8A 0x02**

# UART Data transmission:

*revision: 1.* *UART firmware v1.0*

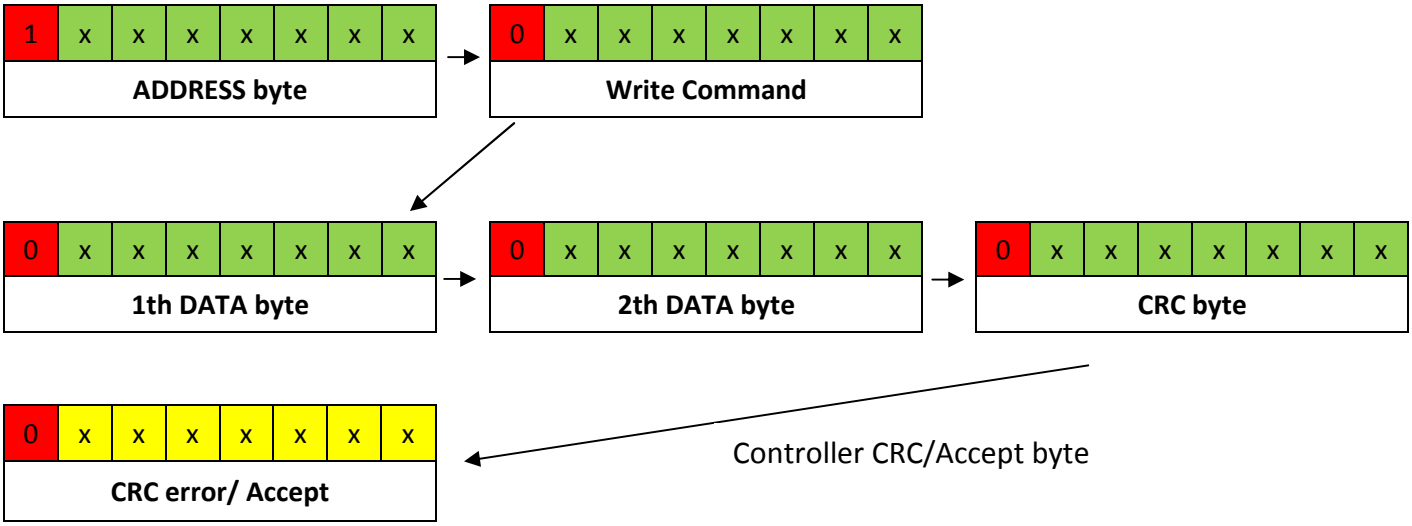
Supported default baud rate: 19200 baud/sec

Supported bit rate: 1 start 8 data bit 1 stop and no parity bit.

**Sending data:**

The controllers are working in SLAVE mode UART communication. It means that every commands are addressable and only the controller which has the same address will accept the command otherwise it goes into stanby mode. When an address match occurs the the controller wakes up and accept the data. As long as the controller does not get other address which is not its own address remains in active mode.

Sending data from master device must be 5 bytes:




**Every ADDRESS byte MSB bit must be "1" any other which is not an ADDRESS byte must be "0".**

For robust and errorless communication CRC byte can be used to check the data transmission errors.

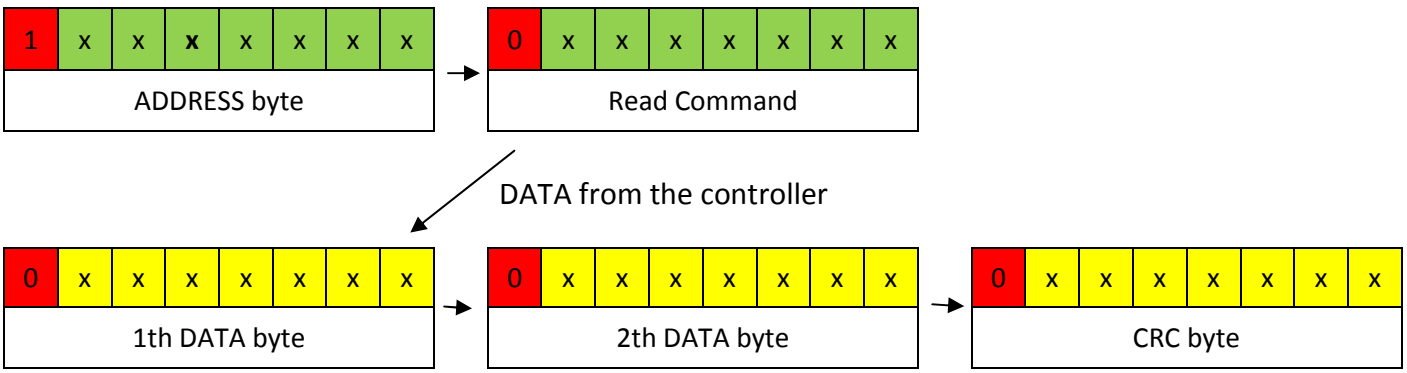
CRC ckeck byte calculated: ((Write command byte + 1th DATA byte + 2th DATA byte) & 0x7F)

Receiving successfully the write command and the DATA bytes the controller send a confirmation byte with his UART address (0x00 | address) or CRC error message with his address (0x80 | adddress).You can use this byte to know when to send the next data.

**Reading data:**

Requesting data from master device must be 2 bytes when the controller received the 2th byte it will send the answer 3 bytes (1th DATA byte ; 2th DATA byte ; CRC)

CRC ckeck byte calculated: ((Read command byte + 1th DATA byte + 2th DATA byte) & 0x7F)



# Controll register map for advanced USART communication:

Model: BR10050 1.1F (100V)

| Register name | Command | | Min Val. | Max val. | Value Range | Resolution | Remark |
|---|---|---|---|---|---|---|---|
| | Read | Write | | | | | |
| speed motor 1 | 0x00 | 0x40 | 0x0000 | 0x0FFF | 97% duty | | Other factors may change the output duty |
| | | | | | | | |
| current limit | 0x02 | 0x42 | 0x0000 | 0x0FFF | 60.3A | 14.7mA | This value is internally limited az 50A. |
| regen limit | 0x03 | 0x43 | 0x0000 | 0x0FFF | 60.3A | 14.7mA | This value is internally limited at 50A. |
| accel limit | 0x04 | 0x44 | 0x0000 | 0x0FFF | 41 sec | 10msec | |
| decel limit | 0x05 | 0x45 | 0x0000 | 0x0FFF | 41 sec | 10msec | |
| turn off low | 0x06 | 0x46 | 0x0000 | 0x0FFF | 158.4V | 38.68mV | |
| turn on low | 0x07 | 0x47 | 0x0000 | 0x0FFF | 158.4V | 38.68mV | |
| turn on high | 0x08 | 0x48 | 0x0000 | 0x0FFF | 158.4V | 38.68mV | Defaul value: 2600 |
| turn off high | 0x09 | 0x49 | 0x0000 | 0x0FFF | 158.4V | 38.68mV | Internally limited on 2701 (49,9V) |
| Heatsink Temperature | 0x0A | | 0x000A | 0x0050 | 80°C | 1°C | Shut down at 75°C |
| Mainboard Temperature | 0x0B | | 0x000A | 0x0050 | 80°C | 1°C | |
| | 0x0F | | | | | | |
| Supply Voltage | 0x10 | | | | 158.4V | 38.68mV | |
| Speed ADC Input | 0x11 | | | | 4.85V | 1.18mV | |
| Load Current 1 | 0x12 | | | | 180A | 58.9mA | 0A=0x07FF |
| | | | | | | | |
| Input Frequency Period Register | 0x14 | | | | | | |
| Input Pulse Width | 0x15 | | | | | | |
| | 0x19 | | | | | | |
| I*R compensation gain | 0x20 | 0x60 | 0x0000 | 0x0FFF | | | |
| top speed | 0x21 | 0x61 | 0x0000 | 0x0FFF | 158.4V | 38.68mV | |
| potentiometer min | 0x22 | 0x62 | 0x0000 | 0x0FFF | 4.85V | 1.18mV | |
| potentiometer max | 0x23 | 0x63 | 0x0001 | 0x0FFF | 4.85V | 1.18mV | |
| | 0x24 | | | | | | |
| Status Register | 0x30 | | | | | | |
| Lock_level_2 Variables | 0x31 | 0x71 | | | | | |
| Save bites 1 | 0x32 | 0x72 | | | | | |
| save bites 2 | 0x33 | 0x73 | | | | | |
| Configuration bits 1 | 0x34 | 0x74 | | | | | |
| Configuration bits 2 | 0x35 | 0x75 | | | | | |
| | | | | | | | |
| UART Address | | 0x7C | 0x0000 | 0x007F | | | Default address 0x00 |
| Current measurement calibration | | 0x7D | 0x0000 | | | 58.9mA | |
| software version | 0x3E | | | | | | |
| hardware version | 0x3F | | | | | | |

# Speed Register:

Type: Write / Read
Write command: 0x40
Read command: 0x00

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

Speed register gives the base value of the output voltage (duty cycle). The final value depending some other factors. The final duty cycle depending on several other factors (current limits, regen modes..)

With default settings the speed register equal to the controller speed input ADC value modified by the acceleration/deceleration settings. Writing into this register this will be the new base value of the output voltage as long as the Speed input ADC has the same value as it was before the overwrite. When the Speed input voltage changes (turning up/down the speed potentiometer) the controller going to overwite the register with the value of the potentiometer.

It is possible to lock the register value (speed potentiometer can't overwrite the speed register) for this feature use the **Lock Variable Register**.

The register set and cleared by UART write or Speed potentiometer input and loses it's value after restarting the controller. Saving the value into the FLASH use the **Save Register**.

Bit 11:0 **SPEED [11:0]:** Speed register value
*Max value is 4095 it is the 97% duty cycle.*
*Min value is 0 it is the 0% duty cycle.*

*Note: final duty cycle may change depending on other factors.*

# Current Limit Register:

Type: Write / Read
Write command: 0x42
Read command: 0x02

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

Current Limit Register sets the value of the current limitation. With default settings the current limit register is equal to the controller's current limit potentiometer setting. Writing into this register overwites the value of the potentiometer and holds the value as long as the potentiometer has the same value as it was before the overwrite. When the potentiometer changes (turning up/down the potentiometer) the controller going to overwite the register with the value of the potentiometer.

It is possible to lock the register value (potentiometer can't overwrite the register) for this feature use the **Lock Variable Register**.

The register set and cleared by UART write or the current limit potentiometer and loses it's value after restarting the controller. Saving the value into the FLASH use the **Save Register**.

Bit 11:0 **CURRENT_LIMIT [11:0]:** Curretn limit register value

*Max value is 4095*

*Min value is 0*

*Note: for the corresponding curret value check the **Register Value Range** table.*

## Regeneraive Current Limit Register:

Type: Write / Read
Write command: 0x43
Read command: 0x03

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

Regenerative Current Limit Register sets the value of the current limitation. With default settings the current limit register is equal to the controller's potentiometer setting. Writing into this register overwites the value of the potentiometer and holds the value as long as the potentiometer has the same value as it was before the overwrite. When the potentiometer changes (turning up/down the potentiometer) the controller going to overwite the register with the value of the potentiometer.

It is possible to lock the register value (potentiometer can't overwrite the register) for this feature use the **Lock Variable Register**.

The register set and cleared by UART write or the potentiometer and loses it's value after restarting the controller. Saving the value into the FLASH use the **Save Register**.

Bit 11:0 **CURRENT_LIMIT [11:0]:** Regenerative current limit register value
*Max value is 4095*
*Min value is 0*

*Note: for the corresponding curret value check the **Register Value Range** table.*

# Acceleration Limit Register:

Type: Write / Read
Write command: 0x44
Read command: 0x04

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

Acceleration Limit Register sets the time how fast the output follows the speed register's value under raising condition. With default settings it is equal to the controller's acceleration (RAMP) potentiometer's setting. Writing into this register overwites the value of the potentiometer and holds the value as long as the potentiometer has the same value as it was before the overwrite. When the potentiometer changes (turning up/down the potentiometer) the controller going to overwite the register with the value of the potentiometer.

It is possible to lock the register value (potentiometer can't overwrite the register) for this feature use the **Lock Variable Register**.

The register set and cleared by UART write or the potentiometer and loses it's value after restarting the controller. Saving the value into the FLASH use the **Save Register**.

Bit 11:0 **CURRENT_LIMIT [11:0]:** Acceleration limit register value
*Max value is 4095*
*Min value is 0*

*Note: for the corresponding time value check the **Register Value Range** table.*

## Deceleration limit register:

Type: Write / Read
Write command: 0x45
Read command: 0x05

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

Deceleration Limit Register sets the time how fast the output follows the speed register's value under falling condition. With default settings it is disabled in the controller.
Saving the value into the FLASH use the **Save Register**.

Bit 11:0 **DECELERATION_LIMIT [11:0]:** Deceleration limit register value
*Max value is 4095*
*Min value is 0*

*Note: for the corresponding time value check the **Register Value Range** table. The minimum value is 1.*

# Turning off minimum voltage register:

Type: Write / Read
Write command: 0x46
Read command: 0x06

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

Turning Off Minimum voltage register sets the voltage level where the controller will shut down if the supply voltage is lower than the register value. With default settings it is disabled in the controller.Writing and locking this register activates the function. When the controller shuting down because of the low supply voltage level the corresponding status register gets an update about the low supply voltage.

Saving the value into the FLASH use the **Save Register**.

Bit 11:0 **TURN_OFF_MIN [11:0]:**

*Max value is 4095*
*Min value is 0*

*Note: for the corresponding voltage level check the **Register Value Range** table.*

# Turning on minimum voltage register:

Type: Write / Read
Write command: 0x47
Read command: 0x07

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

Turning On Minimum voltage register sets the voltage level where the controller will turn back from low supply status if the supply voltage is higher than the register value. With default settings it is disabled in the controller.Writing and locking this register activates the function.

Saving the value into the FLASH use the **Save Register**.

Bit 11:0 **TURN_ON_MIN [11:0]:**

*Max value is 4095*
*Min value is 0*

*Note: for the corresponding voltage level check the **Register Value Range** table.*

## Turning on maximum voltage register:

Type: Write / Read
Write command: 0x48
Read command: 0x08

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

Turning On maximum voltage register sets the voltage level where the controller will turn back from high supply voltage status if the supply voltage is lower than the register value. With default settings it is disabled in the controller.Writing and locking this register activates the function.

Saving the value into the FLASH use the **Save Register**.

Bit 11:0 **TURN_ON_MAX [11:0]:**

*Max value is 4095*
*Min value is 0*

*Note: for the corresponding voltage level check the **Register Value Range** table.*

# Turning off maximum voltage register:

Type: Write / Read
Write command: 0x49
Read command: 0x09

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

Turning Off Maximum voltage register sets the voltage level where the controller will shut down if the supply voltage is higher than the register value. With default settings it is disabled in the controller.The internal maximum voltage settings may rewrite this register. Writing and locking this register activates the function. When the controller shuting down because of the high supply voltage level the corresponding status register gets an update about the high supply voltage.

Saving the value into the FLASH use the **Save Register**.

Bit 11:0 **TURN_OFF_MAX [11:0]:**

*Max value is 4095*
*Min value is 0*

*Note: for the corresponding voltage level check the **Register Value Range** table.*

## Heatsink Temperature Register:

Type: Read
Read Command: 0x0A

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | - | - |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

This register holds the actual heatsink temperature value in °C.

Bit 11:0 **HEATSINK TEMPERATURE [11:0]:**
*Max value is 80*
*Min value is 10*

*Note: for the corresponding temperature value check the **Register Value Range** table.*

## Mainboard Temperature Register:

Type: Read
Read Command: 0x0B

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | - | - |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

This register holds the actual mainboard temperature value in °C.

Bit 11:0 **MAINBOARD TEMPERATURE [11:0]:**
*Max value is 4095*
*Min value is 0*

*Note: for the corresponding temperature value check the **Register Value Range** table.*

## Supply Voltage Register:

Type: Read
Read Command: 0x10

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

This register holds the actual supply voltage value.

Bit 11:0 **SUPPLY VOLTAGE [11:0]:**
*Max value is 4095*
*Min value is 0*

*Note: for the corresponding voltage value check the **Register Value Range** table.*

## Speed ADC Input Register:

Type: Read
Read Command: 0x11

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

Controlling and locking the controller with UART Speed commands the Speed ADC input can serve other task like measuring a sensor or anything else.

Using the Speed ADC input for other task and not for the speed controll function the Speed Register Lock bit must be enabled.

Bit 11:0 **SPEED ADC INPUT [11:0]:**
*Max value is 4095*
*Min value is 0*

*Note: for the corresponding voltage value check the **Register Value Range** table.*

## Load Current Register:

Type: Read
Read Command: 0x12

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

This register holds the actual otput load current.

Bit 11:0 **LOAD CURRENT [11:0]:**
*Max value is 4095*
*Min value is 0*

*Note: for the corresponding curret value check the **Register Value Range** table.*

## Input Frequency Period Register:

Type: Read
Read Command: 0x14

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

The controller PWM/brake input can be used for measuring pulses and other purposes. Like hall sensors encoders etc. Input Frequency Register holds the period's of the measured signal.

Bit 13:0 **INPUT FREQUENCY PERIOD REGISTER [13:0]:**
*Max value is 16383*
*Min value is 0*

*Note: for the corresponding value check the **Register Value Range** table.*

## Input Pulse Width Register:

Type: Read
Read Command: 0x15

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

The controller PWM/brake input can be used for measuring pulses and other purposes. Like hall sensors encoders etc. Input Pulse Width Register holds the width of the captured pulse of the measured signal.

Bit 13:0 **INPUT PULSE WIDTH REGISTER [13:0]:**
*Max value is 16383*
*Min value is 0*

*Note: for the corresponding time value check the **Register Value Range** table.*

# I x R compensation register:

Type: Write / Read
Write Command: 0x60
Read Command: 0x20

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

I x R compensation value can be used for gain the output voltage when the motor internal resistance causing voltage drop and RPM drop. This value will be multiplied with the load current and added to the output voltage. Using the I X R compensation constant from the Register Value Range table you can calculate the needed value.
The [%/A] tell how much compensation is needed for running the motor at the same RPM under load.

Bit 13:0 **I x R COMPENSATION REGISTER [13:0]:**
*Max value is 16383*
*Min value is 0*

*Note: for the corresponding value check the **Register Value Range** table.*

# Top Speed register:

Type: Write / Read
Write Command: 0x61
Read Command: 0x21

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

Top Speed value can be used for making a reference voltage for the speed input or speed register. This can make a supply compensation when the supply voltage is changes or you can limit the maximum output output voltage / motor RPM and does not matter if the speed potentiometer would set higher output voltage the Top Speed will overwrite this value.

For the calculation use the given top speed constant from the ***Register Value Range*** table.

*Giving and example: If the constant for the given controller is 18,51mV than if you need a maximum output voltage of 20V than calculate like this: Top Speed= 20V/0.01805V = 1080.*

*So writing the 1080 in the top speed register you can set maximum 20V output voltage with the potentiometer. If your supply voltag changes between higher ranges for example 25-50V your output voltage remains the same 20V or less depending on your speed potentiometer's setting.*

With TOP Speed settings you can have supply compensation fixing the speed variation with different supply voltages.

Turning on the supply compesation you need to turn on the **Top Speed /Supply compensation** bit in the **Configuration_bits_2** register.

Giving 0 or near 0 value of this register can cause 0V output voltage.

Bit 13:0 **TOP SPEED REGISTER [11:0]:**

*Max value is 4095*
*Min value is 0*

*Note: for the corresponding value check the **Register Value Range** table.*

# Potentiometer min. register:

Type: Write / Read
Write Command: 0x62
Read Command: 0x22

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

With Potentiometer min and max you can define your voltage levels on the speed input what you need for the given 0% output voltage and the given 100% output voltage.
For the calculation use the given potentiometermin/max constant from the ***Register Value Range*** table.

*Giving and example: If the constant for the given controller is 1,184mV than if you need the 0% output starting form the speed input=1V*

*Potentiometer min register = 1V/1,184mV = 844*

Using this function you need to save the potentiometer min/max registers in the flash memory using the ***Save register*** and restart the controller with SW mode setting: 1:OFF 2:OFF.

Bit 13:0 **Potentiometer Min register [11:0]:**
*Max value is 4095*
*Min value is 0*

*Note: for the corresponding value check the **Register Value Range** table.*

# Potentiometer max. register:

Type: Write / Read
Write Command: 0x63
Read Command: 0x23

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

With Potentiometer min and max you can define your voltage levels on the speed input what you need for the given 0% output voltage and the given 100% output voltage.
For the calculation use the given potentiometermin/max constant from the ***Register Value Range*** table.

*Giving and example: If the constant for the given controller is 1,184mV than if you need the 100% output ending at the speed input=3,5V*

*Potentiometer maxregister = 3,5V/1,184mV = 2956*

Using this function you need to save the potentiometer min/max registers in the flash memory using the ***Save register*** and restart the controller with SW mode setting: 1:OFF 2:OFF.

Bit 13:0 **Potentiometer Max register [11:0]:**
*Max value is 4095*
*Min value is 0*

*Note: for the corresponding value check the **Register Value Range** table.*

## Status Register:

Type: Read
Read Command: 0x30

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | - | RM | FM | RPin | FPin |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | BR pin | - | DT | - | OT | OV | UV |

Bit 10 **RM:** Reverse mode:
0: Stop (controller will stop if both Forward mode and Reverse mode is 0)
1: Controller in reverse mode.

Bit 9 **FM:** Forward mode:
0: Stop (controller will stop if both Forward mode and Reverse mode is 0)
1: Controller in forward mode.

Bit 8 **BR Pin:** Brake input pin sate bit.
0: Brake pin is "Low"
1: Brake pin is "High"
*Note: reading this you need to be not in PWM mode.*

Bit 7 **BR Pin:** Brake input pin sate bit.
0: Brake pin is "Low"
1: Brake pin is "High"
*Note: reading this you need to be not in PWM mode.*

Bit 6 **BR Pin:** Brake input pin sate bit.
0: Brake pin is "Low"
1: Brake pin is "High"
*Note: reading this you need to be not in PWM mode.*

Bit 4 **DT:** Drive status bit:
0: Drive mode
1: Brake mode

Bit 2 **OT:** Over temperature status bit.
0:
1: Supply voltage is lower than the Turn on minimum register value.

Bit 1 **OV:** Over voltage status bit.
0:
1: Supply voltage is higher than the Turn off maximum register value.

Bit 0 **UV:** Under voltage status bit.
0:
1: Supply voltage is lower than the Turn on minimum register value.

## Lock Variables Register:

Type: Write / Read
Write Command: 0x71
Read Command: 0x31

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | - | - |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | - | ACC | REG | CUR | SPD |

Locking any register means only USART update can overwrite it's value. The on board potentiometers will be disabled as long as the associated bit is "1".

This is good if you want to deny the possibility to change the value of the registers on board.
Other advantage if you want to use the Speed input for other purposes than you need to lock the SPD register otherwise the value of the speed input will be used for setting the output voltage.

Bit 3 **ACC:** Writing "1" locks the Acceleration Limit Register.

Bit 2 **REG** Writing "1" locks the Regeneration Limit Regitster.

Bit 1 **CUR:** Writing "1" locks the Current Limit Register.

Bit 0 **SPD:** Writing "1" locks the Speed Register.

*Note: This register going to lose is't content after restarting the controller.*

## Save bits_1 register:

Type: Write / Read
Write Command: 0x72
Read Command: 0x32

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | CONF2 | CONF1 | ADDR | - | OFF_H | ON_H | ON_L |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | OFF_L | DEC | ACC | REG | CUR | - | SPD |

These bits are set all the registers you want to save into the flash memory. Writing into this register only designates the register to ba saved. The actual saving command will be done only with writing the Save_bits_2 register. When a memory saving occurs the data will be reloaded from the memory after every restart. Saving a register data automatically locks the particular register and only USART update can rewrite it.

Bit 13 **CONF2:** Configuration 2 Register.
0: no saving
1: Save the given register into flash memory.

Bit 12 **CONF1:** Configuration 1 Register.
Refer to the Bit 13 CONF2 description.

Bit 11 **ADDR:** Configuration 2 Register.
Refer to the Bit 13 CONF2 description.

Bit 9 **OFF_H:** Save turning off maximum value.
Refer to the Bit 13 CONF2 description.

Bit 8 **ON_H:** Save turning on maximum value.
Refer to the Bit 13 CONF2 description.

Bit 7 **ON_L:** Save turning off minimum value.
Refer to the Bit 13 CONF2 description.

Bit 6 **OFF_L:** Save turning off low value.
Refer to the Bit 13 CONF2 description.

Bit 5 **DEC:** Save deceleration value.
Refer to the Bit 13 CONF2 description.

Bit 4 **ACC:** Save acceleration value.
Refer to the Bit 13 CONF2 description.

Bit 3 **REG:** Save regenerative current limit value.
Refer to the Bit 13 CONF2 description.

Bit 2 **CUR:** Save current limit value.
Refer to the Bit 13 CONF2 description.

Bit 0 **SPD:** Save speed value.
Refer to the Bit 13 CONF2 description.

## Save bits_2 register:

Type: Write / Read
Write Command: 0x72
Read Command: 0x33

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | - | - |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | P_MAX | P_MIN | TOP_S | IR_COM | C_CAL |

These bits are set all the registers you want to save into the flash memory. Writing into this register designates the register to ba saved and initiates the saving command. When a memory saving occurs the data will be reloaded from the memory after every restart. Saving a register data automatically locks the particular register and only USART update can rewrite it.

IMPORTANT: Writing into this register as a repeated operation is not recommended**. The FLASH memory rewritable life cycle is limited about 10.000 saving operation! Under the save operation the PWM controller stops and no other operation can be used!**

Bit 4 **P_MAX:** Potentiometer Maximum
0: no saving
1: Save the given register into flash memory.

Bit 3 **P_MIN:** Potentiometer Minimum
Refer to the Bit 8 OFH description.

Bit 2 **TOP_S:** Top Speed/Supply compensation
Refer to the Bit 8 OFH description.

Bit 1 **IR_COM:** I x R compensation
Refer to the Bit 8 OFH description.

Bit 0 **C_CAL:** Current calibration
Refer to the Bit 8 OFH description.

## Configuration bits 1 register:

Type: Write / Read
Write Command: 0x74
Read Command: 0x34

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | - | RD | FD | - | - |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | DM | BM | SRC_2 | SRC_1 | UC |

Bit 10 **RD:** Reverse Direction:
0: Stop
1: Set reverse direction.
*Note: when you are changing direction keep the* **UC** *bit high and* **SRC** *bits in the desired mode.*

Bit 9 **FD:** Forward Direrection:
0: Stop
1: Set forward direction.
*Note: when you are changing direction keep the* **UC** *bit high and* **SRC** *bits in the desired mode.*

Bit 4 **DM:** Selecting operation mode
*This bit set and cleared by UART write and cleared by hardware after restart.*
0: Drive mode.
1: Brake mode or free running mode depending on **BM** bit.

Bit 3 **BM:** Braking mode
*This bit set and cleared by UART write and cleared by hardware after restart.*
0: Manual brake (free running mode)
1: Auto brake (automatic regeneration)

Note: *Manual brake let the motor spinning free after turning down the speed or sending 0 speed command. Auto brake start automatic current regeneration into the supply when the motor internal voltage is higher than the speed.*

Bit 2:1 **SRC[1:0]:** Speed reset controll
*This bit set and cleared by UART write and cleared by hardware after restart.*
00: No speed reset after no UART speed update.
01: Speed reset after 50msec without new UART speed update.
10: Speed reset after 500msec without new UART speed update.
11: Speed reset after 2800msec without new UART speed update.

Bit 0 **UC:** UART controll bit
*This bit set and cleared by UART write and cleared by hardware after restart.*
0: Disable the configuration bits
1: Enable the configuration bits

## Configuration bits 2 register:

Type: Write / Read
Write Command: 0x75
Read Command: 0x35

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | TOP_S | P_OFF |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | PMES | - | - | - | USP_2 | USP_1 | - |

*Note: writing 1 clears the given bit writing 0 makes no changes.*

Bit 8 **TOP_S:** Top Speed/Supply compensation enable bit.
0: Disable
1: Enable

Bit 7 **P_OFF:** Pin terminals OFF .
*This bit is used to disable the Brake function of the PWM/Brake input and using as a logical input.*
0: Enable
1: Disable

Bit 6 **BMC:** Pulse Measurement resolution
*Resolution on the PWM/brake input*
0:
1:

Bit 2:1 **USP [1:0]:** USART communication speed.
Default speed is 19200 Baud/sec
*These bits set the UART baud/sec rate.*
00: 19200 Baud/sec
01:
10:
11:

## USART Address Register:

Type: Write
Write command: 0x7C

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | - | - |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

USART Address bits is a unique 7bit address which marks the given controller. It allows to connect several controllers or even other USART devices to connect to the same RX TX channel. The dafault USART address is 0x00. If you want to controll two or more controllers throught the USART you have to change the address first one by one. After this it is possible to connect more devices to the same USART channel. USART Address will be saved into the FLASH memory and reloaded from the memory after every restart.

The controllers sends out their unique USART address right after the system start. This helps to recover and recognise a forgotten address.

Bit 6:0 **UART ADDRESS [6:0]:**
*Max value is 0x7F*
*Default value: 0x00*

## Current measurement calibration register:

Type: Write
Write command: 0x7D

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

Motor controller boards may have current measurement error 1-15% depending on the model. Calibrating and saving the current measurement register solve the error problem and will use the value for the measurement correction.

For the calibration you need follow the steps:
1. Set maximum speed potnetiomet input
2. Turn donw the current limit
3. Place a pure inductive load to the output. You can use large inductors or a stalled DC- motor.
4. Place a current meter or digital clamp meter measuring the motor current.
5. Turn on the supply
6. Slowly turn up the current limit potentiometer
7. Regarding the controller and you load current capabilities turn ut the current to a high enough level. For example with a 100A motor controller 30A is enogh.
8. When you read the exact current on your own current meter you need to send the equivalent value into the **Current Measurement Calibration** register. Calculating the value use the constant int ***Register Value Range*** table.
9. Save the register with the **Saving bits** register.

*Note: If the deviation between the sent value and the internal current measurement is larger than +/-20% than the controller will ignore the command.*
*Exaple: If you Set 30A load current and your constant is 0,244A tah the register value will be: 30A/0,244A =123*

## Software version register:

Type: Read
Read command: 0x3E

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

## Hardware version register:

Type: Read
Read command: 0x3F

| 1th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 13. bit | 12. bit | 11. bit | 10. bit | 9. bit | 8. bit | 7. bit |

| 2th DATA byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6. bit | 5. bit | 4. bit | 3. bit | 2. bit | 1. bit | 0. bit |

# Examples:

**Example 1: Calling 0x00 address device and writing DATA_to send: 0x05C5 value into the Speed register:**

*Sent bytes:*

| | | | | |
|---|---|---|---|---|
| *0x80* | *0x40* | *0x0B* | *0x45* | *0x10* |

C conversion code before sending out the bytes:

ADDRESS=UNIT_ADDRESS | 0x80;
COMMAND= ***for this check the write commands in the register table***
DATA_1th= (DATA_to_send>>7) & 0x7F;
DATA_2th=DATA_to_send & 0x7F;
CRC=( COMMAND + DATA_1th + DATA_2th) & 0x7F;

**Example 2: Calling 0x00 address device and locking Speed variable:**
Setting the speed you shoud read the **configuration_bits_1 SRC** bit it makes timeout for the speed reset controll if you want to add safety shut down when the controll signal stops or you are using wirelles communication it is useful to reset the speed if there is no new update.

*Sent bytes:*

| | | | | |
|---|---|---|---|---|
| 0x80 | 0x71 | 0x00 | 0x01 | 0x72 |

C conversion code before sending out the bytes:

*Refer to the Example 1.*

If you are just starting the controller you need to set forward or reverse direction with the **configuration_bits_1** register. You can do it like this:
You probably want to overwrite the direction switch inputs so you need to turn on **UC** bit as well. And set FR direction bit.

*Sent bytes:*

| | | | | |
|---|---|---|---|---|
| 0x80 | 0x74 | 0x04 | 0x01 | 0x79 |

**Example 3: Calling 0x02 address device to change the regenerative brkaking mode to "auto regen" from "free running mode"and initiating regeneraitive braking with 0x062C current:**

For this you need to execute 2 write comands. The first set the regenerative current limit.
Sent bytes:

| | | | | |
|---|---|---|---|---|
| 0x82 | 0x43 | 0x0C | 0x2C | 0x7B |

C conversion code before sending out the bytes:

*Refer to the Example 1.*

*Second to set the Configuration bit:.*
*Sent bytes:*

| | | | | |
|---|---|---|---|---|
| 0x82 | 0x74 | 0x00 | 0x19 | 0x0D |

C conversion code before sending out the bytes:

*Refer to the Example 1. and **Set Configuration bits register.***

**Example 4: Calling 0x00 address device to read the load_current_register:**

*Sent bytes:*

| 0x80 | 0x12 |
|---|---|

*(read command)*

Received bytes form the controller:

| 0x13 | 0x36 | 0x0D |
|---|---|---|

C conversion code for the received bytes:

*CRC check:*

```
if(((received_byte_1th + received_byte_2th + Read_command) & 0x7F)== received_byte_3th)
        {
                load_current_register=( received_byte_1th <<7) | (received_byte_2th & 0x7F);
        }
```