See page 9 for quick start

Ethernet
Brushed DC Motor Control Board

FMod-IPDCMOT 48/1.5

User's Manual

Version 1.10

five co
innovative engineering

Version: **1.10**
Last revision: **January 10, 2008**
Printed in **Switzerland**

Trademarks
Windows® is a registered trademark of Microsoft Corporation.
Ethernet® is a registered trademark of Xerox Corporation.
Java® is a registered trademark of Sun Microsystem.
Philips® is a registered trademark of Koninklijke Philips Electronics N.V.
Borland® is a registered trademark of Borland Software Corporation.

Warning
This device is not intended to be used in medical, life-support or space products.

Any failure of this device that may cause serious consequences should be prevented through the implementation of backup systems. The user agrees that protection against consequences resulting from device system failure is the user's responsibility. Changes or modifications to this device not explicitly approved by FiveCo will void the user's authority to operate this device.

Support
Web page: http://www.fiveco.ch/section_motion/support_motion_E.htm
e-mail: support@fiveco.ch

Revision history

Revision	Date	Author	Note	Firmware version	Applet version	Win32 application version
1.5	05.04.05	XG	- First revision history - Easy change IP with broadcast	Since 3.26	3.6	1.2
1.6	21.04.05	XG	- Add IP *SUBNETMASK* register (0x13) - Add *TCPCONNECTIONSOPENED* register (0x1A) - Change L2 to L1 during homing - Change name Reset IP button to SOS button - Add Memory Organization chapter	Since 3.27	3.7	1.3
1.7	06.06.05	XG	- Minor text corrections	Since 3.28	3.7	1.3
1.8	30.09.05	AG	- Text and layout corrections - Add checksum function sample - OEM version removed	Same	Same	Same
1.9	08.12.06	AG XG	- Text and layout corrections - Correct function ID for easy change IP answer frame.	Since 3.31	3.8	1.10
1.10	10.01.08	AG	- New address	Same	Same	Same

Table of Contents

1. Package and operating conditions

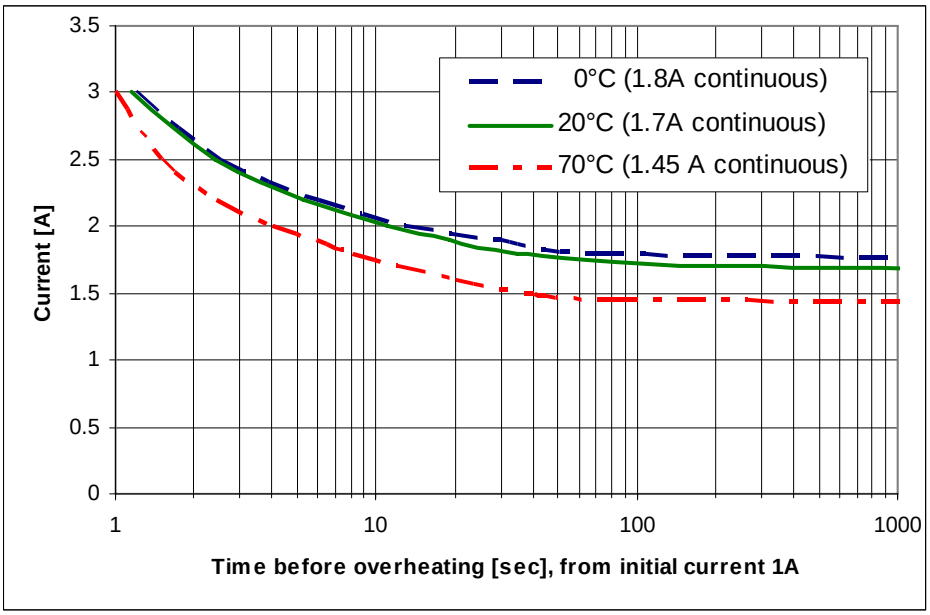
Package contents

- Ethernet Motor Control board IPDCMOT 48/1.5
- CDRom
- This manual
- (optional) Power Supply :
  Power over Ethernet IEEE802.3af Hub-injector
  (110-240 VAC, 50/60Hz -> +48v, 350 mA)

Operating conditions

- Operating temperature 0 – 70 °C
- Supply voltage Vcc 10-48 VDC
- Supply current max ±3 A
- Power consumption 50mA when idle @ Vcc 10 V
  35mA when idle @ Vcc 24 V
  25mA when idle @ Vcc 48 V
- Input capacity (Power +,-) ~220uF, ~50mOhm ESR
- Limit-switch voltage supply 5.0 V (max 2 x 50 mA)
- Max A&B encoder signal 500 kHz = 2Mio pulses/s (x4)
- Output voltage Vcc- 0.7 x Output current
- Max. output current 3 A
- Continuous **OUTPUT** current 1.8 A (0°C)
  1.7 A (20°C)
  1.45 A (70°C)

The measures are made in a closed box, with no airflow



2. Overview

Applications

The FMod-IPDCMOT48/1.5 is a motion control and driver board for DC motors (with brushes). It is particularly interesting not only for its small size and the quality of its regulation system but also because of its communication protocol (Ethernet: TCP/IP-HTTP) which gives it an easy interfacing capability. In addition, the board follows the IEEE 802.3af standard which allows the supply of the module's power through the Ethernet cable.

It can be accessed through a TCP connection (socket) from any computer or through a simple Web page using a standard browser, which allows an easy setup of the controller.

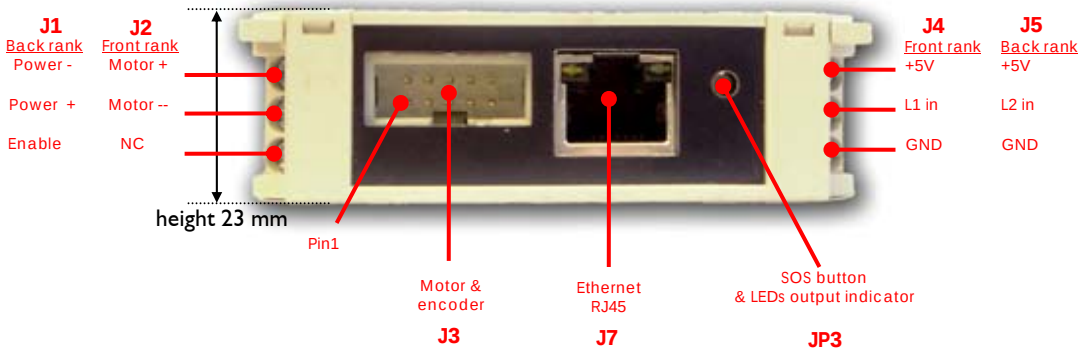
The device connections and dimensions are described in the next pages.

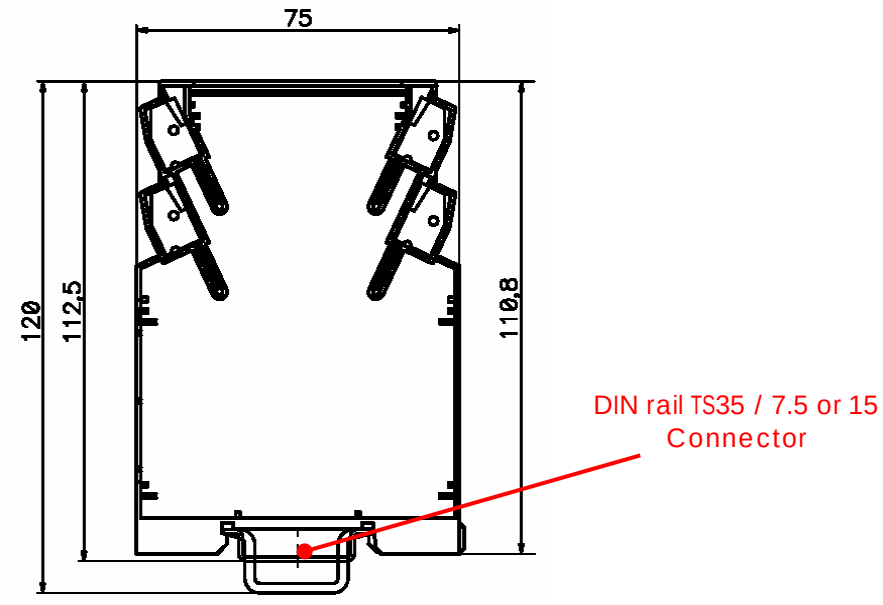
Software operating principle

The PC softwares that must exchange data with this device have to use a dedicated protocol layer on top of the TCP Layer (see chapter 5**Erreur ! Source du renvoi introuvable.**). This protocol is Question & Answer oriented. The PC should send a Question, wait for the Answer and so on.

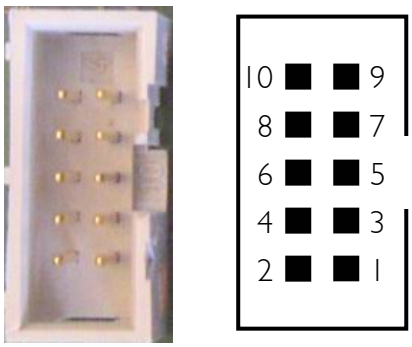
To configure the card's parameters, the protocol uses an Internal Registers Access routine (see chapter 5 and 10).

The Java Applet example and the Borland C++ code sample, available from FiveCo's web site, can help programmers get started with their development.





The following figure displays the pins out of the motor and encoder J3 connector.

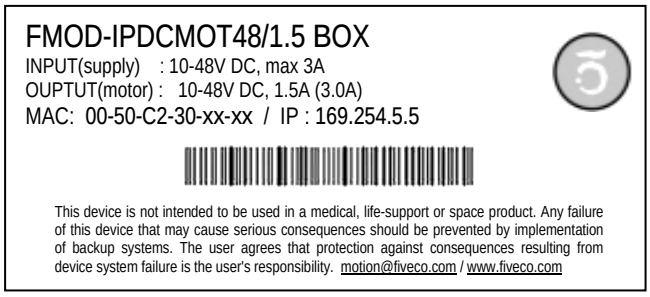


1	Motor + (1A max)
2	5V
3	GND
4	Motor – (1A max)
5	\A
6	A
7	\B
8	B
9	\I
10	I

3. Quick start

This section is intended to help users to quickly plug the module into their system and establish a connection between the computer and the card. Detailed information about hardware and software is provided further in this document.

You can find the board factory communication settings on the label on the board or on the box.



The MAC Address is the 48bits unique identifier on Ethernet networks. The IP Address can be modified. The complete procedure is described further in this manual.

Note: *If the module has already been configured and the IP address has been changed to an unknown value, you can retrieve the SOS IP address (on label) by pressing the "SOS button" during the normal operation of the card.*

Power and network

1. Connect the card to a computer using a RJ45 **cross wired** cable (direct-link), or to an Ethernet-switch with a straight cable.
2. Connect power (10-48V) to the module.
3. Connect 5-48V to the Enable connector through an emergency switch.

Changing IP address

To easily change the factory IP address, user can use the Win32 software provided on the CD-Rom.

1. Plug your new card on your PC network.
2. Start the Win32 application.
3. Click on "File->Easy change IP address".
4. The software scans the network and displays a list of all FiveCo's devices found.
5. Select the MAC address corresponding to your new card.
6. If you have more than one network adapter on your PC, the software will ask you to select the one which is connected to the same network as the FMod-IPDCMOT.
7. The software suggests a new IP address without the last byte. Choose a new IP (**that is not already used on your network!!**) and click the "Change IP address" button.

That's it! The card has a new address and a new subnet mask (the same as your PC). They are automatically saved into EEPROM.

You can now connect to the card with the Win32 software or open its web page by typing its new IP address into a web browser.

Notes:

- The IP address will not be changed if a TCP connection exists with the card.
- The protocol used to change the IP address is described later in this manual.

4. Hardware

Power supply

The card can be powered in two modes: PoE 48VDC (Power over Ethernet, IEEE802.3af) or discrete 10-48V DC.

WARNING: both modes can not be used at the same time !

1. POE (Power Over Ethernet) 48V only, 350 mA on connector J7 (RJ45), pins 4&5(+), 7&8(-). Swapped polarity works too. Compatible with IEEE 802.3af mode B and mode A.

2. 10-48 V with 3A peak current (depending on the motor maximum current) on connector J1.

WARNING: Power supply must manage backward power when braking (power generation) on connector J1 pin1(+) and pin2(-).

> Under 9.0 V driver is automatically set in Brake-Mode.
> Over 54 V driver is automatically set in DriverOpen-Mode.

Motor type

The card drives DC brushed motors (0.1-3A, 10-48V).

With A&B (&Index) channel quadrature encoders with differential line driver (EIA/TIA/RS 422), each logical change of A or B channel defines a pulse.
If the encoder is defined in cycles (or counts) per revolution (CPR), you can multiply by 4 the cycles to obtain the result in pulses (PPR).

Example: an encoder of 500 CPR represents 2000 PPR in the FMod-IPDCMOT.

The motor (+ & -) can be connected to the board on the J3 connector (together with encoder lines) or on the J2 discrete connector. See Overview chapter for more details.

Enable pin

This feature is for security purposes and deactivates the output power.

A voltage in the range [4V ; 48V] on the *Enable pin* lets the driver work normally (regulation on motor).

If the voltage on *Enable pin* drops below 0.8V, the driver (motor) is set to"Brake" mode.

When *Enable pin* goes back to high [4V ; 48V], the user needs to change the mode from "Brake" to the desired one again.

Limit-Switch: stop, reference, stopper type

Both Limit1 (on J4) and Limit2 (on J5) are TTL input lines compatible (logic low [0 - 0.8V], logic high [2-5 V]), but work up to 12 V.

5 VDC sensors shall be preferred and can be powered from the connector (J4 and J5). Open collector (NPN,PNP) and open drain (N,P) output stage sensor can be used.

WARNING: Both Limit-switch sensors (L1 & L2) must be of the same type (NPN,N or PNP,P) because of the internal pull-up or pull-down resistor selection.

LEDs state

- Green: PWM duty cycle [0-75%]
- Yellow: PWM duty cycle [75- 100%[
- Red: PWM at 100% (possible current limitation)

5. TCP Server

General

The board provides an Ethernet port (RJ45 – connector J7) which allows access to all the module's parameters (registers) through a TCP or UDP connection.

Here you will find a small comparison table between these two protocols (non exhaustive):

Features	UDP	TCP
CheckSum (Data integrity)	YES	YES
Multiport (data multiplexing)	YES	YES
Flow control	NO	YES
Acknowledge Data	NO	YES
Multipacket order reconstruction	NO	YES

Two ports are available using the **TCP protocol**:

- Port #80 for HTTP communication.
- Port #8010 Access to the control port

Only one port is accessible through the **UDP protocol**:

- Port #7010 Access to the control port

You will find a detailed description of the different ports over the next pages.

Note: *The board allows up to 4 simultaneous TCP connections. That means for example: that 4 users can connect to the #80 port to see the web page, or 4 users can be connected to the #8010 port and control the I/Os, or even that two see the page and two control the I/Os, etc. In UDP protocol there is no limitation to the number of users connected.*

HTTP port (TCP # 80)

This port is used to access the web page stored on the module.

The user can simply access that port and ask for a particular page, using a standard Web browser (IE, Netscape, Mozilla, Safari), and type the (IP) address of the card:

Ex: type http://169.254.5.5 in the address bar of the browser and the "index.htm" page will be loaded.

Control ports (TCP # 8010 or UDP #7010)

This port is used to access the registers described in the chapter "Registers management" of this manual.

Note: *If you plan to configure all registers using only the onboard Webpage, you can skip this section and go to the Java-Applet section. (The Java-Applet also uses this port to read and write the different settings!)*

TCP/IP works in big endian: most significant byte first, followed by least significant byte(s). The access is done by sending a packet that follows a simple (6 byte header) protocol.

Structure of each packet:	
1) Function ID (2 bytes)	*Code of the function that has to be executed.*
2) Transaction ID (2 bytes)	*Number that defines this packet*
3) Length of the parameters (2 bytes)	*Number of the parameters+data bytes*
4) Parameters (X Byte)	*Parameters + Data*
5) Checksum (2 bytes)	*Control Sum of packet's bytes*

Function ID
The specific code for each function can be found on the next page of this manual.

Transaction ID
The user defines himself the values of the *Transaction ID*. Normally each packet/transaction (communication request) should have a different ID (even though it is not mandatory). When the FMod-IPDMOT 48/1.5 receives a command/packet, it sends back an answer (at each request). This answer contains the same *Transaction ID* as the corresponding command previously sent. In that way, the user is able to check the execution of each command.

Length of the parameters
This 2byte value corresponds to the length (in bytes) of the next section of the packet (parameters only).

Parameters
This part of the packet contains all the parameters (mainly the data that are sent).

Checksum
This 2 bytes value is the Check Sum of all the bytes of the packet (more information on next pages).

READ register(s) value command.

Byte#		Number of bits	Example
0x00	Read (**0x0021**)	16 bits	0x0021
0x02	TransactionID	16 bits	0x1B34
0x04	Number of registers to read (X)	16 bits	0x0001
0x06	X * Registers Addresses	X * 8 bits	0x02
0x06+X	Checksum	16 bits	0x...

The maximum number of registers that can be read at one time is almost 30. The answer sequence should not be greater than 180 bytes. If the number of registers is too big, the FMod-IPDCMOT 48/1.5 will answer only with the value of some of them.

The module **answers** with the following sequence:

Byte#		Number of bits	Example
0x00	Read Answer (**0x0023**)	16 bits	0x0023
0x02	TransactionID (same as demand)	16 bits	0x1B34
0x04	Number of bytes in answer	16 bits	0x0019
0x06	Register address	8 bits	0x02
...	Register value	8–128 bits (16B)	0x12345
The two previous entries are replicated for each register that has been asked for reading			
...	Checksum	16 bits	0x...

WRITE register(s) value command.

Byte#		Number of bits	Example
0x00	Write (**0x0022**)	16 bits	0x0022
0x02	TransactionID	16 bits	0x1B34
0x04	Number of bytes in command	16 bits	0x0003
0x06	Register Addresses	8 bits	0x02
0x07	Register value	8 – 128 bits	0x1234
The two previous entries are replicated for each register that has been asked for reading			
...	Checksum	16 bits	0x...

The max length of this sequence is 180 bytes.

The module **answers** with the following sequence:

Byte#		Number of bits	Example
0x00	Write Answer (**0x0024**)	16 bits	0x0024
0x02	TransactionID (same as demand)	16 bits	0x1B34
0x04	0x0000	16 bits	0x0000
0x06	Checksum	16 bits	0x...

Easy IP address config (UDP # 7010)

A really useful feature of the UDP port #7010 is the "Easy IP config".

The user who wants to design his own software can use this feature to do a "quick start/install" method. Indeed, since this protocol uses a broadcast UDP packet, even if the device is not in the same subnet, it should receive its new IP address and subnet mask.

Procedure:
Send a UDP broadcast message (using a local or direct broadcast IP address) to your network (inside which the FMod-IPDCMOT 48/1.5 is connected) with the following command:

Byte#		Number of bits	Example
0x00	Change IP fct (0x002A)	16 bits	0x002A
0x02	TransactionID	16 bits	0x0000
0x04	Length of params (0x000E)	16 bits	0x000E
0x06	Device Mac Address	6 bytes	0x0050C2308101
0x0C	Device new IP Address	4 bytes	0xC0A81064
0x10	Device new SubnetMask	4 bytes	0xFFFF0000
0x14	Checksum	16 bits	0x...

If the FMod-IPDCMOT 48/1.5 recognizes its MAC address, it will answer this command with a simple acknowledges and change its IP address and subnet mask IF NO TCP CONNECTION IS MADE TO THE BOARD.

Byte#		Number of bits	Example
0x00	Change IP fct ack (0x002B)	16 bits	0x002B
0x02	TransactionID	16 bits	0x0000
0x04	Length of params (0x0000)	16 bits	0x0000
0x14	Checksum	16 bits	0x...

Checksum calculation

This checksum is the same as the IP checksum.

Definition: sum of 1's complement of all 16 bits words of whole message (FiveCo packet) except checksum bytes.

Note: all values are unsigned!

Sequence:

1. Clear accumulator

Loop

x. Only if last word is not made of two bytes, the data byte is the upper byte (big endian)
2. Compute 1's complement of each 16bits word, result is 16bits
3. Convert last result from 16 bits to 32 bits, result is 32bits: 0x0000+last result
4. Add last result to the 32 bits accumulator

Try the Loop

5. Convert accumulator in two 16bits words
6. Add those two 16bits words, result is 16bits word.
7. If an overflow occurs with the last addition (Carry), add 1 to the last result.
8. Last result is the final result

Example (in hexadecimal):

```
 !0x0021  (0XFFDE) → 0x0000FFDE (Read)
+!0x1234  (0xEDCB) → 0x0001EDA9 (TransID)
+!0x0003  (0xFFFC) → 0x0002EDA5 (3 reg to read)
+!0x0A10  (0XF5EF) → 0x0003E394 (reg 0A,10,02)
+!0x02(00)(0XFDFF) → 0x0004E193
```

Note that in this case a last 00 is implicitly used. (02 → 02 00).

```
0x0004 + 0xE193 = 0xE197, (carry=0)
0xE197 + carry  = 0xE197
```

`Checksum = 0xE197`

Here is an example of a checksum calculation function in C:

```
int RetCheckSum(Byte* ByteTab, int Size)
{
    // This function returns the calculated checksum
    unsigned int     Sum=0;
    bool             AddHighByte=true;
    unsigned int     ChecksumCalculated;

    for(int i=0;i<Size;i++)
    {
        if(AddHighByte)
        {
            Sum+=((ByteTab[i])<<8)^0xFF00;
            AddHighByte=false;
        }
        else
        {
            Sum+=(ByteTab[i])^0x00FF;
            AddHighByte=true;
        }
    }
    if (AddHighByte==false)
    Sum+= 0xFF;

    ChecksumCalculated = ((Sum>>16)&0xFFFF)+(Sum&0xFFFF);

    ChecksumCalculated = ((ChecksumCalculated>>16)&0xFFFF)
                  +(ChecksumCalculated&0xFFFF);

    return ChecksumCalculated;
}
```

This function needs a Byte array (ByteTab) containing the command sequence and this array's length (Size) as input. It returns the checksum as an int.

6. Java Applet

A specific Java Applet is provided with the module in order to control all its parameters without having to write any specific software.
When you use a browser to load the HTTP web page loaded on the card, a JAVA applet (included in the onboard webpage) starts a communication between the card and the browser software (PC), and refreshes periodically all the useful registers (read and write sequences).

Overview

To connect to the http server included on the card, simply open your web browser and type the IP address of the module. Here is an example with default address:

"http:// 169.254.5.5"

The applet is downloaded from the module to your computer and runs as a local process (on your computer). You need to use a web browser that is compatible with Java 1.1.

Please note that on a MS Windows® based computer, a delay of some seconds can occur when you download the JAVA applet, due to an OS NetBios issue.

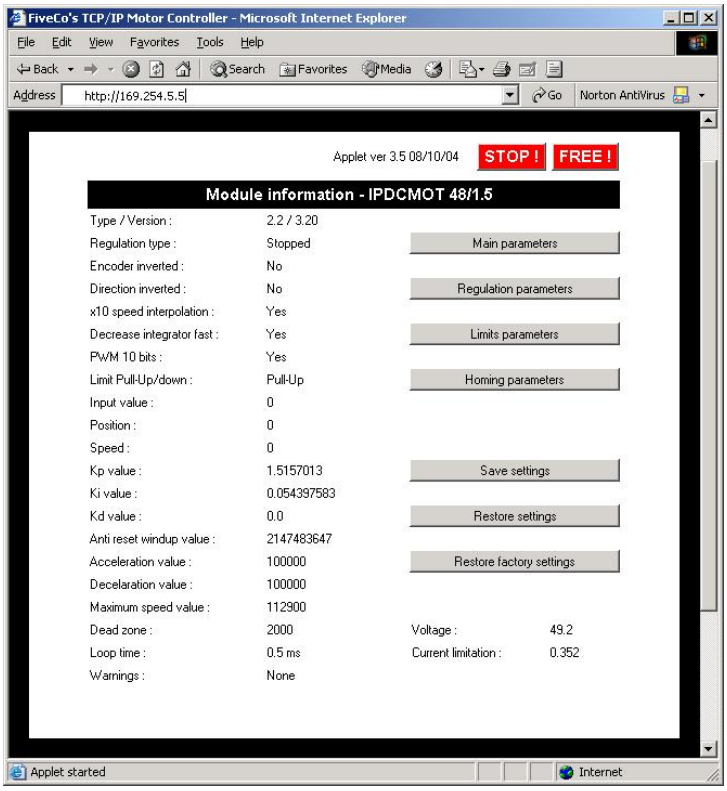
The following paragraphs describe this applet. In order to navigate through the different panels of the applet, you will have to use the buttons present in the applet window.
Note that the "Back" and "Forward" commands of the browser will not be useful to navigate through the applet's panels.

Main Panel

The Main panel shows the information related to the module. The values of the registers are displayed.

Three buttons allow the access to the other three configuration panels.

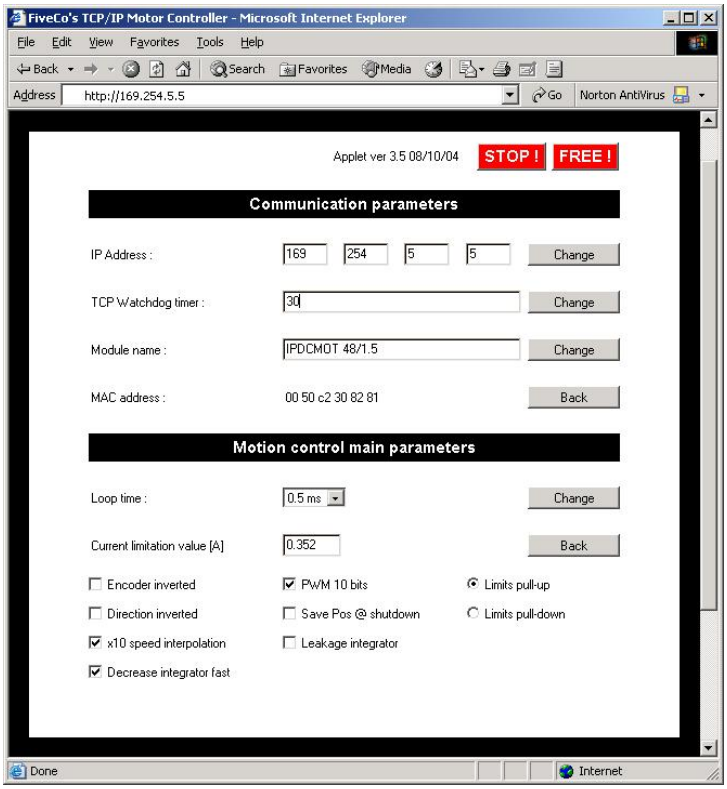
The fourth button "Save actual config on board" is used to send a command to the board in order to save the current configuration on the board, the fifth "Restore config from memory" to restore it from EEPROM and the last one to restore the factory's default settings.



Main Parameters Panel

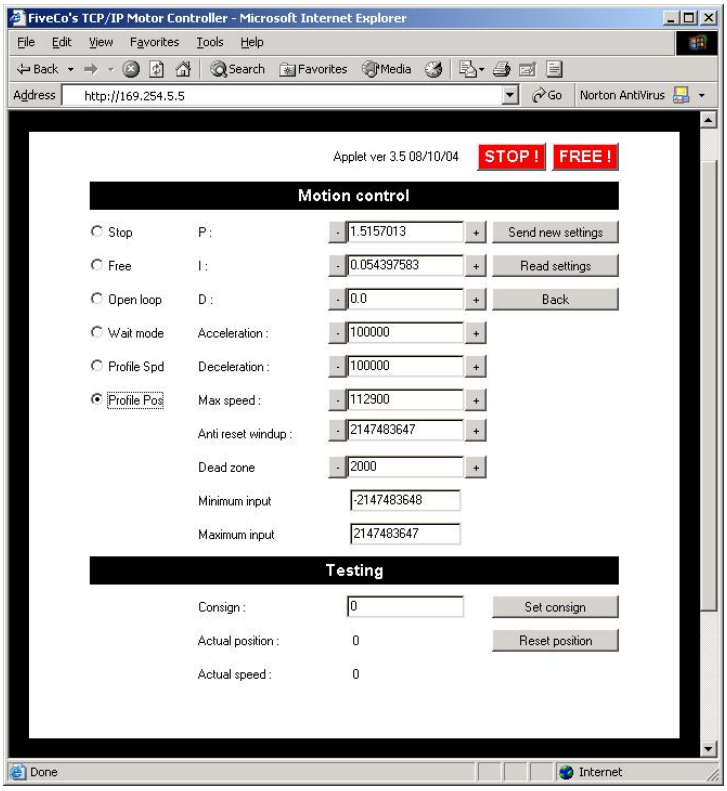
This panel is used to change the board's main parameters:

- **IP address:** (the new one will be valid only when all users have been disconnected. At that moment, the IP address in the address bar of the browser will have to be changed to access the applet again)
- **Subnet mask: useful only for special UDP directed broadcast**
- **TCP Watchdog**: Time before disconnection if the client does not send any packet.
- **Module name** of your choice.
- Module hardware **MAC address** (read only).
- Main **motion control parameters** (see motion control section).



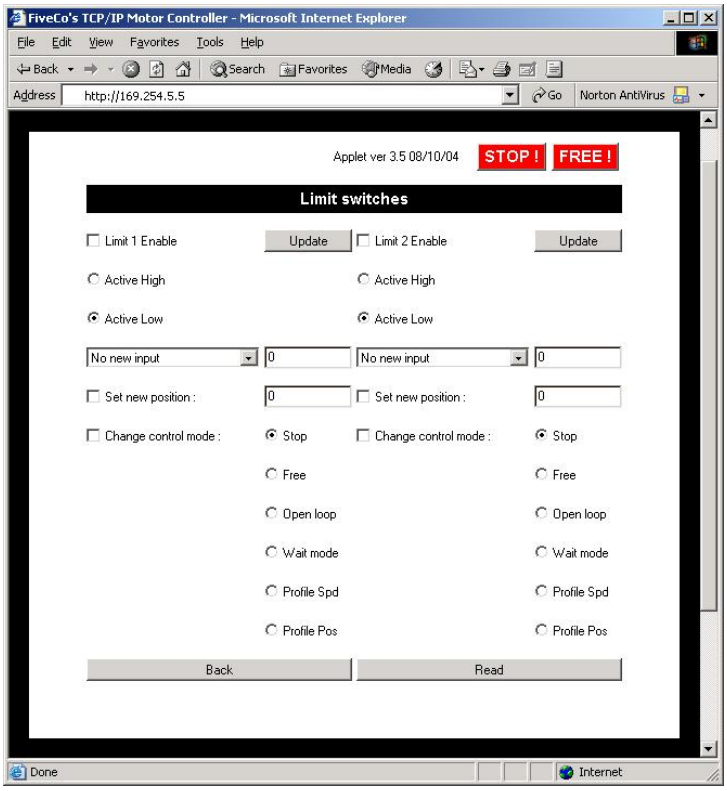
Motion Control Panel

This panel shows the motion control parameters. See chapter related to the use of these parameters.



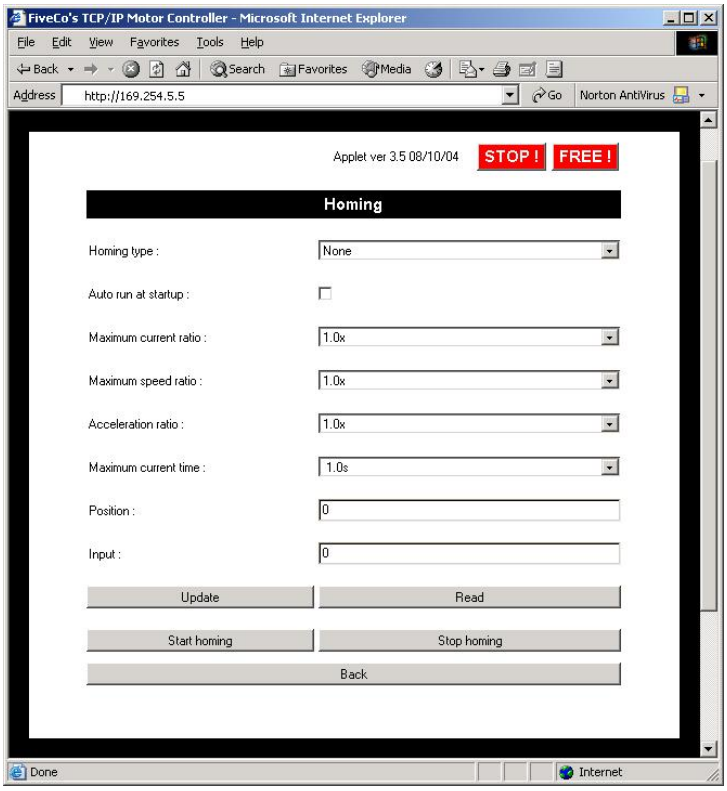
Limits switches Panel

This panel shows the limit control parameters. See chapter related to the use of these parameters.



Homing Panel

This panel shows the homing parameters. Save Parameters (if needed) before executing the "Start homing" function. See chapter related to the use of these parameters.



7. Motion Control Modes

KP, KI, KD depend mainly on the type of motor, the voltage and the encoder resolution. The more the encoder has pulses per turn, the smaller will *KP, KI, KD* values be.

The main *INPUT* (PWM/Speed/Position) is software limited with configurable *INPUTMIN* and *INPUTMAX* values.

List of regulation modes

- Brake
- Driver Open
- Open Loop
- Wait
- Speed Control (Torque mode)
- Position Control

Brake Mode

REGULATIONMODE= 0x00.

Brakes the motor. After 1 second, the motor is fully short-circuited. (At the beginning 50% slow decay mode, 50% driver open, up to 100% slow decay mode, 0% driver open after 1 sec).

Driver Open Mode

REGULATIONMODE= 0x01.

Switches OFF the H-bridge power transistors.
(Internal protection diodes still work.)

Open Loop Mode

REGULATIONMODE= 0x02.

INPUT register is copied to *COMMAND* register which is then converted without regulation to PWM output.

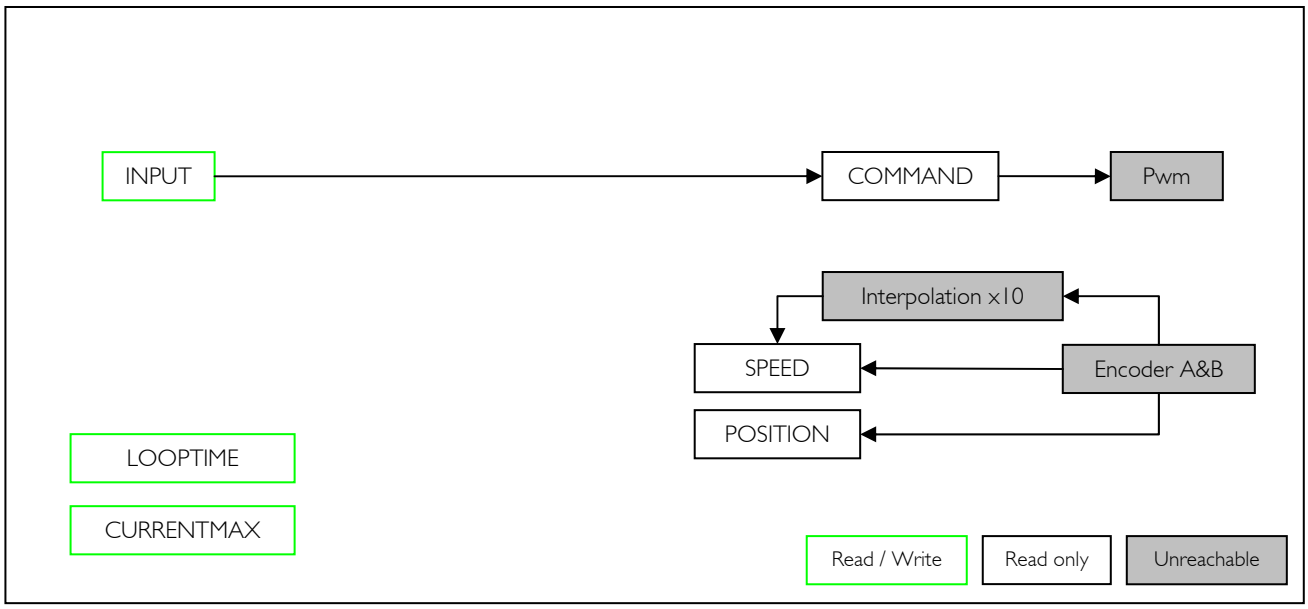
Maximum *INPUT* is 0x0000FFFF (65535). Represents 100% of forward PWM.

INPUT (0) represents 0% of PWM.

Minimum *INPUT* is 0xFFFF0001 (-65535). Represents 100% of backward PWM.

Use this mode if a custom regulation is made on another CPU

With PWM 9bits, the 7 least significant bits are unused.
When PWM 10bits is selected the 6 least significant bits are unused.



Wait Mode

REGULATIONMODE= 0x03.

This mode leaves the PWM and current output unchanged, and halts PID regulation.

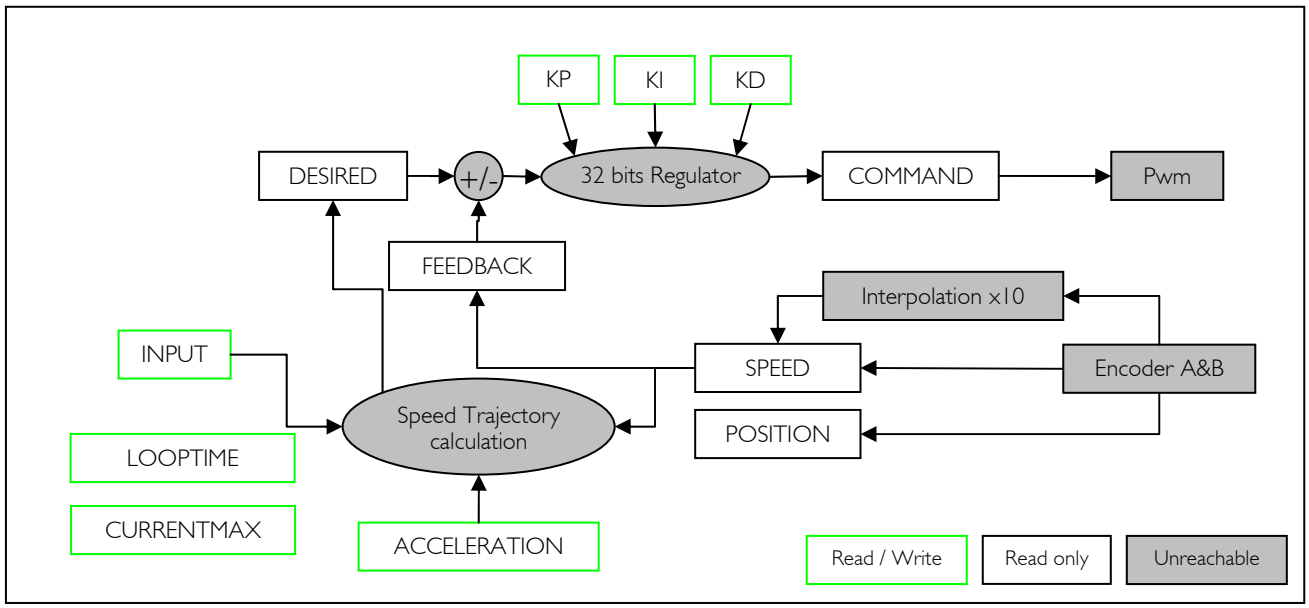
All registers can be updated with no influence on the outputs.

This mode can be used when several registers need to be changed at the same time.

Speed Control Mode

REGULATIONMODE= 0x04.

INPUT is the speed to reach. If immediate *INPUT* is *DESIRED*, set *ACCELERATION* to max.
If smooth control is needed, *DESIRED* value is able to accelerate up/down to *INPUT*. *ACCELERATION* defines the slope between the *INPUT* and *DESIRED* registers. *FEEDBACK* is represented by the effective *SPEED* of the Motor.



How to choose the correct PID parameters in Speed control mode?

Make sure that the motor and the system to which it is connected have no mechanical limit. If they do, disconnect the motor from its system.

- Set *ACCELERATION* to max (2'000'000'000)
- Set *LOOPTIME* at 0.5ms
- Set *CURRENTMAX* to the minimum value between the following values:
  Maximum motor current, maximum torque wanted (converted in current, see datasheet of the motor), maximum current of the power supply, maximum current of the motor driver.
- Clear *KP,KI, KD*
- Set *INPUT* at the maximum speed for your application
- Set *KP* to 0.01 and increase KP by steps of +20% (if oscillations of *SPEED* appear, decrease *KP*) and when the effective *SPEED* is at the 90-92% of *INPUT*, *KP* is correct.
- Remember this value.

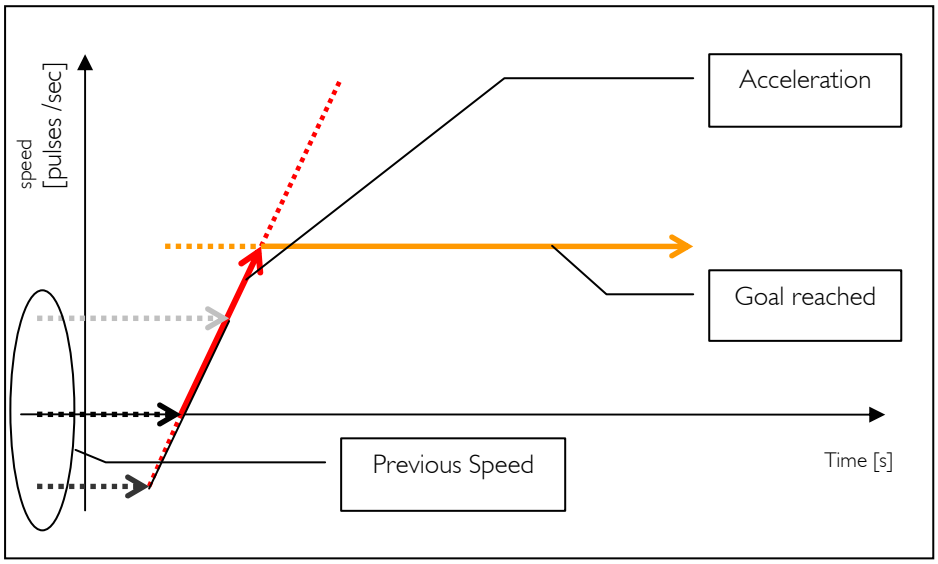
If you have no PID experience, you can stop now. In most cases, regulation already works.

If you need to reach 100% of the INPUT, a correct value for *KI* needs to be found:

- Clear *KP*.
- Divide *INPUT* by 2.
- Set *LOOPTIME* to slow (10ms).
- Set *KI* at 0.001 and increase it by steps of +20% , and when oscillations of *SPEED* of more than 20% appear, decrease it until the oscillations disappear. *KI* is then correct.
- Remember this value.
- Clear *KI*. (this force regulator to clear integrated value *INTEGRALDELTA*).
- Now set *LOOPTIME* at 0.5ms.

- Set *KP* to the remembered value.
- Set *KI* to the remembered value.
- Change *INPUT* to show how the regulation works.

"Speed control Mode" graph example:



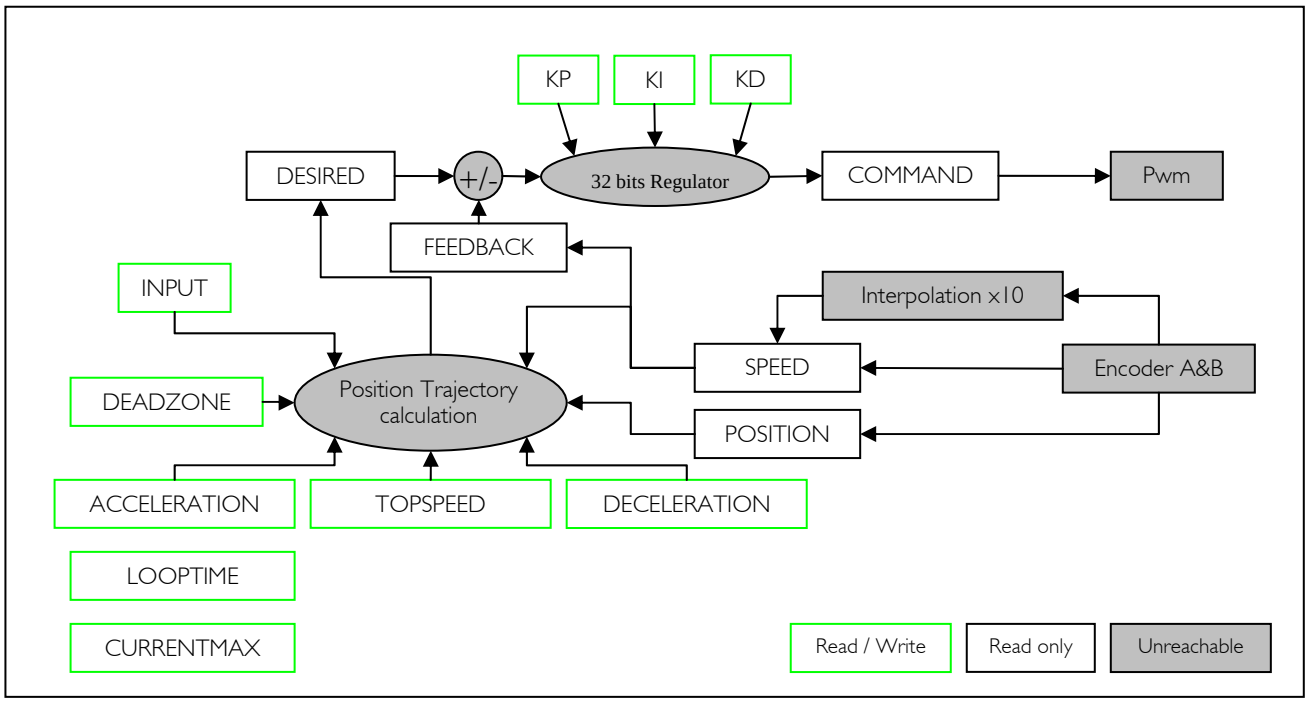
Speed profile in "Speed Control Mode"

Note:
TOPSPEED is not used in "speed control mode", but only in "position control mode". *INPUTMIN* (0x24) and *INPUTMAX* (0x25) registers give more possibilities to replace *TOPSPEED*. After a new *INPUT* (goal) is set, it is possible to change *INPUT*, or *ACCELERATION* without having to wait for the goal to be reached.

Position Control Mode

REGULATIONMODE= 0x05.

INPUT is the position to reach. The trajectory kernel calculates in real-time the *DESIRED* speed for the *PID* regulator and the *FEEDBACK* is represented by the effective *SPEED* and *POSITION* of the Motor.



For the position trajectory, a full trapezoid speed profile consists of 3 different movements:

- *ACCELERATION* from actual speed
- *TOPSPEED*
- *DECELERATION* when coming close to the goal point

DEADZONE is complementary to the deceleration and defines a zone around the goal point (*INPUT* ± *DEADZONE*) where the *DESIRED* speed is forced to zero. It is useful when *DECELERATION* is bigger than effective motor deceleration. It skips oscillations in that case.

How to choose the correct PID parameters in Position control mode?

Make sure that the motor and the system to which it is connected have no mechanical limit. If they do, disconnect the motor from its system.

The position trajectory kernel uses the PID regulator with the speed feedback.

>> See "How choosing the correct PID parameters in Speed mode" to configure PID parameters.

Then

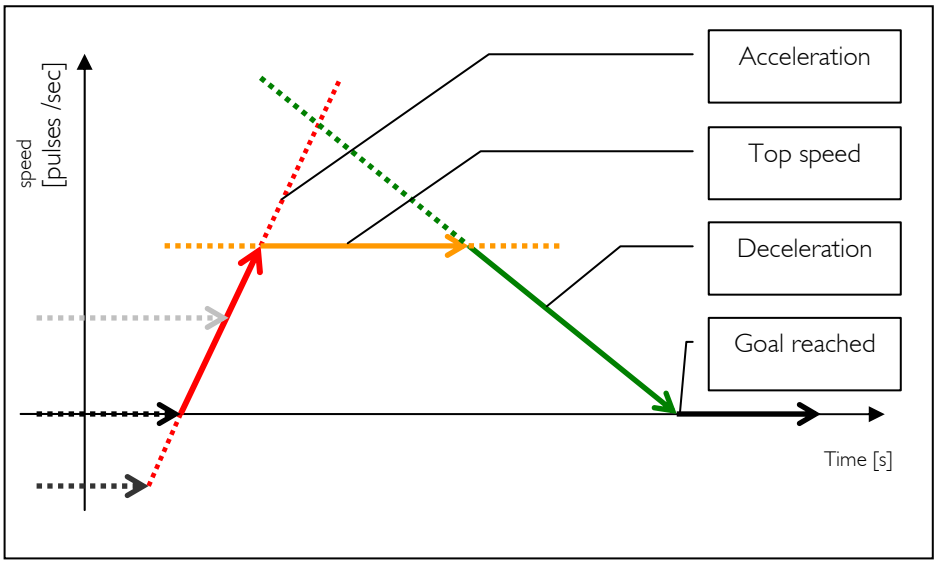
- *KP, KI, KD, CURRENTMAX, LOOPTIME* are set.
- Set *TOPSPEED* to your maximum speed application.
- Set *ACCELERATION* and *DECELERATION* to the same value of *TOPSPEED*. So the acceleration would take 1sec and deceleration another 1 sec.
- Set *DEADZONE* value to 1.
- Test the system by writing different goal position into *INPUT*.
- Run the *SAVEUSERPARAMETERS* function.
- If all the parameters have been correctly configured and tested, connect the motor to its mechanical system.

Note on practical experience:

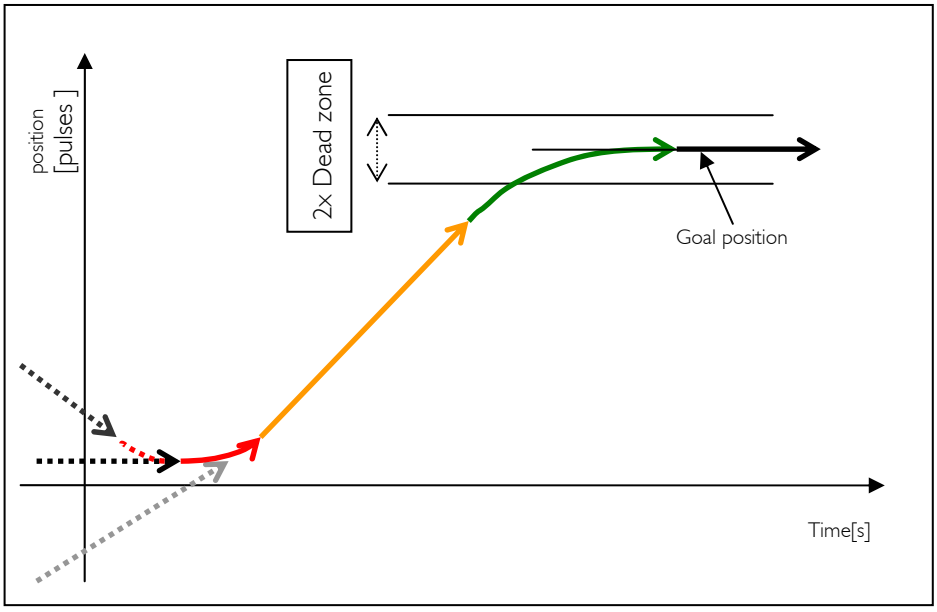
You have discovered the best *KP, KI* and *KD* settings for you motor. During this setup, you had to disconnect the motor from its mechanical system, and controlled the system in "speed control mode" (*REGULATIONMODE*). In most situations, these settings (*KP, KI, KD*) are also used to control regulation of your complete system (motor + mechanical system) in "position control mode" (*REGULATIONMODE*).

Practically speaking, with small DC motors (<200W), only the inertia and the characteristics of the motor itself are responsible for the regulation's parameters.

"Position control mode" graph example:



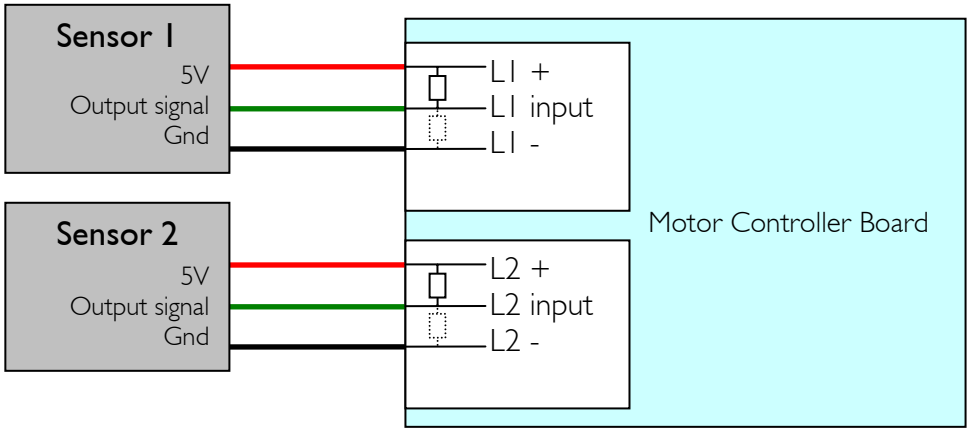
Speed profile in the "position control mode"



Position profile in the "position control mode"

8. Limits - switches

The term "Limit" used in this manual can be understood as a meaning for: "end stroke", "reference", "stopper" or "action". It depends on the external use of each limit signal.
Two independent limits/switch sensors can be connected to the board. Only *Limit1* (J4) can be configured to be the limit for "Homing". Power supply of 5V is provided from the motor controller board, as shown below.



Limits connections

To configure a limit, it is necessary to update a set of registers:

- *OPTIONS* register selects pull-up or pull-down resistors of 1.5KOhm to all Limits.
- *LIMIT(1/2)SETUP* register configures the activity of the limit number 1 or 2.

See "Full Register Description" chapter for more information.

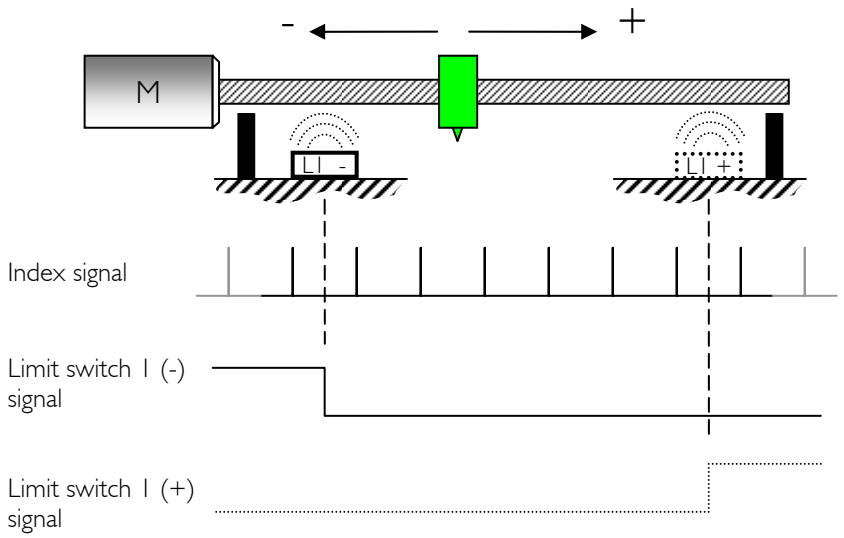
Here are some limit examples:

- Reset: changes *REGULATIONMODE* to desired mode, sets *POSITION* and *INPUT*.
- Thrust: sets a new *INPUT* to come back in normal range.
- Reference: only sets a new *POSITION* when it is reached.
- Stopper: changes *REGULATIONMODE* to Brake mode, to stop the motor.
- Other: any combination of *REGULATIONMODE*, *POSITION*, *INPUT*.

9. Homing (position reference)

The motion control card offers multiple methods to find the home-position (only for positioning system). This method called "homing" is intended as the position reference calibration process for the card.

A mechanical or electrical signal generated by the limit-switch L1 can be



combined (or not) with the index signal to define the home-position. (index signal that can be found on the 3rd channel from the encoders, as well as A/B signals)

Keep in mind that *POSITION* value can be automatically saved in EEPROM when the power shuts-down, and restored at power-up. See *OPTIONS (0x2C)* register for these details.

* alternative site for limit 1 sensor

List of homing method

0. **Actual position is correct, stay there.**
1. **Actual position is home, set new *INPUT*.**
2. **Move - to the first index.**
3. **Move + to the first index.**
4. **Move - to the mechanical limit.**
5. **Move + to the mechanical limit.**
6. **Move - to the mechanical limit with index.**
7. **Move + to the mechanical limit with index.**
8. **Move - to the limit-switch 1.**
9. **Move + to the limit-switch 1.**
10. **Move - to the limit-switch 1 with index.**
11. **Move + to the limit-switch 1 with index.**

All homing methods set *REGULATIONMODE (0x20)* to position control mode. When the homing conditions are verified, *HOMINGPOSITION* register is copied to *POSITION*, and new *INPUT* set with *HOMINGINPUT*.

For the homing sequence, the user can program values for *ACCELERATION*, *TOPSPEED* and *CURRENTMAX* different to the standard "position control mode" ones. The user can also define in the *HOMINGOPTIONS* register the ratio (% of value of these registers) that will be used during homing.
When homing is completed, the standard values are automatically restored from EEPROM.

Example:
If a standard *TOPSPEED* value in position control mode is 100'000pulses/sec and must be reduced to 50% (50'000pulses/sec) during a homing method, then configure to 50% the bits concerning *TOPSPEED* in the *HOMINGOPTIONS* (0x48) register.

When a mechanical limit system is used, another parameter "time" must also be configured. It represents how long the PWM output to the motor must remain saturated (=current max reached & torque max reached) before the home position is accepted.

Note:
During homing, no external *INPUT*, *INPUTOFFSET*, *INPUTOFFSETMEASURED* or *REGULATIONMODE* are accepted. Attempts to write these values will therefore be skipped. The only way to stop or modify homing is to call the *STOPHOMING* (0x4A) function.

Homing method 0: Actual position is correct, stay there

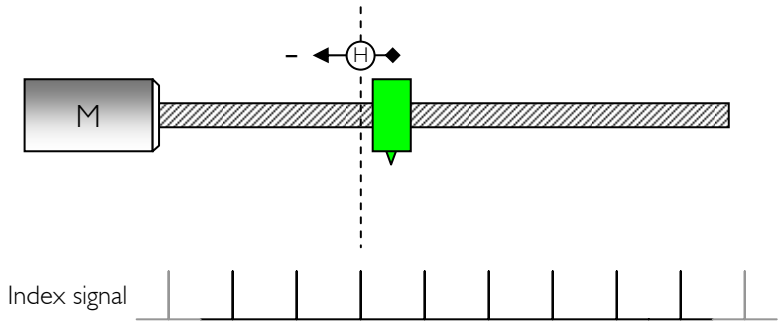
➔ The actual *POSITION* is not overwritten, but it is copied to *INPUT* register.

Homing method 1: Actual position is correct, set new INPUT

➔ The actual *POSITION* is not overwritten.
➔ *HOMINGINPUT* register is copied to *INPUT* register.

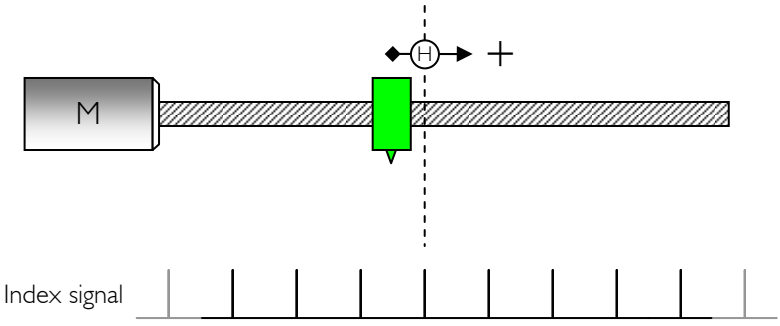
Homing method 2: Move backward (-) to the first index

➔ *HOMINGPOSITION* is copied to *POSITION* when the next index is found.
➔ *HOMINGINPUT* register is copied to *INPUT* register.



Homing method 3: Move forward (+) to the first index

➔ *HOMINGPOSITION* is copied to *POSITION* when the next index is found.
➔ *HOMINGINPUT* register is copied to *INPUT* register.



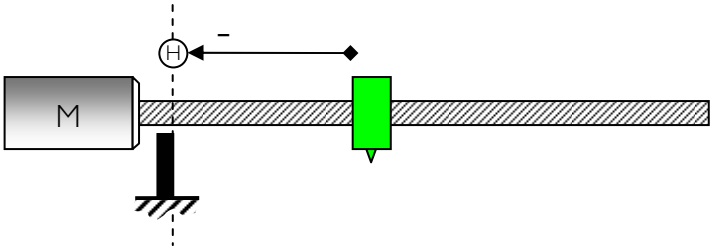
Note:
In Homing Method 2 and 3, even if the "home position" is defined (by the index), the motor will continue to move until position/speed value has reached the goal (*INPUT*).

Homing method 4: Move backward (-) to the mechanical limit

When the module detects a saturated PWM output (torque value control) for more than the time specified in the *HOMINGOPTIONS* register, home is validated:

➔ *HOMINGPOSITION* is copied to *POSITION* register.
➔ *HOMINGINPUT* is copied to *INPUT* register.

The (user-definable) value *of CURRENMAX* and its corresponding ratio (used during the homing sequence) are very important for this method. Please note that a high current (torque) can destroy mechanical parts of the system (e.g. gears).

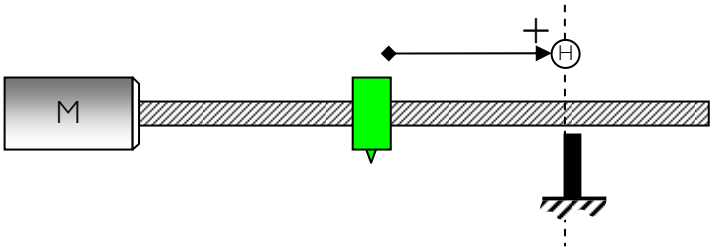


Homing method 5: Move forward (+) to the mechanical limit

When the module detects a saturated PWM output (torque value control) for more than the specified time with *HOMINGOPTIONS*, home is validated:

➔ *HOMINGPOSITION* is copied to *POSITION* register.
➔ *HOMINGINPUT* is copied to *INPUT* register.

The (user-definable) value *of CURRENMAX* and its corresponding ratio (used during the homing sequence) are very important for this method. Please note that a high current (torque) can destroy mechanical parts of the system (e.g. gears).

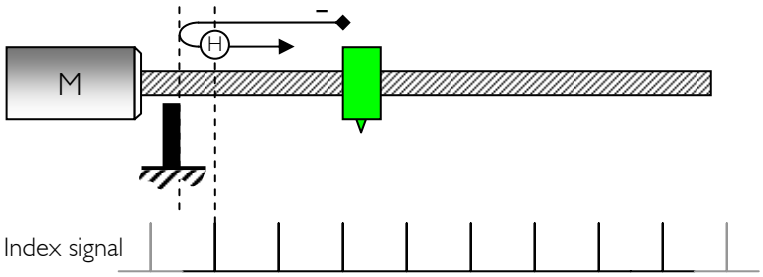


Homing method 6: Move backward (-) to a mechanical limit and index

When the module detects a saturated PWM output for more than the specified time with *HOMINGOPTIONS,* it change direction (forward) until it finds the index, then home is validated:

➔ *HOMINGPOSITION* is copied to *POSITION* register.
➔ *HOMINGINPUT* is copied to *INPUT* register.

CURRENMAX and corresponding homing ratio are very important for this method. Please note that the high current (torque) can destroy mechanical parts of the system (e.g. gears).

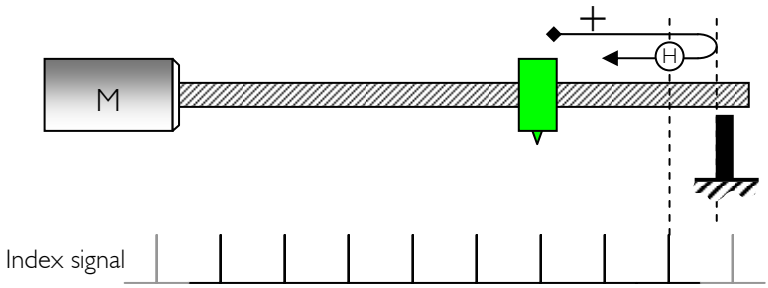


Homing method 7: Move forward (+) to a mechanical limit and index

When the module detects a saturated PWM output for more than the specified time with *HOMINGOPTIONS,* it change direction (backward) until it finds the index, then home is validated:

➔ *HOMINGPOSITION* is copied to *POSITION* register.
➔ *HOMINGINPUT* is copied to *INPUT* register.

CURRENMAX and corresponding homing ratio are very important for this method. Please note that the high current (torque) can destroy mechanical parts of the system (e.g. gears).

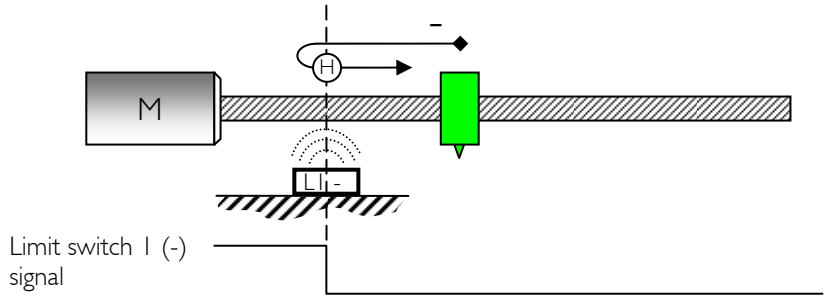


Homing method 8: Move backward (-) to the limit-switch 1

When the module detects the active state of limit-switch 1, it changes direction (forward) until the end of the limit activity, then home is validated:

➔ *HOMINGPOSITION* is copied to *POSITION* register.
➔ *HOMINGINPUT* is copied to *INPUT* register.

During homing with limit-switch activity, the standard actions defined with *LIMIT1OPTIONS* are suspended.

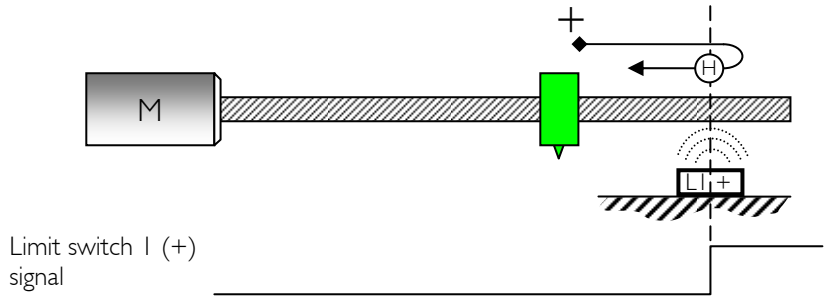


Homing method 9: Move forward (+) to the limit-switch 1

When the module detects the active state of limit-switch 1, it changes direction (backward) until the end of the limit activity, then home is validated:

➔ *HOMINGPOSITION* is copied to *POSITION* register.
➔ *HOMINGINPUT* is copied to *INPUT* register.

During homing with limit-switch activity, the standard actions defined with *LIMIT1OPTIONS* are suspended.

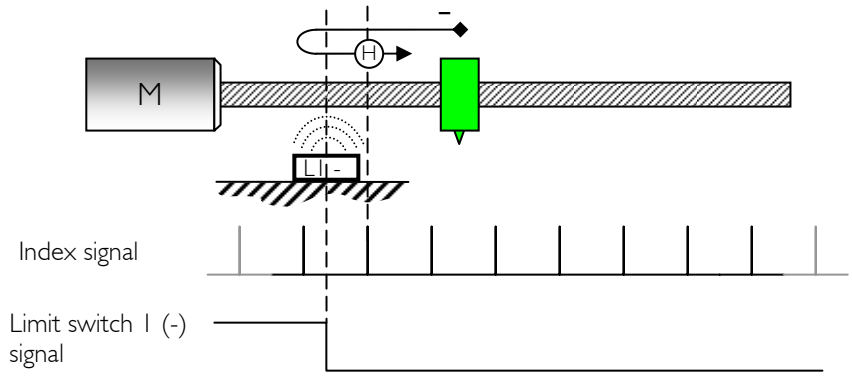


Homing method 10: Move backward (-) to the limit-switch 1 and index

When the module detects the active state of limit-switch 1, it changes direction (forward) until the end of the limit activity. With the first index, home is validated:

➔ *HOMINGPOSITION* is copied to *POSITION* register.
➔ *HOMINGINPUT* is copied to *INPUT* register.

During homing with limit-switch activity, the standard actions defined with *LIMIT1OPTIONS* are suspended.

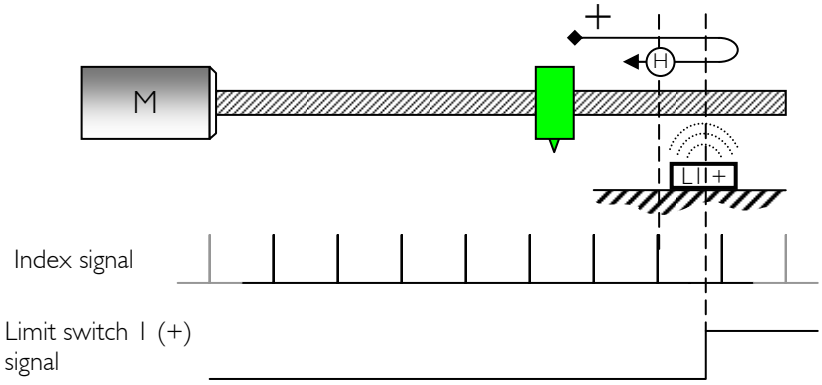


Homing method 11: Move forward (+) to the limit-switch 1 and index

When the module detects the active state of limit-switch 1, it changes direction (backward) until the end of the limit activity. With the first index, home is validated:

➔ *HOMINGPOSITION* is copied to *POSITION* register.
➔ *HOMINGINPUT* is copied to *INPUT* register.

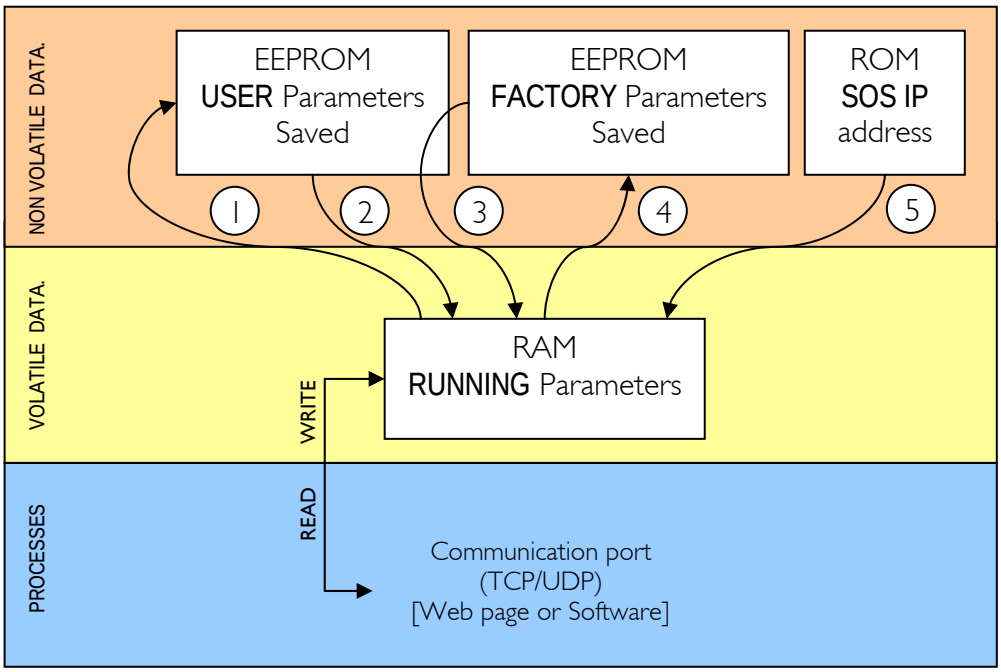
During homing with limit-switch activity, the standard actions defined with *LIMIT1OPTIONS* are suspended.



10. Registers management

Memory Organization

The user must know that a new register value sent through the communication port is loaded to the running parameters in RAM and used for the current process. All these parameters are lost at power-down. It is required to save them to "User Parameters" or "Factory Parameters" with the corresponding function.



Action Number and description:

1	***SaveUserParameters*** (0x03) function
2	During standard power-up or calling ***RestoreUserParameters*** (0x04) function
3	***RestoreFactoryParameters*** (0x05) function
4 + 1	***SaveFactoryParameters*** (0x06) function [For integrators engineers only]
5	By pressing "SOS Button" after power-up
3 + 5 + 1	By pressing "SOS Button" during power-up

List of registers

Address	Bytes	Name
General Information		
0x00	4	*TYPE*
0x01	4	*VERSION*
0x02	0 (fct)	*RESETCPU*
0x03	0 (fct)	*SAVEUSERPARAMETERS*
0x04	0 (fct)	*RESTOREUSERPARAMETERS*
0x05	0 (fct)	*RESTOREFACTORYPARAMETERS*
0x06	0 (fct)	*SAVEFACTORYPARAMETERS*
0x07	4	*VOLTAGE*
0x08	4	*WARNING*
Communication		
0x10 (16)	4	*COMMUNICATIONOPTIONS*
0x11 (17)	6	*ETHERNETMAC*
0x12 (18)	4	*IPADDRESS*
0x13 (19)	4	*SUBNETMASK*
0x14 (20)	1	*TCPTIMEOUT*
0x15 (21)	16	*MODULENAME*
0x1A (26)	1	*TCPCONNECTIONSOPENED*
General configuration		
0x20 (32)	1	*REGULATIONMODE*
0x21 (33)	4	*INPUT*
0x22 (34)	4	*INPUTOFFSET*
0x23 (35)	4	*INPUTOFFSETMEASURED*
0x24 (36)	4	*INPUTMIN*
0x25 (37)	4	*INPUTMAX*
0x26 (38)	4	*POSITION*
0x27 (39)	4	*POSITIONOFFSET*
0x28 (40)	4	*SPEED*
0x2A (42)	4	*CURRENTMAX*
0x2C (44)	4	*OPTIONS*
0x2D (45)	1	*LOOPTIME*
0x2E (46)	4	*OUTPUTVOLTAGEMAX*
PID group		
0x30 (48)	4	*DESIRED*
0x31 (49)	4	*FEEDBACK*
0x32 (50)	4	*COMMAND*
0x33 (51)	4	*KP*
0x34 (52)	4	*KI*
0x35 (53)	4	*KD*
0x36 (54)	4	*ANTIRESETWINDUP*
0x37 (55)	4	*INTEGRALDELTA*
0x38 (56)	4	*DERIVATIONOFDELTA*

List of registers (continued):

Address	Bytes	Name
Trajectory group		
0x40 (64)	4	*ACCELERATION*
0x41 (65)	4	*DECELERATION*
0x42 (66)	4	*TOPSPEED*
0x43 (67)	4	*DEADZONE*
Homing group		
0x48 (72)	4	*HOMINGOPTIONS*
0x49 (73)	0 (fct)	*HOMING*
0x4A (74)	0 (fct)	*STOPHOMING*
0x4B (75)	4	*HOMINGPOSITION*
0x4C (76)	4	*HOMINGINPUT*
Limits group		
0x50 (80)	4	*LIMIT1SETUP*
0x51 (81)	1	*LIMIT1REGULATIONMODE*
0x52 (82)	4	*LIMIT1POSITION*
0x53 (83)	4	*LIMIT1XINPUT*
0x58 (88)	4	*LIMIT2SETUP*
0x59 (89)	1	*LIMIT2REGULATIONMODE*
0x5A (90)	4	*LIMIT2POSITION*
0x5B (91)	4	*LIMIT2XINPUT*

TYPE

Register Address	Register Name	Function	Read/Write Control
0x00	*TYPE*	Product ID	Read only

Register Size	Register structure	
4 Bytes	Unsigned Int 16bits (HH-HL) *TYPE*	Unsigned Int 16bits (LH-LL) *MODEL*

Description:
Product identifier composed of a *Type* and *Model* number.
It defines which kind of peripheral it is.
Normally different *TYPE* modules are not software compatible.

Example:
Module with *TYPE* = 0x00020001 means *Type*=2 (2= Motor driver) , *Model* = 1.

Limits:
None

Active:
Each time the processor is running

VERSION

Register Address	Register Name	Function	Read/Write Control
0x01	*VERSION*	Software ID	Read only

Register Size	Register structure	
4 Bytes	Unsigned Int 16bits (HH-HL) *Version*	Unsigned Int 16bits (LH-LL) *Revision*

Description:
Firmware identifier composed of a *Version* and *Revision* number.
Normally same *Version* with different *Revision* is backward compatible.

Example:
Firmware 0x00030014 = *Version* 3, *Revision* 20 (0x14) is compatible with all earlier revisions of the same version (ver 3.0 to 3.19) but it has new functionalities (which are deactivated by default) or code optimizations.

Limits:
None

Active:
Each time the processor is running

RESET CPU

Function Address	Function Name	Function	Read/Write Control
0x02	*RESETCPU*	Restart processor	Write only

Register Size	Register structure	Unit
0 Byte	none	none

Description:
Reboots the card. The communication will be lost.

Active:
Each time the processor is running.

SAVE USER PARAMETERS

Function Address	Function Name	Function	Read/Write Control
0x03	*SAVEUSERPARAMETERS*	Saves all in EEPROM	Write only

Register Size	Register structure	Unit
0 Byte	none	none

Description:
Saves the following parameters to user EEPROM space:

0x10 *COMMUNICATIONOPTIONS*
0x12 *IPADDRESS*
0x13 *SUBNETMASK*
0x14 *TCPWATCHDOG*
0x15 *MODULENAME*

0x24 *INPUTMIN*
0x25 *INPUTMAX*
0x2A *CURRENTMAX*
0x2C *OPTIONS*
0x2D *LOOPTIME*
0x2E *OUTPUTVOLTAGEMAX*

0x33 *KP*
0x34 *KI*
0x35 *KD*
0x36 *ANTIRESETWINDUP*

0x40 *ACCELERATION*
0x41 *DECELERATION*
0x42 *TOPSPEED*
0x43 *DEADZONE*
0x48 *HOMINGOPTIONS*
0x4B *HOMINGPOSITION*
0x4C *HOMINGINPUT*

0x50 *LIMIT1SETUP*
0x51 *LIMIT1REGULATIONMODE*
0x52 *LIMIT1POSITION*
0x53 *LIMIT1XINPUT*
0x58 *LIMIT2SETUP*
0x59 *LIMIT2REGULATIONMODE*
0x5A *LIMIT2POSITION*
0x5B *LIMIT2XINPUT*

Active:
Each time the processor is running.
For safety purposes, set *REGULATIONMODE* to stop-mode before running this function.
Do not change any of these parameters during 1 sec while saving !

RESTORE USER PARAMETERS

Function Address	Function Name	Function	Read/Write Control
0x04	*RESTOREUSERPARAMETERS*	Restores saved values	Write only

Register Size	Register structure	Unit
0 Byte	none	none

Description:
Restores the user parameters from EEPROM.

See *SAVEUSERPARAMETERS* (0x03) register list.

Active:
Each time the processor is running.
For safety purposes, set *REGULATIONMODE* to stop-mode before running this function.

RESTORE FACTORY PARAMETERS

Function Address	Function Name	Function	Read/Write Control
0x05	*RESTOREFACTORYPARAMETERS*	Factory default	Write only

Register Size	Register structure	Unit
0 Byte	none	none

Description:
Restores factory parameters.

See *SAVEUSERPARAMETERS* (0x03) register list.

Active:
Each time the processor is running
SAVEUSERPARAMETERS must be done after this function for next reboot to use these parameters.
For safety purposes, set *REGULATIONMODE* to stop-mode before running this function.

SAVE FACTORY PARAMETERS

Function Address	Function Name	Function	Read/Write Control
0x06	*SAVEFACTORYPARAMETERS*	Saves factory default	Write only

Register Size	Register structure	Unit
0 Byte	none	none

Description:
This function is reserved for integrator engineers, and not for end-user.
Used when all parameters have been approved for an application. It saves in EEPROM all configurable registers for both factory parameters and user parameters
This function already includes *SAVEUSERPARAMETERS* function.

See *SAVEUSERPARAMETERS* (0x03) register list.

Active:
Each time the processor is running.
For safety purposes, set *REGULATIONMODE* to stop-mode before running this function.
Do not change any of these parameters during 1 sec while saving!

VOLTAGE

Register Address	Register Name	Function	Read/Write Control
0x07	*VOLTAGE*	Power module voltage	Read only

Register Size	Register structure	Unit
4 Bytes	Signed (2's cplt) Int 16 (HH-HL) +16 bits fixed point (LH-LL)	Volt

Description:
Input Voltage

Limits:
Max 0x7FFFFFxx = 32'767.996
Min 0x000000xx = 0.0
Step 0x000001xx = 0.004

Example:
When read 0x00234567 = 2311527 , Voltage = 35.27 (2311527/655636)

Information:
Over 54 V (0x00360000) *REGULATIONMODE* is set in *OpenDriver* mode
Below 9 V (0x00090000) *REGULATIONMODE* is set in *Brake* mode.
Below effective 6.5 V (0x00068000), this value has no meaning.

Active:
Each time the processor is running.

WARNING

Register Address	Register Name	Function	Read/Write Control
0x08	*WARNING*	Bit to bit state	R/W

Register Size	Register structure	Unit
4 Byte	Unsigned Int 32 bits , each bit independent	none

Description:
Each information/warning/error is contained in 2 bits, the first one shows the current state, the next one shows if this state appeared previously.
Only the bits that show the past states can be cleared by writing 0x00000000 to *WARNING* register.

Bits	when set
Warnings.0	Enable pin (if exists) is not activated. *REGULATIONMODE* is set to *Brake* mode.
Warnings.1	Minimum once in the past was the Enable pin (if exists) not activated.
Warnings.2	Undervoltage of the power input.
Warnings.3	Previously
Warnings.4	Overvoltage of the power input.
Warnings.5	Previously
Warnings.6	In speed or position mode only, when the *INPUT+/- DEADZONE* has not been reached.
Warnings.7	Previously
Warnings.8	absolute value of *COMMAND* register is saturated to 0x0000FFFF (65535)
Warnings.9	Previously
Warnings.10	While *HOMING* is running.
Warnings.11	If no home has been found.
Warnings.12-13	Reserved
Warnings.14	Limit1- pin (if exists) has reached its action state.
Warnings.15	Previously
Warnings.16	Limit2+ pin (if exists) has reached its action state.
Warnings.17	Previously
Warnings.18-31	Reserved

Default: bits 31 -> 0
0x00000000

Active:
Each time the processor is running.

COM OPTIONS

Register Address	Register Name	Function	Read/Write Control
0x10 (16)	*COMOPTIONS*	Communication options	Read/Write

Register Size	Register structure	Unit
4 Bytes	32 individual bits	none

Description:
This register is reserved for future use.

ETHERNET MAC

Register Address	Register Name	Function	Read/Write Control
0x11 (17)	*ETHERNETMAC*	Hardware network ID	Read only

Register Size	Register structure	Unit
6 Bytes	6 x Unsigned Bytes	none

Description:
A standard hardware unique identifier (worldwide) for each device on an Ethernet network.

Note:
If the user writes into this register, the MAC address will not be modified. This register is available only for information purposes.

IP ADDRESS

Register Address	Register Name	Function	Read/Write Control
0x12 (18)	*IPADDRESS*	IP network ID	Read/Write

Register Size	Register structure	Unit
4 Bytes	4 x Unsigned Bytes	none

Description:
Network identifier used for TCP/IP and UDP/IP.
The values 255 (0xFF) and 0 (0x00) are reserved for broadcast and network addresses and should not be used in this register.

Notes:
The module will change to a new IP address only when all of its communication ports are closed.
Do not forget to use a *SAVEUSERPARAMETERS* command.

Default value:
169.254.5.5

Example:
For the IP=192.168.16.14 (0xC0, 0xA8, 0x10, 0x0E), write 0xC0A8100E to IPADDRESS.

SUBNET MASK

Register Address	Register Name	Function	Read/Write Control
0x13 (19)	*SUBNETMASK*	IP subnet mask	Read/Write

Register Size	Register structure	Unit
4 Bytes	4 x Unsigned Bytes	none

Description:
Network IP subnet mask used for TCP/IP and UDP/IP.

Notes:
The module will change for a new subnet mask only when all of its communication ports are closed.
Do not forget to use a *SAVEUSERPARAMETERS* command.

If you do not want to use subnets, use the following subnet mask when IP address byte 0 is:
>0 and <=127 : 255.0.0.0 (Class A addresses)
>127 and <=191 : 255.255.0.0 (Class B addresses)
>191 and <=223 : 255.255.255.0 (Class C addresses)

Default value:
255.255.0.0

Example:
For the IP=10.2.6.45 and subnet mask = 255.255.0.0:
IP address class = A → netID = 10, subNetID = 2 and hostID = 6.45

TCP TIMEOUT

Register Address	Register Name	Function	Read/Write Control
0x14 (20)	*TCPTIMEOUT*	Timeout for TCP connection	Read/Write

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	sec

Description:
The TCP timeout is a value (in seconds) after which the user will be disconnected if the board has not been accessed in the meantime.

If the value is 0, the TCP timeout is deactivated. In this case however, if the client crashes during connection, the communication will never be closed on the module's side! Because a maximum of 4 communications are allowed at the same time on the module, one of them will be blocked. If the client crashes four times, all of the 4 communications will be blocked and the module will have to be reset.

The timeout for each TCP/IP connection is reloaded when there is traffic through the port.

Default value:
30

Limitations:
Max value: 255

MODULE NAME

Register Address	Register Name	Function	Read/Write Control
0x15 (21)	*MODULENAME*	Module's ASCII name	Read/Write

Register Size	Register structure	Unit
16 Bytes	16 (only) x Unsigned Bytes (CHAR)	none

Description:
Name and/or description of the module.

Example:
For the name "Hello Module"; extend to 16 byte the name: "Hello Module"+5x space=16 Byte.
So write 0x48656C6C 6F204D6F 64756C65 20202020.

TCP CONNECTIONS OPENED

Register Address	Register Name	Function	Read/Write Control
0x1A (26)	*TCPCONNECTIONSOPENED*	Number of opened TCP connections	Read only

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	none

Description:
Number of users connected to the card using TCP.
Value can be 0 to 4.

REGULATION MODE

Register Address	Register Name	Function	Read/Write Control
0x20 (32)	*REGULATIONMODE*	Select regulation mode	R/W

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	none

Description:
Selects the regulation mode between:

0x00	*Brake*	(brake and stop the motor)
0x01	*DriverOpen*	(disconnect motor pins from ground and power supply)
0x02	*OpenLoop*	(Input reg is directly converted to PWM ouputs)
0x03	*WaitMode*	(continue actual output (PWM), without regulation)
0x04	*SpeedControl*	(acceleration to Input speed, with PID algorithm)
0x05	*PositionControl*	(acceleration, top speed and deceleration ramps with PID)
>0x05	*CloseLoopTypeError*	(set PWM value to zero)

Limits:
Min = 0
Max = 5

If *VOLTAGE* is below 9.0 V, *REGULATIONMODE* is automatically set to *Brake* mode.
if *VOLTAGE* is higher than 54.0 V, *REGULATIONMODE* is automatically set to *DriverOpen* mode.

Default:
After Power ON, *REGULATIONMODE* is in *Brake* mode.

Active:
Each time the processor is running

INPUT

Register Address	Register Name	Function	Read/Write Control
0x21 (33)	*INPUT*	Master register	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	depends

Description:
This register is the input (if needed) for every *REGULATIONMODE*.

Active:
When *REGULATIONMODE = Open-Loop*, *Position Control, Speed Control, or Wait-Mode*

REGULATIONMODE = Open-Loop:
INPUT unit is % of PWM

Max	0x0000FFFF = 65535	PWM = 100% (65535/65536)
Zero	0x00000000 = 0	PWM = 0% (0/65536)
Min	0xFFFF0001 = -65535	PWM =-100% (-65535/65536)

Higher and lower values are automatically saturated when converted to PWM

REGULATIONMODE = Position Control
INPUT is the position to reach, and the unit is pulses

Max	0x7FFFFFFF = 2'147'483'647
Min	0x80000000 = -2'147'483'648

REGULATIONMODE = Speed Control
INPUT is the speed to reach, and the unit is pulses/sec

Max	0x7FFFFFFF = 2'147'483'647
Min	0x80000000 = -2'147'483'648

Default:
0x00000000 = 0

INPUT OFFSET

Register Address	Register Name	Function	Read/Write Control
0x22 (34)	*INPUTOFFSET*	Master register offset	Write only

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	depends

Description:
This register is added to *INPUT* register with each regulation loop, and immediately cleared.
This register enables modification of *INPUT* register without knowing its value.
See *INPUTMIN (0x24)* and *INPUTMAX (0x25)* registers to define the range of the *INPUT*.

Active:
When *HOMING* is not running, else it is ignored and cleared.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x80000000 = -2'147'483'648

Default:
0x00000000 = 0

INPUT OFFSET MEASURED

Register Address	Register Name	Function	Read/Write Control
0x23 (35)	*INPUTOFFSETMEASURED*	New *INPUT* value with local parameter	Write only

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	depends

Description:
This register is added to the measured value *POSITION* or *SPEED*, depending on the running regulation mode. When done *INPUTOFFSETMEASURED* is immediately cleared. The result is copied to *INPUT* register.
This register enables to set a new *INPUT* value without knowing, externally to the motion card, the real-time position or speed value.
See *INPUTMIN (0x24)* and *INPUTMAX (0x25)* registers to define the range of the *INPUT*.

Active:
Only when its value is not equal to 0.
When *HOMING* is not running, else it is ignored and cleared.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x80000000 = -2'147'483'648

Default:
0x00000000 = 0

INPUT MIN

Register Address	Register Name	Function	Read/Write Control
0x24 (36)	*INPUTMIN*	Lowest *INPUT* accepted	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	depends

Description:
This register is a software limitation of *INPUT (0x21)* reg.
If *INPUT < INPUTMIN* then *INPUT= INPUTMIN*.
With complementary *INPUTMAX (0x25)* register, a range of *INPUT* value can be determined.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x80000000 = -2'147'483'648

Default:
Min -> never influences *INPUT* value

Active:
Each time the processor is running, but ignored during homing!

INPUT MAX

Register Address	Register Name	Function	Read/Write Control
0x25 (37)	*INPUTMAX*	Highest *INPUT* accepted	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	depends

Description:
This register is a software limitation of *INPUT (0x21)* reg.
If *INPUT* > *INPUTMAX* then *INPUT= INPUTMAX*.
With complementary *INPUTMIN (0x24)* register, a range of *INPUT* value can be determined.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x80000000 = -2'147'483'648

Default:
Max -> never influences *INPUT* value

Active:
Each time the processor is running, but ignored during homing!

POSITION

Register Address	Register Name	Function	Read/Write Control
0x26 (38)	*POSITION*	Signed pulses counter	R/W

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	Pulse

Description:
Write to this register to set a new position (calibration). See *POSITIONOFFSET (0x27)* register for recalibration.
When read, it shows the evolution of the pulses in an absolute range.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x80000000 = -2'147'483'648

Default:
After Power ON, *POSITION* is cleared (0x00000000), or if *OPTIONS.6* bit is set the *POSITION* before the last shut down is reloaded.

Active:
Each time the processor is running

POSITION OFFSET

Register Address	Register Name	Function	Read/Write Control
0x27 (39)	*POSITIONOFFSET*	Calibrate *POSITION* reg.	Write only (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	Pulse

Description:
Write to this register to shift the *POSITION* register with an offset.
When addition of *POSITIONOFFSET* and *POSITION* has been done, *POSITIONOFFSET* is automatically cleared.
It is preferable to write this register instead of writing *POSITION* reg when motor moves, so pulses will never be lost.

Example: if for an application, it is necessary to reset the position each complete rotation;
when *POSITION* > (1 complete rotation), write – (1 complete rotation) to *POSITIONOFFSET*,
when *POSITION* < - (1 complete rotation) write + (1 complete rotation).

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x80000000 = -2'147'483'648

Active:
Each time the processor is running

SPEED

Register Address	Register Name	Function	Read/Write Control
0x28 (40)	*SPEED*	Time-based pulse count	Read only

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	Pulse/sec

Description:
It represents the number of pulses that could be counted during 1 second.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x80000000 = -2'147'483'648

Accuracy depends on the speed refresh rate, which depends on regulation *LOOPTIME (0x2D)*.

For example: with a refresh rate of 1000Hz, 1 pulse added or missing during the time of 1ms represents +/-1000 pulses during 1 second (1x 1000 = 1000). With a slow refresh rate (*LOOPTIME*), you will have a good speed accuracy, but the regulation loop will perform more slowly.

Example:
The motor runs at maximum speed (PWM saturated), 123'456 pulses/s:

20Hz : 123'456 +-20 0.016% error (max pulses/ speed refresh)
200Hz: : 123'456 +-200 0.16% error
2000Hz : 123'456 +-2000 1.6% error

An encoder with more pulse/rotation allows for a better accuracy!!

If *OPTIONS.2 (0x2C)* bit is set, the *SPEED* has a 10x better accuracy, without changing *LOOPTIME*.

Active:
Each time the processor is running

CURRENT MAX

Register Address	Register Name	Function	Read/Write Control
0x2A (42)	*CURRENTMAX*	Limits output current	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's cplt) Int 16 (HH-HL) +16 bits fixed point (LH-LL)	Ampere

Description:
Limits the output current for the different reasons:

- Motor torque
- Motor heating (Rxl^2)
- Maximum motor current
- Power supply limitation (for example Power Over Ethernet is limited to 0.35 A)

Limits:
Max 0x000300xx = 3.0 A
Min 0x000000xx = 0.0 A
Step 0x000001xx = 0.004 A

Do not exceed 3.0A, because of driver heating and fuse limitation.

Default:
POE 0x00005999 = 22937, current limitation 0.35 A (22937/65536)
Other 0x00018000 = 98304 , current limitation 1.5 A (98304/65536)

Active:
With every *REGULATIONMODE* but during *WaitMode*, update of *CURRENTMAX* is not refreshed.

OPTIONS

Register Address	Register Name	Function	Read/Write Control
0x2C (44)	*OPTIONS*	Bit to bit settings	Write (Read)

Register Size	Register structure	Unit
4 Byte	Unsigned Int 32 bits , each bit independent	none

Description:

Bits	*when set*
Options.0	Swaps encoder pulses-inputs (e.g. if Channel A & B are not correctly wired).
Options.1	Inverts the rotation of the motor (swaps pulses-inputs and PWM-outputs).
Options.2	Better accuracy of speed measurement (x10), calculated in (x10) more time (less realtime).
Options.3	Non linear PID, decreases integration faster than increase, may sometimes cancel oscillations.
Options.4	PWM to 10 bits (39kHz) instead of 9 bits (78kHz).
Options.5	Pull-Up (0=Pull-Down) resistors of 10 kOhm on the *Limit1Signal* and *Limit2Signal* pin
Options.6	Saves *POSITION* reg to EEPROM when shut down appears (*VOLTAGE*<7.0V)
Options.7	Leakage (decrease 1/256 each *LOOPTIME*) of *INTEGRALDELTA* when stable.
Options.8-31	Reserved

Limits:
None

Default:
bits 31 -> 0 : 0x00, 0x00, 0x00, b'11111100'

Active:
Each time the processor is running, *Options.3* and *Options.7* only when PID is in use.

LOOPTIME

Register Address	Register Name	Function	Read/Write Control
0x2D (45)	*LOOPTIME*	Regulation refresh rate	Write (Read)

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	Time

Description:
Refreshes rate of *POSITION*, *SPEED* and regulation loop.

0x00 50 ms, 20 Hz
0x01 20 ms, 50 Hz
0x02 10 ms, 100 Hz
0x03 5 ms, 200 Hz
0x04 2 ms, 500 Hz
0x05 1 ms, 1000 Hz
0x06 500 us, 2000 Hz

Limits:
Max: 6

Default:
6 (500 us, 2000 Hz)

Active:
Each time the processor is running.

OUTPUT VOLTAGE MAX

Register Address	Register Name	Function	Read/Write Control
0x2E (46)	*OUTPUTVOLTAGEMAX*	Limits output PWM	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's cplt) Int 16 (HH-HL) +16 bits fixed point (LH-LL)	Volt

Description:
This register will modify the scale between the *COMMAND* reg. and the PWM output (duty cycle).
If *VOLTAGE* > *OUTPUTVOLTAGEMAX*, then PWM duty cycle is reduced by the same factor (*VOLTAGE/ OUTPUTVOLTAGEMAX)*.

!!! Be careful to avoid the heating of a motor with a lower maximum voltage than the *VOLTAGE* of the card's power supply !!! The best solution is to select a power supply with the same voltage as the motor's recommended voltage.

Limits:

Max	0x7FFFFFFxx = 32'767.996
Min	0x000000xx = 0.0 , but it is strongly recommended that *OUTPUTVOLTAGEMAX* > *VOLTAGE* /2
Step	0x00001xx = 0.004
Disabled	0xFFFFFFFF , with this value , correction is never made

Default:
Disabled 0xFFFFFFFF

Example:

1) Inactive
VOLTAGE = 23.5 V
OUTPUTVOLTAGEMAX = 24.0 V
COMMAND range [-65535 ... 65535]
PWM duty cycle [0 – 100%]

2) Active
VOLTAGE = 46.5 V
OUTPUTVOLTAGEMAX = 24.0 V
COMMAND range [-65535 ... 65535]
PWM duty cycle [0 – 51%]

Information:
To write *OUTPUTVOLTAGEMAX* = 24.0V, send 0x00180000 = 1572864 (24.0 x 65536)

Active:
Each time the processor is running.

DESIRED

Register Address	Register Name	Function	Read/Write Control
0x30 (48)	*DESIRED*	Positive input of PID	Read only

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	Depends

Description:
REGULATIONMODE internally decidse which register is copied to *DESIRED* one. This register is used only for PID overview.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x80000000 = -2'147'483'648

Active:
Updated with every regulation refresh, when PID is selected.

FEEDBACK

Register Address	Register Name	Function	Read/Write Control
0x31 (49)	*FEEDBACK*	Negative input of PID	Read only

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	Depends

Description:
REGULATIONMODE internally decides if *SPEED* or *POSITION* is copied to *FEEDBACK* register. This register is used only for PID overview.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x80000000 = -2'147'483'648

Active:
Updated with every regulation refresh when PID is selected.

COMMAND

Register Address	Register Name	Function	Read/Write Control
0x32 (50)	*COMMAND*	Result of PID	Read only

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	none

Description:
Output of PID and input of PWM driver.

When PID is used:
COMMAND = KP x (DESIRED-FEEDBACK) + KI x INTEGRALDELTA + KD x DERIVATIONOFDELTA.
This reg is optional, and never needed. Only for PID overview.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x80000000 = -2'147'483'648

Active:
Updated with every regulation refresh when PID is selected

KP

Register Address	Register Name	Function	Read/Write Control
0x33 (51)	*KP*	PID Proportional gain	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's cplt) Int 16 (HH-HL) +16 bits fixed point (LH-LL)	none

Description:
P-parameter in PID definition.

Limits:
Max 0x7FFFFFFF = 32'767.9999847
Min 0x00000000 = 0.0
Step 0x00000001 = 0.000015

Example:
To set *KP*= 0.1234 , write 0x00001F97 = 8087 (0.1234 x 65536 = 8087.1424)

When read 0x12345678 = 305419896 , *KP* = 4660.33776 (305419896/65536)

Default:
KP is between 0.5 and 50

Active:
Used with every regulation refresh when PID is selected

KI

Register Address	Register Name	Function	Read/Write Control
0x34 (52)	*KI*	PID Integral gain	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's cplt) Int 16 (HH-HL) +16 bits fixed point (LH-LL)	none

Description:
I-parameter in PID definition.

Limits:
Max 0x7FFFFFFF = 32'767.9999847
Min 0x00000000 = 0.0
Step 0x00000001 = 0.000015

Example:
To set *KI*= 0.1234 , write 0x00001F97 = 8087 (0.1234 x 65536 = 8087.1424)

When read 0x12345678 = 305419896 , *KI* = 4660.33776 (305419896/65536)

Default:
KI is between 0.01 and 1.0

Active:
Used with every regulation refresh when PID is selected

KD

Register Address	Register Name	Function	Read/Write Control
0x35 (53)	*KD*	PID derivative gain	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's cplt) Int 16 (HH-HL) +16 bits fixed point (LH-LL)	none

Description:
D-parameter in PID definition.

Limits:
Max 0x7FFFFFFF = 32'767.9999847
Min 0x00000000 = 0.0
Step 0x00000001 = 0.000015

Example:
To set *KD*= 0.1234 , write 0x00001F97 = 8087 (0.1234 x 65536 = 8087.1424)

When read 0x12345678 = 305419896 , *KD* = 4660.33776 (305419896/65536)

Default:
KD is not used = 0

Active:
Used with every regulation refresh when PID is selected

ANTI-RESET WINDUP

Register Address	Register Name	Function	Read/Write Control
0x36 (54)	*ANTIRESETWINDUP*	Integration saturation	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	none

Description:
During PID loops, the delta (difference of *DESIRED-FEEDBACK*) is integrated *to INTEGRALDELTA*.
If the range of *INTEGRALDELTA* needs to be limited, *ANTIRESETWINDUP* is the maximum value that *INTEGRALDELTA* could reach. *ANTIRESETWINDUP* is an absolute value.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x00000000 = 0

Default:
0x7FFFFFFF. This value should not be modified

Active:
Used for every PID calculation, when *KI* is not 0.

INTEGRAL DELTA

Register Address	Register Name	Function	Read/Write Control
0x37 (55)	*INTEGRALDELTA*	PID Integral result	Read normally (Write)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	none

Description:
During PID loops the delta (difference of *DESIRED-FEEDBACK*) is integrated *in INTEGRALDELTA*.
If the range *INTEGRALDELTA* needs to be limited, *ANTIRESETWINDUP* is the maximum that *INTEGRALDELTA* could reach.

INTEGRALDELTA is cleared when *KI* = 0x00000000 (0).
INTEGRALDELTA x KI is added to PID result (COMMAND)

This register is optional, and never needed. Only for PID overview.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Zero 0x00000000 = 0
Min 0x80000000 = -2'147'483'648

Active:
Updated with every regulation refresh when PID is selected.

DERIVATION OF DELTA

Register Address	Register Name	Function	Read/Write Control
0x38 (56)	*DERIVATIONOFDELTA*	PID derivative result	Read only

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	none

Description:
During PID loops, the delta (difference of *DESIRED-FEEDBACK*) is derivated (delta-deltaold) to *DERIVATIONOFDELTA reg*.

DERIVATIONOFDELTA x KD is added to PID result (COMMAND).
This register is optional, and never needed. Only for PID overview.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Zero 0x00000000 = 0
Min 0x80000000 = -2'147'483'648

Active:
Updated with every regulation refresh when PID is selected.

ACCELERATION

Register Address	Register Name	Function	Read/Write Control
0x40 (64)	*ACCELERATION*	Speed acceleration	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	Pulse/sec²

Description:
When *REGULATIONMODE* is in position or speed control mode, *ACCELERATION* represents the evolution of the *DESIRED* speed over the first segment of the trajectory.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x00000000 = 0

Example:
If a motor (that has a null speed) needs to be at 10'000 Pulse/sec in 0.1 sec, write 0x000186A0 (100'000) to *ACCELERATION*. Calculation used to determine the previous value is the wanted speed value divided by the time allowed/needed to perform that operation (10'000/0.1).

Default:
If you have no idea, set *ACCELERATION* = maximum speed of the motor. The motor will accelerate during approximately 1 second.

Active:
Used when REGULATIONMODE is in Position or Speed Control mode.

DECELERATION

Register Address	Register Name	Function	Read/Write Control
0x41 (65)	*DECELERATION*	Speed deceleration	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	Pulse/sec²

Description:
When *REGULATIONMODE* is in position control mode only, *DECELERATION* represents the evolution of the *DESIRED* speed over the last segment of the trajectory.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x00000000 = 0

Example:
If a motor is at TOPSPEED of 100'000 Pulse/sec and needs to brake to the desired position INPUT in 0.2 second, write 0x0007A120 (500'000) to *DECELERATION*. Calculation to obtain the previous value is the speed before braking divided by the time needed to perform that operation. (100'000/0.2).

Default:
If you have no idea, set *DECELERATION* = *TOPSPEED* . The motor will decelerate during approximately 1 second.

Active:
Used when REGULATIONMODE is in Position Control mode.

TOP SPEED

Register Address	Register Name	Function	Read/Write Control
0x42 (66)	*TOPSPEED*	Maximum speed	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	Pulse/sec

Description:
When *REGULATIONMODE* is in position control mode only, *TOPSPEED* represents the evolution of the constant speed for the middle segment of the trajectory.
If the output of the motor does not accept high speeds (e.g. gear head, or inertial energy,...) use *TOPSPEED*.
Otherwise write the maximum value.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x00000000 = 0

Example:
If a motor is physically limited to 100'000 Pulse/sec, it is used with an output torque of 0 Nm.
It is not very useful for any application.
Assume that *TOPSPEED* is 50'000 Pulse/sec . During the trajectory the motor load could change and the speed value will follow the 3 predefined segments, never being higher than *TOPSPEED*.

Default:
0x7FFFFFFF, no speed limitations.

Active:
Used when REGULATIONMODE is in Position Control mode only.

DEAD ZONE

Register Address	Register Name	Function	Read/Write Control
0x43 (67)	*DEADZONE*	Cancels speed zone	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	Pulse

Description:
When *REGULATIONMODE* is in position control mode:
After the target position *INPUT* has been reached (=*POSITION*), *DEADZONE* represents a range of positions where the trajectory regulator forces the *DESIRED* speed to 0. This activity lasts from *INPUT-DEADZONE* to *INPUT+DEADZONE*.
With smooth trajectories, no mechanical elasticity and good PID settings *DEADZONE* can set down to 1. With a null *DEADZONE*, if *DECELERATION* is too big, overshoots will appear with oscillations. Decrease *DECELERATION* or increase *DEADZONE*.

Faster regulation refresh rates (*LOOPTIME*) improve the regulation and decrease overshoots.
Speed x10 interpolation (*OPTION.2*) increases overshoots.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x00000000 = 0

Example:
To know the effective overshoot of a system, set DEADZONE to 1000 pulses.
Set a new position *INPUT* (e.g. 100'000). See the *POSITION* when movement is completed.
(eg *POSITION*=100'017 –> overshoot of 17pulses).
Write 0x00000011 (17) to *DEADZONE*.

Default:
0x000000001 (1) when no overshoot of goal position is configured (*KP, KI, KD, DECELERATION*).

Active:
Used when REGULATIONMODE is in Position or Speed Control mode only.

When REGULATIONMODE is in Position or Speed Control mode:
If (*INPUT-DEADZONE)* <= measured value (position or speed) >= (*INPUT+DEADZONE*) then *WARNING.6* bit (input not reached) is cleared, else it is set.

HOMING OPTIONS

Register Address	Register Name	Function	Read/Write Control
0x48 (72)	*HOMINGOPTIONS*	Bit to bit settings	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Unsigned Int 32 bits , each bit independent	none

Description:

Homing has to reach a particular condition (chosen between the homing methods) to set the *POSITION* home reference. It is only meaningful in position control mode. This register defines the homing method and different parameters used during homing.

Bits	when set
HomingOptions.0-3	Defines the Homing method (See Table 1).
HomingOptions.4-6	Unused
HomingOptions.7	Auto-homing at power-up (internally call *HOMING* function).
HomingOptions.8-11	Defines a ratio to be multiplied with *CURRENTMAX* reg during homing (See Table 2).
HomingOptions.12-15	Defines a ratio multiplied with *TOPSPEED* reg during homing (See Table 2).
HomingOptions.16-19	Defines a ratio multiplied with *ACCELERATION* reg during homing (See Table 2).
HomingOptions.20-23	Defines the output-saturated-time [in Seconds] in the homing methods configured for current detection and limitation (See Table 2).
HomingOptions.24-31	Unused.

Table 1

4 bits	Homing method
0x0	Already at home and stay
0x1	Already at home and set *INPUT*
0x2	Negative move to the first index
0x3	Positive move to the first index
0x4	Max current detection with negative move
0x5	Max current detection with positive move
0x6	Max current detection with negative move and index
0x7	Max current detection with positive move and index
0x8	Negative move to the limit 1 switch
0x9	Positive move to the limit 1 switch
0xA	Negative move to the limit 1 switch and index
0xB	Positive move to the limit 1 switch and index
0xC	Unused
0xD	Unused
0xE	Unused
0xF	Unused

Table 2

4 bits	Corresponding ratio
0x0	2.3%
0x1	3.1%
0x2	4.7%
0x3	6.3%
0x4	9.4%
0x5	12.5%
0x6	18.8%
0x7	25%
0x8	37.5%
0x9	50%
0xA	75%
0xB	100%
0xC	150%
0xD	200%
0xE	300%
0xF	400%

Default value of the 4 bytes: [bits 31 -> 0]
0x00'BB'BB'00 equivalent to b0000'0000'1011'1011'1011'1011'0000'0000

Active:
If *HomingOptions.7* bit is set upon power up, or every time *HOMING (0x49)* function is called.

HOMING

Function Address	Function Name	Function	Read/Write Control
0x49 (73)	*HOMING*	Starts to find home	Write only

Register Size	Register structure	Unit
0 Bytes	none	none

Description:
When this function is called, it starts to find a home defined by the method described in the *HOMINGOPTIONS register*.

When this function is called:
REGULATIONMODE is internally changed to "position control mode".
WARNING.10-11 bits (homing, no home found) are set to 1.
INPUT is locked and does not accept external commands during homing.

This function can be started automatically after power-up if *HOMINGOPTIONS.7* bit is set.

Active:
Each time the processor is running.

STOP HOMING

Function Address	Function Name	Function	Read/Write Control
0x4A (74)	*STOPHOMING*	Stops a homing sequence	Write only

Register Size	Register structure	Unit
0 Bytes	none	none

Description:
A call of this function stops the homing sequence (if running). At the end of this function, the *RESTOREPARAMETERS* function will be automatically launched. The result is that (only) *CURRENTMAX, ACCELERATION and TOPSPEED* are internally reloaded with their default value. *INPUT* is cleared, *REGULATIONMODE* is set to brake-mode. *WARNING.11* bit (homing) is cleared.

There are three possibilities to stop the homing:

1) When the home is found (*REGULATIONMODE* is set to position control).
2) When *STOPHOMING* is called (*REGULATIONMODE* is set to brake-mode).
3) Power-down the whole card.

Active:
When homing sequence is running, else it has no action.

HOMING POSITION

Register Address	Register Name	Function	Read/Write Control
0x4B (75)	*HOMINGPOSITION*	Sets the Home reference value	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	Pulse

Description:
Write values to this register to set a new reference position during homing. Only for "position control mode".

If *HOMINGOPTIONS (0x48)* is activated and *HOMING* function is running, *HOMINGPOSITION* is copied to *POSITION* register when the conditions are reached. *WARNING.10* bit (no Home) is cleared since a (new) home has been found.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x80000000 = -2'147'483'648

Default:
0x00000000 = 0

Active:
After *HOMING* function, when *HOMINGOPTIONS* conditions are reached.

HOMING INPUT

Register Address	Register Name	Function	Read/Write Control
0x4C (76)	*HOMINGINPUT*	Sets a new goal when home is found	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	Pulse

Description:
Write a value to this register to set a new *INPUT* when homing ends. Only for "position control mode".

If *HOMINGOPTIONS (0x48)* is activated and *HOMING* function is running, when the conditions are reached, *HOMININPUT* is copied into *INPUT* register. *WARNING.11* bit (Homing) is cleared since homing is finished.

Limits:
Max 0x7FFFFFFF = 2'147'483'647
Min 0x80000000 = -2'147'483'648

Default:
0x00000000 = 0

Active:
After *HOMING* function, when *HOMINGOPTIONS* conditions are reached

LIMIT 1 (home) SETUP

Register Address	Register Name	Function	Read/Write Control
0x50 (80)	*LIMIT1SETUP*	Configures limit n° 1	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Unsigned Int 32 bits , each bit independent	none

Bits description:

This register configures the event depending on the "Limit 1s" input pin. (see Overview chapter)

When set to 1:

Bit number	Bit name	Bit description
LIMIT1SETUP .0	*bLimit1Enable*	Activates the limit number1 detection
LIMIT1SETUP .1	*bLimit1ActivHigh*	When clear, it is active low
LIMIT1SETUP .2	*bLimit1RegulationMode*	Copy *LIMIT1REGULATIONMODE* to *REGULATIONMODE*
LIMIT1SETUP .3	*bLimit1Position*	Copy *LIMIT1POSITION* to *POSITION* (once)
LIMIT1SETUP .4	*bLimit1Index*	not used (0), future functionality
LIMIT1SETUP .5	*bLimit1xInputL*	Defines where *LIMIT1xINPUT* must be copied to
LIMIT1SETUP .6	*bLimit1xInputH*	with previous bit
LIMIT1SETUP .5-31	none	not used (0)

bLimit1xInputH , bLimit1xInputL action:

0 0 *LIMIT1xINPUT* unused
0 1 *LIMIT1xINPUT* copied to *INPUT* (each regulation loop)
1 0 *LIMIT1xINPUT* copied to *INPUTOFFSET* (once)
1 1 *LIMIT1xINPUT* copied to *INPUTOFFSETMEASURED* (each regulation loop)

Example:

With a PNP proximity detector (mechanic, optical, magnetic ...), a pull-down resistor (on the limits input pin) can be selected with *OPTIONS.5* bit. If the detector is active high, set *LIMIT1SETUP .1* bit.
If the detector is at reference position, set *LIMIT1SETUP .3* bit and write the reference value to *LIMIT1POSITION* register. Activate *LIMIT1SETUP.0* bit to enable the activity of the limit n° 1 .

To configure a limit, first deactivate the limit, then select pull-up or pull-down resistors, write only necessary registers between *LIMIT1XINPUT*, *LIMIT1POSITION*, *LIMIT1REGULATIONMODE* and after that, write *LIMIT1SETUP* with enable bit set.

Default:
0x00000000 (deactivated)

Active:
When Homing is not running

LIMIT 1 SET REGULATION MODE

Register Address	Register Name	Function	Read/Write Control
0x51 (81)	*LIMIT1REGULATIONMODE*	Limit 1 regulation mode	Write (Read)

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	none

Description:
When "Limit 1s" is reached, if *LIMIT1SETUP.0* is set (enable) and *LIMIT1SETUP.2* is set (new regulation mode), then *LIMIT1REGULATIONMODE* is copied to *REGULATIONMODE*.

See *REGULATIONMODE (0x50)* for other description.

Active:
When Limit1 is reached.

LIMIT 1 SET POSITION

Register Address	Register Name	Function	Read/Write Control
0x52 (82)	*LIMIT1POSITION*	Sets a new position	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	Pulse

Description:
When Limit 1s is reached, if *LIMIT1SETUP.0* is set (enable) and *LIMIT1SETUP.3* is set (new position), then *LIMIT1POSITION* is copied to *POSITION.*

See *POSITION (0x26)* for other description.

Active:
When Limit1 is reached.

LIMIT 1 SET X- INPUT

Register Address	Register Name	Function	Read/Write Control
0x53 (83)	*LIMIT1XINPUT*	Sets a new input	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	depends

Description:
When Limit 1s is reached, if LIMIT1SETUP.0 is set (enable) and LIMIT1SETUP.5-6 not 00, then LIMIT1XINPUT is copied to INPUT or INPUTOFFSET or INPUTOFFSETMEASURED.

See LIMIT1SETUP (0x50) for destination description.

Active:
When Limit1 is reached.

LIMIT 2 SETUP

Register Address	Register Name	Function	Read/Write Control
0x58 (88)	*LIMIT2SETUP*	Configures limit n° 2	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Unsigned Int 32 bits , each bit independent	none

Bits description:

LIMIT2SETUP is the same register as *LIMIT1SETUP* but for limit n° 2 .
See *LIMIT1SETUP (0x50)* for more information.

LIMIT 2 SET REGULATION MODE

Register Address	Register Name	Function	Read/Write Control
0x59 (89)	*LIMIT2REGULATIONMODE*	Limit 2 regulation mode	Write (Read)

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	none

Description:
LIMIT2REGULATIONMODE is the same register as *LIMIT1REGULATIONMODE* but for limit n° 2 .
See *LIMIT1REGULATIONMODE (0x51)* for more information.

LIMIT 2 SET POSITION

Register Address	Register Name	Function	Read/Write Control
0x5A (90)	*LIMIT2POSITION*	Sets a new position	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	Pulse

Description:
LIMIT2POSITION is the same register as *LIMIT1POSITION* but for limit n° 2 .
See *LIMIT1POSITION (0x52)* for more information.

LIMIT 2 SET X- INPUT

Register Address	Register Name	Function	Read/Write Control
0x5B (91)	*LIMIT2XINPUT*	Sets a new input	Write (Read)

Register Size	Register structure	Unit
4 Bytes	Signed (2's complement) Int 32	depends

Description:
LIMIT2XINPUT is the same register as *LIMIT1XINPUT* but for limit n° 2 .
See *LIMIT1XINPUT (0x53)* for more information.

Contact address:

FiveCo - Innovative Engineering
Ch. de la Rueyre 116
CH-1020 Renens
Switzerland
Tel: +41 21 632 60 10
Fax: +41 21 632 60 11

www.fiveco.com
info@fiveco.com