e3PLC (TM) Studio

for Titanio, Platino and Vanadio Drives

Release 1.6 Build 00 (27/06/2022)

Platino BLDC - SERVO - DRIVES

TITANIO VECTOR - STEPPER - DRIVES

VANADIO AC - SERVO - DRIVES

IMPORTANT NOTICE

EVER Elettronica srl
Via del Commercio , 9/11 Loc. S. Grato Z.I.
26900 – LODI - ITALY
Tel. ++39(0)371412318
Fax ++39(0)371412367
E-mail: *support@everelettronica.it*
Web: *www.everelettronica.it*

Release History:

Release	Date	Description
1.0	12-02-2019	First Issue.
1.1	12-11-2019	- Add §9.4 : 'Motor Stall detection' feature - Add objects related to 'Motor Stall detection' feature
1.2	27-01-2020	- Add 'Torque mode' feature - Add objects related to 'Torque mode' feature
1.3	31-05-2021	- Removed BASIC License - Fix size of object Feedback_Limit_Speed - Added objects of Hall Sensors handling. - Added '*Brake Control*' feature and related objects. - Added *Feedback_Sensor_Calibration_mode*. - Modified name and meaning of 'Motor_Poles' object (new name 'Motor_Pole_Pairs') - Added *Motor_Resolution* object. - Added description of bit12, bit14 of *Feedback_Settings* object.
1.4	31-03-2022	- Update of the manual from Titanio eePLC to e3PLC. - Added 'Braking Resistor function' description (*§9.7*) and 'Braking resistor objects'.
1.5	07-06-2022	- Modified description of *Brake_Control_Settings object*
1.6	27-06-2022	- Modified 'Store_Parameters' CANopen Address: from 1010.0H to 1010.1H. - Modified nomenclature of 'Feedback_Source_PPR': now it is 'Feedback_Encoder_PPR'. - Increased 'Feedback_Encoder_PPR' description.

Related Publications

PI-MBUS-300 Rev. J	**Modicon Modbus Protocol Reference Guide**
CiA DS 301 V4.01	**CANopen Application Layer and Communication Profile**

Manual_SW1_Labelling_Realtime_Module_EN

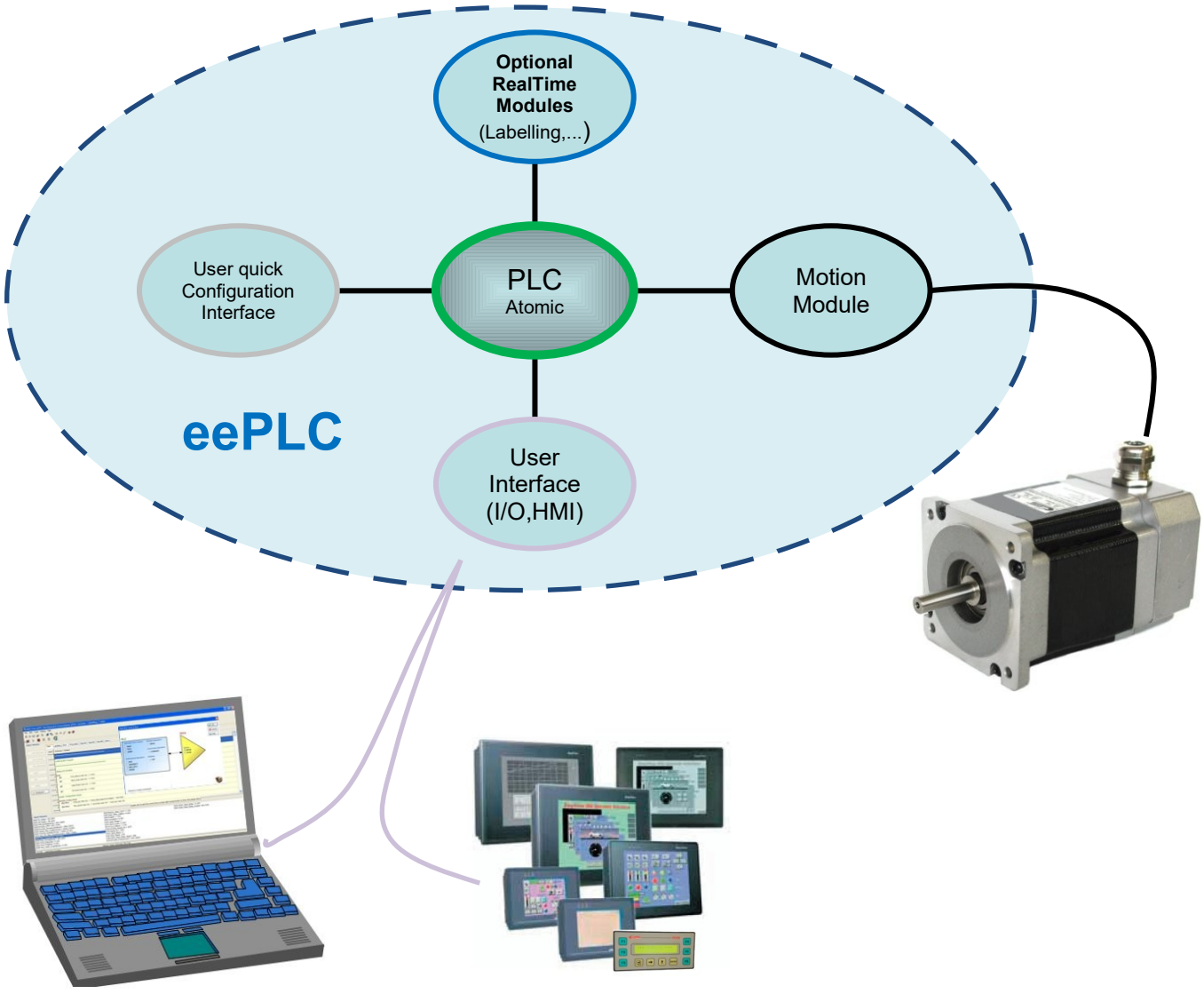
Related Web Sites

www.modbus.org	**Official web site of MODBUS Organization**
www.can-cia.de	**Official web site of CAN in Automation Organization**
www.everelettronica.it	**Official EVER web site**

General Index

1.0 Introduction

The e3PLC is a micro programming language designed for the EVER Titanio-Platino-Vanadio family drives based on the Atomic language for SDM family drives. The scope of this language is to give to the user the freedom to create his own simple application without needing to switch to more complex and more expensive drives. The philosophy of e3PLC is to have few but powerful instructions and to integrate together PLC (with real-time modules for special process handling) and motion functionality. The programming of e3PLC is done by means of an user friendly Personal Computer software supplied by EVER.



1.1 Technical Specifications

The e3PLC technical specifications are described in the table below:

Number of microinstructions	**15**
Max program size (bytes)	**8192**
Instructions Medium Length (bytes)	**6**
Number of user variables	**144**
Support for 4 bytes integer numbers	**YES**
Support for floating point numbers	**LIMITED(*)**
Access to all the drive objects	**YES**
Access to all the drive I/O's	**YES**
Multitasking Support	**YES**
Number of User Tasks	**8**
Expected medium execution time per instruction	**< 100us**

(*) Only limited to **CONVERT** instruction.

1.2 Installation of e3PLC Studio

The e3PLC Studio has the following system requirements:

- **CPU:** i3 class or better.
- **Operating system:** Windows™ 7/8/8.1/10
- **Memory:** The minimum required by the operating system plus 512 MB
- **Hard Disk:** 50Mb free space
- **Comm. Interface:** 1 Serial Interface (RS232/485) or CAN Interface[1] (typically EverElettronica new lines drives are provided with a serial service KIT)

In order to install e3PLC, you need to double click  e3PLC_Studio_Setup.exe that you can find in the USB stick provided by EverElettronica.

Alternatively you can go to the website ' *www.everelettronica.com* ' in the section:

Products → Software → Development Environments → E3PLC

And then download e3PLC Studio.

[1] The CAN interfaces supported are: IXXAT USB-to-CAN Compact, IXXAT iPC-I 320/PCI, IXXAT TinCAN V4. (*www.ixxat.com*), PEAK PCAN-USB, GC USBCAN-I (provided by EVER)

1.3 e3PLC Studio Licensing

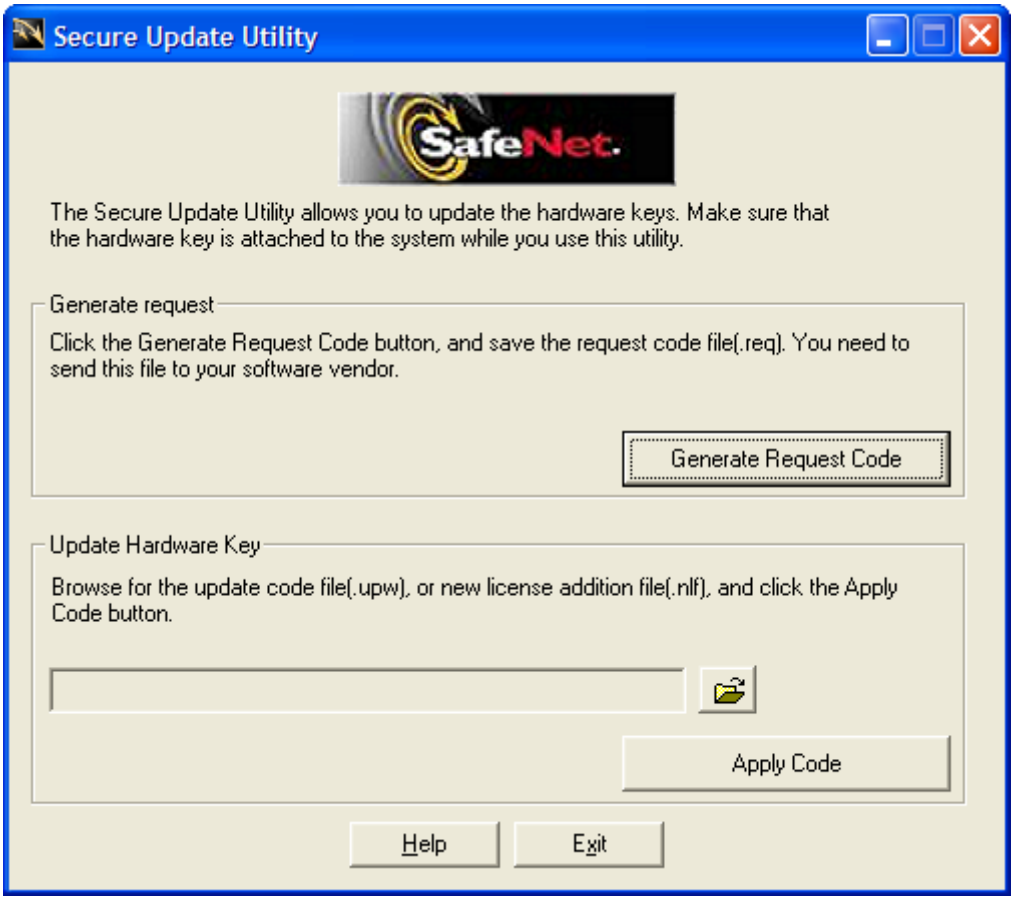
The licensing scheme for the e3PLC Studio is the following:

- ***DEMO VERSION:***
  - No hardware key needed.
  - Limitations:
    - The Development Environment can be used for 40 minutes (every 10 minutes a window will pop up showing the remaining time), then the program will close and it will be necessary to restart it (the user application will be lost if not saved within the trial time).
    - Applications up to 128 bytes long can be downloaded to the drive.
    - The Application Upload is not available.

- ***FULL VERSION:***
  - USB Hardware key required (supplied by EVER) for each computer running the e3PLC Studio.
  - Limitations:
    - None.

1.3.1 License Update or Upgrade

The DEMO VERSION of the e3PLC Studio is automatically activated when a USB Hardware Key is not present or when the license stored in the Key has expired.
The USB Hardware Key can be updated (extending the possible time expiration) or upgraded (from BASIC to FULL VERSION license).
If the Hardware Key is a SafeNET type it is necessary to run the *'Secure Update Utility'* that can be found in the CD-ROM under *'SafeNet/SecureUpdate'* directory:



With the USB Hardware key inserted in the PC, press the *'Generate Request Code'* and send the created .req file to EVER support/customer care. To update/upgrade your license, load the .upw/.nlf file sent by EVER pressing on the  button, then press on the *'Apply Code'* button.

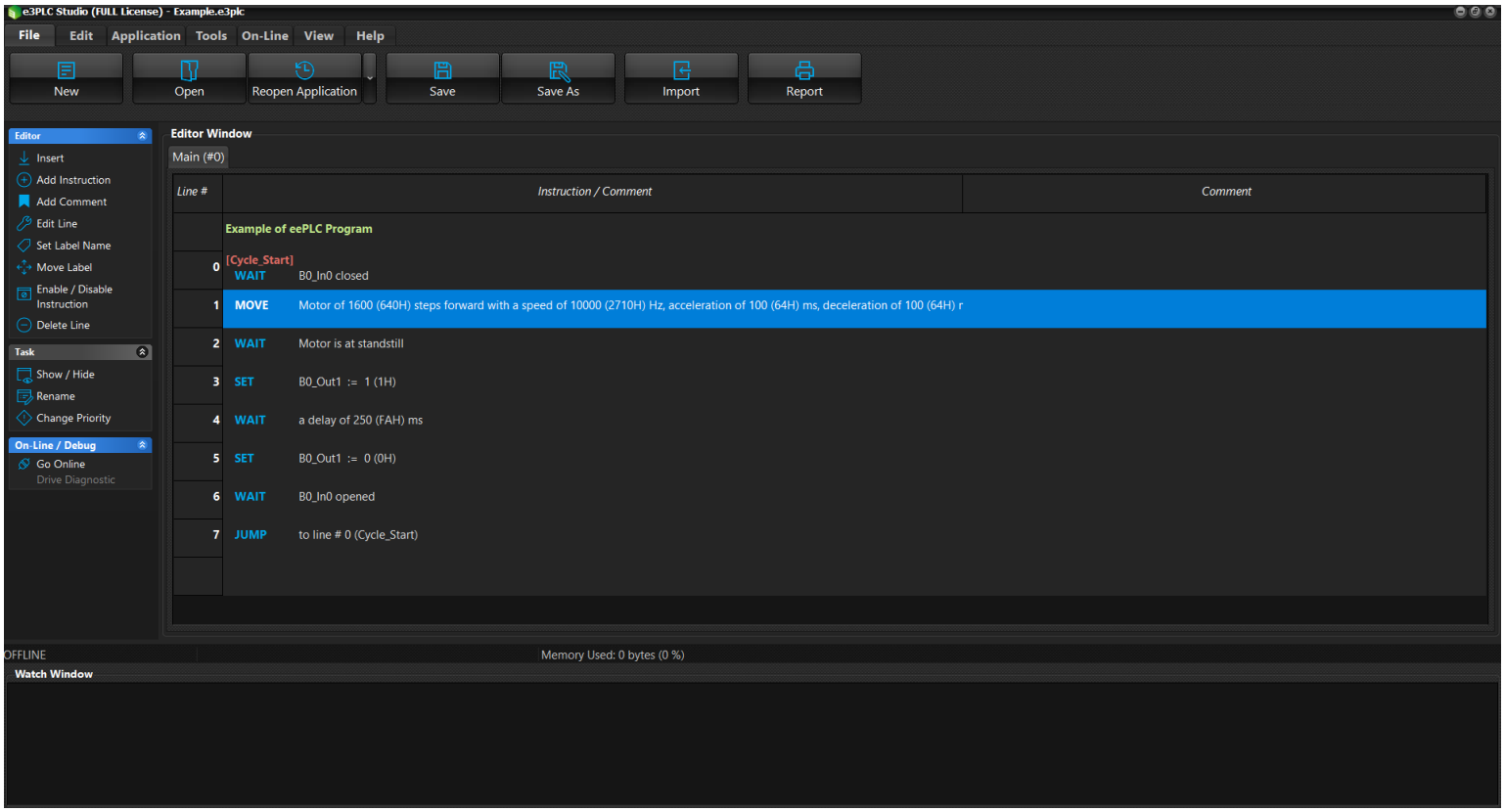
If the Hardware Key is a Keylok II type simply choose '*Help/Update Dongle'* inside the eePLC Studio IDE and select the AUTHORIZE.DAT file supplied by EVER.
Another way to update a Keylok II type Hardware Key is to execute the RemoteUpdateEmailUser.exe that can be found in the CD-ROM under *'Keylok/Secure Update'* directory.

1.4 Executing the e3PLC Studio Development Environment

To start the e3PLC Studio double click on the icon: that is present on the Windows™ Desktop as well as in the program group menu.

It is also possible to start the e3PLC Studio when double clicking on the e3PLC applications (.plc extension), using the Windows™ Resource Explorer, that has the following icon:

Then the e3PLC Studio main window appears:

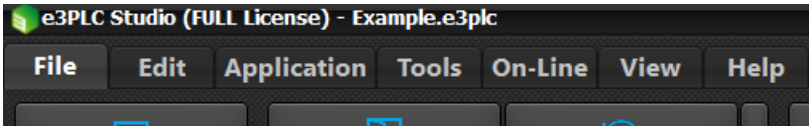


1.5 e3PLC Studio Main window

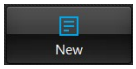
The e3PLC Studio is the main tool to create, debug, load, save, download and upload the user application.

The e3PLC Studio tool bar is divided in 7 bars:

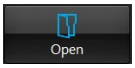
File; Edit; Application; Tools; On-Line; View; Help;



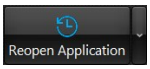
The following commands are available in the e3PLC Studio **File** bar:



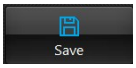
Create a new application (deleting the loaded one)



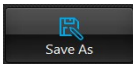
Load an application from the disk



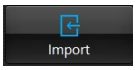
Reopen a recent application



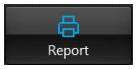
Save the current edited application on the disk



Save the current edited application on the disk for the first time



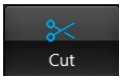
Import an e3PLC application



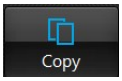
Create a report of the current edited application

The following commands are available in the e3PLC Studio **Edit** bar:

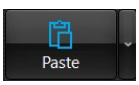
Undo edit operation



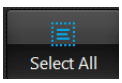
Cut current selected editor line



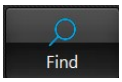
Copy current selected editor line



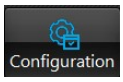
Paste current selected editor line



Select all editor lines

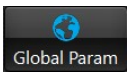


Find an object in the application

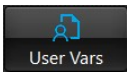


Open Setup Configuration window (see §*1.6*)

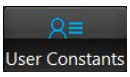
The following commands are available in the e3PLC Studio **Application** bar:



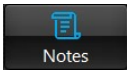
Open the Global Parameters window (see §*1.7*)



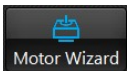
Open the User Variables window (see §*1.8*)



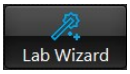
Open the User Variables window (see §*1.9*)



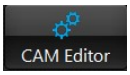
Open the User Notes window (see §*1.10*)



Open the Motor Wizard window (see §*1.11*)

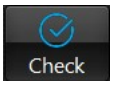


Open the LAB Wizard window (see §*1.12*)

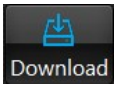


Open the LAB Wizard window (see §*1.13*)

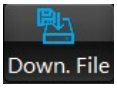
The following commands are available in the e3PLC Studio **Tools** bar:



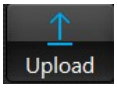
Check (compile) the current edited application (see §*1.14*)



Check the current edited application and download it to drive (see §*1.15*)



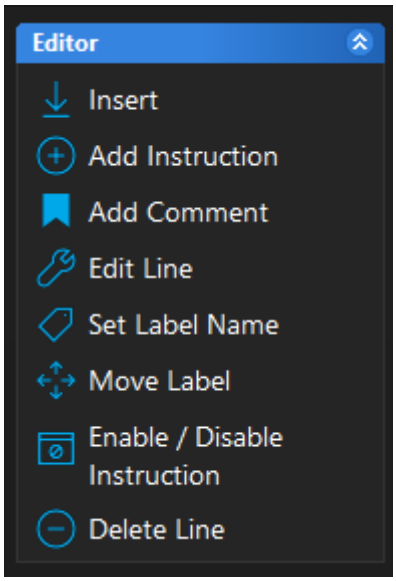
download an application by File to drive (see §*1.16*)



Upload the application stored on the drive (see §*1.17*)

Update drive firmware

The following commands are available next to the e3PLC Studio **Editor Window**:



-To insert a new e3PLC instruction or a new user comment into the current selected line.

-To add a new e3PLC instruction at the end of the current task instructions list

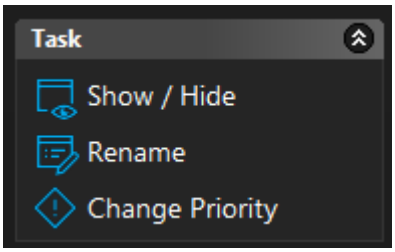
-To add a new user comment line at the end of the current task instructions list.

-To edit the current selected line (e3PLC instruction or user comment). If the line is empty is the same behavior than pressing the *'Insert'* button.
-To assign a label name to the current selected instruction line.

-To Move a label name assigned to an instruction line into another instruction line.

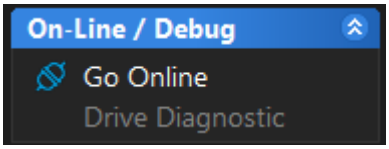
-To uncomment / comment an instruction (and so enable / disable it).

-To delete the current selected line (e3PLC instruction or user comment).



-To open the Show/Hide Tasks window that permits to visualize each of the sixteen available tasks (see §*1.18*).
-To rename the selected task in the Task Tabs.

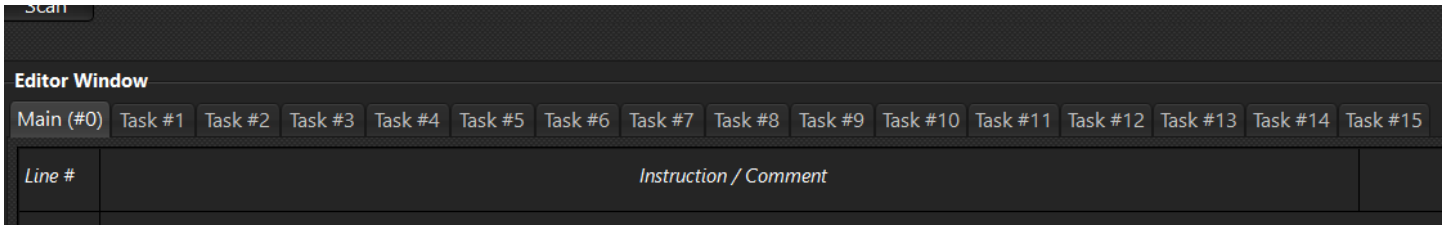
-To change Priority to all tasks simultaneously.



-To go online for debugging (see §*3.0*).

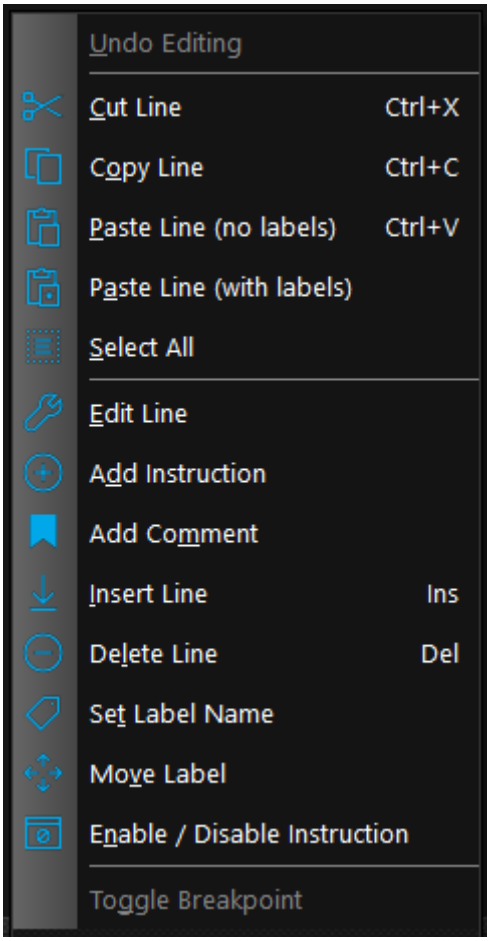
-To open the Drive Diagnostic Window (see §*3.3*).

The task tabs permit to edit the 16 concurrent tasks handled by the e3PLC Application executor on the drive.



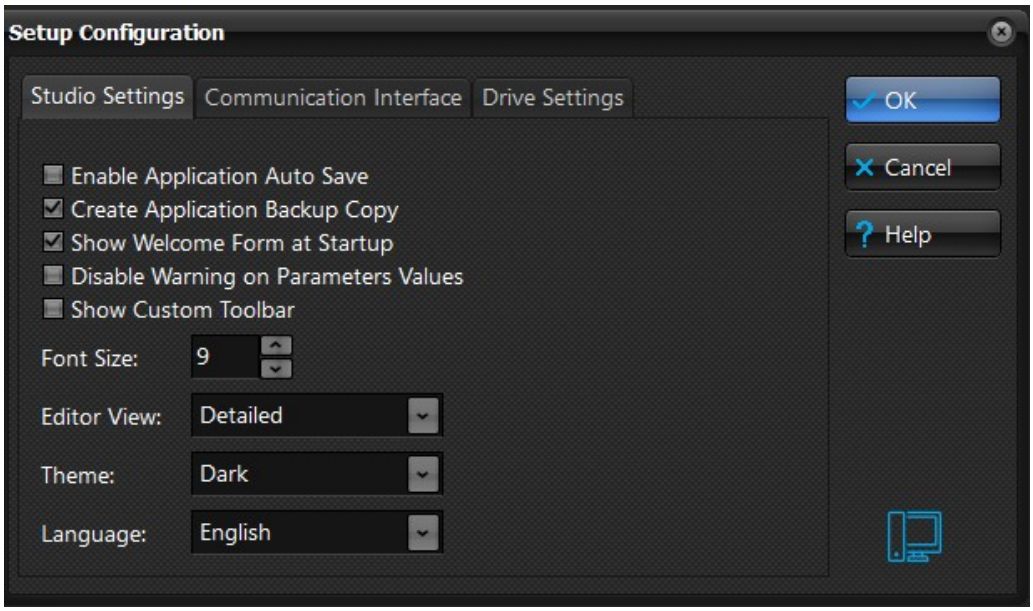
The **Watch Window** is located just below the Editor Window. The dimensions of the Editor and Watch Window can be adjusted by means of the splitter located between the windows. When you roll the mouse arrow over the splitter the mouse pointer changes to this shape:

When pressing the right mouse button inside the Editor Window a context menu shows up with the possible actions that can be executed:



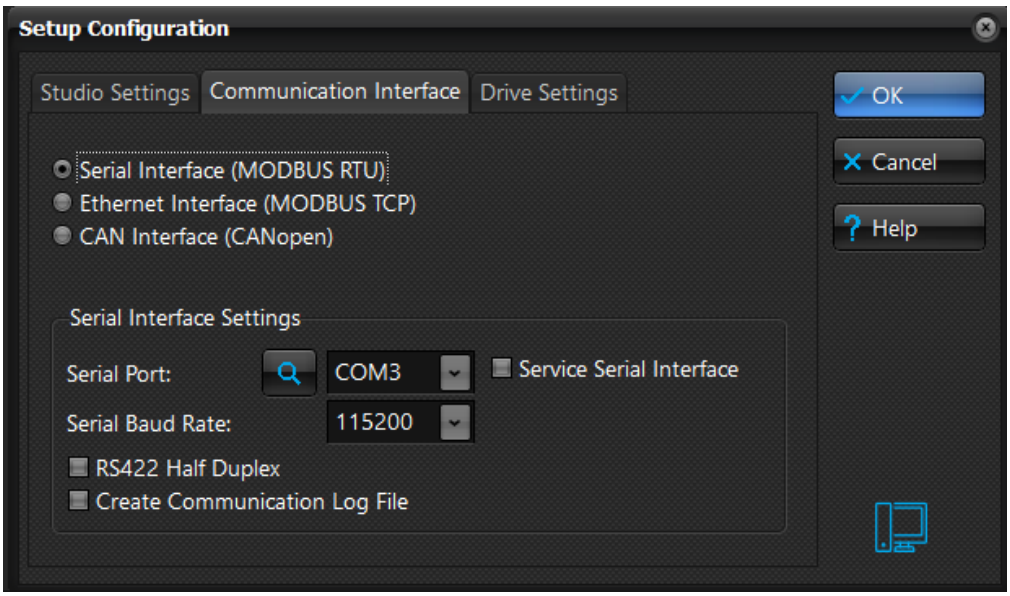
1.6 e3PLC Studio Configuration window

In the e3PLC Studio Configuration window is it possible to change some settings of the application editor as well as the settings of the communication interface necessary for the right connection to the drive.

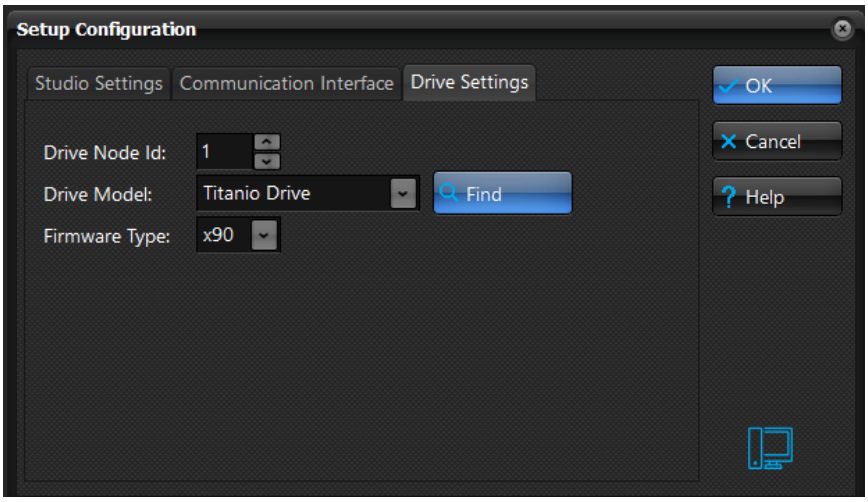


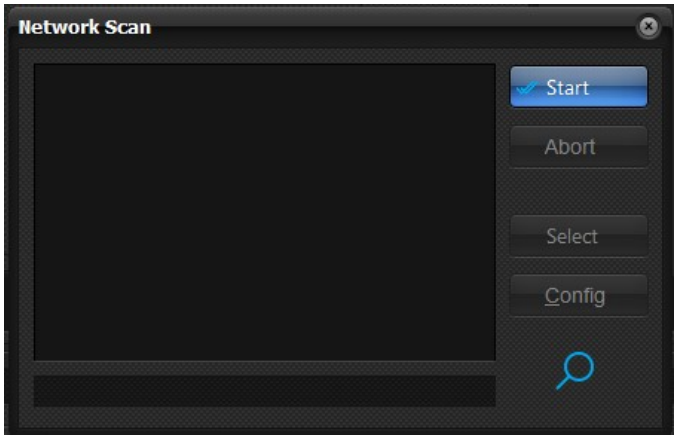
It is very important to set in the right manner:

- Communication Interface (RS232 or CAN)
- The right Serial Port (RS232) or CAN Board (CAN)
- The right Baud Rate
- The right Drive Node Id

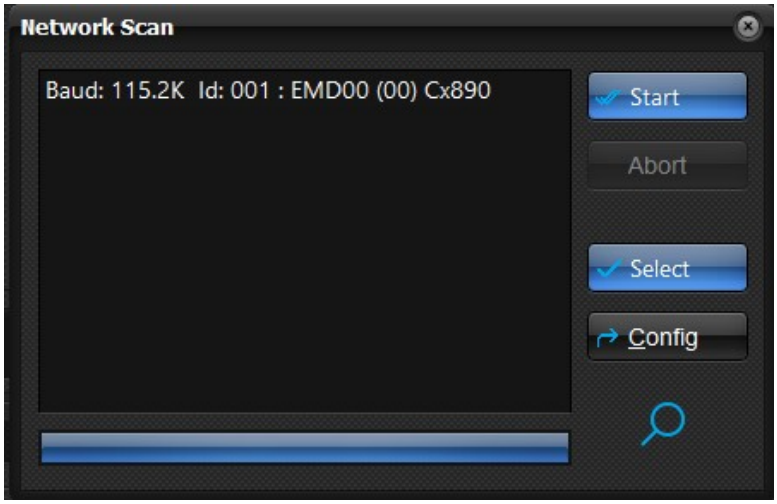


If the Baud Rate, Drive NodeId and Drive Model are not known it is possible to start a network scan by pressing on the 'Find' button:





When pressing on the 'Start' button the network scan procedure will be started. As soon as an EVER drive is detected it will be shown in the list. It is possible to abort the scan any time by pressing the 'Abort' button. In order to select the right drive it is necessary either double clicking on the item in the list, or selecting it and then pressing the 'Select' button. Then the Network Scan window will close and the new Baud Rate, Drive NodeId and Drive Model will be set to the e3PLC Studio Configuration Window.



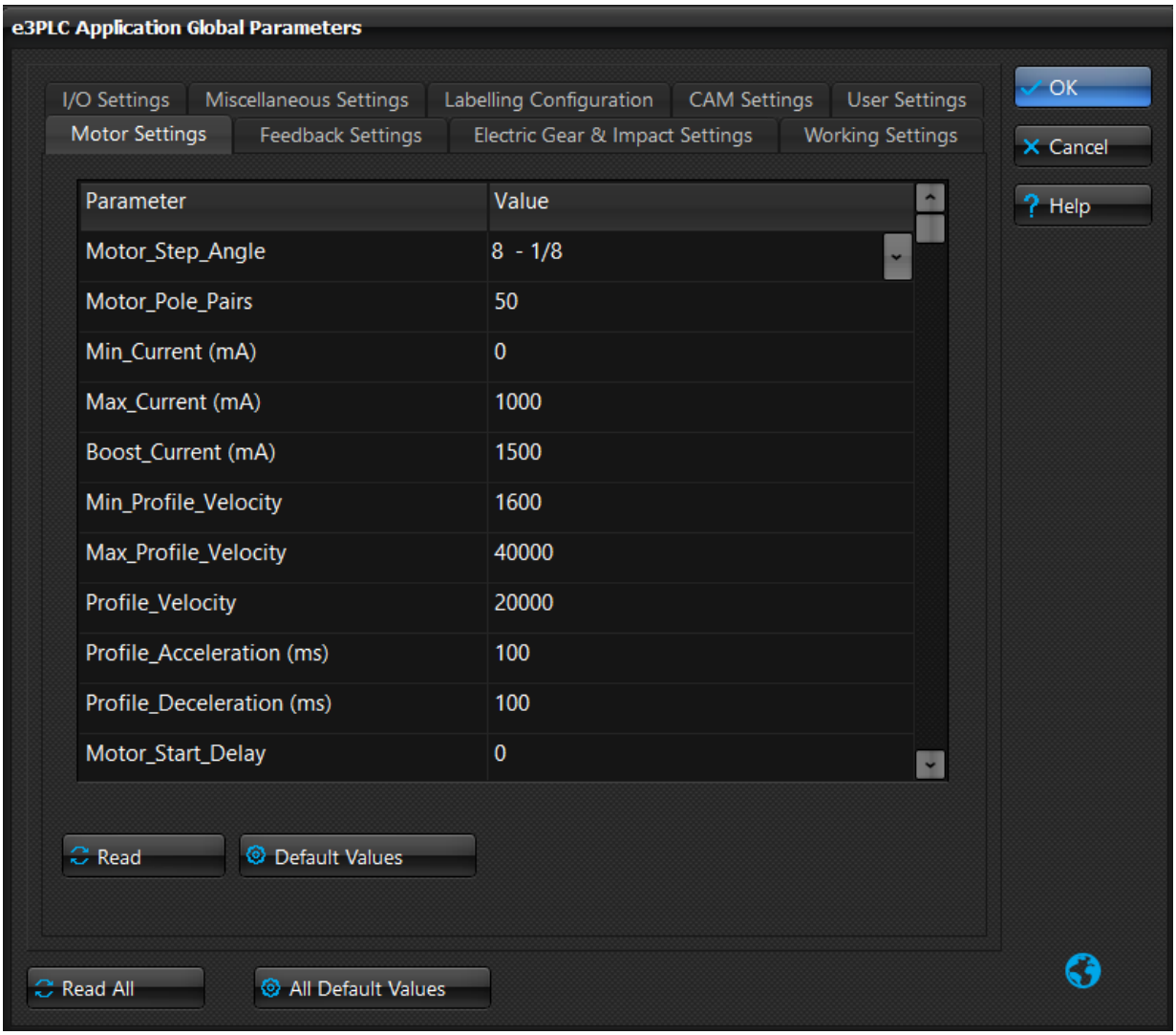
1.7 e3PLC Studio Application Global Parameters window

In the e3PLC Application Global Parameters window it is possible to set the drive's main parameters.

!!! It is the first task to perform before starting to build the application !!!

It is very important to set the right currents according to the used motor.

The parameters are grouped by categories. The explanation about the meaning and possible values of the parameters can be found in § *7.0*.

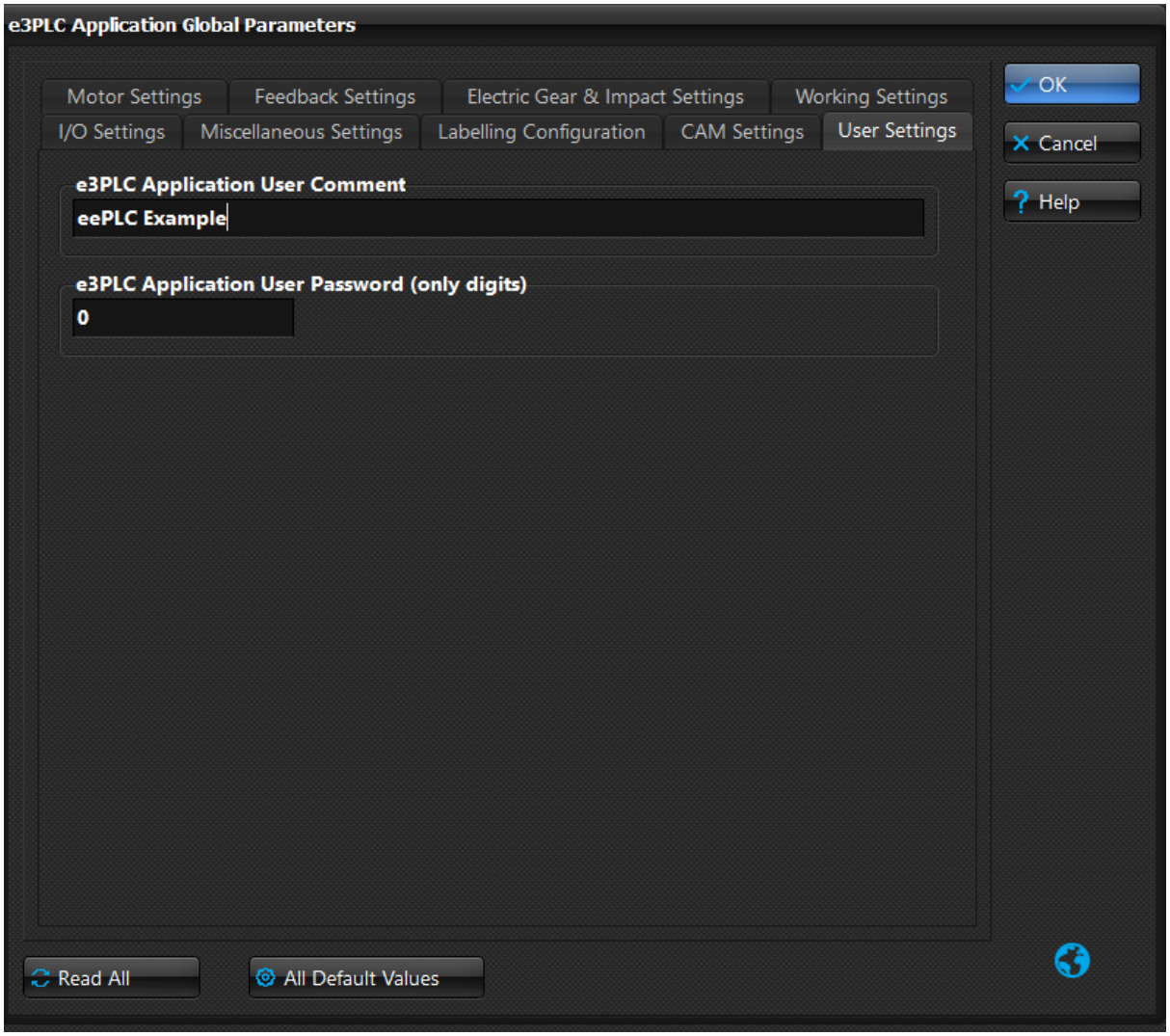


The parameter values will be stored together with the application instruction in the e3PLC application file (.plc).
It is possible to read or write all the parameters (Read All / Write All buttons) from the drive or only the parameters of a single category (Read / Write) to the concerning drive.

When pressing on the 'Default Values' button, the default parameter values will be set in the e3PLC Application Global Parameters window (they are not automatically sent to the drive!!. To send them to the drive it is necessary to press Write/Write All). It is a starting point for who is not very familiar with the parameters of the Titanio family drives.

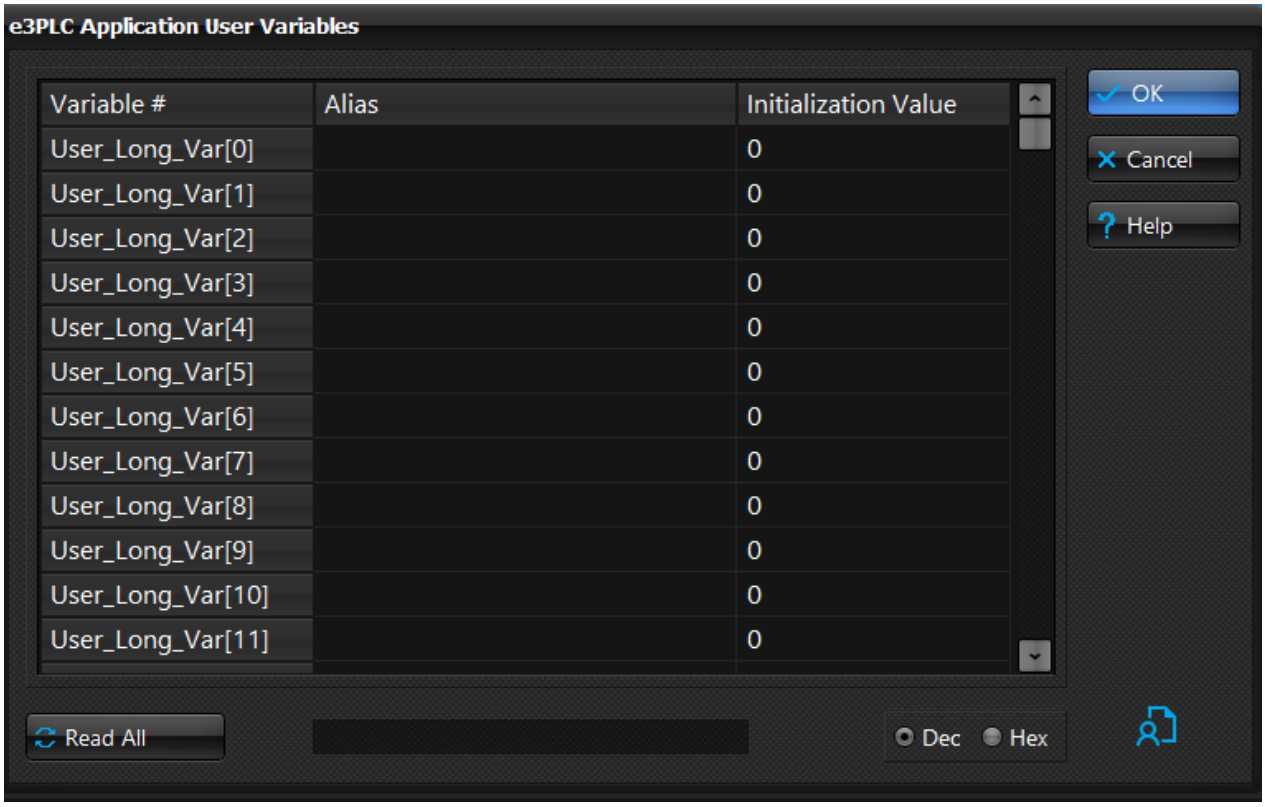
The 'Store in NVRAM' button will write the drive's '*Store_Parameters*' object that saves the current parameters value to a non volatile RAM so, at the next drive switch on the saved values will be restored.

Under the 'User Settings' tab it is possible to set either an application comment (useful to recognize the application running in the drive' or the user password (to prevent unauthorized uploading of the application stored on the drive).



1.8 e3PLC Studio Application User Variables window

In the e3PLC Studio Application User Variables window it is possible to set the user variables aliases (symbolic name) and Initialization values.



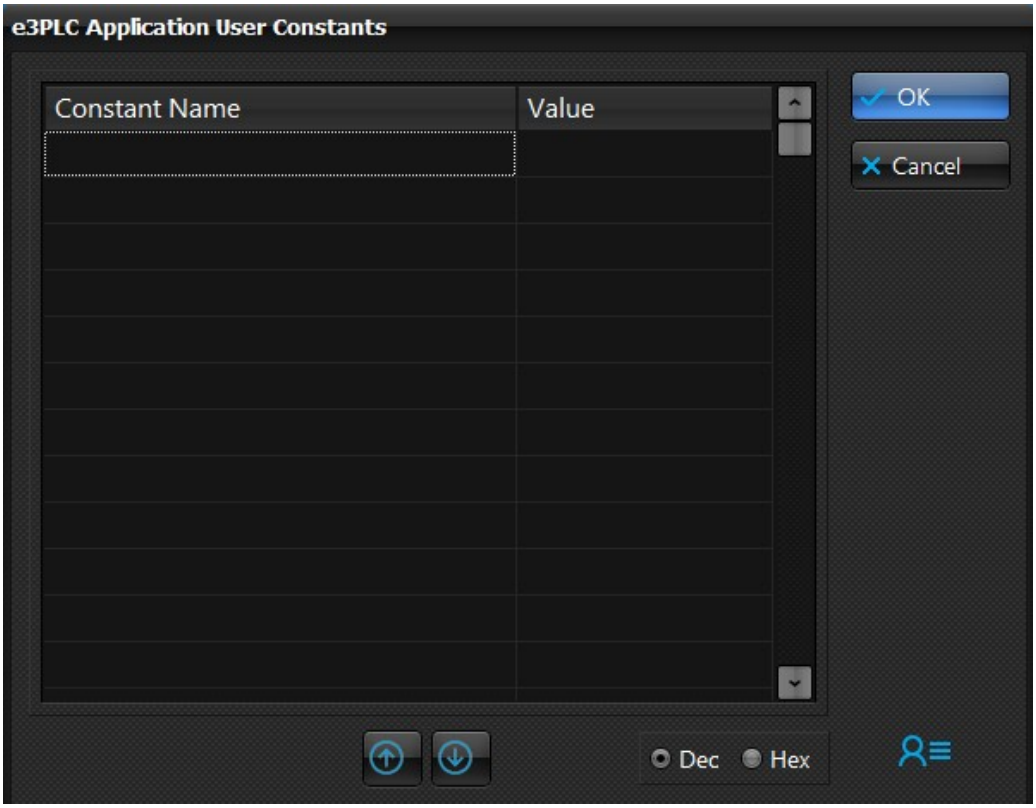
Unlike the Global Parameters, the user variables do not have a specific function. The function will be determined by the user in the application.
To use a variable inside the e3PLC user application an alias is required.
It is possible to read the current values from the drive by pressing the 'Read All' button.
It is possible to write the current edited values to the drive RAM by pressing the 'Write All' button.
It is possible to save the current drive RAM content to the NVRAM by pressing the 'Store in NVRAM' button.
The edited user variables values and aliases will be stored together with the application instruction in the e3PLC application file (.plc)
The Initialization Value is the value that could be sent to the drive during the Application Download (see §*1.15*)

Pressing the right mouse key inside the variables definition grid a menu with an option will pop up giving the possibility to exchange two user variables among them.

Only the 32 bit signed integer (*User_Long_Vars*) drive variables are fully supported by the e3PLC Studio so far.

1.9 e3PLC Studio Application User Constant window

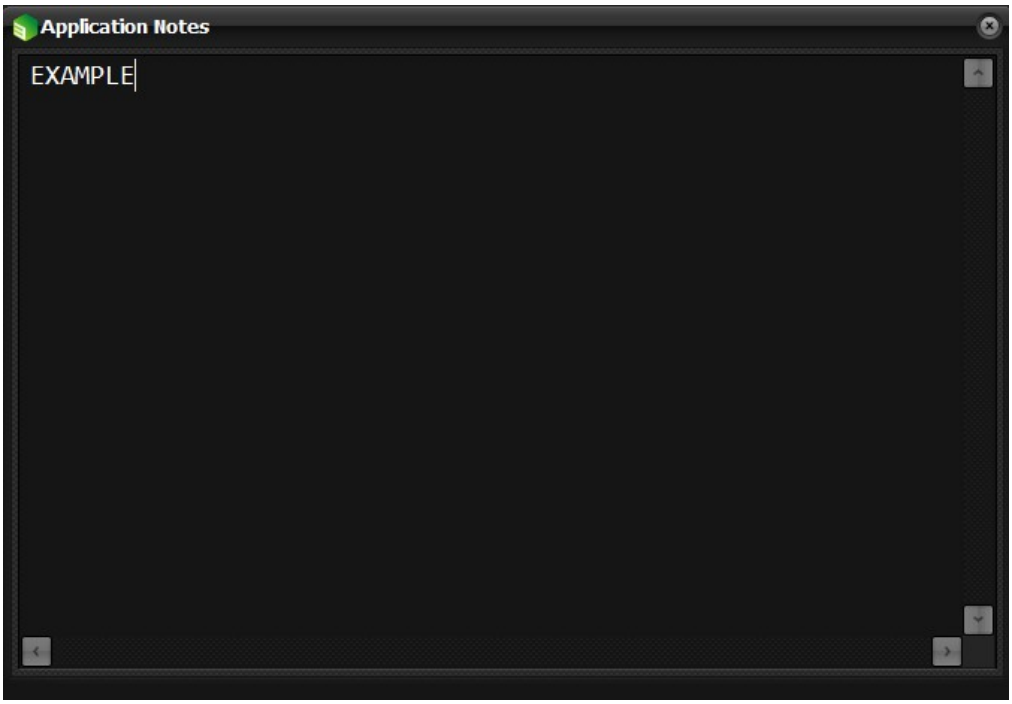
In the e3PLC Studio Application User Constant window it is possible to set the user constant name (symbolic name) and values.



To use a constant inside the e3PLC user application a name is required.
It's not allowed to modify constant values inside the e3PLC application.

1.10 e3PLC Studio Application Notes window

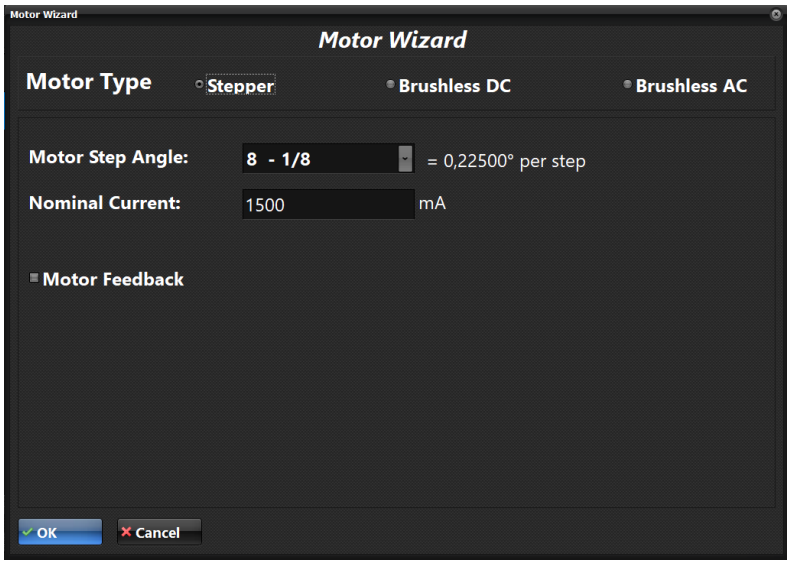
In the e3PLC Studio Application Notes window it is possible to write Notes about the application.

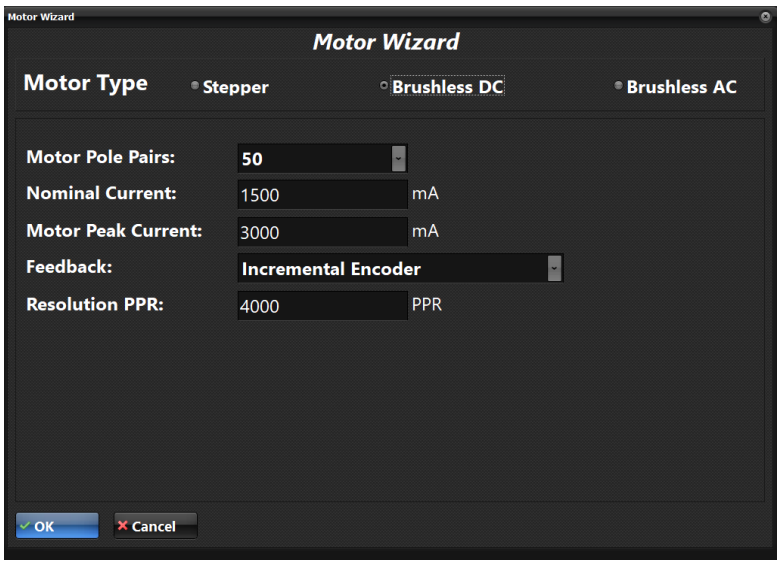


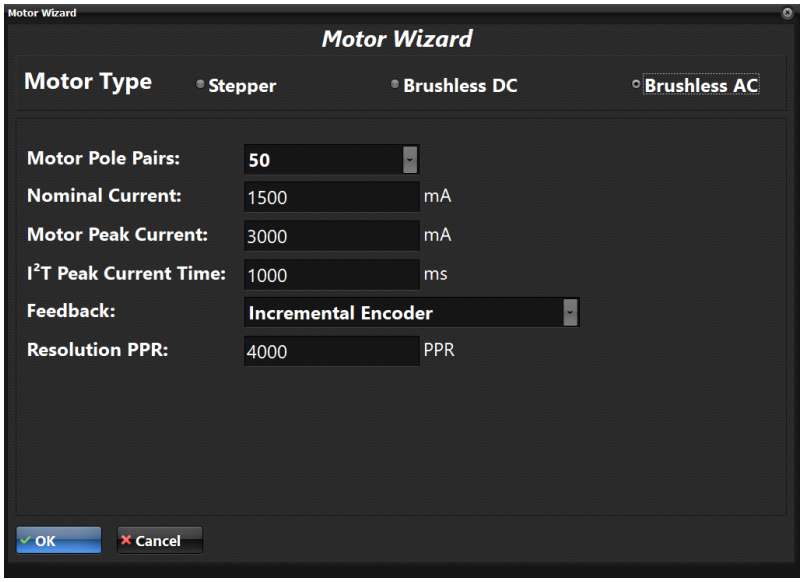
The notes will be automatically saved at Application Notes Window closing.

1.11 e3PLC Studio Motor Wizard window

In the e3PLC Studio Motor Wizard window it is possible to set at first the motor type (STEPPER, Brushless DC, Brushless AC), and then it is possible to set some of the main motor parameters.





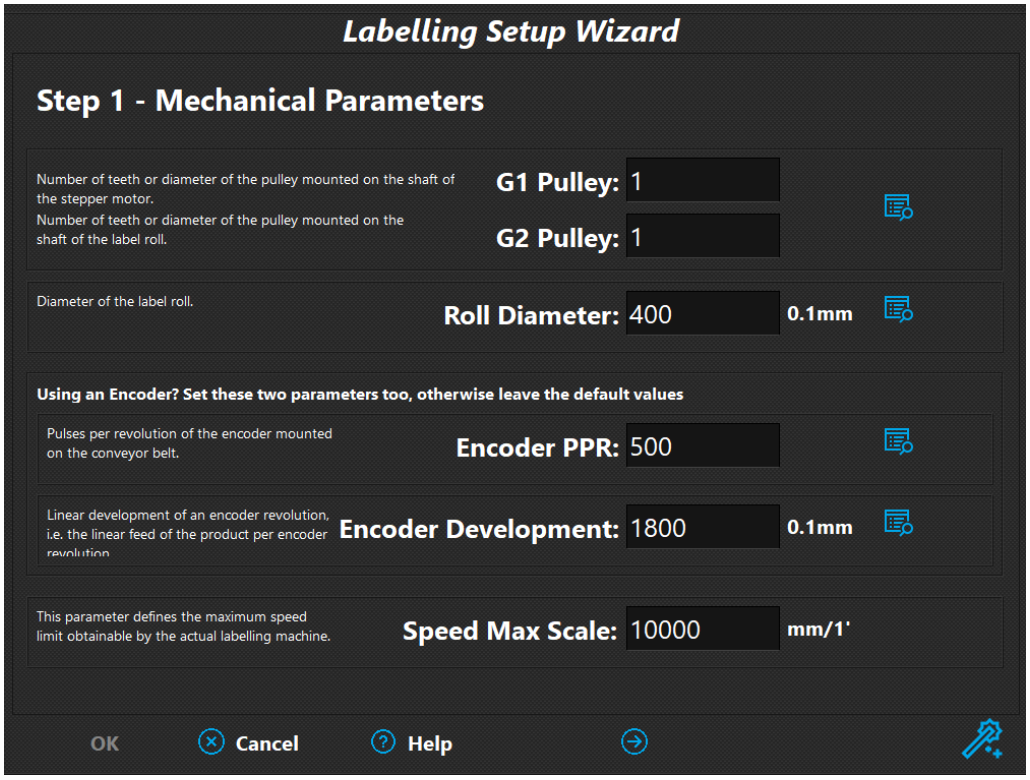


For the description of all parameters, please consult the e3PLC Object Dictionary (§ *7.0*).

Please Note: setting the *Nominal_Current*, this current is automatically assigned also to *Max_Current* , to *Boost_Current* and to *Feedback_Boost_Current*.
The *Min_Current* is automatically set as 30% of Nominal_Current.

1.12 e3PLC Studio LAB Wizard window

In the e3PLC Studio LAB Wizard window it is possible to set all Labelling parameters.

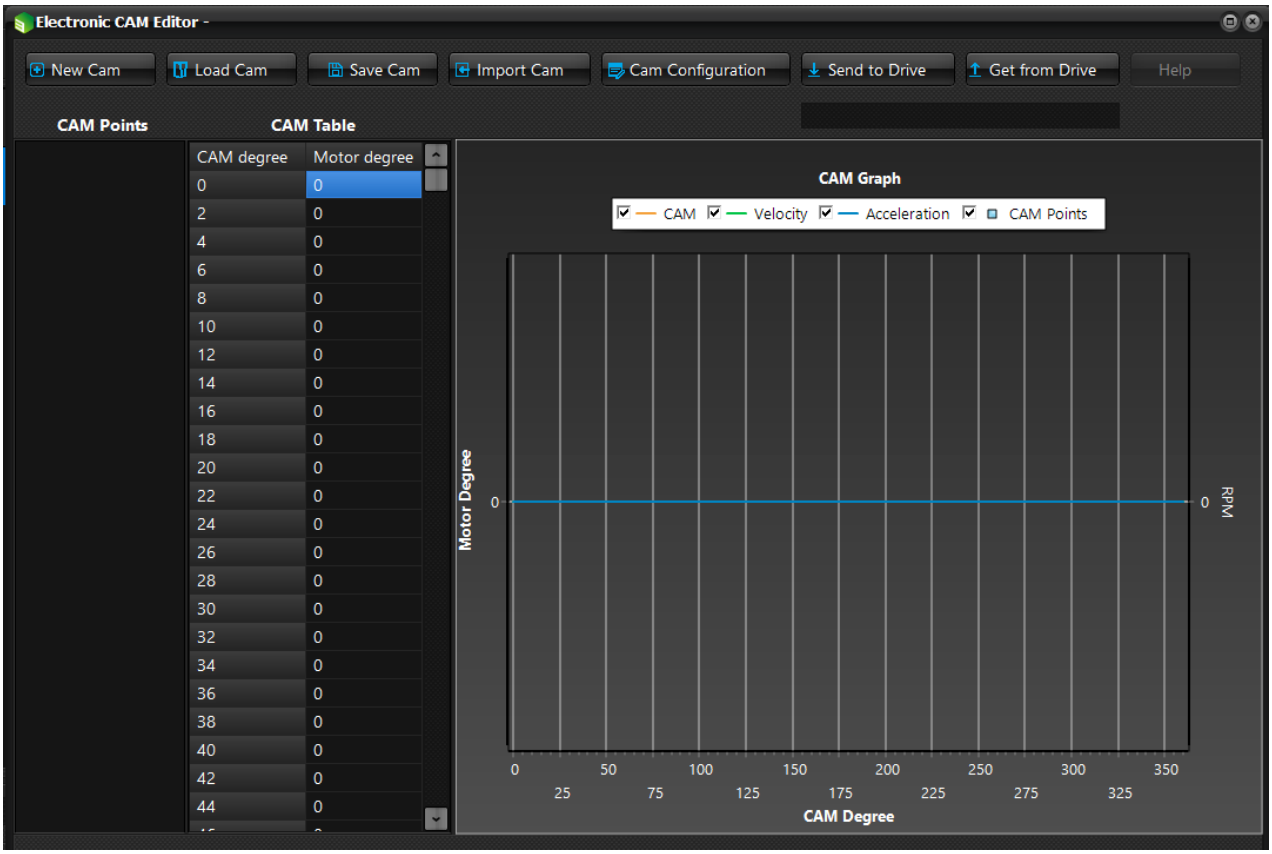


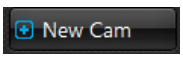
For the description of all parameters, please consult the Manual_SW1_Labelling_Realtime_Module_EN.

Please note: using  it is possible to change page and continue to set Labelling Parameters.

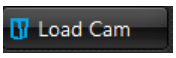
1.13 e3PLC Studio CAM Editor window

Through the CAM Editor window integrated into the e3PLC environment you can parameterize the CAM module and define the desired CAM profile fast and easy.



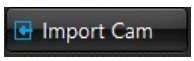


Create a New Electronic CAM

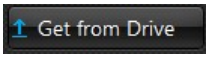


Load a CAM from PC

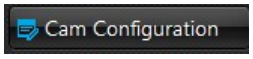
Save a CAM in PC



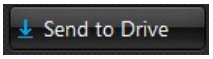
Import ECAM-4 format file



Get the CAM from the connected Drive



Open CAM configuration window

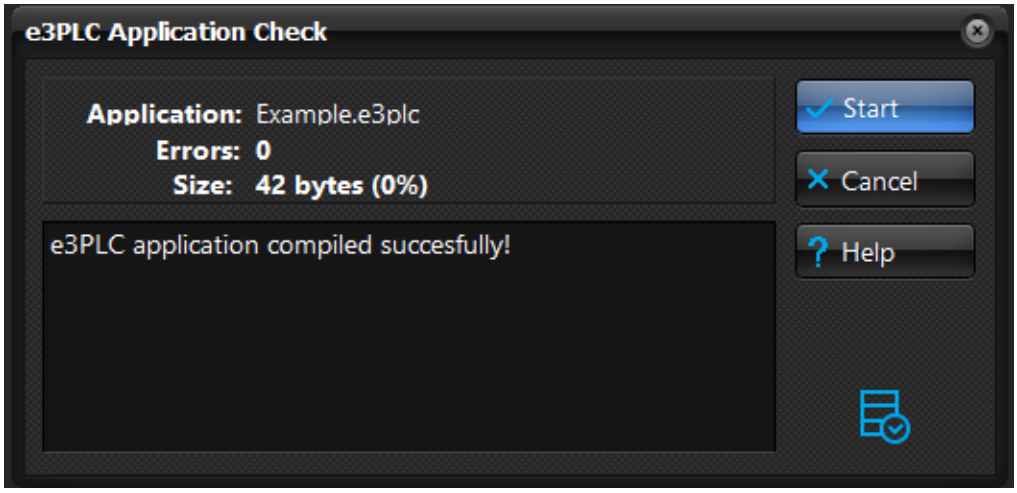


Send CAM to the connected Drive

More information about the Electronic_CAM_Module can be found in the *'CAM Realtime Module Manual for eePLC Studio for Titanio Drives'*.

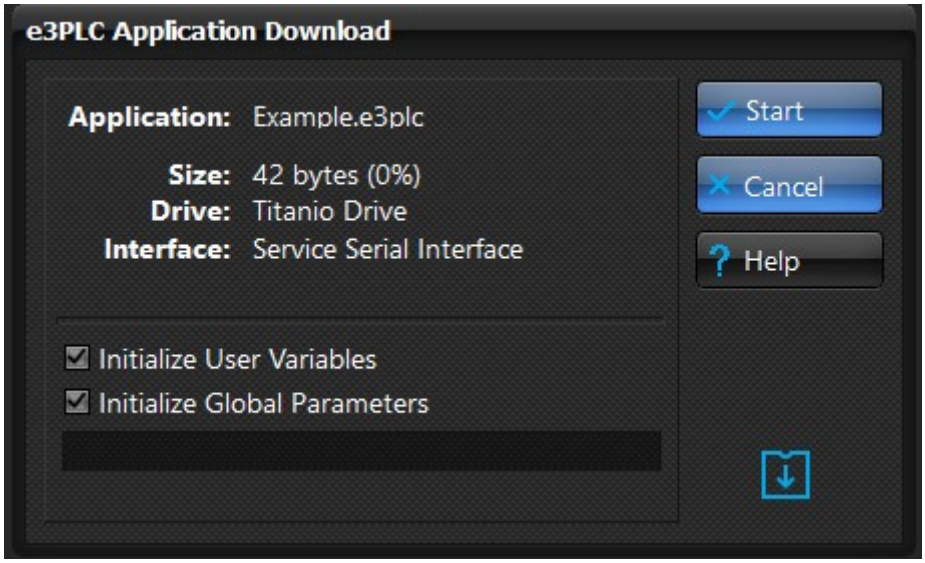
1.14 e3PLC Studio Application Check window

The e3PLC Studio Application Check window permits to check the user application on correctness and size.
When pressing the 'Start' button the check procedure will be performed.
The possible errors are shown in the list. When double clicking on an error the application line where the error was found will be selected.



1.15 e3PLC Studio Application Download window

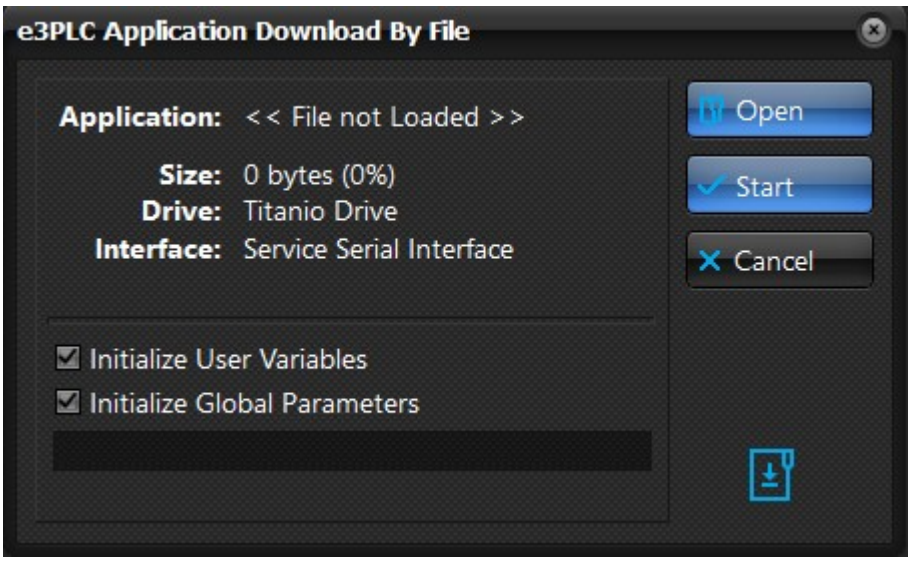
The e3PLC Studio Application Download window permits to download the user application to the RAM and NVRAM drive.
Before opening the Download window a check of the user application is performed (see §*1.14*). If no error is found, the Download window is showed up, otherwise a message box suggests to execute an application check.



If the 'Initialize User Variables' checkbox is checked the User Variables initialization values set in the IDE (see §*1.8*) will be sent together with the user application, otherwise the current user variables values stored in the drive are retained.
If the 'Initialize Global Parameters' checkbox is checked the Global Parameters values set in the IDE (see §*1.7*) will be sent together with the user application, otherwise the current parameters values stored in the drive are retained.
When pressing the 'Start' button the download procedure will be performed.

1.16 e3PLC Studio Application Download by File window

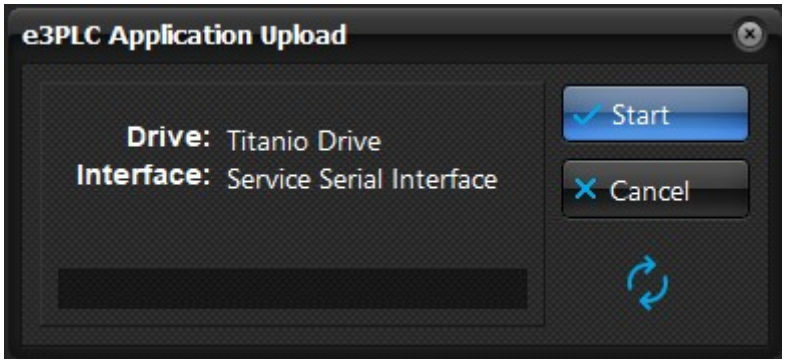
The e3PLC Studio Application Download by File window permits to select and open a file '.plcobj' and to download it to the RAM and NVRAM drive.



If the 'Initialize User Variables' checkbox is checked the User Variables initialization values set in the IDE (see §*1.8*) will be sent together with the user application, otherwise the current user variables values stored in the drive are retained.
If the 'Initialize Global Parameters' checkbox is checked the Global Parameters values set in the IDE (see §*1.7*) will be sent together with the user application, otherwise the current parameters values stored in the drive are retained.
When pressing the 'Start' button the download procedure will be performed.

1.17 e3PLC Studio Application Upload window

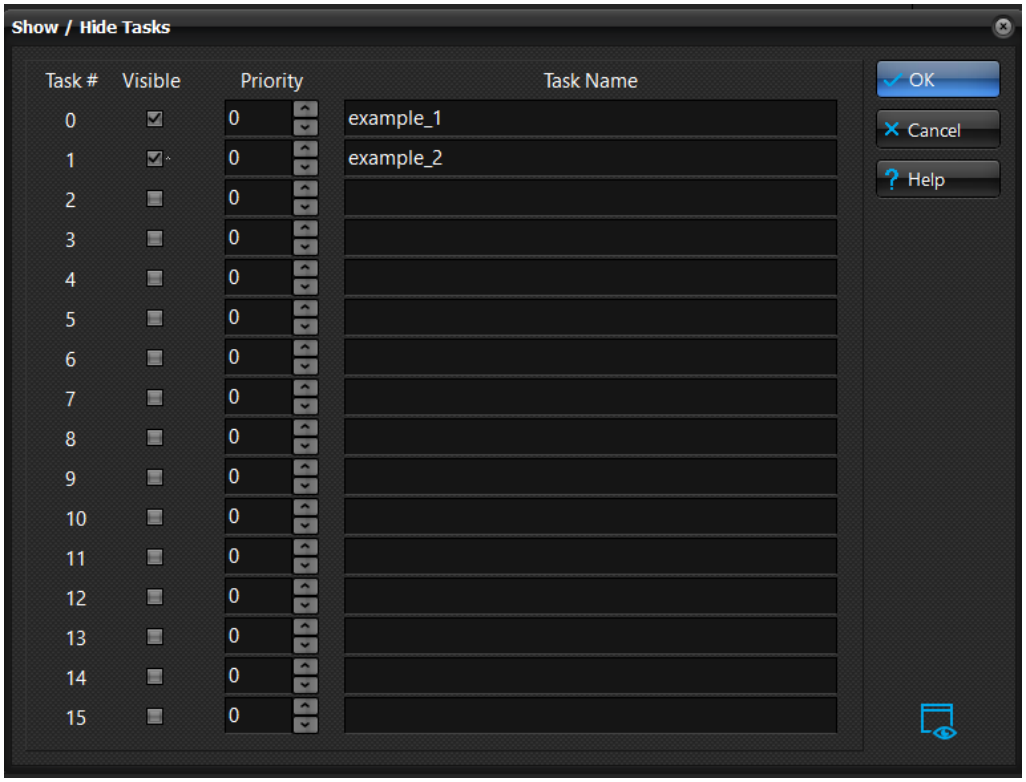
The e3PLC Application Upload window permits to upload the user application stored on the drive.



If the stored program is password protected (see §*1.7*) it will be requested to insert the correct password, otherwise the upload process will be aborted.

1.18 e3PLC Studio Show/Hide Tasks window window

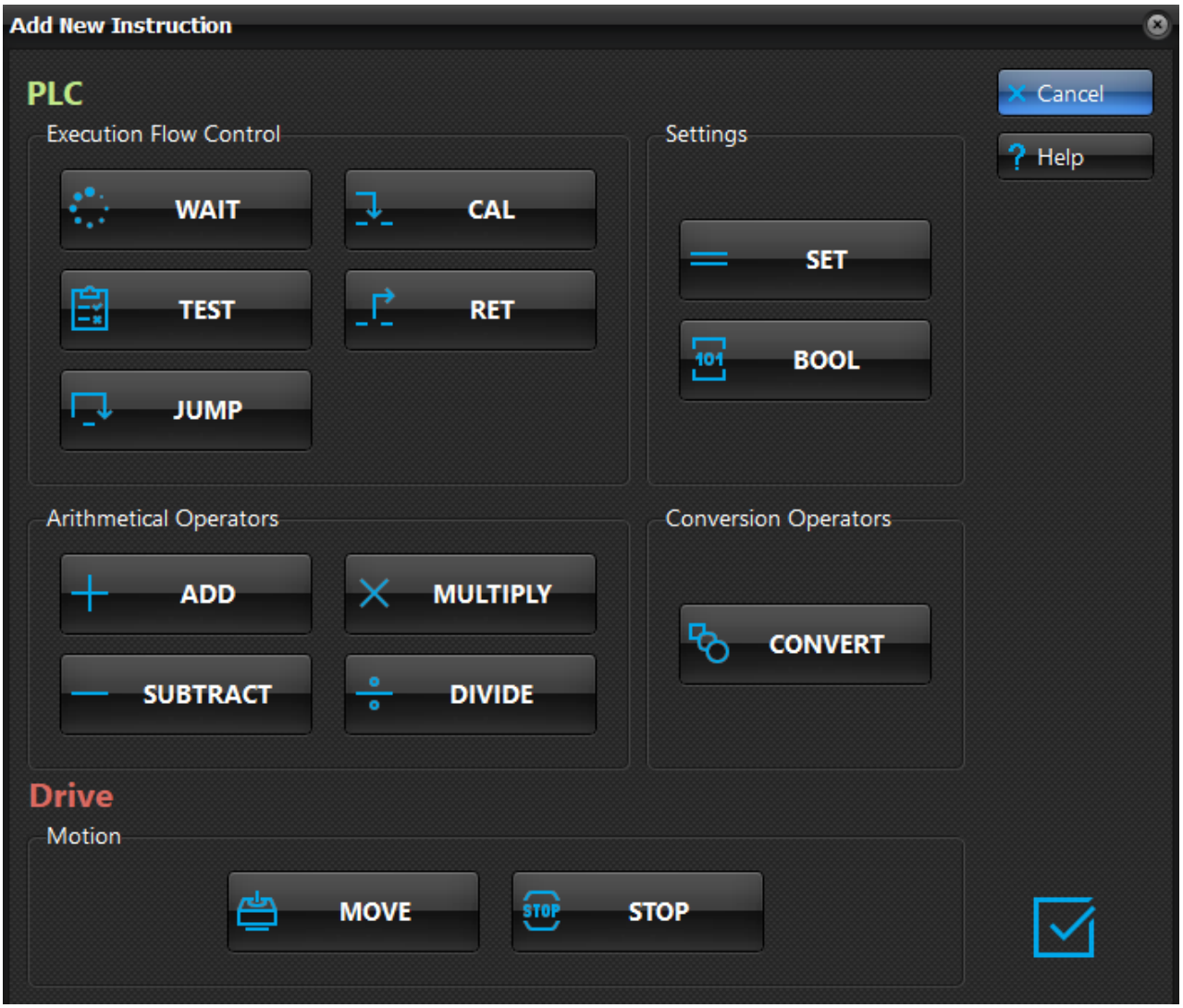
The e3PLC Application Upload window permits to make V**isible** each task, change it's P**riority** and define a T**ask Name**.



2.0 Building the e3PLC Application

This chapter describes how to build an e3PLC application. The whole application building is performed by means of windows, lists and buttons. The user only needs to type when inserting constant numeric values.

To add a new instruction either press the 'Add Instruction' button (see §*1.5*) or double click on an empty line in the current task. Then a window with the available instructions will appear.



Clicking on the desired instruction, the corresponding window will appear.

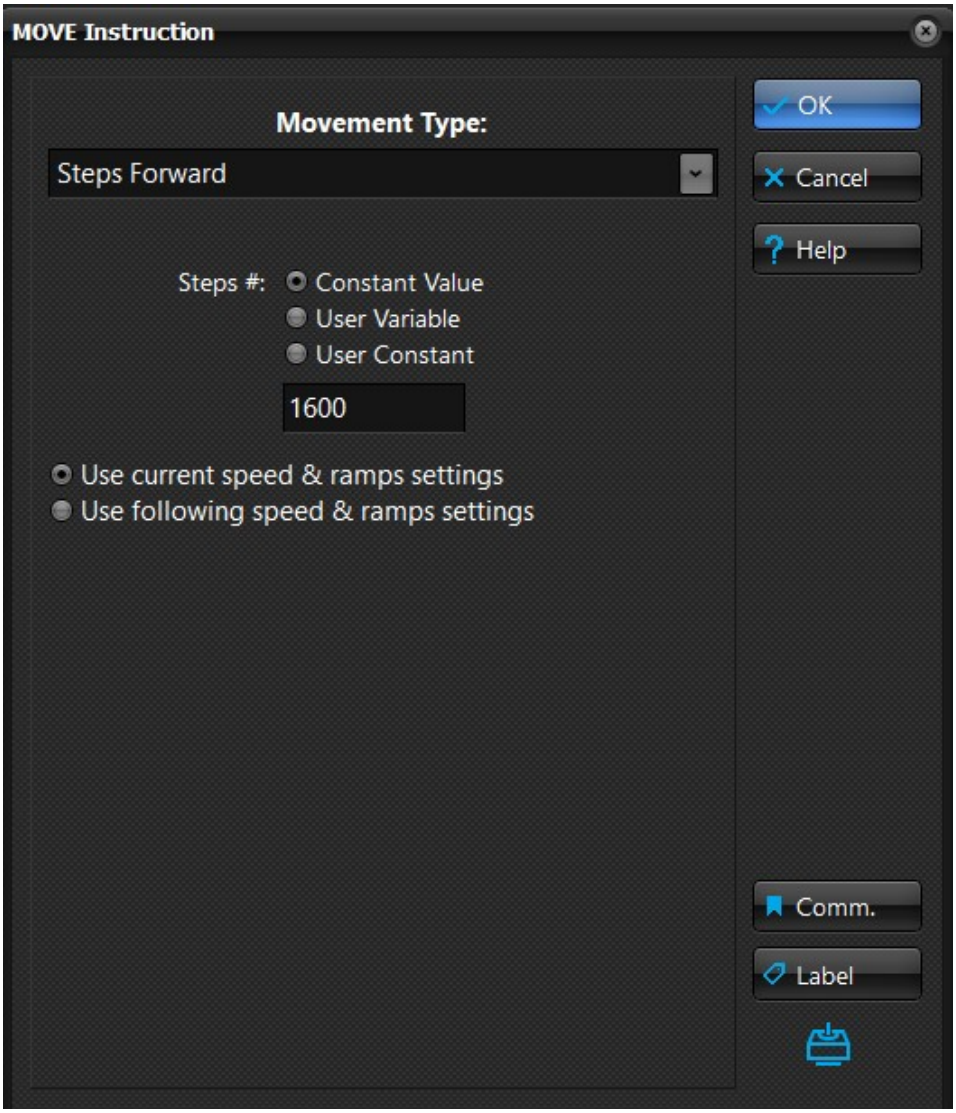
The explanation of the various instruction windows follows.

2.1 Motion Instructions

The motion instructions are used to control the movement and the stop of the motor.

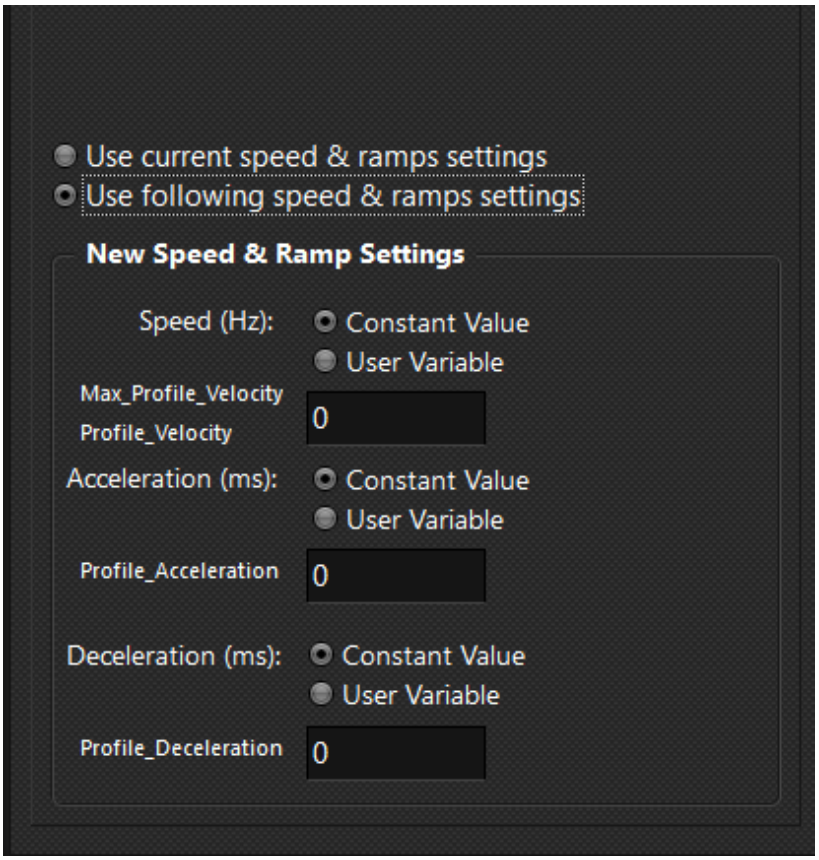
2.1.1 MOVE Instruction

The MOVE instruction starts a motor movement. For the explanation of the various movement types refers to §*8.3*.



When the *'Use current speed & ramps settings'* radio button is checked the motion profile is given by the current values of the motion parameters.

When the 'Use following speed & ramps settings' radio button is checked the following frame opens:



Then, just before the movement is started the following global parameters are changed:

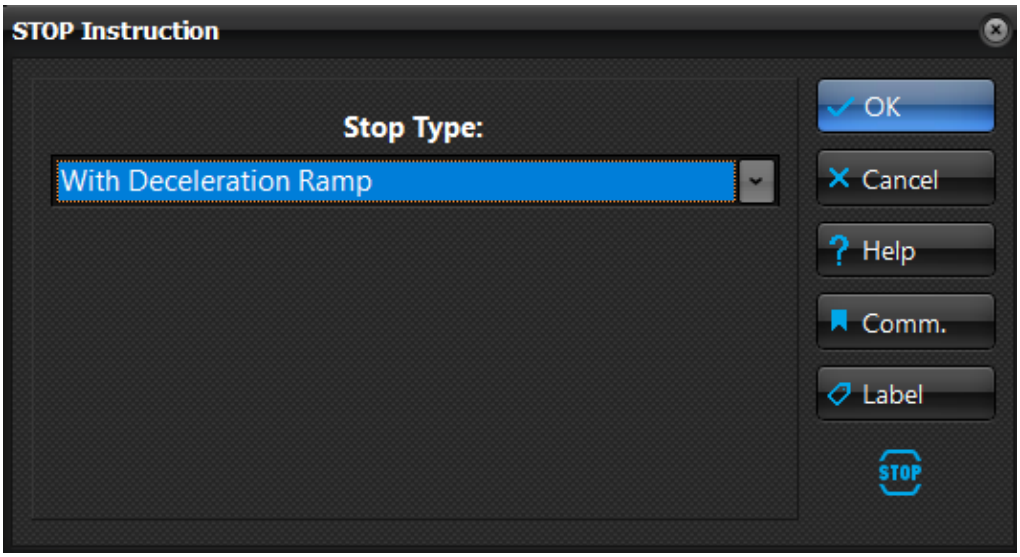
- The *'Max_Profile_Velocity'* and *'Profile_Velocity'* objects are set equal to the Speed parameter value.
- The *'Profile_Acceleration'* object is set equal to the Acceleration parameter value.
- The *'Profile_Deceleration'* object is set equal to the Deceleration parameter value.

The MOVE instruction doesn't wait for the end of the movement before passing the execution to the next application instruction. To wait until the motor movement is completed it is necessary to insert a WAIT for motor into the standstill instruction just after the MOVE instruction.

2.1.2 STOP Instruction

The STOP instruction stops the current motor movement. There are three types of motor stop that can be chosen:

- **Stop with deceleration ramp**. In this case the motor will stop using the deceleration ramp specified with the *Profile_Deceleration* object. This is the preferred method to stop a running motor.
- **Stop without deceleration ramp**. In this case the motor will stop immediately. This method should be used only when the motor is running at very low speed, since it could lose steps due the load inertia.
- **Stop with # of steps**. In this case the motor will stop within the specified steps number. If the specified steps are fewer than the current deceleration ramp steps, the deceleration ramp will be steeper.



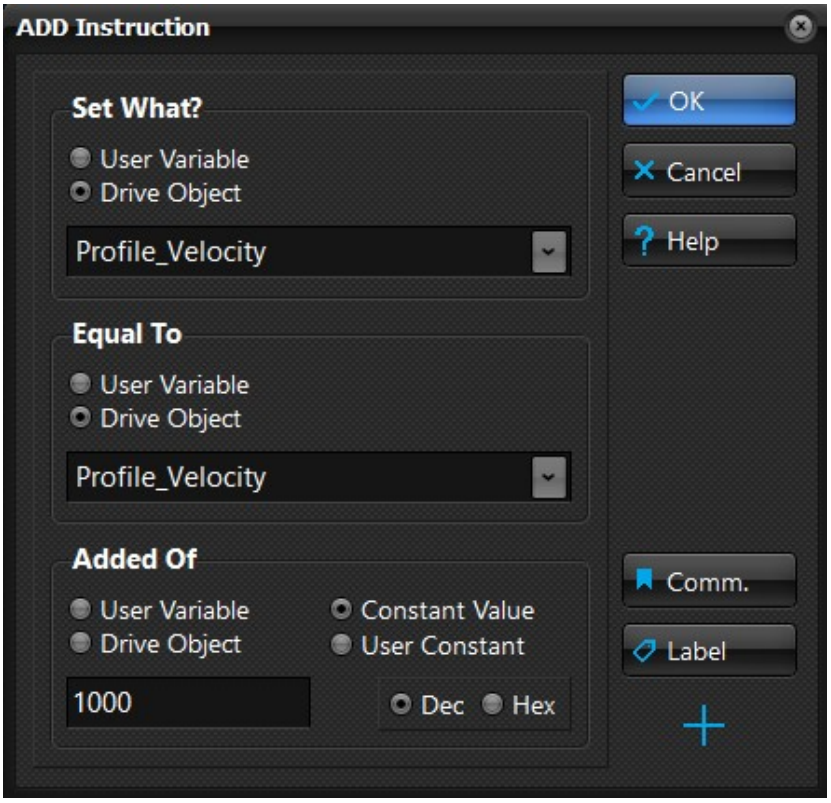
The stop types with Trigger Input or with Sync Object act in the same way as the movement types (see §*8.3.3* and §*8.3.4*). The STOP instruction doesn't wait for the end of the movement before passing the execution to the next application instruction. To wait until the motor movement is completed it is necessary to insert a WAIT for motor into the standstill instruction just after the MOVE instruction.

2.2 Arithmetical Operators

The Arithmetical Operators perform the four arithmetical operations (addition, subtraction, multiplication, division) on Global Parameters or User Variables.

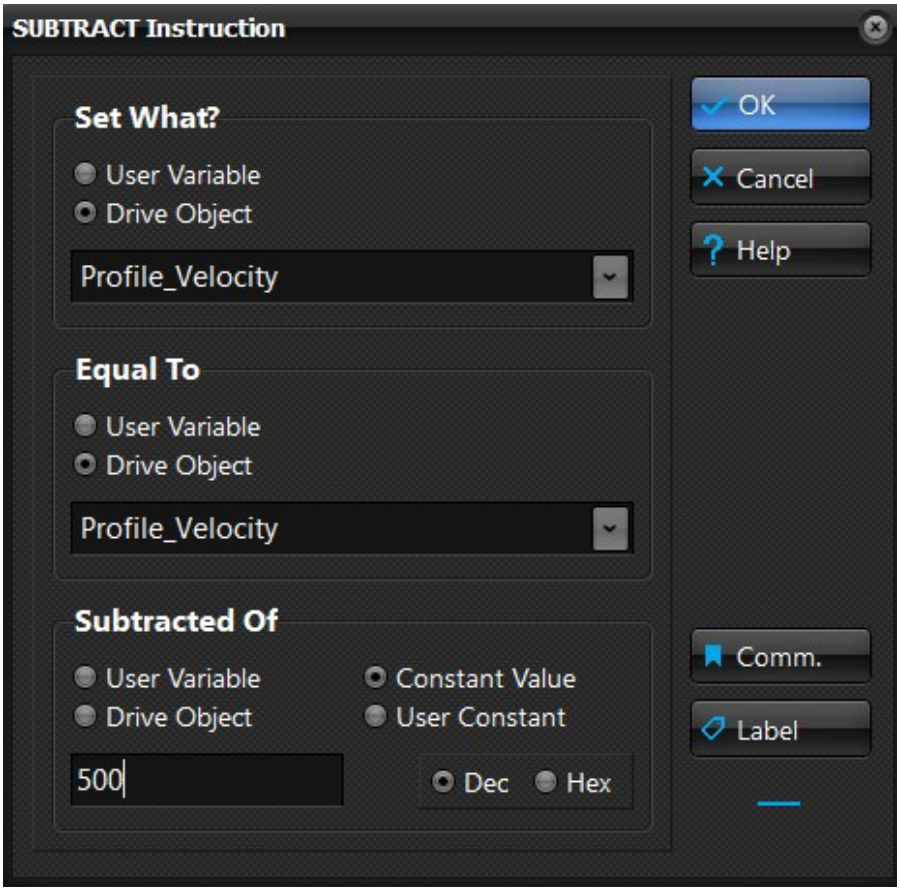
2.2.1 ADD Instruction

The ADD Instruction performs an integer addition between two arguments (the second one could be also a numerical constant) storing the result in a destination parameter.



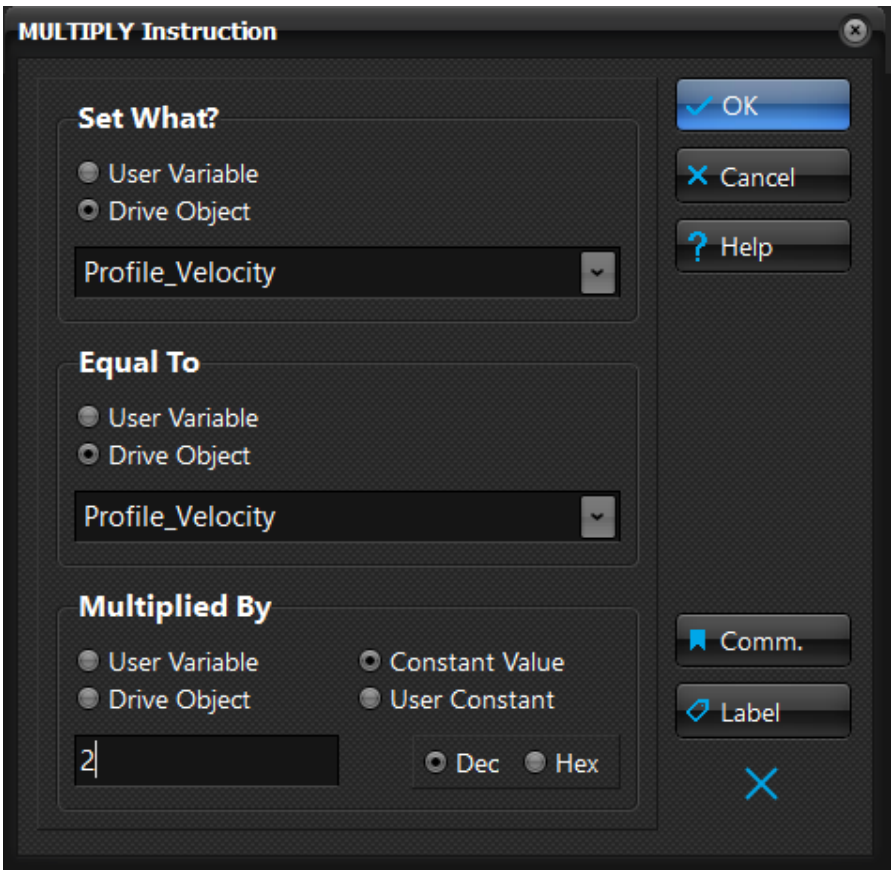
2.2.2 SUBTRACT Instruction

The SUBTRACT Instruction performs an integer subtraction between two arguments (the second one could be also a numerical constant) storing the result in a destination parameter.



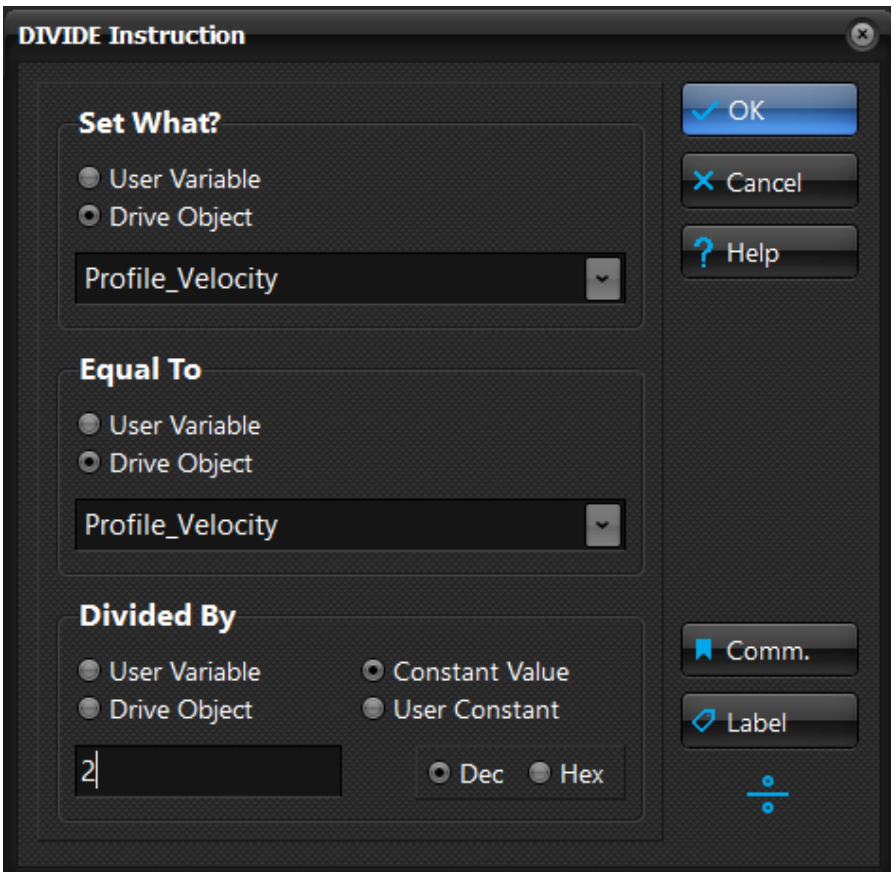
2.2.3 MULTIPLY Instruction

The MULTIPLY Instruction performs an integer multiplication between two arguments (the second one could also be a numerical constant) storing the result in a destination parameter.



2.2.4 DIVIDE Instruction

The DIVIDE Instruction performs an integer division between two arguments (the second one could be also a numerical constant) storing the result in a destination parameter.



If the second operand is a constant the e3PLC Studio performs a by zero division check.
If the second operand is a User Variable or a Drive Object the e3PLC drive's firmware will perform a by zero division check issuing an exception if the check is positive.

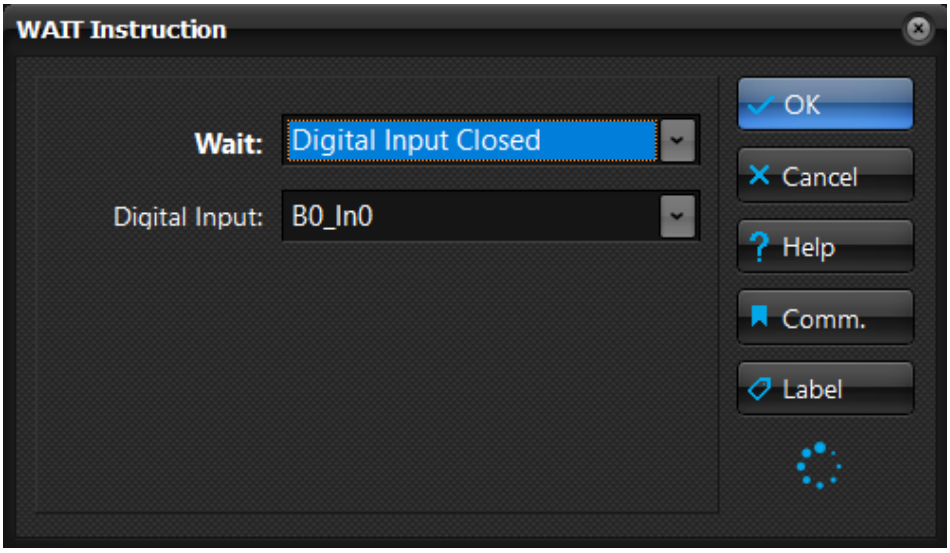
2.3 Application Flow Control Instructions

The Application Flow Control Instructions are used to change the execution flow of a task in a conditional or unconditional manner.

2.3.1 WAIT Instruction

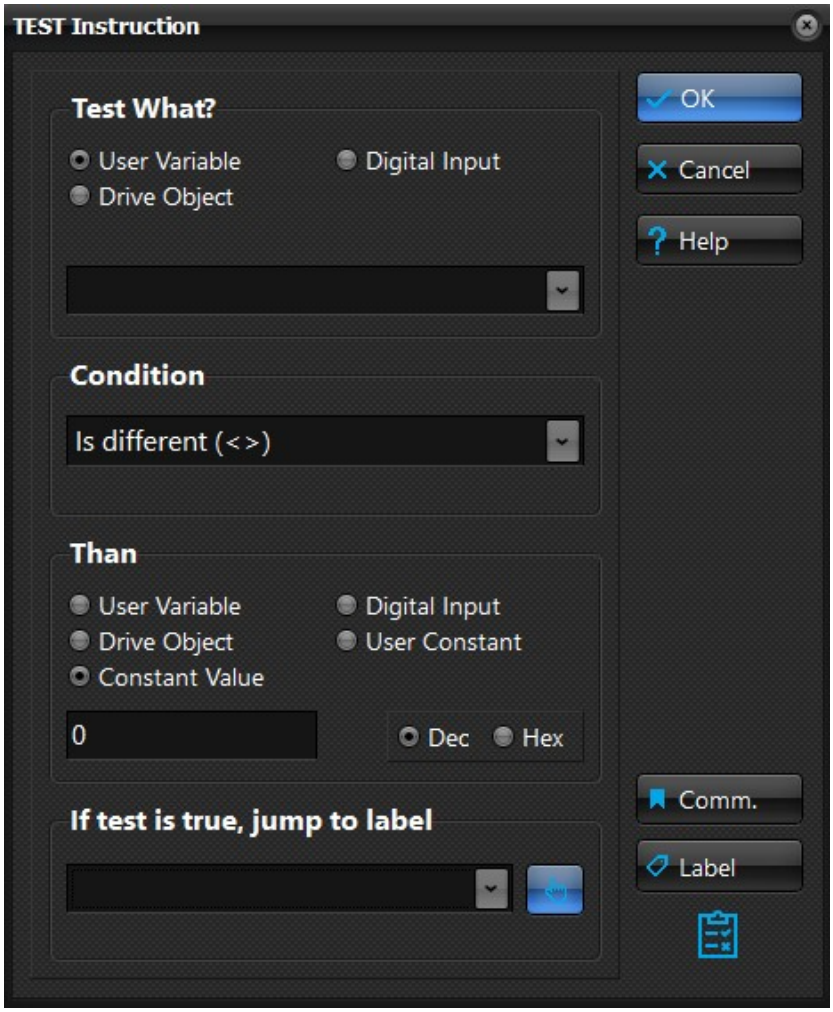
The WAIT instruction halts the current task, the execution won't take place until the wait for condition is satisfied. It is possible to wait until:

- Time delay elapses
- The motor is running
- The motor is at standstill
- A Drive Digital Input is open
- A Drive Digital Input is closed



2.3.2 TEST Instruction

The TEST instruction performs a check on the current value of a User Variable, a Drive Object or a Digital Input. If the TEST condition is met, the execution of the current task will continue from the specified label, otherwise the execution will continue with the next instruction just after the TEST instruction.



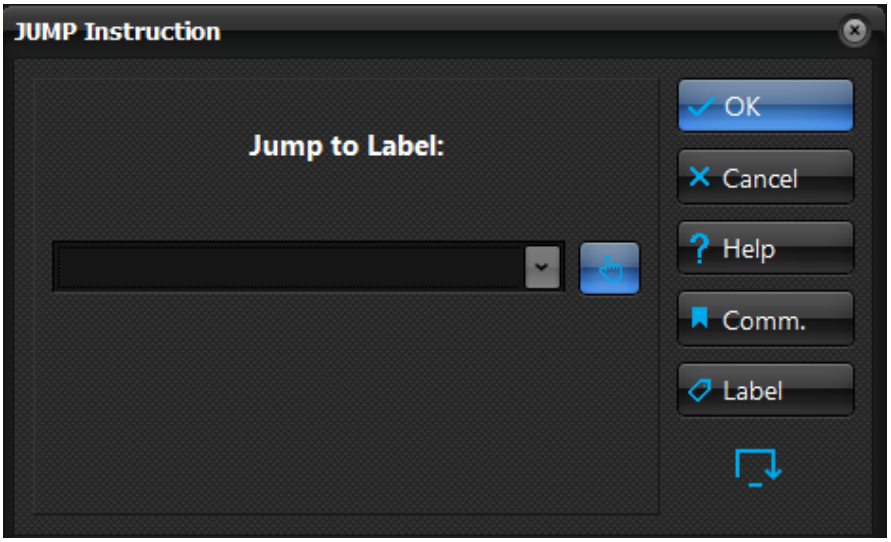
When testing a Digital Input it is necessary to keep in mind that the only two values that are allowed are 0 (input open) or 1 (input closed).

Since the label to jump could not be defined yet when inserting the TEST instruction in the application, it is possible to leave the list box blank and afterwards edit this instruction again in order to specify when the right label will be defined.

If the jump to label list box is left blank, when checking the application correctness (see § *1.14*) a 'label not found' error is issued.

2.3.3 JUMP Instruction

The JUMP instruction will change the execution flow of the current task only to the label specified.

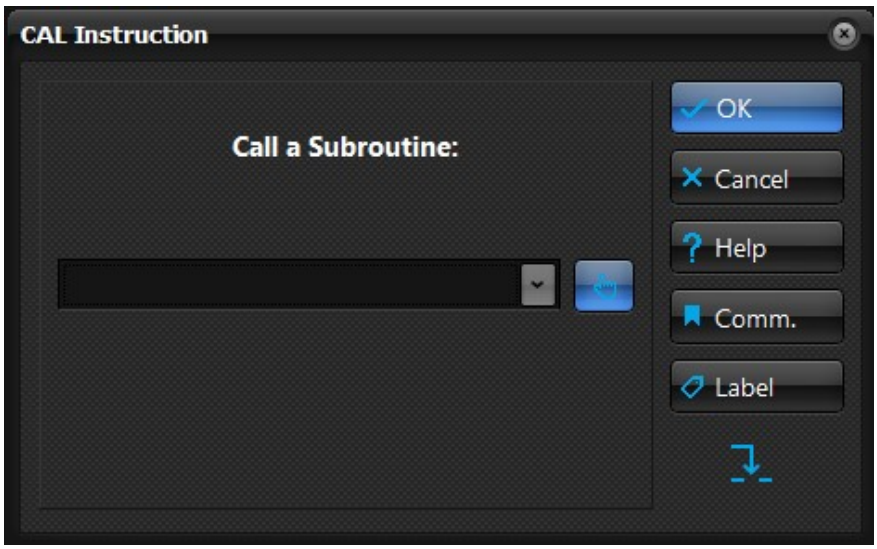


Since the label to jump could not be yet defined when inserting the JUMP instruction in the application, it is possible to leave blank the list box and afterwards edit again this instruction to specify the right label when will be defined.

If the 'jump to label' list box is left blank, a 'label not found' error is issued when checking the application correctness (see §*1.14*).

2.3.4 CAL Instruction

The CAL instruction will change the execution flow of the current task only to the subroutine specified. The current task flow will be restored when the RET instruction is executed inside the called subroutine.



Since the label of the subroutine could not be yet defined when inserting the CAL instruction in the application, it is possible to leave blank the list box and afterward edit again this instruction to specify the right label when will be defined.

If the 'CAL a Subroutine' list box is left blank, a 'label not found' error is issued when checking the application correctness (see §*1.14*).

The subroutine can be defined outside the current task. To define a subroutine just assign a Label Name to a valid instruction line (see §*1.6*).

2.3.5 RET Instruction

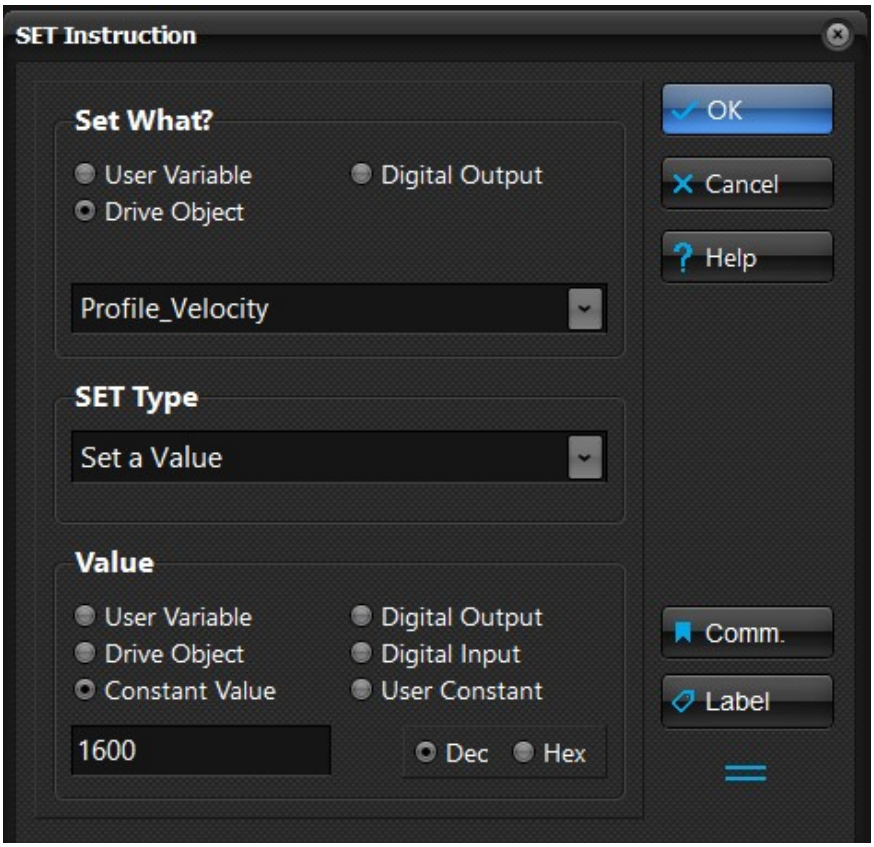
The RET instruction will restore the execution flow of the current task only to the next instruction after the previous executed CAL instruction. The RET instruction doesn't show any setting window since it doesn't need any argument. If the RET instruction is executed by the drive without any previous CAL instruction, the drive will go in emergency condition showing a 'Too many RET' error in the Alarm History.

2.4 Settings Instructions

The Settings Instructions are used to change the value of the drive's resource.

2.4.1 SET Instruction

The SET instruction sets a destination parameter (User Variable, Drive Object or Digital Output) equal to the source parameter. It is possible also to set or reset a single bit. The read only drive objects cannot be chosen as destination parameter.



When setting a Digital Output it is necessary to keep in mind that the values different than 0 are considered equal to 1 (output closed).

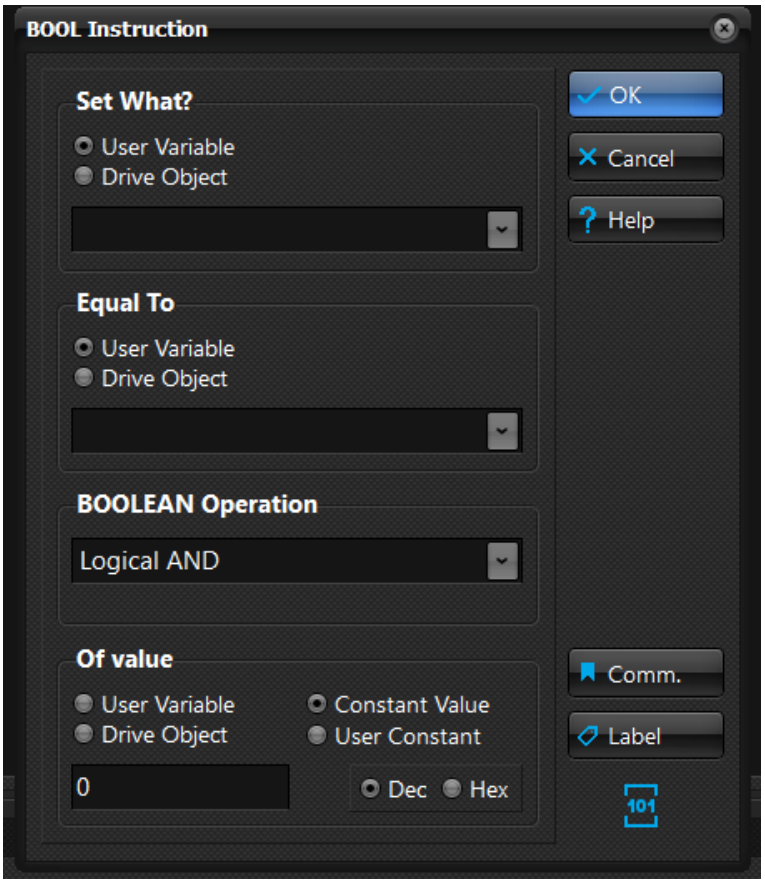
2.5 Boolean Operators

The Boolean Operators permit to perform operations according to the boolean logic.

2.5.1 BOOL Instruction

The BOOL Instruction can be used to perform one of the following operation on a User Variable or Drive Object:

- Logical AND
- Logical OR
- Logical Shift Right
- Logical Shift Left



The result of the boolean operation can be stored in a User Variable or Drive Object.

The parameter of the boolean operation (value of the AND/OR operation or the number of the shifts) can also be a numerical constant.

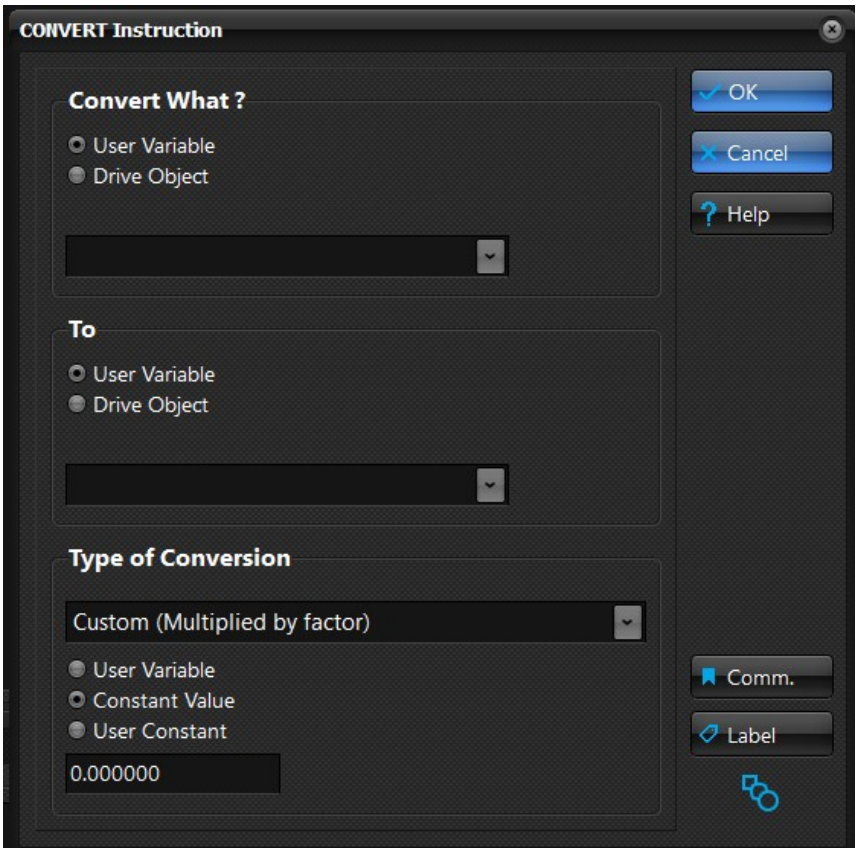
2.6 Conversion Operators

The Conversion Operators permit to perform several conversions from various measure units.

2.6.1 CONVERT Instruction

The CONVERT instruction can be used to perform the following conversions:

- RPM to Hertz conversion
- Hertz to RPM conversion
- Custom conversion (multiplied or divide by a factor)
- Labelling Conversions



The CONVERT instruction is the only e3PLC instruction that perform the calculation using the floating point (REAL) arithmetic. Thus even though the conversion result is converted to an integer (losing the decimal part), the source user variable and the conversion factor (user variable or constant value) can be REAL type.

3.0 Debugging the e3PLC Application

The e3PLC Studio allows to debug the user application very easily. To do that it is necessary to go ONLINE with the drive pressing the 

button (Go ONLINE). To go back to the OFFLINE condition just press the 

button.

The e3PLC Studio issues a warning if the edited application has not been checked (see §*1.14*) or if the Application User Comment (see §*1.7*) is different than the one stored in the drive (applications could be different). This is to prevent weird results when debugging the application.

In the ONLINE condition it is not possible to edit the application in the editor window.

In the e3PLC Studio status bar it is possible to control the current condition (ONLINE or OFFLINE) and the current condition of the e3PLC Executor on the drive (Application Stopped or Running)

3.1 Controlling the e3PLC Application Execution

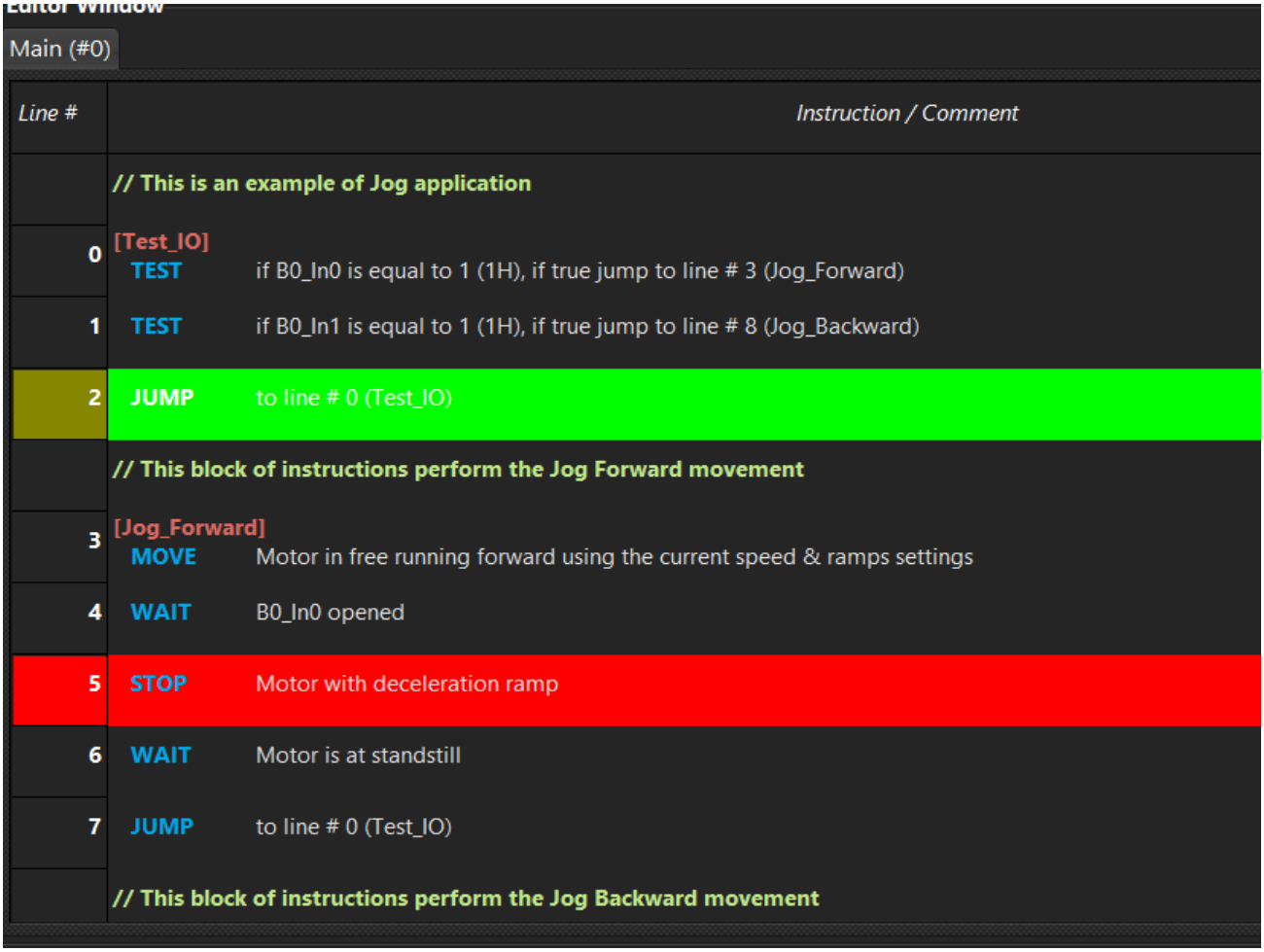
When being in the ONLINE condition the application execution toolbar is visible, showing the possible actions:

(START) When the application execution is stopped it starts the execution.

(STOP) When the application is running it stops the execution.

(STEP) When the application execution is stopped it executes one program instruction.

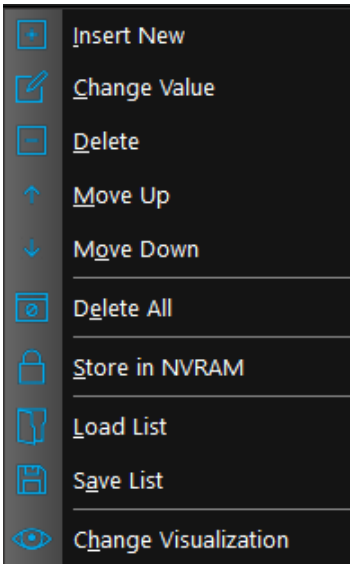
(BEGIN) When the application execution is stopped it resets the application execution. When pressing the **(START)** button after the **(BEGIN)** button the execution will start from the beginning.



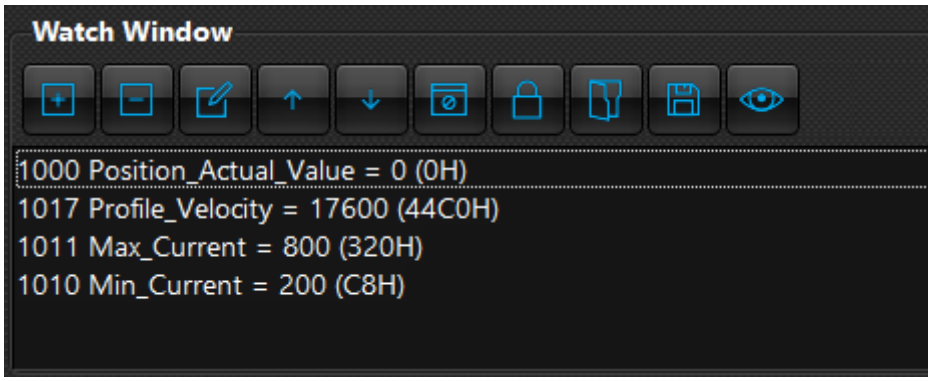
When the application execution is stopped a green color highlights the next instruction line that will be executed when the **(START)** button is pressed.
The red color highlights a line where a breakpoint is active.
When the application is running yellow boxes are drawn aside the lines that are in execution.

3.2 Watch Window

When being in the ONLINE condition the Watch Window will be enabled and the contained items are updated every 500ms. To add or remove an item to be watched (Drive Object or User Variable) just press the left mouse button inside the Watch Window area. A context menu will appear with the possible operations.

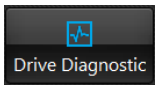


It is also possible to directly change a value of an item by double clicking on it.
It is also possible to save or load a list (.awtc extension) from the disk.

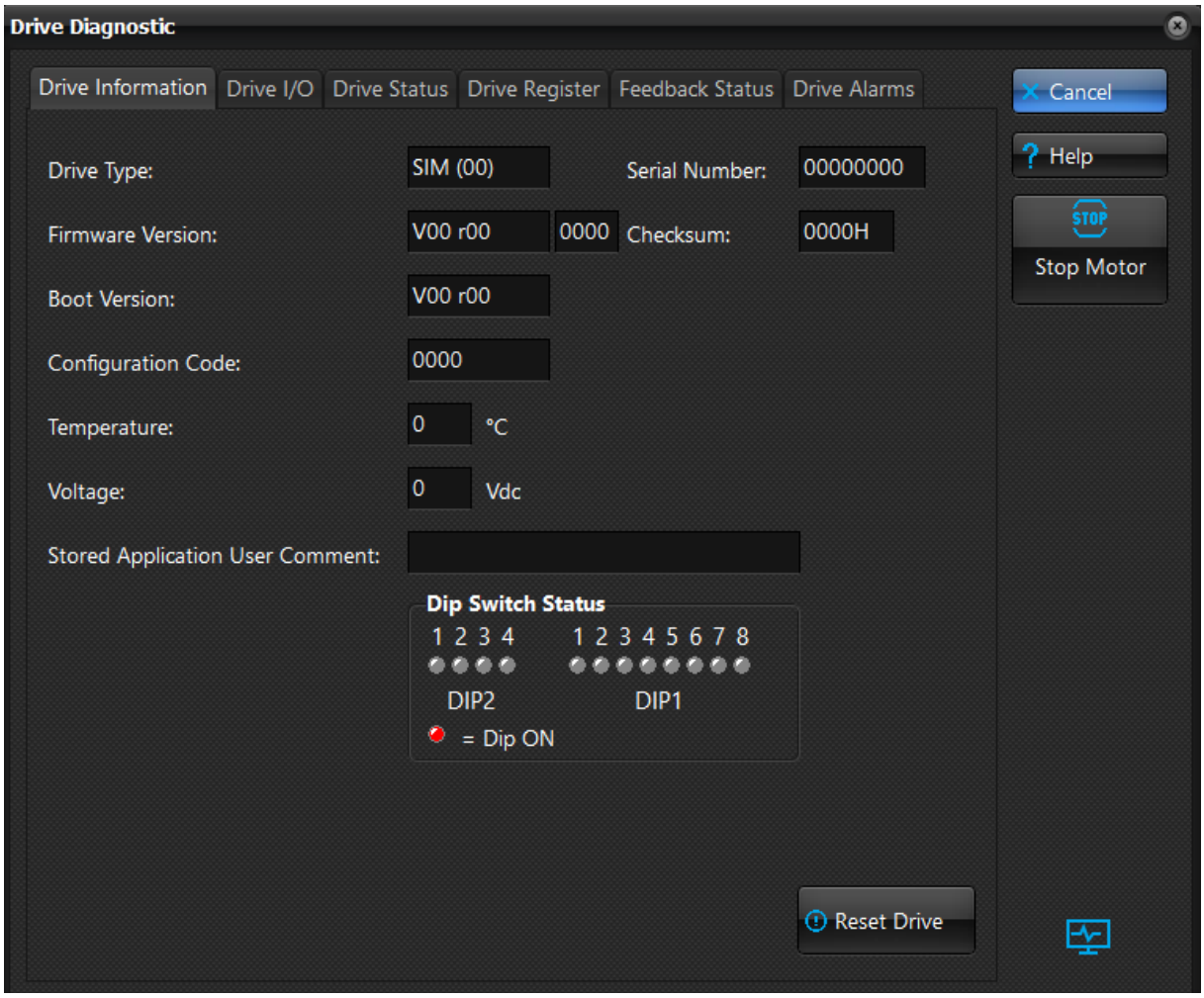


3.3 Drive Diagnostic Window

In the ONLINE condition, the 'DriveDiagnosticWindow' button in the toolbar with the application execution controls permits to open the Drive Diagnostic Window:



With this window it is possible to get various drive information as well as to simulate the digital inputs, force a Drive reset and read or clear the drive alarms history.



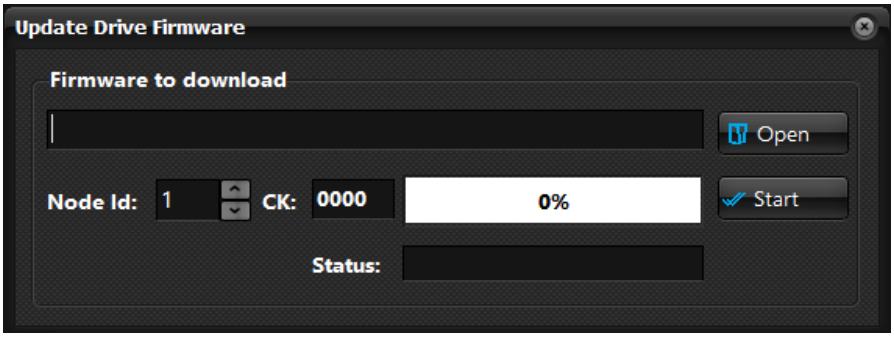
This window is automatically closed when switching to OFFLINE condition.

4.0 e3PLC Studio Additional Tools

The e3PLC Studio provides additional tools useful for drive services or profiling.

4.1 Drive Firmware Update Window

It is possible to update the drive's internal firmware by selecting the *'Fw Update'* item in the '*Tools'* main menu voice.



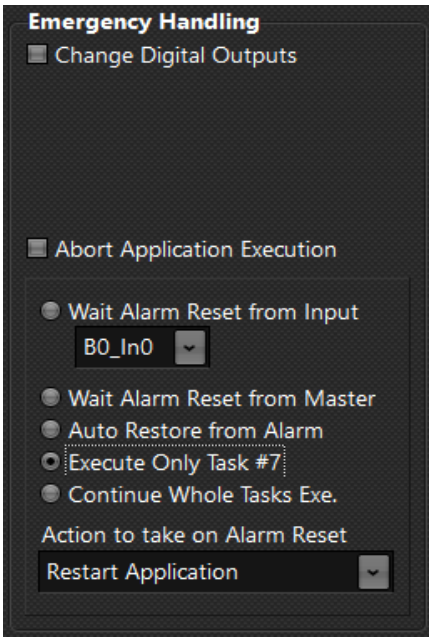
!! The firmware update is a critical process that could makes the drive unusable !!
Perform this operation only under EVER supervision.

5.0 Emergency Handling

The user can select how to handle the emergency situations that could occur during the execution of the e3PLC Application.
When an emergency condition is detected by the e3PLC Firmware the application execution is halted (the motor movement is aborted as well) and the actions set by the user in the Emergency Handling frame (see e3PLC Studio Application Global Parameters §*1.7*) are performed:

It is possible to:

- Set the Digital Outputs state
- Abort the application execution
- Wait for an alarm reset that can be taken from:
  - Digital Input
  - Fieldbus (Modbus or CANopen). See 'Master_Register' object.
  - Drive itself (Auto Restore)
- Execute the task #7 as Emergency Task
- Continue the execution of whole tasks
- Then it is possible to restart the application execution from the beginning (Reset Application Execution) checked, or continuing from the instruction where the emergency condition was detected.



The Emergency Handling set by the user will be ignored in case the following conditions occur:

- The drive is in emergency condition since the power up
- No user program is stored in the NVRAM drive or the program is corrupted
- An emergency that inhibit the e3PLC Executor

If Task #7 is set as Emergency Task, the restore from alarm happens as soon as the task end.

6.0 Application Execution

At switch-on the drive will retrieve the startup parameters (settings, motor currents, ramps...) stored in NVRAM while checking if a valid user application is stored. If so, after about 3 seconds, the program starts its execution from the first line of the first defined task. Within this 3 seconds it is possible to abort the execution of the application by means of the e3PLC Studio going in ONLINE condition and pressing the (STOP) button. While executing the e3PLC user application the drive will continue to act as a CANopen/MODBUS standard slave and all the checking enabled in the '*Drive_Working_Settings*' object are active.

7.0 e3PLC Object Dictionary

The Drive's objects that are available in the e3PLC Firmwares Releases are explained below. The objects are listed in alphabetical order.

Name: **Analog_In[0÷1]**
Address: **1110H,1111H**
CANopen Index.Sub: **6404.1H,6404.2H**
Type: WORD
Access: r
Unit: mV
Range: -10000 ÷ 10000
Default Value: --
Store Supported: No

Description: It contains the value of the drive's analog input 0 and 1.

Notes: The number of available analog inputs depends on the version of the drive currently in use.

Name: **Analog_In[0÷1]_Max_Scale_mV**
Address: **1131H,1141H**
CANopen Index.Sub: **2250.2H,2251.2H**
Type: WORD
Access: rw
Unit: mV
Range: -10000 ÷ 10000
Default Value: 0
Store Supported: Yes

Description: Specify the maximum value of the analog input (#0 or #1) used for the computation of the Analog_In[x]_Out according to the following formula:

$$Analog\_In[x]\_Out = (Analog\_In[x] - Analog\_In[x]\_Min\_mV) * \frac{(Analog\_In[x]\_Max\_Out - Analog\_In[x]\_Min\_Out)}{(Analog\_In[x]\_Max\_mV - Analog\_In[x]\_Min\_mV)} + Analog\_In[x]\_Min\_Out$$

Notes: See *Analog_In[x]_Type*.

Name: **Analog_In[0÷1]_Max_Scale_Out**
Address: **1134H,1144H**
CANopen Index.Sub: **2250.4H,2251.4H**
Type: DWORD
Access: rw
Unit: --
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: Yes

Description: Specify the maximum value of the analog input output value Analog_In[x]_Out that is computed according to the following formula:

$$Analog\_In[x]\_Out = (Analog\_In[x] - Analog\_In[x]\_Min\_mV) * \frac{(Analog\_In[x]\_Max\_Out - Analog\_In[x]\_Min\_Out)}{(Analog\_In[x]\_Max\_mV - Analog\_In[x]\_Min\_mV)} + Analog\_In[x]\_Min\_Out$$

Notes:

Name: **Analog_In[0÷1]_Min_Scale_mV**
Address: **1130H,1140H**
CANopen Index.Sub: **2250.1H,2251.1H**
Type: WORD
Access: rw
Unit: mV
Range: -10000 ÷ 10000
Default Value: 0
Store Supported: Yes

Description: Specify the minimum value of the analog input (#0 or #1) used for the computation of the Analog_In[x]_Out according to the following formula:

$$Analog\_In[x]\_Out = (Analog\_In[x] - Analog\_In[x]\_Min\_mV) * \frac{(Analog\_In[x]\_Max\_Out - Analog\_In[x]\_Min\_Out)}{(Analog\_In[x]\_Max\_mV - Analog\_In[x]\_Min\_mV)} + Analog\_In[x]\_Min\_Out$$

Notes: See *Analog_In[x]_Type.*

Name: **Analog_In[0÷1]_Min_Scale_Out**
Address: **1132H,1142H**
CANopen Index.Sub: **2250.3H,2251.3H**
Type: DWORD
Access: rw
Unit: **--**
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: Yes

Description: Specify the minimum value of the analog input output value Analog_In[x]_Out that is computed according to the following formula:

$$Analog\_In[x]\_Out = (Analog\_In[x] - Analog\_In[x]\_Min\_mV) * \frac{(Analog\_In[x]\_Max\_Out - Analog\_In[x]\_Min\_Out)}{(Analog\_In[x]\_Max\_mV - Analog\_In[x]\_Min\_mV)} + Analog\_In[x]\_Min\_Out$$

Notes:

Name: **Analog_In[0÷1]_Out**
Address: **1136H,1146H**
CANopen Index.Sub: **2250.5H,2251.5H**
Type: DWORD
Access: r
Unit: **--**
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: No

Description: It contains the value of the Analog Input (#0 or #1) converted to the user measure unit according to the following formula:

$$Analog\_In[x]\_Out = (Analog\_In[x] - Analog\_In[x]\_Min\_mV) * \frac{(Analog\_In[x]\_Max\_Out - Analog\_In[x]\_Min\_Out)}{(Analog\_In[x]\_Max\_mV - Analog\_In[x]\_Min\_mV)} + Analog\_In[x]\_Min\_Out$$

Notes:

Name: **Analog_In[0]_K_Filter**
Address: **1120H**
CANopen Index.Sub: **2200.06H**
Type: WORD
Access: rw
Unit: **--**
Range: 0 ÷ 10000
Default Value: --
Store Supported: Yes

Description: It contains the value of the K constant used for filtering the analog input #0. The formula for filtering the analog inputs is the following:

$$Analog\_Input\_Filtered_{(n)} = Analog\_Input\_Filtered_{(n-1)} + \frac{(Analog\_Input_{(n)} - Analog\_Input\_Filtered_{(n-1)})}{Analog\_Inputs\_K\_Filter}$$

High value of K lead to a more filtered analog input value. With K = 1 the filter is disabled and the Analog_In[0] objects return the instant value of analog input #0 without any filtering. With K = 0 the analog input #0 is disabled.

Notes:

Name: **Analog_In[0]_Type**
Address: **1128H**
CANopen Index.Sub: **2200.EH**
Type: WORD
Access: rw
Unit: --
Range: 0 ; 1

0 = Differential ±10V
1 = Potentiometer

Default Value: --
Store Supported: Yes

Description: This object contains Analog Input0 Type. See '*C Appendix*'.

Notes:

Name: **Analog_In[1]_K_Filter**
Address: **112AH**
CANopen Index.Sub: **2200.10H**
Type: WORD
Access: rw
Unit: **--**
Range: 0 ÷ 10000
Default Value: --
Store Supported: Yes

Description: It contains the value of the K constant used for filtering the analog input #1. The formula for filtering the analog inputs is the following:

$$Analog\_Input\_Filtered_{(n)} = Analog\_Input\_Filtered_{(n-1)} + \frac{(Analog\_Input_{(n)} - Analog\_Input\_Filtered_{(n-1)})}{Analog\_Inputs\_K\_Filter}$$

High value of K lead to a more filtered analog input value. With K = 1 the filter is disabled and the Analog_In[1] objects returns the instant value of analog input #1 without any filtering. With K = 0 the analog input #1 is disabled.

Notes:

Name: **Analog_In[1]_Type**
Address: **1129H**
CANopen Index.Sub: **2200.FH**
Type: WORD
Access: rw
Unit: --
Range: 0 ; 1

0 = Differential ±10V
1 = Potentiometer

Default Value: --
Store Supported: Yes

Description: This object contains Analog Input1 Type. See '*C Appendix*'.

Notes:

Name: **Analog_Out[0÷1]**
Address: **1112H,1113H**
CANopen Index.Sub: **6414.1H,6414.2H**
Type: WORD
Access: rw
Unit: mV
Range: 0 ÷ 10000
Default Value: 0
Store Supported: No

Description: This object is used to set/read the value of the analog outputs**.**

Notes: The number of available analog outputs depends on the version of the drive currently in use.

Name: **Analog_Speed_Max_Scale_Hz**
Address: **1126H**
CANopen Index.Sub: **2200.DH**
Type: DWORD
Access: rw
Unit: Hertz
Range: -Max_Profile_Velocity ÷ Max_Profile_Velocity
Default Value: 20000
Store Supported: Yes

Description: It contains the value of the motor speed when Analog Input #0 value is equal to *Analog_Speed_Max_Scale_mV* value while electric gear from analog input is enabled.

Notes: See *Motor_Gear_Type* and *Drive_Working_Settings* objects.

Name: **Analog_Speed_Max_Scale_mV**
Address: **1122H**
CANopen Index.Sub: **2200.AH**
Type: WORD
Access: rw
Unit: mV
Range: -10000 ÷ 10000
Default Value: 10000
Store Supported: Yes

Description: It contains the value the Analog Input #0 should match to move the motor at *Analog_Speed_Max_Scale_Hz* speed while electric gear from analog input is enabled.

Notes: See *Motor_Gear_Type* and *Drive_Working_Settings* objects. See *Analog_In[x]_Type*.

Name: **Analog_Speed_Min_Scale_Hz**
Address: **1124H**
CANopen Index.Sub: **2200.CH**
Type: DWORD
Access: rw
Unit: Hertz
Range: -Max_Profile_Velocity ÷ Max_Profile_Velocity
Default Value: 0
Store Supported: Yes

Description: It contains the value of the motor speed when Analog Input #0 value is equal to *Analog_Speed_Min_Scale_mV* value while electric gear from analog input is enabled.

Notes: See *Motor_Gear_Type* and *Drive_Working_Settings* objects.

Name: **Analog_Speed_Min_Scale_mV**
Address: **1121H**
CANopen Index.Sub: **2200.9H**
Type: WORD
Access: rw
Unit: mV
Range: -10000 ÷ 10000
Default Value: 0
Store Supported: Yes

Description: It contains the value the Analog Input #0 should match to move the motor at *Analog_Speed_Min_Scale_Hz* speed while electric gear from analog input is enabled.

Notes: See *Motor_Gear_Type* and *Drive_Working_Settings* objects. See *Analog_In[x]_Type*.

Name: **Analog_Speed_Tolerance_0V**
Address: **1123H**
CANopen Index.Sub: **2200.BH**
Type: WORD
Access: rw
Unit: mV
Range: 0 ÷ 10000
Default Value: 0
Store Supported: Yes

Description: It contains the tolerance value for Analog Input #0 nearby 0V while electric gear from analog input is enabled.

Notes: See *Motor_Gear_Type* and *Drive_Working_Settings* objects.

Name: B0_Digital_Inputs
Address: 1100H
CANopen Index.Sub: 6000.1H
Type: WORD
Access: r
Unit: --
Range: 0 (all inputs are open) ÷ 2^n-1 (all inputs are closed, where n is the number of digital inputs available)
Default Value: --
Store Supported: No

Description: It contains the status of all the inputs on bank 0 of the drive.

Notes: See *A Appendix*.
A numeric value is associated to the input, following the procedure laid out hereafter:

INPUTS	VALUE
Input 0	1
Input 1	2
Input 2	4
Input 3	8
Input 4	16
Input 5	32
Input n	2^n

Name: B0_Digital_Inputs_Falling_Edge
Address: 1106H
CANopen Index.Sub: 2200.13H
Type: WORD
Access: rw
Unit: --
Range: 0 ÷ 2^n-1 (where n is the number of digital inputs available)
Default Value: 0

Bit #	Input	Description	Default Value
0	B0_In0	1 = falling edge detected 0 = no falling edge detected	0
1	B0_In1	1 = falling edge detected 0 = no falling edge detected	0
2	B0_In2	1 = falling edge detected 0 = no falling edge detected	0
3	B0_In3	1 = falling edge detected 0 = no falling edge detected	0
n	B0_Inn	1 = falling edge detected 0 = no falling edge detected	0

Store Supported: --

Description: It contains the detection of the falling edges of the digital inputs of bank 0. The firmware will set the corresponding bit every time detects a falling edge on each input of bank 0. The user software is responsible of reset of the bits to start again the detection of falling edges.

Notes: See *A Appendix*.

Name: **B0_Digital_Inputs_Rising_Edge**
Address: **1104H**
CANopen Index.Sub: **2200.12H**
Type: WORD
Access: rw
Unit: **--**
Range: 0 ÷ 2^n-1 (where n is the number of digital inputs available)
Default Value: 0

Bit #	Input	Description	Default Value
0	B0_In0	1 = rising edge detected 0 = no falling edge detected	0
1	B0_In1	1 = rising edge detected 0 = no falling edge detected	0
2	B0_In2	1 = rising edge detected 0 = no falling edge detected	0
3	B0_In3	1 = rising edge detected 0 = no falling edge detected	0
n	B0_Inn	1 = rising edge detected 0 = no falling edge detected	0

Store Supported: --

Description: It contains the detection of the rising edges of the digital inputs of bank 0. The firmware will set the corresponding bit every time detects a rising edge on each input of bank 0. The user software is responsible of reset of the bits to start again the detection of rising edges.

Notes: See *A Appendix*.

Name: **B0_Digital_Inputs_Polarity**
Address: **112DH**
CANopen Index.Sub: **2200.7H**
Type: WORD
Access: rw
Unit: **--**
Range: 0 (all inputs have normal polarity) ÷ 2^n-1 (all inputs have inverted polarity, where n is the number of digital inputs available)
Default Value: 0

Bit #	Input	Description	Default Value
0	B0_In0	1 = inverted polarity 0 = normal polarity	0
1	B0_In1	1 = inverted polarity 0 = normal polarity	0
2	B0_In2	1 = inverted polarity 0 = normal polarity	0
3	B0_In3	1 = inverted polarity 0 = normal polarity	0
n	B0_Inn	1 = inverted polarity 0 = normal polarity	0

Store Supported: Yes

Description: It contains the polarity of all the digital inputs on bank 0 of the drive. When the polarity is normal, the input status (*B0_Digital_Inputs*) is 1 if the voltage is supplied to the input (5V or 24V) and 0 if no voltage is supplied to the input. When the polarity is inverted, the input status (*B0_Digital_Inputs*) is 0 if the voltage is supplied to the input (5V or 24V) and 1 if no voltage is supplied to the input.

Notes: See *A Appendix*.

Name: **B0_Digital_Outputs**
Address: **1101H**
CANopen Index.Sub: **6200.1H**
Type: WORD
Access: r/w
Unit: **--**
Range: 0 (all outputs are open) ÷ 2^n-1 (all outputs are closed, where n is the number of digital outputs available)
Default Value: --
Store Supported: No

Description: It contains the status of all the outputs on bank 0 of the drive.

Notes: See *A Appendix*.
A numeric value is associated to the outputs, following the procedure laid out hereafter:

OUTPUTS	VALUE
Output 0	1
Output 1	2
Output 2	4
Output 3	8
Output 4	16
Output 5	32
Output n	2^n

Name: **B0_Digital_Outputs_Polarity**
Address: **112EH**
CANopen Index.Sub: **2200.8H**
Type: WORD
Access: rw
Unit: **--**
Range: 0 (all outputs have normal polarity) ÷ 2^n-1 (all outputs have inverted polarity, where n is the number of digital outputs available)
Default Value: 0

Bit #	Output	Description	Default Value
0	B0_Out0	1 = inverted polarity 0 = normal polarity	0
1	B0_Out1	1 = inverted polarity 0 = normal polarity	0
n	B0_Outn	1 = inverted polarity 0 = normal polarity	0

Store Supported: Yes

Description: It contains the polarity of all the digital outputs on bank 0 of the drive. When the polarity is normal, the output will be 24V if status (*B0_Digital_Outputs*) = 1 and 0V if status = 0. When the polarity is inverted, the output will be 0V if status (*B0_Digital_Outputs*) = 1 and 24V if status = 0.

Notes: See *A Appendix*.

Name: B1_Digital_Inputs
Address: 1102H
CANopen Index.Sub: 6100.2H
Type: WORD
Access: r
Unit: --
Range: 0 (all inputs are open) ÷ 2^n-1 (all inputs are closed, where n is the number of digital inputs available)
Default Value: --
Store Supported: No

Description: It contains the status of all the inputs on bank 1 of the drive.

Notes: See *A Appendix*.
A numeric value is associated to the input, following the procedure laid out hereafter:

INPUTS	VALUE
Input 0	1
Input 1	2
Input 2	4
Input 3	8
Input 4	16
Input 5	32
Input n	2^n

Name: B1_Digital_Inputs_Falling_Edge
Address: 1107H
CANopen Index.Sub: 2200.15H
Type: WORD
Access: rw
Unit: --
Range: 0 ÷ 2^n-1 (where n is the number of digital inputs available)
Default Value: 0

Bit #	Input	Description	Default Value
0	B1_In0	1 = falling edge detected 0 = no falling edge detected	0
1	B1_In1	1 = falling edge detected 0 = no falling edge detected	0
2	B1_In2	1 = falling edge detected 0 = no falling edge detected	0
3	B1_In3	1 = falling edge detected 0 = no falling edge detected	0
n	B1_Inn	1 = falling edge detected 0 = no falling edge detected	0

Store Supported: --

Description: It contains the detection of the falling edges of the digital inputs of bank 1. The firmware will set the corresponding bit every time detects a falling edge on each input of bank 1. The user software is responsible of reset of the bits to start again the detection of falling edges.

Notes: See *A Appendix*.

Name: **B1_Digital_Inputs_Rising_Edge**
Address: **1105H**
CANopen Index.Sub: **2200.14H**
Type: WORD
Access: rw
Unit: **--**
Range: 0 ÷ 2^n-1 (where n is the number of digital inputs available)
Default Value: 0

Bit #	Input	Description	Default Value
0	B1_In0	1 = rising edge detected 0 = no rising edge detected	0
1	B1_In1	1 = rising edge detected 0 = no rising edge detected	0
2	B1_In2	1 = rising edge detected 0 = no rising edge detected	0
3	B1_In3	1 = rising edge detected 0 = no rising edge detected	0
n	B1_Inn	1 = rising edge detected 0 = no rising edge detected	0

Store Supported: --

Description: It contains the detection of the rising edges of the digital inputs of bank 1. The firmware will set the corresponding bit every time detects a rising edge on each input of bank 1. The user software is responsible of reset of the bits to start again the detection of rising edges.

Notes: See *A Appendix*.

Name: **B1_Digital_Inputs_Polarity**
Address: **112BH**
CANopen Index.Sub: **2200.16H**
Type: WORD
Access: rw
Unit: **--**
Range: 0 (all inputs have normal polarity) ÷ 2^n-1 (all inputs have inverted polarity, where n is the number of digital inputs available)
Default Value: 0

Bit #	Input	Description	Default Value
0	B1_In0	1 = inverted polarity 0 = normal polarity	0
1	B1_In1	1 = inverted polarity 0 = normal polarity	0
2	B1_In2	1 = inverted polarity 0 = normal polarity	0
3	B1_In3	1 = inverted polarity 0 = normal polarity	0
n	B1_Inn	1 = inverted polarity 0 = normal polarity	0

Store Supported: Yes

Description: It contains the polarity of all the digital inputs on bank 1 of the drive. When the polarity is normal, the input status (*B1_Digital_Inputs*) is 1 if the voltage is supplied to the input (5V or 24V) and 0 if no voltage is supplied to the input. When the polarity is inverted, the input status (*B1_Digital_Inputs*) is 0 if the voltage is supplied to the input (5V or 24V) and 1 if no voltage is supplied to the input.

Notes: See *A Appendix*.

Name: **B1_Digital_Outputs**
Address: **1103H**
CANopen Index.Sub: **6300.2H**
Type: WORD
Access: r/w
Unit: **--**
Range: 0 (all outputs are open) ÷ 2^n-1 (all outputs are closed, where n is the number of digital outputs available)
Default Value: --
Store Supported: No

Description: It contains the status of all the outputs on bank 1 of the drive.

Notes: See *A Appendix*.
A numeric value is associated to the outputs, following the procedure laid out hereafter:

OUTPUTS	VALUE
Output 0	1
Output 1	2
Output 2	4
Output 3	8
Output 4	16
Output 5	32
Output n	2^n

Notes:

Name: **B1_Digital_Outputs_Polarity**
Address: **112CH**
CANopen Index.Sub: **2200.17H**
Type: WORD
Access: rw
Unit: **--**
Range: 0 (all outputs have normal polarity) ÷ 2^n-1 (all outputs have inverted polarity, where n is the number of digital outputs available)
Default Value: 0

Bit #	Output	Description	Default Value
0	B1_Out0	1 = inverted polarity 0 = normal polarity	0
1	B1_Out1	1 = inverted polarity 0 = normal polarity	0
n	B1_Outn	1 = inverted polarity 0 = normal polarity	0

Store Supported: Yes

Description: It contains the polarity of all the digital outputs on bank 1 of the drive. When the polarity is normal, the output will be 24V if status (*B1_Digital_Outputs*) = 1 and 0V if status = 0. When the polarity is inverted, the output will be 0V if status (*B1_Digital_Outputs*) = 1 and 24V if status = 0.

Notes: See *A Appendix*.

Name: **Baud_Rate**
Address: **40B9H**
CANopen Index.Sub: **4000.8H**
Type: WORD
Access: r or rw
Unit: --
Range: 0 ÷ 7 (Modbus: 0 = 115.2k, 1= 57.6k, 2 = 38.4k, 3 = 19.2k, 4 = 9.6k, 5 = 4.8k, 6 = 2.4k, 7 = 1.2)
(CAN: 0 = 1M, 1 = 500K, 2 = 250K, 3 = 125K)
Default Value: --
Store Supported: Yes

Description: This object contains the drive's NodeId. On drives with dip-switches/rotoswitches the object is read-only. On drives without dip-switches/rotoswitches this object can be changed using the e3PLC Studio or following the procedure explained at §*10.1.2* or §*11.1.2*.
Notes:

Name: **BiSS_Encoder_Actual_Value**
Address: **2A00H**
CANopen Index.Sub: **2A00.0H**
Type: DWORD
Access: r
Unit: IU
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: No

Description: This object contains the BiSS_Encoder_Internal_Value normalized to 32 bits and subtracted by the BiSS_Encoder_Offset_Value.

BiSS_Encoder_Actual_Value = BiSS_Encoder_Internal_Value - BiSS_Encoder_Offset_Value

Since the BiSS_Encoder_Internal_Value cannot be changed, it is possible to get the desired value acting on the Offset. For Instance if a particular position motor position have to become the 0 position, just set the *BiSS_Encoder_Offset_Value* = BiSS_Encoder_Internal_Value. When using a BiSS Absolute Encoder it is suggested to set the motor resolution (see *Motor_Step_Angle* object) to -1 to have the same resolution between motor increments and encoder increments. Setting at startup the *Position_Actual_Value* object equal to *BiSS_Encoder_Actual_Value* object the following motor movement to target issued will be relative to the actual absolute encoder position.

Notes: This object is available only on drive models fitted with BiSS Encoder input.

Name: BiSS_Encoder_Config
Address: 2A04H
CANopen Index.Sub: 2A04.0H
Type: DWORD
Access: r
Unit: IU
Range: 0 ÷ FFFFFFFFH
Default Value: 0 (BiSS Encoder disabled)

Byte 3	Byte 2	Byte 1	Byte 0
Reserved	Reserved	Multiturn Bits #	SingleTurn Bits #

Store Supported: Yes

Description: This object contains the configuration for the BiSS Encoder. For the correct reading of the BiSS Encoder connected to the drive it is necessary to specify the right resolution (# of bits of either single turn or multiturn). For Instance if the connected BiSS Encoder has a resolution of 17 bits single turn and 16 bits multiturn, The *BiSS_Encoder_Config* object must be set equal to 4113 (1011H).

Notes: This object is available only on drive models fitted with BiSS Encoder input.

Name: BiSS_Encoder_Internal_Value
Address: 2A0AH
CANopen Index.Sub: 2A0A.0H
Type: DWORD
Access: r
Unit: IU
Range: -2147483648 ÷ 2147483647
Default Value: --
Store Supported: No

Description: This object contains the actual BiSS Encoder value normalized to 32 bits. Regardless of the Encoder multiturn and singleturn resolution, this object contains in the low word the single turn position and in the high word the number of turns.

Notes: This object is available only on drive models fitted with BiSS Encoder input.

Name: BiSS_Encoder_Offset_Value
Address: 2A08H
CANopen Index.Sub: 2A08.0H
Type: DWORD
Access: r
Unit: IU
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: Yes

Description: This object contains the offset used to calculate the *BiSS_Encoder_Actual_Value* object.

Notes: This object is available only on drive models fitted with BiSS Encoder input.

Name: BiSS_Encoder_RxErr
Address: 2A06H
CANopen Index.Sub: 2A06.0H
Type: WORD
Access: rw
Unit: --
Range: 0 ÷ 65535
Default Value: 0
Store Supported: No

Description: This object contains the number of BiSS Encoder receive errors. This object should be always 0 otherwise there are some communication errors with the Encoder. Check the cabling and the cable length to solve communication errors. The error counter can be cleared setting it to 0.

Notes: This object is available only on drive models fitted with BiSS Encoder input.

Name: BiSS_Encoder_Status
Address: 2A02H
CANopen Index.Sub: 2A02.0H
Type: WORD
Access: r
Unit: --
Range: 0 ÷ 65535

Bit #	Description
0	1 = Ok, Good 0 = Warning Condition
1	1 = Ok, Data Valid 0 = Error (HW Failure detected)
2 ÷ 15	Reserved

Default Value: 0
Store Supported: No

Description: This object contains the value of diagnostic bits of BiSS Encoder received communication frame. If the Encoder is correctly working the value of this object should be always equal to 3 otherwise there are some communication errors with the Encoder or the Encoder is faulty.

Notes: This object is available only on drive models fitted with BiSS Encoder input.

Name: Boot_Version
Address: 4099H
CANopen Index.Sub: 4004.2H
Type: WORD
Access: r
Unit: --
Range: 0000H ÷ FFFFH
Default Value: --
Store Supported: --

Description: It contains the current version of the boot. The MSB contains the version, while the LSB contains the release (Example: value 0105H means V01r05).

Notes:

Name: Brake_Control_Settings
Address: 2C00H
CANopen Index.Sub: 2C00.0H
Type: WORD
Access: rw
Unit: --
Range: 0 ÷ 65535
Default Value: 0
Store Supported: Yes

Description: This object defines settings regarding the *Brake Control* (*§9.5*) functionality.

Bit #	Name	Description	Default Value
0	**Automatic Brake Handling**	0 = Disabled	0
		1 = Enabled	0
1	**Brake Control Type**	0 = Type 0	0
		1 = Type 1	0
2	**Brake Digital Output use Mode**	0 = Mode 0	0
		1 = Mode1	0
3	**Brake Digital Output Bank**	0 = The Brake Digital Output is one of digital outputs of Bank #0	0
		1 = The Brake Digital Output is one of digital outputs of Bank #1	0
4÷7	**Brake Digital Output**	Digital output used for the Brake	0
8÷15	**Reserved**	Reserved	0

Bit Explanation:

Automatic Brake Handling :

When this bit is set (1) the Automatic Brake Handling is enabled.
The digital output, defined by bit4÷bit7 and bit3 of *Brake_Control_Settings* object, is used for Brake Handling according to the rules defined by bit1 and bit2 of of *Brake_Control_Settings* object.

When this bit is reset (0) the Automatic Handling is disabled and settings of bit1÷bit7 are not considered.

Brake Control Type :

Brake Control Type = 0

The Brake is close (active) :

- if the drive is in Emergency condition.
- if the drive is not in Emergency condition and bit2 of *Brake_Control_Settings* object is equal to 1 and the digital output (used for the Brake and defined by bit4÷bit7 and bit3 of *Brake_Control_Settings* object) is reset by *B0_Digital_Outputs* object (if bit3 = 0 of *Brake_Control_Settings* object) *or* *B1_Digital_Outputs* object (if bit3 = 1 of *Brake_Control_Settings* object).

The Brake is open (released) :

- if the drive is not in Emergency condition and bit2 of *Brake_Control_Settings* object is equal to 0.

- if the drive is not in Emergency condition and bit2 *of* *Brake_Control_Settings* object is equal to 1 and the digital output (used for the Brake and defined by bit4÷bit7 and bit3 of *Brake_Control_Settings* object) is set by *B0_Digital_Outputs* object (if bit3 = 0 of *Brake_Control_Settings* object*)* or *B1_Digital_Outputs* object (if bit3 = 1 of *Brake_Control_Settings* object*)*.

Brake Control Type = 1

The Brake is close (active) :

- if the drive is in Emergency condition.
- if the drive is not in Emergency condition and bit2 of *Brake_Control_Settings* object is equal to 1 and the digital output (used for the Brake and defined by bit4÷bit7 and bit3 of *Brake_Control_Settings* object) is reset by *B0_Digital_Outputs* object (if bit3 = 0 of *Brake_Control_Settings* object) *or* *B1_Digital_Outputs* object (if bit3 = 1 of *Brake_Control_Settings* object*)*.
- if the drive is not in Emergency condition and the motor is at standstill and bit2 of *Brake_Control_Settings* object is equal to 0.

The Brake is open (released) :

- if the drive is not in Emergency condition and bit2 *of* *Brake_Control_Settings* object is equal to 0 and the motor is running.
- if the drive is not in Emergency condition and bit2 *of* *Brake_Control_Settings* object is equal to 1 and the digital output (used for the Brake and defined by bit4÷bit7 and bit3 of *Brake_Control_Settings* object) is set by *B0_Digital_Outputs* object (if bit3 = 0 of *Brake_Control_Settings* object*)* or *B1_Digital_Outputs* object (if bit3 = 1 of *Brake_Control_Settings* object*)*.

Brake Digital Output use Mode :

When this bit is set (1), according to the Brake Control Type (bit1 of *Brake_Control_Settings* object), the state of Digital Output (used for the Brake and defined by bit4÷bit7 and bit3 of *Brake_Control_Settings* object), can be changed by *B0_Digital_Outputs* object (if bit3 = 0 of *Brake_Control_Settings* object) *or* *B1_Digital_Outputs* object (if bit3 = 1 of *Brake_Control_Settings* object).

When this bit is reset (0) the state of Digital Output (used for the Brake and defined by bit4÷bit7 and bit3 of *Brake_Control_Settings* object) can not be changed by *B0_Digital_Outputs* *object* (if bit3 = 0 of *Brake_Control_Settings* object) *or* *B1_Digital_Outputs* object (if bit3 = 1 of *Brake_Control_Settings* object).

Brake Digital Output Bank :

When this bit is set (1) the Digital Output, used for the Brake and defined by bit4÷bit7 and bit3 of *Brake_Control_Settings* object, is related to digital outputs of Bank#1.

When this bit is set (0) the Digital Output, used for the Brake and defined by bit4÷bit7 and bit3 of *Brake_Control_Settings* object, is related to digital outputs of Bank#0.

Brake Digital Output :

The bit4÷bit7 define Digital output used for the Brake. It can be related to digital outputs of Bank#0 (if bit3 = 0 of *Brake_Control_Settings* object) or Bank#1 (if bit3 = 1 of *Brake_Control_Settings* object).

If the Digital Output used for the Brake is B0_OUT0 (digital output #0 of Bank#0) is necessary to set (1) the bit2 of *Drive_Working_Settings_Extended* object.
If the Digital Output used for the Brake is B0_OUT1 (digital output #1 of Bank#0) is necessary to set (1) the bit7 of *Drive_Working_Settings* object.

Notes:

- This object is available with firmware V03r20 or superior.
- See §9.5
- See *2C01.0H*, *2C02.0H*, *2C03.0H*, *2C04.0H*

Name: **Brake_Control_Time1_Close_Brake**
Address: **2C01H**
CANopen Index.Sub: **2C01.0H**
Data Type: WORD
Access: rw
Unit: ms
Range: 0 ÷ 1000
Default Value: 150
Store Supported: Yes

Description: This object is used for the *Brake Control* (*§9.5*) functionality and defines the time between motor standstill and closing of Brake.

Notes:

- This object is available with firmware V03r20 or superior.
- See *§9.5*
- See *2C00.0H*, *2C02.0H*, *2C03.0H*, *2C04.0H*

Name: **Brake_Control_Time2_Close_Brake**
Address: **2C02H**
CANopen Index.Sub: **2C02.0H**
Data Type: WORD
Access: rw
Unit: ms
Range: 0 ÷ 1000
Default Value: 150
Store Supported: Yes

Description: This object is used for the *Brake Control* (*§9.5*) functionality and defines the time between closing of Brake and switching off of motor current.

Notes:

- This object is available with firmware V03r20 or superior.
- See *§9.5*
- See *2C00.0H*, *2C01.0H*, *2C03.0H*, *2C04.0H*

Name: **Brake_Control_Time1_Open_Brake**
Address: **2C03H**
CANopen Index.Sub: **2C03.0H**
Data Type: WORD
Access: rw
Unit: ms
Range: 0 ÷ 1000
Default Value: 150
Store Supported: Yes

Description: This object is used for the *Brake Control* (*§9.5*) functionality and defines the time between the switching on of motor current and the release of the Brake. During this time motor movement are not allowed.

Notes:

- This object is available with firmware V03r20 or superior.
- See *§9.5*
- See *2C00.0H*, *2C01.0H*, *2C02.0H*, *2C04.0H*

Name: **Brake_Control_Time2_Open_Brake**
Address: **2C04H**
CANopen Index.Sub: **2C04.0H**
Data Type: WORD
Access: rw
Unit: ms
Range: 0 ÷ 1000
Default Value: 150
Store Supported: Yes

Description: This object is used for the *Brake Control* (*§9.5*) functionality and defines the time for the release of the Brake. During this time motor movement are not allowed.

Notes:

- This object is available with firmware V03r20 or superior.
- See *§9.5*
- See *2C00.0H*, *2C01.0H*, *2C02.0H*, *2C03.0H*

Name: Braking_Resistor_Value
Address: 2B80H
CANopen Index.Sub: 2B80.0H
Data Type: UWORD
Access: rw
Unit: Ohm
Range: 0 ÷ 65535
Default Value: 50
Store Supported: Yes

Description: This object defines the ohmic value of the braking resistor.

Notes: *- This object is available with firmware version V03r18 or superior.*
- See §9.7

Name: Braking_Resistor_Power
Address: 2B81H
CANopen Index.Sub: 2B81.0H
Data Type: UWORD
Access: rw
Unit: Watt
Range: 0 ÷ 65535
Default Value: 50
Store Supported: Yes

Description: This object defines the rated power of the braking resistor.

Notes: *- This object is available with firmware version V03r18 or superior.*
- See §9.7

Name: Braking_Threshold_ON
Address: 2B82H
CANopen Index.Sub: 2B82.0H
Data Type: UWORD
Access: rw
Unit: Volt
Range: 0 ÷ 65535
Default Value: 380
Store Supported: Yes

Description: This object defines the DC bus voltage threshold above which the braking resistor is activated.

Notes: *- This object is available with firmware version V03r18 or superior.*
- See §9.7

Name: Braking_Threshold_OFF
Address: 2B83H
CANopen Index.Sub: 2B83.0H
Data Type: UWORD
Access: rw
Unit: Volt
Range: 0 ÷ 65535
Default Value: 370
Store Supported: Yes

Description: This object defines the DC bus voltage threshold below which the braking resistor is deactivated.

Notes: *- This object is available with firmware version V03r18 or superior.*
- See §9.7

Name: **Braking_Resistor_Overload_Time**
Address: **2B84H**
CANopen Index.Sub: **2B84.0H**
Data Type: UWORD
Access: rw
Unit: 0.1ms
Range: 0 ÷ 65535
Default Value: 0
Store Supported: Yes

Description: This object defines the maximum time the braking resistor can withstand the peak of the power.

Notes: *- This object is available with firmware version V03r18 or superior.*
- See §9.7

Name: CANopen_RX_PDO_Status
Address: ----
CANopen Index.Sub: 2300.1H
Type: WORD
Access: rw
Unit: --
Range: 0000H ÷ FFFFH

Bit #	Name	Description	Default Value
0	RX_PDO1	Receive PDO #1	0
1	RX_PDO2	Receive PDO #2	0
2	RX_PDO3	Receive PDO #3	0
3	Reserved	Reserved	0
4	Reserved	Reserved	0
5	Reserved	Reserved	0
6	Reserved	Reserved	0
7	Reserved	Reserved	0
8	Reserved	Reserved	0
9	Reserved	Reserved	0
10	Reserved	Reserved	0
11	Reserved	Reserved	0
12	Reserved	Reserved	0
13	Reserved	Reserved	0
14	Reserved	Reserved	0
15	Reserved	Reserved	0

Default Value: 0
Store Supported: --

Description: It contains the reception status of PDOs. Every time a PDO is received the corresponding bit is set. The reset of the bits is done by the user program.

Notes: This object can be used only on drives fitted with CAN interface (see §*11.0*).

Name: **CANopen_RX_PDO1_Data[0]**
Address: ----
CANopen Index.Sub: 2300.2H
Type: DWORD
Access: rw
Unit: --
Range: 00000000H ÷ FFFFFFFFH
Default Value: 0
Store Supported: --

Description: It contains the low DWORD of the RX PDO #1.

Notes: This object can be used only on drives fitted with CAN interface (see §*11.0*).

Name: **CANopen_RX_PDO1_Data[1]**
Address: ----
CANopen Index.Sub: 2300.3H
Type: DWORD
Access: rw
Unit: --
Range: 00000000H ÷ FFFFFFFFH
Default Value: 0
Store Supported: --

Description: It contains the high DWORD of the RX PDO #1.

Notes: This object can be used only on drives fitted with CAN interface (see §*11.0*).

Name: **CANopen_RX_PDO2_Data[0]**
Address: ----
CANopen Index.Sub: 2300.4H
Type: DWORD
Access: rw
Unit: --
Range: 00000000H ÷ FFFFFFFFH
Default Value: 0
Store Supported: --

Description: It contains the low DWORD of the RX PDO #2.

Notes: This object can be used only on drives fitted with CAN interface (see §*11.0*).

Name: **CANopen_RX_PDO2_Data[1]**
Address: ----
CANopen Index.Sub: 2300.5H
Type: DWORD
Access: rw
Unit: --
Range: 00000000H ÷ FFFFFFFFH
Default Value: 0
Store Supported: --

Description: It contains the high DWORD of the RX PDO #2.

Notes: This object can be used only on drives fitted with CAN interface (see §*11.0*).

Name: **CANopen_RX_PDO3_Data[0]**
Address: ----
CANopen Index.Sub: 2300.6H
Type: DWORD
Access: rw
Unit: --
Range: 00000000H ÷ FFFFFFFFH
Default Value: 0
Store Supported: --

Description: It contains the low DWORD of the RX PDO #3.

Notes: This object can be used only on drives fitted with CAN interface (see §*11.0*).

Name: **CANopen_RX_PDO3_Data[1]**
Address: ----
CANopen Index.Sub: 2300.7H
Type: DWORD
Access: rw
Unit: --
Range: 00000000H ÷ FFFFFFFFH
Default Value: 0
Store Supported: --

Description: It contains the high DWORD of the RX PDO #3.

Notes: This object can be used only on drives fitted with CAN interface (see §*11.0*).

Name: CANopen_TX_PDO_Command
Address: ----
CANopen Index.Sub: 2301.7H
Type: DWORD
Access: w
Unit: --
Range: 00000000H ÷ FFFFFFFFH

Byte 3	Byte 2	Byte 1	Byte 0 CMD	Description
NA		Target NodeId: 1 ÷ 127	1	Send to Target RX PDO #1
NA		Target NodeId: 1 ÷ 127	2	Send to Target RX PDO #2
NA			3	Send to Targets RX PDO #3
NA			4	Send TX PDO #2
Length: 0 ÷ 8	CobId: 0 ÷ 2047		5	Send Free CAN Frame

Default Value: 0
Store Supported: --

Description: This object can be used to send data to RX PDO #1 ÷ 3 of other drives on CAN network, as well as to send TX PDO #2 or a free CAN frame.

Notes: This object can be used only on drives fitted with CAN interface (see §*11.0*).

Name: CANopen_TX_PDO_SendData[0]
Address: ----
CANopen Index.Sub: 2301.5H
Type: DWORD
Access: rw
Unit: --
Range: 00000000H ÷ FFFFFFFFH
Default Value: 0
Store Supported: --

Description: It contains the low DWORD of the data sent using the CANopen_TX_PDO_Command object (except the TX PDO #2).

Notes: This object can be used only on drives fitted with CAN interface (see §*11.0*).

Name: CANopen_TX_PDO_SendData[1]
Address: ----
CANopen Index.Sub: 2301.6H
Type: DWORD
Access: rw
Unit: --
Range: 00000000H ÷ FFFFFFFFH
Default Value: 0
Store Supported: --

Description: It contains the high DWORD of the data sent using the CANopen_TX_PDO_Command object (except the TX PDO #2).

Notes: This object can be used only on drives fitted with CAN interface (see §*11.0*).

Name: **CANopen_TX_PDO2_Data[0]**
Address: ----
CANopen Index.Sub: 2301.3H
Type: DWORD
Access: rw
Unit: --
Range: 00000000H ÷ FFFFFFFFH
Default Value: 0
Store Supported: --

Description: It contains the low DWORD of the TX PDO #2.

Notes: This object can be used only on drives fitted with CAN interface (see §*11.0*).

Name: **CANopen_TX_PDO2_Data[1]**
Address: ----
CANopen Index.Sub: 2301.4H
Type: DWORD
Access: rw
Unit: --
Range: 00000000H ÷ FFFFFFFFH
Default Value: 0
Store Supported: --

Description: It contains the high DWORD of the TX PDO #2.

Notes: This object can be used only on drives fitted with CAN interface (see §*11.0*).

Name: Clockout_Prescaler
Address: 1147H
CANopen Index.Sub: 2252.00H
Type: WORD
Access: rw
Unit: --
Range: 0 ÷ FFFFH
Default Value: 0H
Store Supported: Yes

Description: This object is used to :

- Enable the clockout generation depending on the motor position
- Enable replication of Incremental Encoder Input signals (Enc#0)

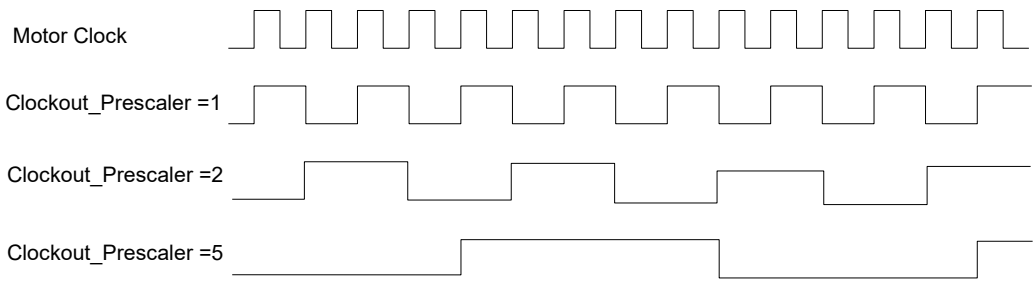
Depending on 'Clockout_Prescaler' value B0_OUT0 and B0_OUT1 digital outputs are used to generate clockout signals.

Clockout_Prescaler = 0

Clockout generation is disabled.

1 (0001h) ≤ *Clockout_Prescaler* ≤ 32767 (7FFFh)

The clock is generated on output B0_OUT1 while the motor is moving.
Clockout output is toggled every Clockout_Prescaler motor microsteps. The Max Clockout Frequency is limited to 10Khz so the max Motor Clock Frequency has to be less or equal to (10Kz * Clockout_Prescaler *2).



The Motor Clock Frequency depend on the motor speed and *Motor_Step_Angle* object so for example if the motor runs at the speed of 200 rpm we have :

Motor_Clock_Frequency = 666.66 Hz if Motor_Step_Angle = 1
Motor_Clock_Frequency = 1333.33 Hz if Motor_Step_Angle = 2
Motor_Clock_Frequency = 2666.66 Hz if Motor_Step_Angle = 4

Motor_Clock_Frequency = 85333.33 Hz if Motor_Step_Angle = 128

See also the means of *Working_Setting_Extended.Clockout_Init* object bit.

Clockout_Prescaler = 65535 (FFFFh) or 65296 (FF10h)

The clock is generated on outputs B0_Out0 and B0_Out1 in quadrature mode while the motor is moving with resolution of 200 incs/rev.

Clockout_Prescaler = 65297 (FF11h)

The clock is generated on outputs B0_Out0 and B0_Out1 in quadrature mode while the motor is moving with resolution of 400 incs/rev.

Clockout_Prescaler = 65298 (FF12h)

The clock is generated on outputs B0_Out0 and B0_Out1 in quadrature mode while the motor is moving with resolution of 500 incs/rev.

Clockout_Prescaler = 65299 (FF13h)

The clock is generated on outputs B0_Out0 and B0_Out1 in quadrature mode while the motor is moving with resolution of 800 incs/rev.

Clockout_Prescaler = 65300 (FF14h)

The clock is generated on outputs B0_Out0 and B0_Out1 in quadrature mode while the motor is moving with resolution of 1000 incs/rev.

Clockout_Prescaler = 65301 (FF15h)

The clock is generated on outputs B0_Out0 and B0_Out1 in quadrature mode while the motor is moving with resolution of 1600 incs/rev.

Clockout_Prescaler = 65302 (FF16h)

The clock is generated on outputs B0_Out0 and B0_Out1 in quadrature mode while the motor is moving with resolution of 2000 incs/rev.

Clockout_Prescaler = 65534 (FFFEh)

When motor is running B0_OUT1 = ON.
When motor is stopped B0_OUT1 = OFF

Clockout_Prescaler = 65280 (FF00h)

The Incremental encoder digital input signals (Enc#0) are replicated on B0_OUT0 and B0_OUT1 digital outputs.

This function is not available on all Titanio drives. Refer to specific hardware manual of the used drive.

Notes:

The Clockout feature is available only for Drives with firmware V02R27 or superior.

See also *A Appendix Multiplexed I/O Allocations.*

When Clockout function is active the state of B0_OUT0 and B0_OUT1 digital outputs are not affected from Drive_Working_Settings.Disable_Digital_Outputs_FW_Handling and Drive_Working_Setting_Extended.Disable_Fault_Output settings.

When the Clockout function is active, the B0_OUT0 and B0_OUT1 outputs are driven by Clockout_Generator so write into object B0_Digital_Outputs has no effect on B0_OUT0 and B0_OUT1. Read B0_Digital_Outputs object returns state of digital outputs of Clockout_Generator.

Name: Counter_Config[0]
Address: 2210H
CANopen Index.Sub: 2210.1H
Type: WORD
Access: r
Unit: --
Range: 0 = quadrature mode

Default Value: 0
Store Supported: Yes

Description: This register is used to set the configuration of hardware counter 0.
See *A Appendix*.

Notes:

Name: Counter_Config[1]
Address: 2211H
CANopen Index.Sub: 2210.2H
Type: WORD
Access: rw
Unit: --
Range: 0 ÷ 2
0 = quadrature mode
1 = up mode
2 = up/down mode

Default Value: 2
Store Supported: Yes

Description: This register is used to set the configuration of hardware counter 1.
See *A Appendix*.

Notes:

Name: Current_Actual_Value
Address: 1008H
CANopen Index.Sub: 6078.0H
Type: WORD
Access: r
Unit: mA
Range: 0 ÷ (max drive current)
Default Value: 0
Store Supported: No

Description: This object contains the current value of the current supplied to the motor.

Notes:

Name: Dips
Address: 100DH
CANopen Index.Sub: 2004.0H
Type: WORD
Access: r
Unit: --
Range: 0 ÷ 4095 (0FFFH)
Default Value: --
Store Supported: No

Description: This object contains the current status of drive's dips switches (only for drives fitted with dips switches).

Notes:

Name: Drive_Configuration_Code
Address: 40BAH
CANopen Index.Sub: 4004.7H
Type: WORD
Access: r
Unit: --
Range: --
Default Value: 0x90H (where x depends on the drive's fitted fieldbus)
Store Supported: --

Description: It contains the information about the configuration code (firmware type) stored in the drive. The configuration of the e3PLC firmware is 0490H for the Modbus RTU versions, 0390H for the CANopen version, 0690H for the EtherCAT versions and 0890H for the Modbus TCP versions.

Notes:

Name: **Direct_Command_Parameter_1**
Address: **4101H**
CANopen Index.Sub: **4001.2H**
Type: UINT8
Access: r/w
Unit: **--**
Range: 00 ÷ FFH
Default Value: 0
Store Supported: No

Description: This object contains the parameters #1 of the Direct_Command object.

Notes: See *Direct_Command_CMD* object for more details.

Name: **Direct_Command_Parameter_2**
Address: **4102H**
CANopen Index.Sub: **4001.3H**
Type: INT32
Access: r/w
Unit: **--**
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: No

Description: This object contains the parameters #2 of the Direct_Command object.

Notes: See *Direct_Command_CMD* object for more details.

Name: **Direct_Command_Parameter_3**
Address: **4104H**
CANopen Index.Sub: **4001.4H**
Type: UINT16
Access: r/w
Unit: **--**
Range: 0 ÷ 10000
Default Value: 0
Store Supported: No

Description: This object permits to change the Profile_Velocity object to a value that is computed as follows:

$$\left(\frac{Max\_Profile\_Velocity}{10000}\right) * Direct\_Command\_Parameter3$$

If the Direct_Command_Parameter_3 is set to 0 or >10000 the Profile_Velocity object remain unchanged.

Notes: See *Direct_Command_CMD* object for more details.

Name: Direct_Command_CMD
Address: 4105H
CANopen Index.Sub: 4001.1H
Type: UINT8
Access: rw
Unit: --
Range: 0 (Stop motor movement) ÷ 1H (Start motor movement)
Default Value: --
Store Supported: No

Description: This object enable the master to start or stop the Titanio drive's motor. According to the desired command (move or stop) the Parameters #1, #2 assume the following meaning:

STOP Command (Direct_Command_CMD = 0H)

Par #1 Value	Par #1 Description	Par #2 Value	Par #2 Description
0	Stop with no ramp	Not used	Not used
1	Stop with ramp	Not used	Not used
2	Stop with steps	# Steps	Number of stop steps
20	SYNC Stop with no ramp	Not Used	Not used
21	SYNC Stop with ramp	Not used	Not used
22	SYNC Stop with steps	# Steps	Number of stop steps
30	TRIGGER Stop with no ramp	Not used	Not used
31	TRIGGER Stop with ramp	Not used	Not used
32	TRIGGER Stop with steps	# Steps	Number of stop steps

MOVE Command (Direct_Command_CMD = 1H)

Par #1 Value	Par #1 Description	Par #2 Value	Par #2 Description
0	Move free running forward	Not used	Not used
1	Move free running backward	Not used	Not used
2	Move steps forward	# Steps	Number of movement steps
3	Move steps backward	# Steps	Number of movement steps
4	Move to Target position	Position	Position to reach
5	Move absolute steps	# Steps	Number of absolute steps (if steps < 0 move backward, if steps > 0 move forward)
10	Homing Forward Movement	Not used	Not used
11	Homing Backward Movement	Not used	Not used
16	Homing Forward Torque Movement	Not used	Not used
17	Homing Backward Torque Movement	Not used	Not used
20	Move free running forward with SYNC start	Not used	Not used
21	Move free running backward with SYNC start	Not used	Not used
22	Move steps forward with SYNC start	# Steps	Number of movement steps
23	Move steps backward with SYNC start	# Steps	Number of movement steps
24	Move to Target position with SYNC start	Position	Position to reach
25	Move absolute steps with SYNC start	# Steps	Number of absolute steps (if steps < 0 move backward, if steps > 0 move forward)
30	Move free running forward with TRIGGER start	Not used	Not used
31	Move free running backward with TRIGGER start	Not used	Not used
32	Move steps forward with TRIGGER start	# Steps	Number of movement steps
33	Move steps backward with TRIGGER start	# Steps	Number of movement steps
34	Move to Target position with TRIGGER start	Position	Position to reach
35	Move absolute steps with TRIGGER start	# Steps	Number of absolute steps (if steps < 0 move backward, if steps > 0 move forward)
127	Feedback Sensor Calibration mode	Start / Stop procedure	0 → 1 : Start Feedback Sensor Calibration 1 → 0 : Stop Feedback Sensor Calibration

Notes: The Parameters objects (*Direct_Command_Parameter_x*) must be configured first to write this object. If a parameter # is not used for the desired command it is not necessary to configure it. For more details about motor movement refers to §*8.3*.

Name: Drive_Inputs_Level
Address: 2200H
CANopen Index.Sub: 2200.1H
Type: WORD
Access: rw
Unit: --
Range: 0 ÷ FFFFH
Default Value: 0018H
Store Supported: Yes

Description: This register is used to parametrize drive digital inputs working level.

Bit #	Name	Description	Default Value
0	Forward_Limit_Switch_Level	1 = Active high 0 = Active low	0
1	Backward_Limit_Switch_Level	1 = Active high 0 = Active low	0
2	Fast_Stop_Level	1 = Active high 0 = Active low	0
3	Start_Trigger_Level	1 = Active high 0 = Active low	1
4	Stop_Trigger_Level	1 = Active high 0 = Active low	1
5	Reserved	Reserved	0
6	Reserved	Reserved	0
7	Reserved	Reserved	0
8	Reserved	Reserved	0
9	Reserved	Reserved	0
10	Reserved	Reserved	0
11	Start_Trigger_Edge	1 = Input checked on edge 0 = Input checked on level	0
12	Stop_Trigger_Edge	1 = Input checked on edge 0 = Input checked on level	0
13	Reserved	Reserved	0
14	Reserved	Reserved	0
15	Reserved	Reserved	0

Bit Explanation:

Forward_Limit_Switch_Level: When this bit is set (1) the drive will consider the forward limit switch intervention when the corresponding digital input (see *Drive_Inputs_Setting* object) will close.
When this bit is reset (0) the drive will consider the forward limit switch intervention when the corresponding digital input (see *Drive_Inputs_Setting* object) will open.

Backward_Limit_Switch_Level: When this bit is set (1) the drive will consider the backward limit switch intervention when the corresponding digital input (see *Drive_Inputs_Setting* object) will close.
When this bit is reset (0) the drive will consider the backward limit switch intervention when the corresponding digital input (see *Drive_Inputs_Setting* object) will open.

Fast_Stop_Level: When this bit is set (1) the drive will consider the fast stop input intervention when the corresponding digital input (see *Drive_Inputs_Setting* object) will close.
When this bit is reset (0) the drive will consider the fast stop input intervention when the corresponding digital input (see *Drive_Inputs_Setting* object) will open.

Start_Trigger_Level: When this bit is set (1) the drive will consider the start trigger input intervention when the corresponding digital input (see *Drive_Inputs_Setting* object) will close.
When this bit is reset (0) the drive will consider the start trigger input intervention when the corresponding digital input (see *Drive_Inputs_Setting* object) will open.

Stop_Trigger_Level: When this bit is set (1) the drive will consider the stop trigger input intervention when the corresponding digital input (see *Drive_Inputs_Setting* object) will close.
When this bit is reset (0) the drive will consider the stop trigger input intervention when the corresponding digital input (see *Drive_Inputs_Setting* object) will open.

Notes:

Name: Drive_Inputs_Setting
Address: 2204H
CANopen Index.Sub: 2200.3H
Type: DWORD
Access: rw
Unit: --
Range: 0 ÷ FFFFFFFFH
Default Value: 00010210H
Store Supported: Yes

Description: This register is used to parametrize drive digital inputs bank #0 allocation. For any function can be assigned one digital input that have to be specified in the four bits of register concerning that function.

	Bit 31÷28	Bit 27÷24	Bit 23÷20	Bit 19÷16	Bit 15÷12	Bit 11÷8	Bit 7÷4	Bit 0÷3
				Trigger Stop	Trigger Start	Fast Stop	Backward Limit Switch	Forward Limit Switch
Range				**0,1,2,3,5,6,7**	**0,1,2,3,5,6,7**	**0÷7**	**0÷7**	**0÷7**
Default Value	0	0	0	1	0	2	1	0

Notes: See also A Appendix.

Name: Drive_Register
Address: 1202H
CANopen Index.Sub: 4000.3H
Type: WORD
Access: r
Unit: --
Range: 0 ÷ FFFFH
Default Value: --
Store Supported: No

Description: This register contains info about drive motion status and enable the master network node guarding check.

Bit #	Name	Description	Default Value
0	Motor_Running	1 = Drive's motor is running. 0 = Drive's motor is stopped.	0
1	Motor_Direction	0 = Motor direction forward 1 = Motor direction backward	0
2	Motor_Feedback_Error	1 = Drive is in Feedback Error condition 0 = Ok	0
3	Motor_Busy	1 = Motor is busy doing cycle 0 = Cycle completed - idle status	0
4	Motor_Impacted	1 = Motor / Position transducer displacement detected 0 = No displacement detected or checking not enabled	0
5	Motor_Overrun	1 = Reached or faulty limit switch 0 = Ok	0
6	Motor_Limit_Switch_Not_Found	1 = Limit switch not found during homing movement 0 = Ok	0
7	Motor_Movement_Not_Executed	1 = Last movement not executed 0 = Ok	0
8	Drive_Protection	1 = Drive is in protection condition, check Error_Register 0 = Drive is in normal condition	0
9	SYNC_Armed	1 = Drive is waiting for SYNC signal 0 = Ok	0
10	Start_Trigger_Armed	1 = drive is waiting for Start TRIGGER signal 0 = Ok	0
11	Stop_Trigger_Armed	1 = drive is waiting for Stop TRIGGER signal 0 = Ok	0
12	Target_Reached	1 = Target Reached 0 = Target not Reached	0
13	Motor_Standby	1= Motor in standby condition 0= Ok	0
14	Master_Watchdog_Timeouted	1 = Master watchdog timeouted 0 = Master watchdog ok or disabled	0
15	Drive_Watchdog	Drive watchdog bit	0

Bit Explanation:

Motor_Running: This bit comes set (1) when the motor is running. If the electric gear is enabled, and the motor is at a standstill because the external frequency is null but the movement is not yet over, the bit is equal to 1.
This bit comes reset (0) when the motor stops.

Motor_Direction: This bit comes set (1) when the motor direction is backward.
This bit comes reset (0) when the motor direction is forward.
If the motor is at a standstill, the bit value corresponds to the direction of the last motor step executed.

Motor_Feedback_Error: This bit comes set (1) when the drive is in feedback error condition.
This bit comes reset (0) when the feedback error condition is cleared or feedback feature is not enabled (see §8.2 for details).

Motor_Busy: This bit comes set (1) when the motion cycle is in progress.
This bit comes reset (0) when the motion cycle has ended.
This bit is basically the same as Motor_Running except for Homing movements.

Motor_Impacted: This bit comes set (1) when the object *Impact_Actual_Displacement* is higher than *Impact_Max_Displacement* and the bit *Impact_Motor_Check* in *Drive_Working_Settings* object is set. The bit remain set until the bit *Master_Alarm_Reset* of *Master_Register* object comes set.
This bit comes reset (0) when either the object *Impact_Actual_Displacement* is less than *Impact_Max_Displacement* object or the bit *Impact_Motor_Check* in *Drive_Working_Settings* object is not set.

Motor_Overrun: This bit comes set (1) when a limit switch intervention has been detected and one of the bits *xxx_Limit_Switch_Check* of *Drive_Working_Settings* object is set. The bit remain set until the bit *Master_Alarm_Reset* of *Master_Registe* object comes set.
This bit comes reset (0) when no limit switches intervention has been detected or both the bits *xxx_Limit_Switch_Check* of *Drive_Working_Settings object* are not set.

Motor_Limit_Switch_Not_Found: This bit comes set (1) when during a homing movement the limit switch has not been found within *Homing_Overrun* object steps. The bit remain set until the bit *Master_Alarm_Reset* of *Master_Registe* object comes set.
This bit comes reset (0) when during a homing movement the limit switch has been found withing *Homing_Overrun* object steps.

Motor_Movement_Not_Executed: This bit comes set (1) when the drive cannot perform a movement because either a protection is active (see *Error_Register* object) or the bit *Master_Motor_Enable* of *Master_Register* object is reset. The bit remain set until the bit *Master_Alarm_Reset* of *Master_Register* object comes set.
This bit comes reset (0) while drive can perform movements.

Drive_Protection: This bit comes set (1) when the *Error_Register* object is <> 0. The bit remain set until the bit *Master_Alarm_Reset* of *Master_Register* object comes set.
This bit comes reset (0) while the *Error_Register* object is = 0.

SYNC_Armed: This bit comes set (1) when the drive have a pending MOVE/STOP with SYNC command and the *Motor_SYNC* object has not yet been set.
This bit comes reset (0) when the drive have no pending MOVE/STOP with SYNC command.

Start_Trigger_Armed: This bit comes set (1) when the drive have a pending MOVE with Trigger command and the start trigger (see *Drive_Inputs_Setting* object) has not yet been detected.
This bit comes reset (0) when the drive have no pending MOVE with Trigger command.

Stop_Trigger_Armed: This bit comes set (1) when the drive have a pending STOP with Trigger command and the stop trigger (see *Drive_Inputs_Setting* object) has not yet been detected.
This bit comes reset (0) when the drive have no pending STOP with Trigger command.

Target_Reached: This bit comes set (1) when the drive's current position is inside the defined position window (see *Position_Window* and *Position_Window_Time* objects)
This bit comes reset (0) when the drive has not yet reached the target position.

Motor_Standby: This bit comes set (1) when the motor is running and the electric gear feature is enabled, but the motor is at a standstill because the external frequency is null.
This bit comes reset (0) when the external frequency is not null or the electric gear feature is disabled.

Master_Watchdog_Timeouted: This bit comes set (1) when the master did not wrote the *Master_Register* object within *Master_Watchdog_Timeout*. The bit remain set until the bit *Master_Alarm_Reset* of *Master_Register* object comes set.
This bit comes reset (0) while the master refresh the *Master_Register* object within *Master_Watchdog_Timeout*.

Drive_Watchdog: This bit toggles (0/1) every *Drive_Watchdog_Time*.

Notes:

Name: Drive_Register_Extended
Address: 1206H
CANopen Index.Sub: 4000.AH
Type: WORD
Access: ro
Unit: --
Range: --
Default Value: --
Store Supported: No

Description: This object defines the additional functionality of the Drive_Register object.

Bit #	Name	Description	Default Value
0	Drive_Not_Ready	1 = Drive not ready 0 = Drive ready	0
1	Parameters_Change_Not_Allowed	1 = Parameters change not allowed 0 = Parameters change allowed	0
2	Store_In_Progress	1 = Store in NVRAM in progress 0 = Store in NVRAM completed	0
3	Stop_Trigger_Fault	1 = Stop Trigger Fault (not detected) 0 = Ok	0
4	Torque_Enable_Input	1 = Torque Enable Input is on 0 = Torque Enable Input is off	0
5	Fast_Stop_Active	1 = Fast Stop Active 0 = Fast Stop not Active	0
6	Motor_Stall_detection_enabled	1 = Motor Stall detection feature is enabled 0 = Motor Stall detection feature is disabled	0
7	Reserved	Reserved	0
8	Reserved	Reserved	0
9	Reserved	Reserved	0
10	Reserved	Reserved	0
11	Reserved	Reserved	0
12	Reserved	Reserved	0
13	Reserved	Reserved	0
14	Reserved	Reserved	0
15	Reserved	Reserved	0

Bit Explanation:

Drive_Not_Ready: This bit comes set (1) in the following situations: during emergency condition or when the bit #0 of *Master_Register* is 0 (motor disabled).
This bit comes reset (0) when the drive is ready to move the motor.

Parameters_Change_Not Allowed:
This bit comes set (1) when the drive is in initialization process and parameters could not be changed.
This bit comes reset (0) when the drive is ready to accept parameters change.

Store_In_Progress: This bit comes set (1) when a store in NVRAM is in progress. The drive should not be switched off or reset during this procedure.
This bit comes reset (0) when the drive is not storing data in NVRAM.

Stop_Trigger_Fault: This bit comes set (1) when a STOP with trigger input has been issued but the movement ended without detecting the trigger input.
This bit comes reset (0) when a new STOP command is issued.

Torque_Enable_Input: This bit comes set (1) when the Torque Enable Input is ON (24Vdc present) (check current drive version for availability of Torque Enable Input)
This bit comes reset (0) when the Torque Enable Input is OFF (0Vdc present) or when not available on current drive version.

Fast_Stop_Active: This bit comes set (1) when the fast stop input is active.
This bit comes reset (0) when the fast stop input is inactive or not used (bit *Fast_Stop_From_Input* of *Drive_Working_Setting* object = 0) .

Motor_Stall_detection_enabled [(1)] **:**
This bit comes set (1) when the motor stall detection feature is active.
This bit comes reset (0) when the motor stall detection feature is not active or not used.

Notes:

- [(1)] This bit is available only with firmware version V02r74 or superior (see §*9.4* for more details about 'Motor Stall detection' feature).

Name: Drive_Temperature_Actual_Value
Address: 100CH
CANopen Index.Sub: 200A.0H
Type: WORD
Access: r
Unit: °C
Range: 0 ÷ Maximum Drive Temperature
Default Value: --
Store Supported: No

Description: This object contains the current drive's temperature.

Notes:

Name: Drive_Type
Address: 40BCH
CANopen Index.Sub: 4004.6H
Type: WORD
Access: r
Unit: --
Range: --
Default Value: --
Store Supported: --

Description: It contains the information about the drive type and hardware version. The MSB contains the drive board family, while the LSB contains the hardware version.

MSB Value	Description
30H	CSMD1 Board
31H	TWD00 Board
32H	IMD00 Board
33H	ILD00 Board
34H	ISD00 Board
35H	IMD01 Board
36H	ISX00 Board
37H	IMD02 Board
38H	EMD00 Board
39H	EM00CM Module
3AH	DDC02 Board
3BH	ETH02 Board
3CH	ETH01 Board
3DH	IMD04 Board
3EH	IMD03 Board
3FH	IMD05 Board
40H	EMD01 Board
41H	EM01CM Module
42H	IMD07 Board
43H	ETH03 Board
44H	IBD00 Board
45H	IMD08 Board

Notes:

Name: **Drive_Voltage_Actual_Value**
Address: **100BH**
CANopen Index.Sub: **2009.0H**
Type: WORD
Access: r
Unit: Volts
Range: Minimum Drive Voltage ÷ Maximum Drive Voltage
Default Value: --
Store Supported: No

Description: This object contains the current voltage that powers the drive.

Notes:

Name: **Drive_Watchdog_Time**
Address: **1204H**
CANopen Index.Sub: **4000.5H**
Type: WORD
Access: r/w
Unit: milliseconds
Range: 1 ÷ 65535
Default Value: 500
Store Supported: Yes

Description: This object set the frequency of the drive's watchdog toggle (bit #15 of *Drive_Register* object).

Notes:

Name: Drive_Working_Settings
Address: 2202H
CANopen Index.Sub: 2200.2H
Type: WORD
Access: rw
Unit: --
Range: 0 ÷ FFFFH
Default Value: 0840H
Store Supported: Yes

Description: This register is used to parametrize drive working modalities.

Bit #	Name	Description	Default Value
0	Forward_Limit_Switch_Check	1 = Forward limit switch check enabled 0 = Forward limit switch check disabled	0
1	Backward_Limit_Switch_Check	1 = Backward limit switch check enabled 0 = Backward limit switch check disabled	0
2	Fast_Stop_From_Input	1 = Fast Stop Input enabled 0 = Fast Stop Input disabled	0
3	Limit_Switches_Motor_Action	0= instant stop motor; 1= stop motor with ramp	0
4	Feedback_Motor_Check	1 = Enable Feedback Feature 0 = Disable Feedback Feature	0
5	Impact_Motor_Check	1 = Check of motor impact enabled 0 = Check of motor impact disabled	0
6	Automatic_Motor_Current_Reduction	1 = Set min_current when motor stops 0 = disable	1
7	Disable_Digital_Outputs_FW_Handling	1 = Disable Digital_Outputs handling by firmware (All DO user free) 0 = Enable Digital_Outputs handling by firmware (Not all DO user free)	0
8	Fast_Stop_Motor_Action	0= instant stop motor; 1= stop motor with ramp	0
9	Fast_Stop_Power_Action	0= nothing ; 1=open transistor	0
10	Motor_Rotation_Direction	1 = counter clockwise rotation when motor move forward 0 = clockwise rotation when motor move forward	0
11	Motor_Gear_Init	1 = Restart external reference acquisition at each motor start 0 = Continuous external reference acquisition	1
12	Motor_Gear	1 = Electric Gear Enabled 0 = Electric Gear Disabled	0
13	Feedback_Error_Motor_Action	1 = Stop Motor when feedback error 0 = Don't stop the motor when feedback error	0
14	Electric_Gear_Modality	1 = Synchronous Electric gear 0 = Asynchronous Electric gear	0
15	Master_Watchdog_Timeout_Action	1 = If master watchdog timeouts stop current motor movement 0 = If master watchdog timeouts do not anything else except setting the corresponding Drive_Register object bit	0

Bit Explanation:

Forward_Limit_Switch_Check: When this bit is set (1) the drive will check continuously for the intervention of forward limit switch (see *Drive_Inputs_Setting* and *Drive_Inputs_Level* objects). If the limit switch intervene and the motor is running forward, the motor will stop without deceleration ramp and the *Motor_Overrun* bit of *Drive_Register* object will be set.
When this bit is reset (0) the drive will not check for the forward limit switch at all.

Backward_Limit_Switch_Check: When this bit is set (1) the drive will check continuously for the intervention of backward limit switch (see *Drive_Inputs_Setting and Drive_Inputs_Level* objects). If the limit switch intervene and the motor is running backward, the motor will stop without deceleration ramp and the *Motor_Overrun* bit of *Drive_Register* object will be set.
When this bit is reset (0) the drive will not check for the backward limit switch at all.

Fast_Stop_From_Input: When this bit is set (1) the drive will check continuously for the intervention of fast stop input (see *Drive_Inputs_Setting and Drive_Inputs_Level* objects). If the fast stop intervene and the motor is running, the motor will stop (according to bit *Fast_Stop_Motor_Action*) and the *Fast_Stop_Active* bit of *Drive_Register* object will be set.
When this bit is reset (0) the drive will not check for the fast stop input at all.

Limit_Switches_Motor_Action: When this bit is set (1) the drive will stop the motor with a deceleration ramp when a limit switch intervene.
When this bit is reset (0) the drive will stop immediately the motor when a limit switch intervene.

Feedback_Motor_Check: When this bit is set (1) the drive will enable the feedback feature to close the motion control loop. This will optimize the motor efficiency (See §*8.2* for details).
When this bit is reset (0) the feedback feature is disabled.

The 'Feedack' feature (bit4 of *Drive_Working_Settings* object*)* and 'Motor Stall detection' feature (bit13 of *Drive_Working_Settings_Extended object)* cannot be both active at the same time otherwise an alarm is issued (bit5 of *Error_Register* object and bit14 of *Feedback_Status*). See Note(1).

Impact_Motor_Check: When this bit is set (1) the drive will check continuously for the displacement (*Impact_Actual_Displacement* object) between the motor position (*Position_Actual_Value* object) and the impact source position. If the difference between these positions is greater than *Impact_Max_Displacement* object, the motor will be stopped and *Motor_Impacted* bit of *Drive_Register* object is set.
When this bit is reset (0) the impact check feature is disabled.

Automatic_Motor_Current_Reduction: When this bit is set (1) the drive will automatically set the motor current equal to *Min_Current* object.
When this bit is reset (0) the drive will continue to set the motor current equal to *Max_Current* object even though the motor is stopped.

Disable_Digital_Outputs_FW_Handling: When this bit is set (1) the firmware will not set/reset automatically the digital outputs assigned to Ready/Busy function. Whole digital outputs are available to the user (except Fault output).
When this bit is reset (0) the firmware will set/reset automatically the digital outputs assigned to Ready/Busy function. The remaining digital outputs are available to the user (except the Fault output). See *A Appendix*.

Fast_Stop_Motor_Action: When this bit is set (1) the drive will stop the motor with a deceleration ramp when the emergency fast stop intervene.
When this bit is reset (0) the drive will stop immediately the motor when the emergency fast stop intervene.

Fast_Stop_Power_Action: When this bit is set (1) the drive will open the transistors (no current, no voltage powered to the motor) if fast stop happened.
When this bit is reset (0) the drive will continue to power the motor even after a fast stop command.

Motor_Rotation_Direction: When this bit is set (1) the motor will rotate counter clockwise when the motion direction is forward (*Position_Actual_Value* object increase) and clockwise when the motion direction is backward (*Position_Actual_Value* object decrease).
When this bit is reset (0) the drive motor will rotate clockwise when the motion direction is forward (*Position_Actual_Value* object increase) and counter clockwise when the motion direction is backward (*Position_Actual_Value* object decrease).

Motor_Gear_Init: It selects the initialization options for the electric gear.
If the bit is set to 1, when the motor is running with the electric gear function enabled, acquisition of external pulses is re-initialized at every motor start.
If the bit is set to 0, when the motor runs with the electric gear function enabled, it will also recover the pulses acquired while the motor was at a standstill (up to 32768 pulses).

Motor_Gear: When this bit is set (1) the electric gear modality is enabled.
When this bit is reset (0) the electric gear modality is disabled.

Feedback_Error_Motor_Action: When this bit is set (1) the drive will stop any movement in progress if a feedback error is detected. The bit Motor_Feedback_Error of *Drive_Register* object is set. See §*8.2* for details.
When this bit is reset (0) the drive will not take any action, if a feedback error is detected, except setting the bit Motor_Feedback_Error of *Drive_Register* object. See § *8.2* for details.

Electric_Gear_Modality: When this bit is set (1) the Synchronous electric gear modality is enabled.
When this bit is reset (0) the Asynchronous electric gear modality is enabled.

Master_Watchdog_Timeout_Action: When this bit is set (1) the drive will stop any movement in progress if the bit *Master_Watchdog* of *Master_Register* object has not been refreshed within *Master_Watchdog_Timeout* object time. The bit *Master_Watchdog_Timeouted* of *Drive_Register* object will also be set.
When this bit is reset (0) the drive will not take any action, if the bit *Master_Watchdog* of *Master_Register* object has not been refreshed within *Master_Watchdog_Timeout* object time, except setting the bit *Master_Watchdog_Timeouted* of *Drive_Register* object.

Notes:

- See also *Drive_Working_Setting_Extended object.*

- *(1) See §**9.4** for more details about 'Motor Stall detection' feature.*

Name: Drive_Working_Settings_Extended
Address: 2203H
CANopen Index.Sub: 2200.11H
Type: WORD
Access: rw
Unit: --
Range: 0 ÷ FFFFH
Default Value: 0H
Store Supported: Yes

Description: This register is used to parametrize drive working modalities.

Bit #	Name	Description	Default Value
0	Clockout_Init	1 = Reset counter at the end of each motor movement 0 = Keep motor position at the end of each motor movement	1
1	Reserved	Reserved	0
2	Disable_Fault_Output	1 = Fault Output (B0_Out0) is disabled and free for user application 0 = Fault Output (B0_Out0) is enabled	0
3	CANopen_Auto_Operational	1 = The drive will go automatically in Operational state at switch on 0 = The drive will remain in Pre-Operational state at switch on	1
4	Reserved	Reserved	0
5	Disable_Auto_Motor_Enable	1 = The motor will remain disabled at switch-on 0 = The motor will be automatically enabled at switch on	0
6	Encoder0_Rotation_Direction	1 = The encoder #0 counting is reversed 0 = The encoder #0 counting is not reversed	0
7	Encoder1_Rotation_Direction	1 = The encoder #1 counting is reversed 0 = The encoder #1 counting is not reversed	0
8	Reserved	Reserved	0
9	RL_Motor_Detection	1 = Enable the Motor Parameters (RL) Detection 0 = Disable the Motor Parameters (RL) Detection	1
10	Enable_I2T_Protection	1 = Enable the I2T Protection 0 = Disable the I2T Protection	0
11	Reserved	Reserved	0
12	Reserved	Reserved	0
13	Motor_Stall_detection	1 = Enable 'Motor Stall Detection' feature 0 = Disable 'Motor Stall Detection' feature	0
14	Reserved	Reserved	0
15	Reserved	Reserved	0

Bit Explanation:

Clockout_Init: When this bit is set (1) the clockout generation restart from begin at each motor movement, so the first clock out pulse will be out after Clockout_Prescaler Motor Clocks. When this bit is reset (0) the clockout generation continue from the last motor position at each next motor movement.
The clockout generation is enabled with the object Clockout_Prescaler (see *Clockout_Prescaler* object).

Disable_Fault_Output: When this bit is set (1) the B0_Out0 digital output is free for the user application
When this bit is reset (0) the B0_Out0 is automatically handled by the drive's firmware as Fault Output.

CANopen_Auto_Operational: When this bit is set (1) the drive will go automatically in Operational state at switch-on (CANopen drives).
When this bit is reset (0) the drive will remain in Pre-Operational state at switch-on and an external NMT Master should send the command to enter in Operational state.

Disable_Auto_Motor_Enable: When this bit is set (1) the motor will remain disabled at switch on (*Master_Register* Bit #0 = 0)
When this bit is reset (0) the motor will be automatically enabled at switch on (*Master_Register* bit #0 = 1).

EncoderX_Rotation_Direction: When this bit is set (1) the Encoder #X counting is reversed.
When this bit is reset (0) the Encoder #X counting is not reversed.

RL_Motor_Detection : When this bit is set (1) the Motor Parameters (Resistance and Inductance) are detected automatically at switch-on (*Nominal_Current* object need to be set).
When this bit is reset (0) the Motor Parameters (Resistance and inductance) are not detected automatically and must be set on objects *Motor_R* and *Motor_L*.

Enable_I2T_Protection : When this bit is set (1) the I2T Protection is enabled.
When this bit is reset (0) the I2T Protection is disabled.

Motor_Stall_detection [(1)] : When this bit is set (1) the 'Motor Stall detection' feature is enabled.
When this bit is reset (0) the 'Motor Stall detection' feature is disabled.

The 'Feedack' feature (bit4 of *Drive_Working_Settings* object*)* and 'Motor Stall detection' feature (bit13 of *Drive_Working_Settings_Extended object)* cannot be both active at the same time otherwise an alarm is issued (bit5 of *Error_Register* object and bit14 of *Feedback_Status*) .

Notes:

[(1)] The 'Motor Stall detection' feature is available only with firmware version V02r74 or superior.

Name: eePLC_Emergency_Inserted
Address: 59A0H
CANopen Index.Sub: 59A0.0H
Type: WORD
Access: r
Unit: --
Range: --
Default Value: --
Store Supported: No

Description: This object contains the state of the internal emergency of e3PLC firmware.

Bit #	Name	Description	Default value
0	Protection_Drive_Current	1 = Drive is in current protection condition 0 = Ok	0
1	Protection_Open_Phase	1 = Drive is in open phase protection condition 0 = Ok	0
2	Protection_Drive_Voltage	1 = Drive is in voltage protection condition 0 = Ok	0
3	Protection_Drive_Thermal	1 = Drive is in thermal protection condition 0 = Ok	0
4	eePLC_User_Program	1 = eePLC User Program Error 0 = Ok	0
5	Missing_Torque_Enable	1 = Torque Enable Input Missing (only for drive with Torque Enable Input) 0 = Ok	0
6	Watchdog_Occurred	1 = Watchdog Occurred 0 = Ok	0
7	EEprom_Fail	1 = EEprom Fail 0 = Ok	0
8	Internal_SW_Error	1 = Drive is in internal software error condition. 0 = Ok	0
9	Feature_Unavailable	1= eePLC Feature Unavailable 0 = Ok	0
10	Feedback_Error	1 = Motor Feedback Error 0 = Ok	0
11	Missing Calibration	1 = Missing Calibration 0 = Ok	0
12	Protection_EEprom_Write_Overrun	1 = EEprom Write Overrun. 0 = Ok.	0
13	eePLC_Software_Protection	1 = eePLC software protection 0 = Ok	0
14	Protection_Current_Regulation	1 = Drive is in current regulation out of range protection condition Verify Motor Currents settings 0 = Ok	0
15	Protection_Open_Transistor	1 = Drive is in open transistor condition. 0 = Ok	0

Notes:

Name: eePLC_User_Free_Timer[0÷3]
Address: 5A00H,5A01H,5A02H,5A03H
CANopen Index.Sub: 5A00H,5A01H,5A02H,5A03H
Type: UWORD
Access: rw
Unit: ms
Range: 0 ÷ 65535
Default Value: 0
Store Supported: No

Description: This objects are user free timer that can be used in the eePLC application. They work in countdown mode.

Notes:

Name: **eePLC_User_Settings**
Address: **5986H**
CANopen Index.Sub: **5986.0H**
Type: WORD
Access: rw
Unit: **--**
Range: **--**
Default Value: --
Store Supported: No

Description: This object is used to activate some functionality during the application execution.

Bit #	Name	Description	Default Value
0	eePLC_Software_Protection	1 = the drive will be in emergency condition and a P+7 is showed on drive display. See § *5.0* and *B Appendix* 0 = no action	0
1	Reserved	Reserved	0
2	Reserved	Reserved	0
3	Reserved	Reserved	0
4	Reserved	Reserved	0
5	Reserved	Reserved	0
6	Reserved	Reserved	0
7	Reserved	Reserved	0
8	Reserved	Reserved	0
9	Reserved	Reserved	0
10	Reserved	Reserved	0
11	Reserved	Reserved	0
12	Reserved	Reserved	0
13	Reserved	Reserved	0
14	Reserved	Reserved	0
15	Reserved	Reserved	0

Notes:

Name: **eePLC_Warning_Inserted**
Address: **59A2H**
CANopen Index.Sub: **59A2.0H**
Type: WORD
Access: ro
Unit: **--**
Range: **--**
Default Value: --
Store Supported: No

Description: This object contains the state of the internal warnings of the e3PLC Firmware.

Bit #	Name	Description	Default value
0	Warning_Voltage	1 = Drive voltage is near limit 0 = Ok	0
1	Warning_Temperature	1 = Drive temperature is near limit 0 = Ok	0
2	Warning_EEprom_Near_Write_Overrun	1 = EEprom near Write Overrun. 0 = Ok.	0
3	Warning_EEprom_Near_EOL	1 = EEprom near End of Life 0 = Ok.	0
4	Reserved	Reserved	0
5	Reserved	Reserved	0
6	Reserved	Reserved	0
7	Reserved	Reserved	0
8	Missing_Nominal_Current	1 = *Nominal Current* not set and Motor_RL_Detection enabled 0 = Ok	0
9	Reserved	Reserved	0
10	Reserved	Reserved	0
11	Reserved	Reserved	0
12	Reserved	Reserved	0
13	Reserved	Reserved	0
14	Reserved	Reserved	0
15	Reserved	Reserved	0

Notes:

Name: **Electric_Gear_Ext_Speed_Ref**
Address: **1034H**
CANopen Index.Sub: **2024.0H**
Type: DWORD
Access: r/w
Unit: **--**
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: No

Description: It defines the speed reference when the electric gear speed type is enabled. The motor reference speed will be the result of the multiplication between this object and the Motor gear ratio object.

Notes: See *Gear_Ratio_Motor_Revs* and *Gear_Ratio_Shaft_Revs* objects.

Name: **Encoder_Actual_Value[0÷1]**
Address: **1002H,1004H**
CANopen Index.Sub: **2007.0H,2008.0H**
Type: DWORD
Access: r/w
Unit: Encoder steps
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: No

Description: This object contains the encoder # 0 and #1 current position. The position can be cleared writing a 0 in the object.

Notes: The number of encoders available depends on the version of the drive currently in use.

Name: **Encoder_Frequency[0÷1]**
Address: **2214H,2216H**
CANopen Index.Sub: **2211.1H,2211.2H**
Type: DWORD
Access: r
Unit: Hz
Range: -200000 ÷ 200000
Default Value: 0
Store Supported: No

Description: This object contains the encoder # 0 and #1 frequency.

Notes: The number of encoders available depends on the version of the drive currently in use.

Name: Error_Register
Address: 1201H
CANopen Index.Sub: 4000.2H
Type: WORD
Access: ro
Unit: --
Range: --
Default Value: --
Store Supported: No

Description: This object contains the state of the various drive errors.

Bit #	Name	Description	Default value
0	Protection_Drive_Thermal	1 = Drive is in thermal protection condition 0 = Ok	0
1	Protection_Drive_Voltage	1 = Drive is in voltage protection condition 0 = Ok	0
2	Protection_Drive_Current	1 = Drive is in current protection condition 0 = Ok	0
3	Protection_Open_Phase	1 = Drive is in open phase protection condition 0 = Ok	0
4	Missing_Torque_Enable	1 = Torque Enable Input Missing (only for drive with Torque Enable Input) 0 = Ok	0
5	Feedback_Error (1)	1 = Motor Feedback Error 0 = Ok	0
6	Protection_Current_Regulation	1 = Drive is in current regulation out of range protection condition. Verify Motor Currents 0 = Ok	0
7	Protection_Open_Transistor	1 = Drive is in open transistor condition 0 = Ok	0
8	eePLC_User_Program	1 = eePLC User Program Error 0 = Ok	0
9	Internal_SW_Error	1 = Drive is in internal software error condition 0 = Ok	0
10	eePLC_Software_Protection	1 = eePLC software protection 0 = Ok	0
11	Missing_Calibration	1 = Missing Calibration 0 = Ok	0
12	Watchdog_Occurred	1 = Internal Watchdog Occurred 0 = Ok	0
13	Eeprom_Fail	1 = Eeprom Failure 0 = Ok	0
14	I2T_Protection	1 = I2T Protection 0 = Ok	0
15	Feature_Unavailable	1 = Feature unavailable 0 = Ok	0

Notes:

(1) See *Feedback_Status* object for more details about type of Feedback Error.

Name: **Firmware_Checksum**
Address: **4098H**
CANopen Index.Sub: **4004.1H**
Type: WORD
Access: r
Unit: **--**
Range: 0000H ÷ FFFFH
Default Value: **--**
Store Supported: **--**

Description: It contains the checksum of the current firmware.

Notes:

Name: **Firmware_Version**
Address: **40BBH**
CANopen Index.Sub: **4004.0H**
Type: WORD
Access: r
Unit: **--**
Range: 0000H ÷ FFFFH
Default Value: **--**
Store Supported: **--**

Description: It contains the current version of the firmware. The MSB contains the version, while the LSB contains the release (Example: value 0105H means V01r05).

Notes:

Name: **Feedback_Actual_Position_Error**
Address: **281AH**
CANopen Index.Sub: **2230.2H**
Type: Integer32
Access: ro
Unit: IU (one motor turn = 65536 IU)
Range: Integer32
Default Value: --
Store Supported: No

Description: This object returns the actual position displacement between MOTOR_REF_Position and the MOTOR_ACTUAL_Position. The value is refreshed if the feedback is actived and the unit are IU.

Feedback_Actual_Position_Error = MOTOR_REF_Position – MOTOR_ACTUAL_Position

Notes: See §*8.2* for details.

Name: **Feedback_Actual_Velocity_Error**
Address: **2830H**
CANopen Index.Sub: **2230.1FH**
Type: Integer32
Access: ro
Unit: IU/sec (one motor turn = 65536 IU)
Range: Integer32
Default Value: --
Store Supported: No

Description: This object returns the actual velocity displacement between the MOTOR_REF_Speed and the MOTOR_ACTUAL_Speed. The value is refreshed if the feedback is actived and the unit are IU/sec.

Feedback_Actual_Velocity_Error = MOTOR_REF_Velocity – MOTOR_ACTUAL_Speed

Notes: See §*8.2* for details.

Name: **Feedback_Boost_Current**
Address: **2816H**
CANopen Index.Sub: **2230.12H**
Type: Unsigned16
Access: rw
Unit: mA
Range: 0 ÷ (max drive current)
Default Value: 1500
Store Supported: Yes

Description: This value is used for Feedback feature and define Maximum current supplied to the motor. The parameter id used also to limit the motor torque to a defined value.

Notes: See §*8.2* for details.

Name: **Feedback_Calibration_Current**
Address: **282AH**
CANopen Index.Sub: **2230.1CH**
Type: Unsigned16
Access: rw
Unit: mA
Range: 0 ÷ (max drive current)
Default Value: 1000
Store Supported: Yes

Description: This value is used for Feedback feature and define current used during the calibration procedure. This value has to be enough to permit the motor to move in the point of maximun torque, in general it should be set to the maximun current that the motor can accept.

Notes: See § *8.2* for details.

Name: **Feedback_Calibration_Phase**
Address: **2824H**
CANopen Index.Sub: **2230.19H**
Type: Unsigned16
Access: ro
Unit: --
Range: 0÷50
Default Value: 0
Store Supported: No

Description: This object return the feedback calibration phase.

Value	Description
0	Feedback disactive
1÷49	Feedback calibration in execution
65535	Feedback actived

Notes: See §*8.2* for details.

Name: **Feedback_Calibration_Speed**
Address: **2826H**
CANopen Index.Sub: **2230.1AH**
Type: Unsigned16
Access: rw
Unit: 0.01 rpm
Range: 0 ÷ 10000
Default Value: 500 (5 rpm)
Store Supported: Yes

Description: This value is used for Feedback feature and define motor speed used during the calibration procedure. The value has to be set at low speed value generally 5 rpm is a good choice.

Notes: See §*8.2* for details.

Name: Feedback_Current_Filter_Time
Address: 2817H
CANopen Index.Sub: 2230.0EH
Type: Unsigned16
Access: rw
Unit: us
Range: 0 ÷ 10000
Default Value: 100 (100 us)
Store Supported: Yes

Description: This value is used for Feedback feature and define the filter on output current(Iq_out).

In general values from 100 to 500 are a good choice. If it's necessary a quicker velocity reaction decrease the value, if it is necessary to reduce the motor noise at very low speed increase the value.

Notes: See §*8.2* for details.

Name: Feedback_Encoder_Filter_Time
Address: 282CH
CANopen Index.Sub: 2230.1DH
Type: Unsigned16
Access: rw
Unit: us
Range: 0 or 50 ÷ 20000
Default Value: 900 (0.9 ms)
Store Supported: Yes

Description: This value is used for Feedback feature and define the filter on encoder speed detection.

Value	Description
0	Automatic filter activation
50÷20000	Valid filter value from 50 us to 20000 us

In general value from 500 to 2000 are a good choice. If it's necessary a quicker velocity reaction decrease the value, if it is necessary to reduce the motor noise at very low speed increase the value.

Notes: See §*8.2* for details.

Name: Feedback_Iq_min
Address: 2813H
CANopen Index.Sub: 2230.FH
Type: Unsigned16
Access: rw
Unit: mA
Range: Unsigned16
Default Value: 500
Store Supported: Yes

Description: This value is used for Feedback feature and define the minimum amplitude of the phase's current. Can be considered also as minimun current applied at the motor.

Notes: See §*8.2* for details.

Name: **Feedback_Kalfas**
Address: **2819H**
CANopen Index.Sub: **2230.DH**
Type: Unsigned16
Access: rw
Unit: --
Range: Unsigned16
Default Value: 0
Store Supported: Yes

Description: This value is used for Feedback feature. It's used to change the advance angle depending by speed. For speed higher of 1500 rpm set the value 60000.

Notes: See §*8.2* for details.

Name: **Feedback_Ki**
Address: **2804H**
CANopen Index.Sub: **2230.BH**
Type: Unsigned32
Access: rw
Unit: --
Range: Unsigned32
Default Value: 1000
Store Supported: Yes

Description: This value is used for Feedback feature and define the intergral Gain of PID regulator.
The value has to be adjust to reduce the position error (Feedback_Actual_Velocity_Error) or velocity error (Feedback_Actual_Velocity_Error) at the end of acceleration or deceleration.
Pay attention that a not coherent value (big value) can introduce motor's vibration.

Notes: See §*8.2* for details.

Name: **Feedback_Ki_Limit**
Address: **2814H**
CANopen Index.Sub: **2230.CH**
Type: Unsigned16
Access: rw
Unit: mA
Range: 0 ÷ (max drive current)
Default Value: 1500
Store Supported: Yes

Description: This value is used for Feedback feature to limit PID's integral(anti-windup)

Notes: See §*8.2* for details.

Name: **Feedback_Kfbw_Acc**
Address: **280CH**
CANopen Index.Sub: **2230.15H**
Type: Unsigned32
Access: rw
Unit: --
Range: Unsigned32
Default Value: 14000
Store Supported: Yes

Description: This value is used for Feedback feature and defines the gain used to calculate the Feedforward current, while the motor is accelerating backward. The value has to be adjust to reduce the Feedback_Actual_Position_Error and the Feedback_Actual_Velocity_Error during acceleration in backward direction
Pay attention that a not coherent value (big value) can introduce motor's vibration.

Notes: See §*8.2* for details.

Name: **Feedback_Kfbw_Dec**
Address: **280EH**
CANopen Index.Sub: **2230.16H**
Type: Unsigned32
Access: rw
Unit: --
Range: Unsigned32
Default Value: 14000
Store Supported: Yes

Description: This value is used for Feedback feature and defines the gain used to calculate the Feedforward current, while the motor is decelerating backward. The value has to be adjust to reduce the Feedback_Actual_Position_Error and the Feedback_Actual_Velocity_Error during deceleration in backward direction.
Pay attention that a not coherent value (big value) can introduce motor's vibration.

Notes: See §*8.2* for details.

Name: **Feedback_Kffw_Acc**
Address: **280CH**
CANopen Index.Sub: **2230.6H**
Type: Unsigned32
Access: rw
Unit: --
Range: Unsigned32
Default Value: 14000
Store Supported: Yes

Description: This value is used for Feedback feature and defines the gain used to calculate the Feedforward current, while the motor is accelerating forward. The value has to be adjust to reduce the Feedback_Actual_Position_Error and the Feedback_Actual_Velocity_Error during acceleration in forkward direction
Pay attention that a not coherent value (big value) can introduce motor's vibration.

Notes: See §*8.2* for details.

Name: **Feedback_Kffw_Dec**
Address: **280EH**
CANopen Index.Sub: **2230.7H**
Type: Unsigned32
Access: rw
Unit: --
Range: Unsigned32
Default Value: 14000
Store Supported: Yes

Description: This value is used for Feedback feature and defines the gain used to calculate the Feedforward current, while the motor is decelerating forward. The value has to be adjust to reduce the Feedback_Actual_Position_Error and the Feedback_Actual_Velocity_Error during acceleration in forward direction.
Pay attention that a not coherent value (big value) can introduce motor's vibration.

Notes: See §*8.2* for details.

Name: **Feedback_Kp**
Address: **2800H**
CANopen Index.Sub: **2230.4H**
Type: Unsigned32
Access: rw
Unit: --
Range: Unsigned32
Default Value: 20000
Store Supported: Yes

Description: This value is used for Feedback feature and define the position Gain of PID's regulator.The value has to be adjust to keep the Feedback_Actual_Position_Error inside the tollerance of application. The value can be set to 0, when is necessary only a velocity controller and has not to be considered the postion error.
Pay attention that a not coherent value (big value) can introduce motor's vibration.

Notes: See §*8.2* for details.

Name: **Feedback_Kv**
Address: **2802H**
CANopen Index.Sub: **2230.5H**
Type: Unsigned32
Access: rw
Unit: --
Range: Unsigned32
Default Value: 8000
Store Supported: Yes

Description: This value is used for Feedback feature and define the velocity Gain of PID's regulator.The value has to be adjust to keep the Feedback_Velocity_Position_Error inside the tollerance of application.
Pay attention that a not coherent value (big value) can introduce motor's vibration.

Notes: See §*8.2* for details.

Name: **Feedback_Limit_Speed**
Address: **2828H**
CANopen Index.Sub: **2230.1BH**
Type: Unsigned16
Access: rw
Unit: rpm
Range: Unsigned16
Default Value: 10000
Store Supported: Yes

Description: This value is used for Feedback feature and define the maximun motor speed. If the speed became higher of the limit , the motor current is reduced to limit the speed.

Notes: See §*8.2* for details.

Name: **Feedback_Position_Error_Limit**
Address: **2808H**
CANopen Index.Sub: **2230.1H**
Type: Unsigned32
Access: rw
Unit: IU (one motor turn = 65536 IU)
Range: Unsigned32
Default Value: 10000
Store Supported: Yes

Description: This value is used for Feedback feature and define the maximum allowed displacement between motor ref position and motor actual position detected by encoder. When the displacement limit is reached, the motor is stopped depending by object *Feedback_Settings.Type*,

Notes: See §*8.2* for details.

Name: Feedback_Settings
Address: 2820H
CANopen Index.Sub: 2230.17H
Type: Unsigned16
Access: rw
Unit: rpm
Range: Unsigned16
Default Value: 0
Store Supported: Yes

Description: This value is used for Feedback feature and define some important settings about the feedback functionality. This object is managed at bit and the following tables show the meaning of each bits.

Feedback_Settings															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Sensor Type				Calibration Mode	Kff_Switch			Type			

Bit	Value	Description
0 ..15	Type: Mode	Mode 0: Compatibility Mode
		Mode 1: Stop motor if position error reach the limit and show drive Alarm
		Mode 2: Stop motor if position error reach the limit
		Mode 3: Stop motor if velocity error reach the limit and show drive Alarm
		Mode 4: Stop motor if velocity error reach the limit
		Mode 5: Torque mode. In this mode, the torque (Target_torque) is preset as a set value and reached via a ramp function (Torque_slope). (Note: This function is available only with firmware version V02r82 or superior.)
		Mode 6 to 15 : Reserved
4	Not used	
5	Not used	
6	Kff_Switch	Define the managing of the feedforward current. See the *Feedback Diagram*
7	Calibration Mode	0= the Strong Calibration is executed each time the Close Loop has to be activated for example after the power is supplied or after an Reset Alarm that disabled the close loop
		1= the Strong Calibration, is executed only the first time the power is supplied or when happen an Alarm that require the Strong Calibration (example encoder misalignment), in all other cases after the Reset_Alarm is sent, the Little Calibration will be done. See the *Feedback Calibration Procedure*
8..11	Sensor Type	0 = Incremental encoder (Enc#0)
		1 = Hall sensor
		2 = Incremental encoder (Enc#0) + Hall sensors
		3 = Incremental encoder (Enc#0) + Zero enc
		4 = Multi-Turn Absolute encoder multi turn SSI
		5 = Multi-Turn encoder SSI + incremental encoder
		6 = Multi-Turn Absolute encoder BISS
		7 = Multi-Turn Absolute encoder BISS + incremental encoder
		8 = Single-Turn Magnetic encoder
		9 to 15 Reserved

Bit	Value	Description
12	Calibration Type	0 = Full Feedback Calibration procedure.
		1 = Light Feedback Calibration procedure.
13	Torque Limit Speed Enable (1)	0= Torque Limit Speed disabled 1= Torque Limit Speed enabled This function is used only for 'Mode 5' . (Note: This function is available only with firmware version V02r82 or superior.)
14	Absolute Encoder Calibrated (2)	0 = The Sensor Calibration procedure for Absoluter encoder BiSS is performed every time.
		1 = The calibrated values in NVRAM for Absoluter encoder BiSS can be used. **Note :** This bit was introduced for compatibility reasons with previous firmware versions (the default value of this bit is equal to 0) and is used only for 'Sensor Type = 6'.
15	Not used	

Notes:

- (1) This function is available only with firmware version V02r82 or superior.
- (2) This function is available only with firmware version V03r18 or superior.

- See §*8.2* for details.

Name: Feedback_Status
Address: 2822H
CANopen Index.Sub: 2230.18H
Type: Unsigned16
Access: ro
Unit: --
Range: Unsigned16
Default Value: --
Store Supported: No

Description: This object returns some important information about the Feedback status.
This object is managed at bit and the following tables show the meaning of each bits.

Bit	Value	Description
0	Calibration in Progress	0= Feedback Calibration NOT in progres
		1= Feedback Calibration in progress. See the object *Feedback_Calibration_Phase* and the Feedback_Calibration_Procedure
1	Calibration_Error	0=Ok
		1= Feedback Calibration Procedure failed
2	Encoder Direction Error	0=Ok
		1= The bit is set during the Calibration Procedure, if the system detected that the encoder is counting in the wrong sense. The Encoder Phases has to be exchanged
3	Encoder Not Present	0=Ok
		1= The bit is set during the Calibration Procedure, if the system detected that the encoder is not present. Check if the encoder wiring is correct
4	Hall Sensor Fail	0=Ok
		1= The bit is set during the Calibration Procedure, if the system detected that the Hall sensors are not working well. Check if the Hall sensor wiring is correct
5	Calibration Aborted	0=Ok
		1= The bit is set if the Calibration Procedure, it was aborted before of the conclusion.
6	Torque Limit Speed Reached (2)	0= Torque limit speed reached.
		1= Torque limit speed not reached.
7	Not used	
8	Encoder Fault	0=Ok
		1= Detected problem on feedback encoder. The reasons can be:: The drive is suppling the max current (*Feedback_Boost_Current*) but the encoder is not counting from time more of TIMEOUT(encoder disconnected or broken) The encoder is counting in reverse direction respect at the expected direction for a time more of TIMEOUT(encoder misalignment, check the encoder wiring or the mechanical mounting of it)
9	Following Error	0 = OK
		1= Following Error The bit is set if the system detect the following Error in particular: In Mode 0,1,2 is set if |Fbd_Actual_Poition_Error| >= Fdb_Position_Error_Limit In Mode 3,4 is set if |Fbd_Actual_Velocity_Error| >= Fdb_Velocity_Error_Limit
10		0=OK

Bit	Value	Description
	Motor Stall detected (1)	1= Motor Stall detected. This Error can be issued when 'Motor Stall' feature is enabled (bit13 of *Drive_Working_Settings_Extended object)* and motor stall is detected. The bit5 (Feedback_Error) of *Error_Register* object is also set. See §*9.4* for more details about 'Motor Stall detection' feature.
11	**GAIN out of range**	0=OK
		1= GAIN out of range. The bit is set if the system detected that the set Feedback GAIN sent the PID regulator out of control
12	**Calibration attained**	0 = Calibration not attained
		1 = Calibration attained
13	**Absolute Encoder Error**	0 = Ok
		1 = Absolute Encoder Error
14	**Motor Stall Conflict** (1)	0=OK
		1= Error : tried to enable 'Feedback' feature with 'Motor Stall detection' feature already enabled or vice versa. The 'Feedback' feature (bit4 of *Drive_Working_Settings* object*)* and the 'Motor Stall detection' feature (bit13 of *Drive_Working_Settings_Extended object)* cannot be both active at the same time. The bit5 (Feedback_Error) of *Error_Register* object is also set. See §*9.4* for more details about 'Motor Stall detection' feature.
15	**Active**	0= Feedback not active
		1= Feedback Active(the system is working in closed loop)

Notes:

- See §*8.2* for details.

- (1) This bit is available only with firmware version V02r74 or superior.
- (2) This bit is available only with firmware version V02r82 or superior.

Name: **Feedback_Encoder_PPR**
Address: **280AH**
CANopen Index.Sub: **2230.3H**
Type: Unsigned32
Access: rw
Unit: Inc
Range: Unsigned32
Default Value: 1600
Store Supported: Yes

Description: This value is used for Feedback feature and define the quadrature pulses per revolution of incremental encoder installed on motor rear shaft.
Please Note: typically the PPR number indicated in Ever Motor Datasheets needs to be multiplied by 4; this is because of the drive quadrature count.

Notes: See §*8.2* for details.

Name: **Feedback_Velocity_Error_Limit**
Address: **282EH**
CANopen Index.Sub: **2230.1EH**
Type: Unsigned32
Access: rw
Unit: 0.01 rpm
Range: Unsigned32
Default Value: 1000
Store Supported: Yes

Description: This value is used for Feedback feature and define the maximum allowed displacement between motor ref speed and motor actual speed detected by encoder. When the displacement limit is reached, the motor is stopped depending by object *Feedback_Settings.Type*,

Notes: See §*8.2* for details.

Name: **Gear_Ratio_Motor_Revs**
Address: **1022H**
CANopen Index.Sub: **6091.1H**
Type: DWORD
Access: r/w
Unit: Motor Revolutions
Range: -2147483648 ÷ 2147483647
Default Value: 1
Store Supported: Yes

Description: It sets the ratio between the motor and the encoder when electric gear feature is enabled. Each time this object is stored the *Motor_Gear_Ratio* object is updated according to the following formula:

$$Motor\_Gear\_Ratio = \frac{Gear\_Ratio\_Motor\_Revs}{Gear\_Ratio\_Shaft\_Revs}$$

Notes: The motor reference position will be equal to Encoder_Actual_Value[x] * Motor_Gear_Ratio.

Name: **Gear_Ratio_Shaft_Revs**
Address: **1024H**
CANopen Index.Sub: **6091.2H**
Type: DWORD
Access: r/w
Unit: Shaft Revolutions
Range: -2147483648 ÷ 2147483647
Default Value: 1
Store Supported: Yes

Description: It sets the ratio between the motor and the encoder when electric gear feature is enabled. Each time this object is stored the *Motor_Gear_Ratio* object is updated according to the following formula:

$$Motor\_Gear\_Ratio = \frac{Gear\_Ratio\_Motor\_Revs}{Gear\_Ratio\_Shaft\_Revs}$$

Notes: The motor reference position will be equal to Encoder_Actual_Value[x] * Motor_Gear_Ratio.

Name: Hall_Sensors_Status
Address: 2900H
CANopen Index.Sub: 2900.0H
Type: Unsigned16
Access: ro
Unit: --
Range: 0 ÷ 7
Default Value: 0
Store Supported: No

Description: This object returns state of Hall Sensors:

A	B	C	Status	Note
0	0	0	0	Not valid
0	0	1	1	
0	1	0	2	
0	1	1	3	
1	0	0	4	
1	0	1	5	
1	1	0	6	
1	1	1	7	Not valid

The Status 0 and Status 7 are not valid combinations. One of these two status means issues on Hall Sensors connection and / or issues concerning Hall Sensors.

Notes:

Name: Hall_Sensors_Position
Address: 2901.0H
CANopen Index.Sub: 2901.0H
Type: Unsigned16
Access: ro
Unit: 0.1 degree
Range: 0,600,1200,1800,2400,3000
Default Value: 0
Store Supported: No

Description: This object returns position of Hall Sensors.

Notes:

Name: Hall_Sensors_Sequence_Settings
Address: 290A.0H
CANopen Index.Sub: 290A.0H
Type: Unsigned32
Access: rw
Unit: --
Range: --
Default Value: 326451h
Store Supported: Yes

Description: This value defines Hall Sensors sequence.

Notes:

Name: Hall_Sensors_Sequence_Detected
Address: 290D.0H
CANopen Index.Sub: 290D.0H
Type: Unsigned32
Access: ro
Unit: --
Range: --
Default Value: 0
Store Supported: No

Description: This value returns Hall Sensors sequence detected after that Hall Sensors Procedure is done.

Notes:

Name: **Homing_Offset[0÷1]**
Address: **2220H,2222H**
CANopen Index.Sub: **2220.1H, 2220.2H**
Type: DWORD
Access: rw
Unit: Motor steps
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: Yes

Description: This object is used to define the offset position concerning the homing movement forward (*Homing_Offset[0]*) and backward (*Homing_Offset[1]*).

Notes: See §*8.3.2* for details.

Name: **Homing_Overrun[0÷1]**
Address: **2224H,2226H**
CANopen Index.Sub: **2220.3H, 2220.4H**
Type: DWORD
Access: rw
Unit: Motor steps
Range: 0 ÷ 4294967296
Default Value: 0 (no homing overrun check)
Store Supported: Yes

Description: This object is used to define the maximum number of steps the motor will perform while looking for limit switch when performing a homing movement. If this object is 0, the drive will run indefinitely until the limit switch is found. If the limit switch is not found within *Homing_Overrun[x]* steps, the motor will be stopped with ramp and the *Motor_Limit_Switch_Not_Found* bit of *Drive_Register* will be set.

Notes: See §*8.3.2* for details.

Name: **Homing_Preset_Position**
Address: **222EH**
CANopen Index.Sub: **2220.8H**
Type: DWORD
Access: rw
Unit: Motor steps
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: Yes

Description: This object is used to define the motor current position (*Position_Actual_Value* object) at the end of the Homing procedure.

Notes: See §*8.3.2* for details.

Name: **Homing_Speed_Out**
Address: **222AH**
CANopen Index.Sub: **2220.6H**
Type: DWORD
Access: rw
Unit: Hz
Range: 0 ÷ Max_Profile_Velocity
Default Value: 0
Store Supported: Yes

Description: This object is used to define the exit speed during Homing procedure.

Notes: If this object is set to 0 the exit speed will be equal to *Min_Profile_Velocity* object.
See §***8.3.2*** for details.

Name: **Homing_Status_Register**
Address: **2228H**
CANopen Index.Sub: **2220.5H**
Type: WORD
Access: ro
Unit: **--**
Range: **--**
Default Value: --
Store Supported: No

Description: This object contains the state of the Homing procedure.

Bit #	Name	Description	Default value
0	Busy	1 = Homing procedure is in progress 0 = Homing procedure idle	0
1	Limit_Switch_Not_Found	1 = Limit Swiitch not found during Homing procedure 0 = Ok	0
2	Abort	1 = Homing procedure aborted 0 = Ok	0
3	Reserved	Reserved	0
4	Reserved	Reserved	0
5	Reserved	Reserved	0
6	Reserved	Reserved	0
7	Reserved	Reserved	0
8	Reserved	Reserved	0
9	Reserved	Reserved	0
10	Reserved	Reserved	0
11	Reserved	Reserved	0
12	Reserved	Reserved	0
13	Reserved	Reserved	0
14	Reserved	Reserved	0
15	Reserved	Reserved	0

Notes: See §*8.3.2* for details.

Name: Homing_Torque_Current_Limit
Address: 222CH
CANopen Index.Sub: 2220.7H
Type: WORD
Access: rw
Unit: mA
Range: 0 ÷ Max drive current
Default Value: 0
Store Supported: Yes

Description: This object is used to defines the maximum current used for Homing Torque modes.

Notes: See §*8.3.2* for details.

Name: Impact_Actual_Displacement
Address: 2244H
CANopen Index.Sub: 2240.4H
Type: DWORD
Access: rw
Unit: Motor steps
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: No

Description: This object stores the actual displacement between motor and the source used for motor impact check feature.

Notes: See §*9.1* for more details about Motor Impact Check feature.

Name: Impact_Factor
Address: 2246H
CANopen Index.Sub: 2240.5H
Type: DWORD (float)
Access: rw
Unit: --
Range: -2147483648.0 ÷ 2147483647.0
Default Value: 1.0
Store Supported: Yes

Description: This object stores the conversion ratio between motor steps and increments unit of source used for motor impact check feature.

Notes: See §*9.1* for more details about Motor Impact Check feature. With the encoder mounted on motor shaft Impact_Factor = Motor steps per revolution / Encoder pulses per revolution. Note that the drive counts the encoder pulses on quadrature, so the encoder pulses per revolution = encoder resolution * 4.

Name: **Impact_Max_Displacement**
Address: **2242H**
CANopen Index.Sub: **2240.3H**
Type: DWORD
Access: rw
Unit: Motor steps
Range: 0 ÷ 4294967296
Default Value: 0
Store Supported: Yes

Description: Maximum allowed displacement between motor and encoder for the impact feature. If the difference between impact source value and motor (*Position_Actual_Value* object) is higher than *Impact_Max_Displacement*, the motor will be stopped (could be already blocked) and the bit *Motor_Impacted* of *Drive_Register* object will be set.

Notes: See §*9.1* for more details about Motor Impact Check feature.

Name: **Impact_Source**
Address: **2248H**
CANopen Index.Sub: **2240.3H**
Type: WORD
Access: rw
Unit:
Range: 0 = Encoder #0
1 = Encoder #1
10 = Biss Encoder (Only for drive models with BiSS interface)

Default Value: 0
Store Supported: Yes

Description: This object defines the encoder number used for motor impact check feature.

Notes: See §*9.1* for more details about Motor Impact Check feature.

Name: **Master_Register**
Address: **1203H**
CANopen Index.Sub: **4000.4H**
Type: WORD
Access: rw
Unit: --
Range: 0 ÷ FFFFH
Default Value: 0001H
Store Supported: No

Description: This register can be used by the Master Network to send fast stop commands, to reset alarms and to refresh the Master Watchdog checked by the drive.

Bit #	Name	Description	Default Value
0	Master_Motor_Enable	1 = Motor movements are allowed 0 = Motor movements not allowed	1
1	Master_Motor_Fast_Stop	1 = Force an motor fast stop. 0 = No action	0
2	Master_Alarm_Reset	1 = Force to reset alarm condition 0 = No action	0
3	Reserved	Reserved	0
4	Reset_Feedback_Displacement	1 = Reset Feedback Displacement 0 = No action	0
5	Feedback_Calibration_Strong	1 = Force Feedback Calibration Strong 0 = No action	0
6	Reserved	Reserved	0
7	Reserved	Reserved	0
8	Reserved	Reserved	0
9	Reserved	Reserved	0
10	Reserved	Reserved	0
11	Reserved	Reserved	0
12	Reserved	Reserved	0
13	Reserved	Reserved	0
14	Reserved	Reserved	0
15	Master_Watchdog	Master watchdog bit	0

Bit Explanation:

Master_Motor_Enable: When the Master sets (1) this bit the motor movements are allowed (from digital inputs and from fieldbus) unless an emergency occurs.
If the Master resets (0) this bit no motor movement will be allowed and the bit *Motor_Movement_Not_Executed* in the *Drive_Register* object will be set if a motor movement command is received via fieldbus or via digital inputs. Also the power to the motor is no supplied any more (the feedback feature is disabled as well). When the Master sets this bit again it is necessary to waits 3-4 seconds before issuing new movement commands. It is suggested also to perform an homing procedure since the motor position (*Position_Actual_Value* object) is not valid any more.

Master_Motor_Fast_Stop: When the Master sets (1) this bit the motor will perform a immediate stop (without ramp) if the motor is running.

Master_Alarm_Reset: When the Master sets (1) this bit the *Error_Register* object, and the bits #4,5,6,7,8,14 of *Drive_Register* object are cleared.

Reset_Feedback_Displacement: When the Master sets (1) this bit the Feedback Displacement is reset. See §*8.2* for details.

Feedback_Calibration_Strong: When the Master sets (1) this bit the Feedback Calibration Strong is forced. See § *8.2* for details.

Master_Watchdog: The master have to toggle (0/1) this bit each time it writes the *Master_Register* object. If drive don't receive a write access to this object within the *Master_Watchdog_Timeout* the bit *Master_Watchdog_Timouted* of *Drive_Register* object is set.

Notes: The checking for the *Master_Watchdog* bit starts the first time the master perform a write access to this object. The *Master_Motor_Enable* bit is set (1) by default to enable motor movement when the drive acts as standalone or when the master do not want to handle the *Master_Register* object. The default value of bit #0 depends on the value of the *Drive_Working_Settings_Extended* bit #5.

Name: **Master_Watchdog_Timeout**
Address: **1205H**
CANopen Index.Sub: **4000.6H**
Type: WORD
Access: r/w
Unit: milliseconds
Range: 0 ÷ 65535
Default Value: 0
Store Supported: Yes

Description: This object set the toggle frequency of the *Master_Watchdog* bit of *Drive_Register* object. When the network master write the *Master_Register* object the drive starts the check of the *Master_Watchdog* bit. If the corresponding bit of the *Master_Register* object do not toggle within the *Master_Watchdog_Timeout* the Bit #15 of *Drive_Register* object is set. If this object is set = 0 the master watchdog checking is disabled.

Notes: If the *Master_Watchdog_Timeout_Action* bit of *Drive_Working_Settings* object is set, when a Master Watchdog timeout occur, the drive will stop immediately the movement in progress and will set the *Master_Watchdog_Timeouted* bit in the *Drive_Register* object. The Network Master then have to set the *Master_Alarm_Reset* bit of the *Master_Register* object to recover from the error condition.

Name: **Max_Profile_Velocity**
Address: **1015H**
CANopen Index.Sub: **607F.0H**
Type: DWORD
Access: r/w
Unit: Hertz
Range: (250**Motor_SPR* / *65536*) ÷ (3276800**Motor_SPR* / *65536*)
Default Value: 40000
Store Supported: Yes

Description: It sets the maximum motor velocity for the acceleration and deceleration ramp.

Notes: This object can only be set with the motor at a standstill;
This object must be higher than the *Min_Profile_Velocity* object.
If the value to be set is lower than the minimum value range, the minimum value is stored.
The *Profile_Velocity* object becomes equal to the value of the *Max_Profile_Velocity* object .

Name: **Max_Torque**
Address: **2B01H**
CANopen Index.Sub: **6072.0H**
Type: UINT16
Access: r/w
Unit: ‰ Nominal_Current
Range: UINT16
Default Value: 1000
Store Supported: Yes

Description: This value is used for Feedback feature and indicate the configured maximum permissible torque in the motor. The Max torque is proportional to '*Nominal_Current*' (Motor rated current) of the motor. This object is used only if '*Feedback_Settings*' (Type) = 5.

The value is defined as thousandths of the torque, e.g. '500' means '50%' of the rated torque.

Notes: See §*8.2* for details.

This object is available only with firmware version V02r82 or superior.

Name: **Min_Current**
Address: **1010H**
CANopen Index.Sub: **2005.1H**
Type: WORD
Access: r/w
Unit: mA
Range: 0 ÷ (max drive current)
Default Value: 0
Store Supported: Yes

Description: It sets the motor's reduced current. The drive automatically reduces the current after a current reduction time (*Time_RWC* object) at the end of the movement

Notes: The current set is the Irms current. The peak current is Irms * 1.4.

Name: **Max_Current**
Address: **1011H**
CANopen Index.Sub: **2005.2H**
Type: WORD
Access: r/w
Unit: mA
Range: 0 ÷ (max drive current)
Default Value: 0
Store Supported: Yes

Description: It sets the motor's current when running at constant speed. During acceleration and deceleration, the *Boost_Current* is automatically set.

Notes: The current set is the Irms current. The peak current is Irms * 1.4.

Name: **Boost_Current**
Address: **1012H**
CANopen Index.Sub: **2005.3H**
Type: WORD
Access: r/w
Unit: mA
Range: 0 ÷ (max drive current)
Default Value: 0
Store Supported: Yes

Description: It sets the boost current in the motor. The boost current is enabled when the motor accelerates and decelerates.

Notes: The current set is the Irms current. The peak current is Irms * 1.4.

Name: **Nominal_Current**
Address: **1013H**
CANopen Index.Sub: **2005.4H**
Type: WORD
Access: r/w
Unit: mA
Range: 0 ÷ (max drive current)
Default Value: 0
Store Supported: Yes

Description: It's the motor's nominal current , this parameter is used for the automatic detection of motor R,L. This parameter is generally set one time at switch-on when the motor is standstill, anyway each time the parameter is set to a different value, a new detection of R,L parameter is executed.

Notes: The current set is the Irms current. The peak current is Irms * 1.4.

Name: **Min_Profile_Velocity**
Address: **1014H**
CANopen Index.Sub: **2010.0H**
Type: WORD
Access: r/w
Unit: Hertz
Range: 0 ÷ (150000**Motor_SPR* / *65536*)
Default Value: 0
Store Supported: Yes

Description: It sets the minimum motor velocity for the acceleration and deceleration ramp.

Notes: This object can only be set with the motor at a standstill;
This object cannot be higher than the *Max_Profile_Velocity* object.
If the value to be set is lower than the minimum value range, the minimum value is stored.
If the value to be set is higher than the maximum value range, the maximum value is stored.

Name: **Motor_Gear_Kp**
Address: **1033H**
CANopen Index.Sub: **2021.3H**
Type: WORD
Access: r/w
Unit: **--**
Range: 0 ÷ 1000
Default Value: 20
Store Supported: Yes

Description: Kp constant used for electric gear feature. Kp is used for position correction. Increasing Kp value the drive will accelerate the motor to compensate position error.

Notes:

Name: Motor_Gear_Type
Address: 1032H
CANopen Index.Sub: 2021.2H
Type: WORD
Access: r/w
Unit: --
Range: 0 ÷ 30
Default Value: 11
Store Supported: Yes

Description: It sets type of electric gear type and the reference source. With monodirectional type the direction of the movement is that defined by *Direct_Command_Parameter_1*. With bidirectional type the direction of the movement depends on the source rotation direction. The following values can be set:

Value	*Description*
3	Monodirectional speed (Electric_Gear_Ext_Speed_Ref object)
4	Bidirectional speed (Electric_Gear_Ext_Speed_Ref object)
5	Bidirectional speed from Analog Input #0
6	Monodirectional speed from Analog Input #0
10	Monodirectional from Counter #1
11	Bidirectional from Counter #1
20	Ck/dir from Counter #1
22	Bidirectional from CAM position (use only with CAM mode)
30	Clock Up & Clock Down Counter #1

Notes:

Name: Motor_Pole_Pairs
Address: 1027H
CANopen Index.Sub: 2012.2H
Type: WORD
Access: r/w
Unit: # of motor Pole Pairs
Range: 2; 3; 4; 5; 6; 8; 10; 11; 12; 15; 25; 45; 50; 100
Default Value: 50 (typical value for stepper motors)
Store Supported: Yes

Description: It sets the number of motor pole pairs.

Notes: This object can only be set with the motor at a standstill.
If the value set is not valid the default value is stored.

- With firmware V03r16 or lower the motor resolution is defined by mean of *Motor_Step_Angle* and *Motor_Pole_Pairs* objects (in this case *Motor_Resolution* object is read only and returns the motor resolution value).

- With firmware V03r17 or superior the motor resolution can be defined by mean of *Motor_Resolution* object or *Motor_Step_Angle*, *Motor_Pole_Pairs* objects. If *Motor_Resolution* object value is 0 then the motor resolution is defined by mean of *Motor_Step_Angle* and *Motor_Pole_Pairs* objects (for compatibility reasons). A value of *Motor_Resolution* object different from zero defines directly the motor resolution (*Motor_Step_Angle* object is not considered and only *Motor_Pole_Pairs* object have to be defined).

- See *2012.1H* , *60EF.0H* objects.

Name: **Motor_Resolution**
CANopen Index.Sub: **60EF.0H**
Type: Unsigned32
Access: ro [1]
rw [2]
Unit: Increments
Range: 200[1] ÷ 65536
0 [2] ÷ 65536
Default Value: 200 [1]
0 [2]
Store Supported: No [1]
Yes [2]

Description: This objects is the Motor Resolution.

Notes: - [1] *with firmware V03r16 or lower.*
- [2] *with firmware V03r17 or superior.*

- This object is available with firmware V03r10 or superior.
- This object can only be set with the motor at a standstill.

- With firmware V03r16 or lower the motor resolution is defined by mean of *Motor_Step_Angle* and *Motor_Pole_Pairs* objects (in this case *Motor_Resolution* object is read only and returns the motor resolution value).

- With firmware V03r17 or superior the motor resolution can be defined by mean of *Motor_Resolution* object or *Motor_Step_Angle*, *Motor_Pole_Pairs* objects. If *Motor_Resolution* object value is 0 then the motor resolution is defined by mean of *Motor_Step_Angle* and *Motor_Pole_Pairs* objects (for compatibility reasons). A value of *Motor_Resolution* object different from zero defines directly the motor resolution (*Motor_Step_Angle* object is not considered and only *Motor_Pole_Pairs* object have to be defined).

- *See 2012.1*H*, 2012.2*H

Name: **Motor_R**
Address: **2000H**
CANopen Index.Sub: **2005.6H**
Type: DWORD
Access: rw
Unit: ohm $x10^{-3}$
Range: DWORD
Default Value: 380
Store Supported: Yes

Description: This object sets the motor R (resistance). This value is used when bit #9 of *Drive_Working_Settings_Extended* object is equal to 0.

Notes:

Name: **Motor_L**
Address: **2002H**
CANopen Index.Sub: **2005.7H**
Type: DWORD
Access: rw
Unit: H $x10^{-6}$
Range: DWORD
Default Value: 2420
Store Supported: Yes

Description: This object sets the motor L (inductance). This value is used when bit #9 of *Drive_Working_Settings_Extended* object is equal to 0.

Notes:

Name: **Motor_R_Detected**
Address: **2004H**
CANopen Index.Sub: **2005.8H**
Type: DWORD
Access: r
Unit: ohm $x10^{-3}$
Range: Unsigned32
Default Value: **--**
Store Supported: No

Description: This object returns the motor R (resistance) when bit #9 of *Drive_Working_Settings_Extended* object is equal to 1.

Notes:

Name: **Motor_L_Detected**
Address: **2006H**
CANopen Index.Sub: **2005.9H**
Type: DWORD
Access: r
Unit: H $x10^{-6}$
Range: Unsigned32
Default Value: --
Store Supported: No

Description: This object returns the motor L (inductance) when bit #9 of *Drive_Working_Settings_Extended* object is equal to 1.

Notes:

Name: **Motor_Stall_Actual_Err_Angle**
Address: **2009H**
CANopen Index.Sub: **2005.BH**
Type: WORD
Access: ro
Unit: (0.01 rad)
Range: -32768÷32767
Default Value: **--**
Store Supported: No

Description: This object returns the actual displacement between theoretical angle and estimated angle of motor rotor position for 'Motor Stall detection' feature.

Notes: See §*9.4* for more details about 'Motor Stall detection' feature.

This object is available only with firmware version V02r74 or superior.

Name: **Motor_Stall_Filter_Time**
Address: **200AH**
CANopen Index.Sub: **2005.CH**
Type: UWORD
Access: rw
Unit: microseconds
Range: 0÷65535
Default Value: 200
Store Supported: Yes

Description: This object defines the time of the software filter used to calculate the estimated angle of the motor rotor position for 'Motor Stall detection' feature.

Notes: See §*9.4* for more details about 'Motor Stall detection' feature.

This object is available only with firmware version V02r74 or superior.

Name: **Motor_Stall_Max_Err_Angle**
Address: **2008H**
CANopen Index.Sub: **2005.AH**
Type: UWORD
Access: rw
Unit: (0.01 rad)
Range: 0÷65535
Default Value: 1256
Store Supported: Yes

Description: This object defines the maximum allowed displacement between theoretical angle and estimated angle of motor rotor position for 'Motor Stall detection' feature. When the displacement limit is reached, the motor is stopped (*Min_Current* object value is applied to motor) and alarm is issued.

Notes: See §*9.4* for more details about 'Motor Stall detection' feature.

This object is available only with firmware version V02r74 or superior.

Name: **Motor_Step_Angle**
Address: **1026H**
CANopen Index.Sub: **2012.1H**
Type: WORD
Access: r/w
Unit: --
Range: 1; 2; 4; 5; 8; 10; 16; 25; 32; 50; 64; 125; 128; 250; 256; 65535
Default Value: 1
Store Supported: Yes

Description: The *Motor_Step_Angle* and *Motor_Pole_Pairs* are used to define the motor resolution .

- If '*Motor_Step_Angle*' value is different from 65535, the number of motor Increments per revolution are computed as follow.

  TITANIO drives (Stepper motors) :

  *Motor_Resolution (Inc/rev) = (Motor_Pole_Pairs * 4) * Motor_Step_Angle;*

  PLATINO *drives (BLDC motors) &* ***VANADIO*** *drives (BLAC motors) :*

  *Motor_Resolution (Inc/rev) = (Motor_Pole_Pairs * 6) * Motor_Step_Angle;*

- If '*Motor_Step_Angle*' value is 65535 the number of motor Increments per revolution is :

  Motor_Resolution (Inc/rev) = 65536

Notes: This object can only be set with the motor at a standstill.
If the value set is not valid the default value is stored.

- See *2012.2H* , *60EF.0H* objects.

- With firmware V03r16 or lower the motor resolution is defined by mean of *Motor_Step_Angle* and *Motor_Pole_Pairs* objects (in this case *Motor_Resolution* object is read only and returns the motor resolution value).

- With firmware V03r17 or superior the motor resolution can be defined by mean of *Motor_Resolution* object or *Motor_Step_Angle*, *Motor_Pole_Pairs* objects. If *Motor_Resolution* object value is 0 then the motor resolution is defined by mean of *Motor_Step_Angle* and *Motor_Pole_Pairs* objects (for compatibility reasons). A value of *Motor_Resolution* object different from zero defines directly the motor resolution (*Motor_Step_Angle* object is not considered and only *Motor_Pole_Pairs* object have to be defined).

Name: Motor_SPR
Address: 1028H
CANopen Index.Sub: 2012.3H
Type: WORD
Access: r
Unit: --
Range:
Default Value: --
Store Supported: No

Description: Return the motor resolution as microsteps every motor turn.

Notes:

Name: **Motor_Start_Delay**
Address: **223AH**
CANopen Index.Sub: **2025.1H**
Type: DWORD
Access: r/w
Unit: microseconds
Range: 0 ÷ 2^{31}
Default Value: 0
Store Supported: Yes

Description: It defines defines the delay on the motor start. This object can be used to have repetitive motor starts. Each time the motor has to start (**MOVE** command executed, or *Motor_SYNC* object written or Trigger Start input detected) *Motor_Start_Delay* microseconds are waited.

Notes: This object should not be used together with *Motor_Start_Delay_Pulses* object.

Name: **Motor_Start_Delay_Pulses**
Address: **223CH**
CANopen Index.Sub: **2025.2H**
Type: WORD
Access: r/w
Unit: pulses
Range: 0 ÷ 65535
Default Value: 0
Store Supported: Yes

Description: It defines defines the delay on the motor start expressed in pulses received from Counter #1. This object is generally used when electric gear feature is enabled. This object can be used to have repetitive motor starts. Each time the motor has to start (**MOVE** command executed, or *Motor_SYNC* object written or Trigger Start input detected) *Motor_Start_Delay_Pulses* pulses are waited from Counter #1.

Notes: See *Counter_Config[1]* object for details on Counter #1 configuration.
This object should not be used together with *Motor_Start_Delay* object.

Name: **Motor_Stop_Trigger_Count**
Address: **2250H**
CANopen Index.Sub: **2026.1H**
Type: WORD
Access: r/w
Unit: edges
Range: 0 ÷ 65535
Default Value: 0
Store Supported: No

Description: It defines defines the number of stop edges if a STOP with trigger input has been issued. Value = 0 means STOP at the first edge detected, Value = 1 means STOP after two edges detected, and so on.

Notes:

Name: **Motor_Stop_Trigger_Min_Position**
Address: **2252H**
CANopen Index.Sub: **2026.2H**
Type: DWORD
Access: r/w
Unit: motor steps
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: No

Description: It defines the min position used to filter the stop trigger signal. See also *Motor_Stop_Trigger_Options* for more details.

Notes:

Name: **Motor_Stop_Trigger_Max_Position**
Address: **2254H**
CANopen Index.Sub: **2026.3H**
Type: DWORD
Access: r/w
Unit: motor steps
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: No

Description: It defines the max position used to filter the stop trigger signal. See also *Motor_Stop_Trigger_Options* for more details.

Notes:

Name: **Motor_Stop_Trigger_Options**
Address: **2251H**
CANopen Index.Sub: **2026.4H**
Type: WORD
Access: r/w
Unit: **--**
Range: **Byte High:** 0 ÷ 20 ms **Byte Low:** 0 ÷ 5
Default Value: 0
Store Supported: No

Description:
Byte low defines the stop trigger filter type.
Byte high defines the time in ms used only if Byte low =5
Byte low =0: No filter on stop trigger signal
Byte low =1: Inhibit stop signal if Position_Actual_Value < Motor_Stop_Trigger_Min_Position
Byte low =2: Inhibit stop signal if Position_Actual_Value > Motor_Stop_Trigger_Max_Position
Byte low =3: Inhibit stop signal if
Position_Actual_Value < Motor_Stop_Trigger_Min_Position OR
Position_Actual_Value > Motor_Stop_Trigger_Max_Position

Byte low =4: Inhibit stop signal if
Position_Actual_Value > Motor_Stop_Trigger_Min_Position AND
Position_Actual_Value < Motor_Stop_Trigger_Max_Position

Byte low =5: Inhibit stop signal if (available from version V01r91or higher)
T < Byte_High ms

Notes:

Name: **Motor_SYNC**
Address: **2280H**
CANopen Index.Sub: **2280.0H**
Type: WORD
Access: w
Unit: --
Range: 0 ÷ 1
Default Value: 0
Store Supported: No

Description: This object is used to synchronize the start and the stop of movement that must be synchronized between more axes.

Notes: See §*8.3.4* for more details about synchronized movements.

Name: Node_Id
Address: 40B8H
CANopen Index.Sub: 4000.07H
Type: WORD
Access: r (rw)
Unit: --
Range: 1 ÷ 127
Default Value: 1
Store Supported: Yes

Description: This object contains the drive's NodeId. On drives with dip-switches/rotoswitches the object is read-only. On drives without dip-switches/rotoswitches this object can be changed using the e3PLC Studio or following the procedure explained at §*10.1.2* or §*11.1.2*.

Notes:

Name: **Position_Actual_Value**
Address: **1000H**
CANopen Index.Sub: **6063.0H**
Type: DWORD
Access: r/w
Unit: Motor Steps
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: No

Description: This object contains the motor's current position.

Notes:

Name: **Position_Window**
Address: **2810H**
CANopen Index.Sub: **6067.0H**
Type: DWORD
Access: r/w
Unit: Motor Steps
Range: -1 ÷ 2147483647
Default Value: 0
Store Supported: Yes

Description: This object set the symmetrical range of accepted positions relative to the target position if the actual position is within the position window the target position shall be regarded as having been reached (*Drive_Register.Target_Reached* = 1). If the value of position window is -1 the position window control shall be switched off.

Notes:

Name: **Position_Window_Time**
Address: **2812H**
CANopen Index.Sub: **6068.0H**
Type: WORD
Access: r/w
Unit: ms
Range: 0 ÷ 65535
Default Value: 0
Store Supported: Yes

Description: This object shall indicate the configured time, during which the actual position within the position window is measured. The value shall be given in ms.

Notes:

Name: **Profile_Acceleration**
Address: **1019H**
CANopen Index.Sub: **6083.0H**
Type: WORD (DWORD on CANopen)
Access: r/w
Unit: Milliseconds
Range: 5 ÷ 32000
Default Value: 100
Store Supported: Yes

Description: It sets the time of the acceleration ramp needed by the motor to go from the minimum movement frequency (*Min_Profile_Velocity* object) to the maximum movement frequency (*Max_Profile_Velocity* object).

Notes: This object can also be changed while the motor is running;
If the value to be set is lower than the minimum value range, the minimum value is stored.
If the value to be set is higher than the maximum value range, the maximum value is stored.
The number of acceleration steps can be calculated using the following formula:

$$Steps = \frac{Profile\_Acceleration}{2000} * (Min\_Profile\_Velocity + Max\_Profile\_Velocity)$$

Name: **Profile_Deceleration**
Address: **101AH**
CANopen Index.Sub: **6084.0H**
Type: WORD (DWORD on CANopen)
Access: r/w
Unit: Milliseconds
Range: 5 ÷ 32000
Default Value: 100
Store Supported: Yes

Description: It sets the time of the deceleration ramp needed by the motor to go from the maximum movement frequency (*Max_Profile_Velocity* object) to the minimum movement frequency (*Max_Profile_Velocity* object).

Notes: This object can also be changed while the motor is running;
If the value to be set is lower than the minimum value range, the minimum value is stored.
If the value to be set is higher than the maximum value range, the maximum value is stored.
The number of deceleration steps can be calculated using the following formula:

$$Steps = \frac{Profile\_Deceleration}{2000} * (Min\_Profile\_Velocity + Max\_Profile\_Velocity)$$

Name: **Profile_Velocity**
Address: **1017H**
CANopen Index.Sub: **6081.0H**
Type: DWORD
Access: r/w
Unit: Hertz
Range: 1 ÷ *Max_Profile_Velocity* object included
Default Value: 20000
Store Supported: Yes

Description: It sets the motor velocity.

Notes: This object can also be set while the motor is running.
This object must be lower than or equal to the *Max_Profile_Velocity* object.

Name: Realtime_Modules_Enable
Address: 2207H
CANopen Index.Sub: 2207.0H
Type: WORD
Access: rw
Unit: --
Range: 0 ÷ FFFFH
Default Value: 0000H
Store Supported: No

Description: This object can be used to start or stop the firmware built-in realtime modules.

Bit #	Name	Description	Default Value
0	Labelling_Realtime_Module	1 = Labelling Realtime Module enabled 0 = Labelling Realtime Module disabled	0
1	Electronic_CAM_Module	1 = Electronic CAM Module enabled 0 = Electronic CAM Module disabled	0
2	Reserved	Reserved	0
3	Reserved	Reserved	0
4	Reserved	Reserved	0
5	Reserved	Reserved	0
6	Reserved	Reserved	0
7	Reserved	Reserved	0
8	Reserved	Reserved	0
9	Reserved	Reserved	0
10	Reserved	Reserved	0
11	Reserved	Reserved	0
12	Reserved	Reserved	0
13	Reserved	Reserved	0
14	Reserved	Reserved	0
15	Reserved	Reserved	0

Bit Explanation:

Labelling_Realtime_Module:
More information about the Labelling_Realtime_Module can be found in the *'Labelling Realtime Module Manual for eePLC Studio for Titanio Drives'*.

Electronic_CAM_Module:
More information about the Electronic_CAM_Module can be found in the *'CAM Realtime Module Manual for eePLC Studio for Titanio Drives'*.

Notes:

Name: **RotoSwitches**
Address: **100EH**
CANopen Index.Sub: **2027.0H**
Type: WORD
Access: r
Unit: --
Range: 0 ÷ 65535 (FFFFH)
Default Value: --
Store Supported: No

Description: This object contains the current status of drive's RotoSwitches (only for drives fitted with RotoSwitches)**:**

Notes:

Name: Serial_Interface_Parameters
Address: ----
CANopen Index.Sub: ----
Type: WORD
Access: rw
Unit: --
Range: 0000H – 0102H

Serial Data Format

Byte 0 Value	Description
0	8N1 (Standard)
1	8E1
2	8O1

Modbus DWORD Order

Byte 1 Value	Description
0	(Standard Big Endian) MSW, LSW
1	(Little Endian) LSW, MSW

Default Value: 0000H
Store Supported: No

Description: This object is used to change the settings of the Serial Communication Interface. This object can be changed only inside the user's application and it's execution is dependent to the status of U0 dip (see §*10.1.1*). If U0 dip is OFF the execution of this object is not allowed and it will not change the standard settings of the Serial Communication Interface. The U0 dip must be ON to allow the working of this object. For drives not fitted with dips-switches or rotoswitches this objects is always executed. The Service Serial Communication interface is not affected by the value of this object.

Notes: The e3PLC Studio cannot communicate with the drive while a different than standard settings are used. Set U0 dip OFF to communicate with the PC again or use the Service Serial Communication.

Name: Start_Trigger_Input_Filter
Address: 1150H
CANopen Index.Sub: 2200.18H
Type: DWORD
Access: rw
Unit: microseconds
Range: 0 ÷ FFFFFFFFH

Default Value: 00500050H
Store Supported: Yes

Description: This object is used to set the filter for 'Start Trigger Input'

	Start_Trigger_Input_Filter	
	(Digital Input High Level)	(Digital Input Low Level)
	Bit 31÷16	Bit 15÷0
Range	0÷65535	0÷65535
Default Value	50h	50h

Notes:

Name: Stop_Trigger_Input_Filter
Address: 1152H
CANopen Index.Sub: 2200.19H
Type: DWORD
Access: rw
Unit: microseconds
Range: 0 ÷ FFFFFFFFH

Default Value: 00500050H
Store Supported: Yes

Description: This object is used to set the filter for 'Stop Trigger Input'

	Stop_Trigger_Input_Filter	
	(Digital Input High Level)	(Digital Input Low Level)
	Bit 31÷16	Bit 15÷0
Range	0÷65535	0÷65535
Default Value	50h	50h

Notes:

Name: Store_Parameters
Address: 2300H
CANopen Index.Sub: 1010.1H
Type: DWORD
Access: w
Unit: --
Range:

Value	Domains
65766173H	Whole domains
1001H	Variables_Stored_Block_1 domain (User_Long_Vars[0÷15])
1002H	Variables_Stored_Block_2 domain (User_Long_Vars[16÷31])
1003H	Variables_Stored_Block_4 domain (User_Float_Vars[0÷15])
1004H	Variables_Stored_Block_3 domain (User_Long_Vars[32÷47])
1005H	Variables_Stored_Block_5 domain (User_Long_Vars[48÷63])
1006H	Variables_Stored_Block_6 domain (User_Long_Vars[64÷79])
1007H	Variables_Stored_Block_7 domain (User_Long_Vars[80÷95])
1008H	Variables_Stored_Block_8 domain (User_Long_Vars[96÷111])
1009H	Variables_Stored_Block_9 domain (User_Long_Vars[112÷127])
1010H	Whole standard objects (Objects_Stored_Block_x domains) and whole labelling objects (Labelling_Objects_Stored_Block_x domains)

Default Value: --
Store Supported: --

Description: This object supports the saving of drive parameters in non volatile memory. On reception of the correct signature the drive stores the current parameters values in non volatile memory. At the next drive switch on the parameters starting value will be equal to the value stored in non volatile memory.

Notes: The storing process takes about 4÷5 (worst case) seconds to be completed. If a further store parameters command is sent before the completion of the previous one, the drive will not answer to the communication interface until the previous storing process is completed. The non volatile ram is really written only if data have changed compared to the previous store process. It is possible to know if a store procedure is in process by checking the bit #2 of *Drive_Register_Extended* object.

!!!WARNING!!! : The non volatile memory can be written for a limited number of times (typically 100,000 times), when reached that limit F4 errors (see B Appendix) can occur, and the drive should be sent to EVER for reparation.

If a write overrun is detected it will be signaled on *eePLC_Warning_Inserted* object first as Warning_Write_Overrun and later as Protection_Write_Overrun on *eePLC_Emergency_Interted* that inhibits any further write on the EEprom

Name: **Target_torque**
Address: **2B00H**
CANopen Index.Sub: **6071.0H**
Type: INT16
Access: r/w
Unit: ‰ Nominal_Current
Range: INT16
Default Value: 0
Store Supported: No

Description: This value is used for Feedback feature and indicate the input value for the torque controller. The Target torque is proportional to '*Nominal_Current*' (Motor rated current) of the motor. This object is used only if '*Feedback_Settings*' (Type) = 5.

The value is defined as thousandths of the torque, e.g. '500' means '50%' of the rated torque.

Notes: - See §*8.2* for details.
- This object is available only with firmware version V02r82 or superior.

Name: **Torque_actual_value**
Address: **2B03H**
CANopen Index.Sub: **6077.0H**
Type: INT16
Access: ro
Unit: ‰ Nominal_Current
Range: INT16
Default Value: 0
Store Supported: No

Description: This value is used for Feedback feature and provides the actual value of the torque. The value is proportional to '*Nominal_Current*' (Motor rated current) of the motor. This object is used only if '*Feedback_Settings*' (Type) = 5.

The value is defined as thousandths of the torque, e.g. '500' means '50%' of the rated torque.

Notes: - See §*8.2* for details.
- This object is available only with firmware version V02r82 or superior.

Name: **Torque_demand**
Address: **2B02H**
CANopen Index.Sub: **6074.0H**
Type: INT16
Access: ro
Unit: ‰ Nominal_Current
Range: INT16
Default Value: 0
Store Supported: No

Description: This value is used for Feedback feature and provides the actual value of the trajectory generator. The value is proportional to '*Nominal_Current*' (Motor rated current) of the motor. This object is used only if '*Feedback_Settings*' (Type) = 5.

The value is defined as thousandths of the torque, e.g. '500' means '50%' of the rated torque.

Notes: - See §*8.2* for details.
- This object is available only with firmware version V02r82 or superior.

Name: **Torque_slope**
Address: **2B04H**
CANopen Index.Sub: **6087.0H**
Type: UINT32
Access: rw
Unit: ‰ Nominal_Current / sec
Range: UINT32
Default Value: 10000
Store Supported: Yes

Description: This value is used for Feedback feature and indicates the slope in Torque mode. The value is proportional to '*Nominal_Current*' (Motor rated current) of the motor. This object is used only if '*Feedback_Settings*' (Type) = 5.

Notes: - See §*8.2* for details.
- This object is available only with firmware version V02r82 or superior.

Name: Task_Control
Address: 59A6
CANopen Index.Sub: 59A6.0
Type: WORD
Access: wo
Unit: --
Range: 0000H – 0107H

Byte 1	Byte 0
Task Command	Task Number

Task Command

Byte 1 Value	Description
0	Kill Task
1	Restart Task

Task Number

Byte 0 Value	Description
0 - 7	Task Number

Default Value: ----
Store Supported: No

Description: This object can be used to kill or restart a task.

Notes: If whole tasks are killed the error 'End of Program Execution' is raised (F+6 on display).

Name: Tasks_Status
Address: 59A7
CANopen Index.Sub: 59A7.0
Type: WORD
Access: ro
Unit: --
Range: 0000H – 00FFH

Bit #	Name	Description
0	Task #0 Status	1 = task running 0 = task killed
1	Task #1 Status	1 = task running 0 = task killed
2	Task #2 Status	1 = task running 0 = task killed
3	Task #3 Status	1 = task running 0 = task killed
4	Task #4 Status	1 = task running 0 = task killed
5	Task #5 Status	1 = task running 0 = task killed
6	Task #6 Status	1 = task running 0 = task killed
7	Task #7 Status	1 = task running 0 = task killed

Default Value: --
Store Supported: No

Description: This object returns the status of whole application tasks.

Notes:

Name: **Variable_Index**
Address: **2209H**
CANopen Index.Sub: **2208.0H**
Type: WORD
Access: rw
Unit: **--**
Range: 0 ÷ (User_Long_Vars # + User_Float_Vars #)
Default Value: 0
Store Supported: No

Description: This object permits to indexing a User_Long_Var or a User_Float_Var.

Notes:

Name: **Variable_Index_Value**
Address: **220AH**
CANopen Index.Sub: **2209.0H**
Type: DWORD
Access: rw
Unit: **--**
Range: -2147483648 ÷ 2147483647 (User_Long_Vars) or $3.4*10^{-38}$ ÷ $3.4*10^{38}$ (User_Float_Vars)
Default Value: **--**
Store Supported: No

Description: This object contains the value of the User_Long_Var or the User_Float_Var indexed by Variable_Index object.

Notes:

Name: **Velocity_Actual_Value**
Address: **1009H**
CANopen Index.Sub: **606C.0H**
Type: DWORD
Access: r
Unit: Hz (Stepper motor)
0.01 rpm (DC Brushless)
Range: 1 ÷ Max_Profile_Velocity
Default Value: 0
Store Supported: No

Description: This object contains the current motor velocity.

Notes:

Name: Velocity_demand_value
Address: 2B06H
CANopen Index.Sub: 606B.0H
Type: INT32
Access: ro
Unit: Hz (Stepper motor)
0.01 rpm (DC Brushless)
Range: INT32
Default Value: --
Store Supported: No

Description: This object provides the output value of the trajectory generator.

Notes: This object is available only with firmware version V02r82 or superior.

Name: **User_Long_Var[0 ÷ 127]**
Address: **0000H ÷ 00FEH**
CANopen Index.Sub: **2100.0H ÷ 217F.0H**
Type: DWORD
Access: r/w
Unit: --
Range: -2147483648 ÷ 2147483647
Default Value: 0
Store Supported: Yes

Description: They are the integer (32 bit) user variable.

Notes:

Name: **User_Float_Var[0 ÷ 15]**
Address: **0100H ÷ 011EH**
CANopen Index.Sub: **2180.0H ÷ 218F.0H**
Type: REAL (DWORD)
Access: r/w
Unit: --
Range: $3.4*10^{-38}$ ÷ $3.4*10^{38}$
Default Value: 0.0
Store Supported: Yes

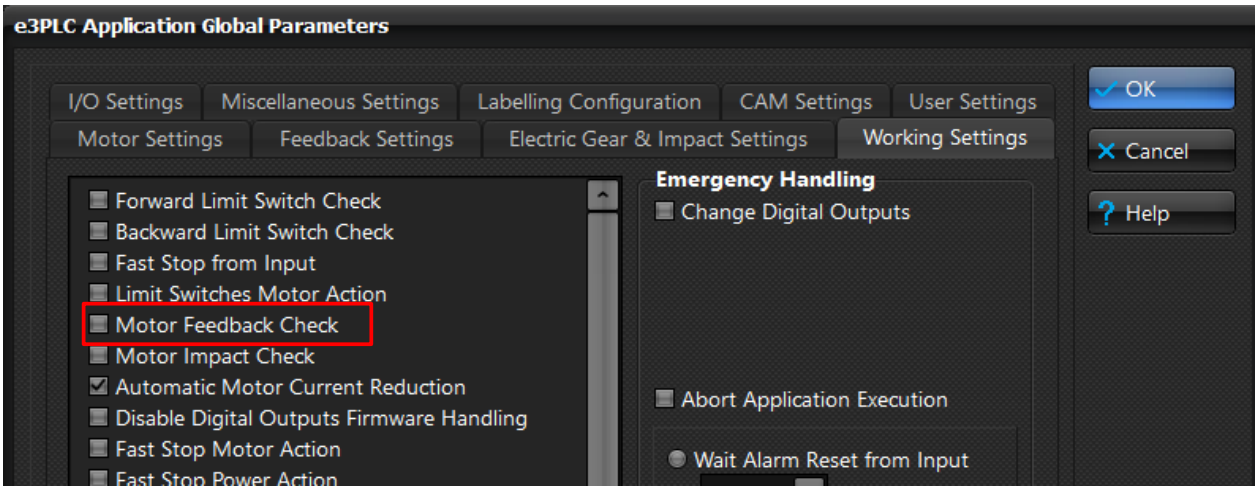
Description: They are the floating point (32 bit single precision) user variable.

Notes:

8.0 Motor management

The stepping motor allows operators to perform accurate positioning.
The stepper motor can be controlled in two way:
- Open Loop Modality (traditional use of stepper motor without feedback sensor)
- Close Loop Modality (newest use of stepper motor as a Servo Motor)

The Open/Closed Loop modality can be select through the bit *Drive_Working_Settings.Motor Feedback_Check*



8.1 Open Loop Modality

This modality is working if the bit *Drive_Working_Settings.Motor Feedback_Check =0*
This is the traditional and simplest use of stepper motor. For this modality can be use a simple stepper motor without double shaft and without feedback sensor(incremental encoder).

Below the list of Objects necessary for the configuration of the drive to works in Open Loop.

Object name	Note
Motor Poles	Mandatory
Motor_Step_Angle	Mandatory
Min_Current	Mandatory
Max_Current	Mandatory
Boost_Current	Mandatory
Nominal_Current	Mandatory
Min_Profile_Velocity	Mandatory
Max_Profile_Velocity	Mandatory
Profile_Velocity	Mandatory
Motor_R	The values can be omitted if it's enabled the automatic motor parameters detection. (bit *Drive_Working_Settings.Motor_RL_Detetection* =1).
Motor_L	
Motor_Start_Delay	Optional set it to 0. To define only if necessary delay the start of the motor
Motor_Start_Delay_Pulse	Optional set it to 0. To define only if necessary delay the start of the motor respect the master encoder rotation
Drive_Working_Settings	This Object permit the settings of some working features
Drive_Working_Settings_Extended	This Object permit the settings of some working features

Diagram of the elements making up the movement profile

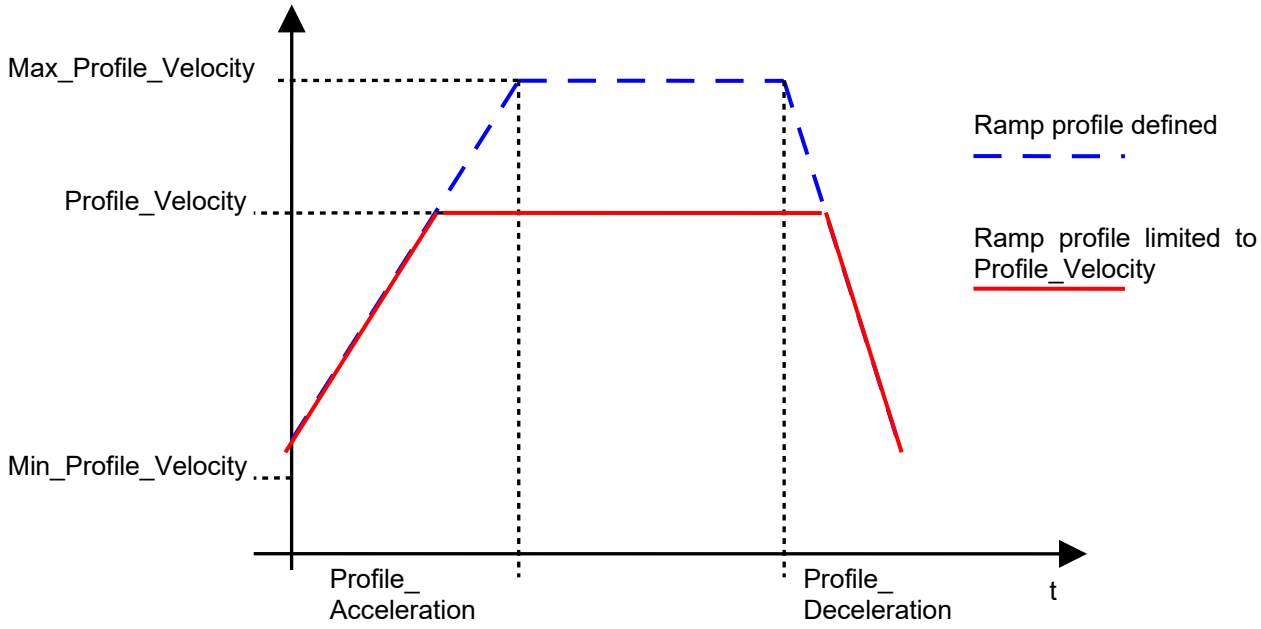
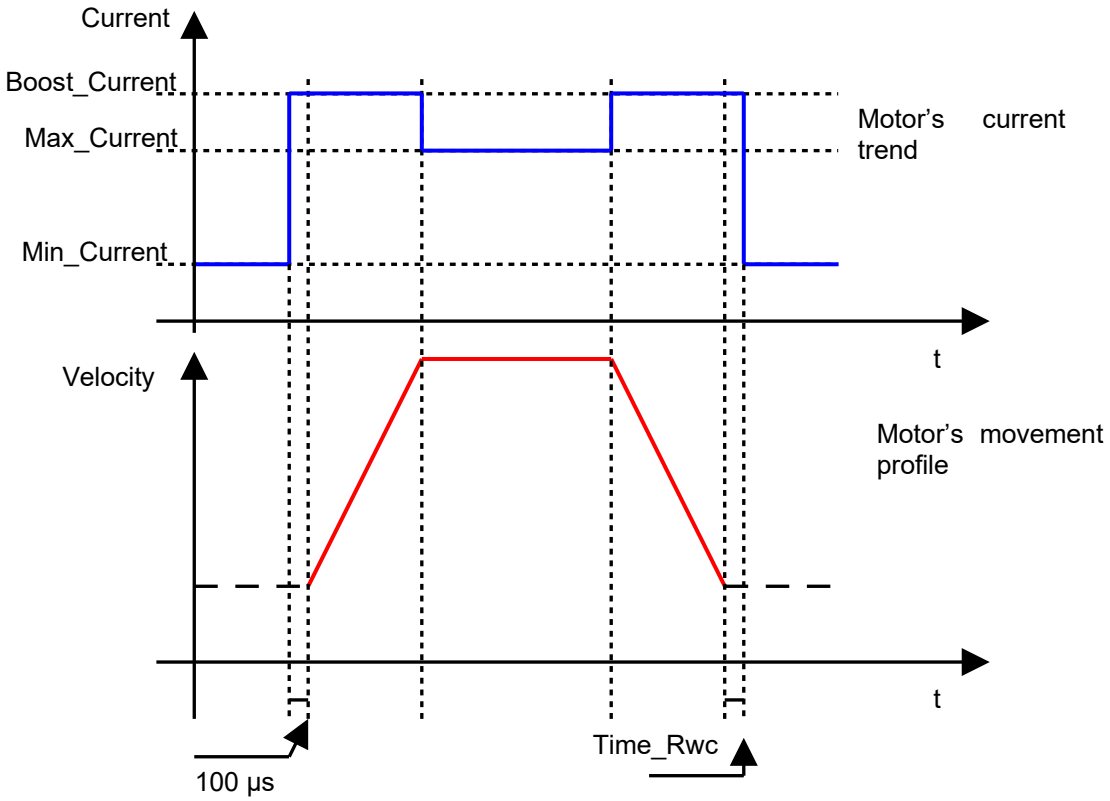


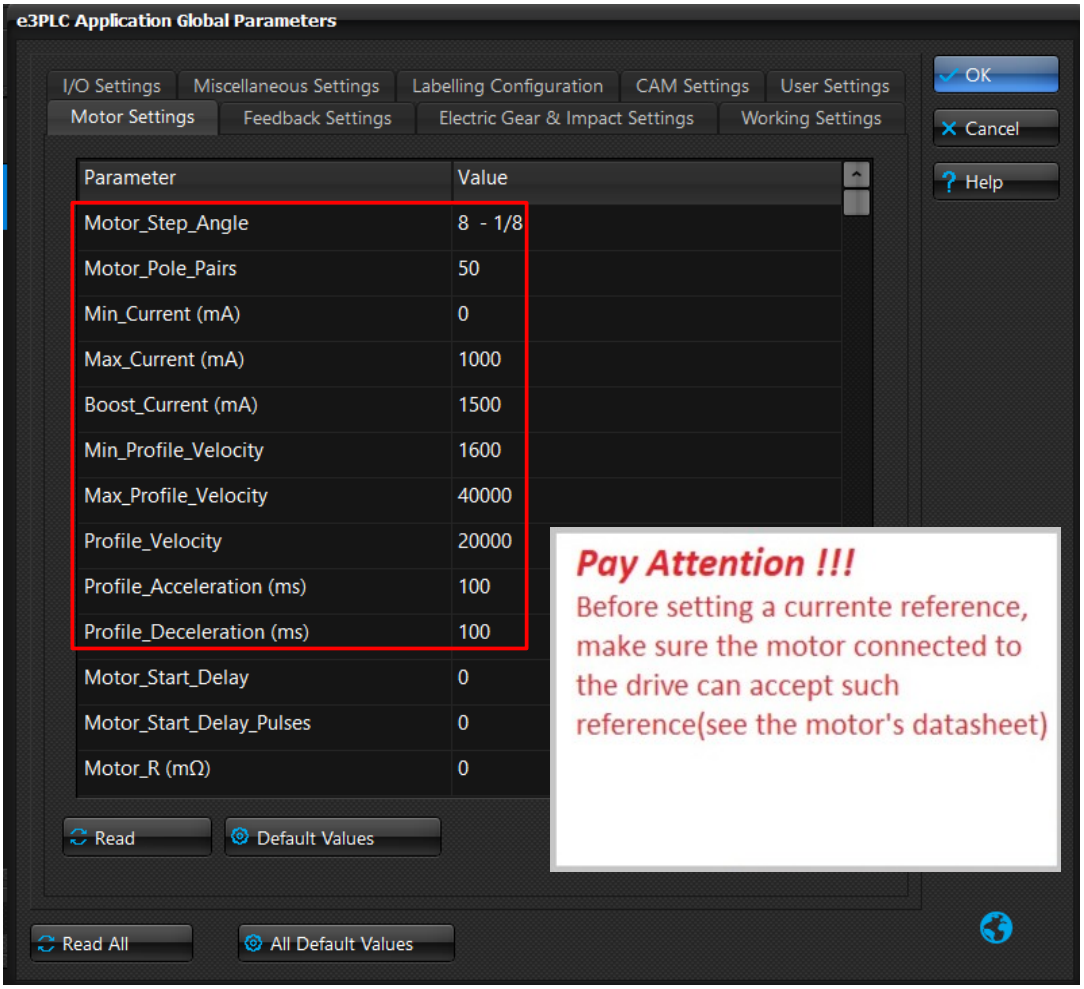
Diagram of the motor's current reference in Open Loop

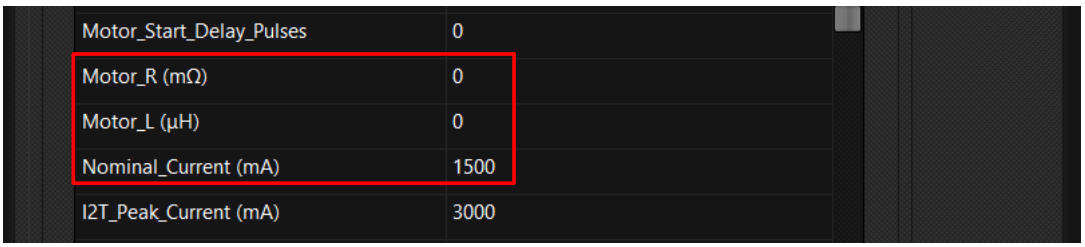


8.1.2 Open Loop Global Parameters Settings

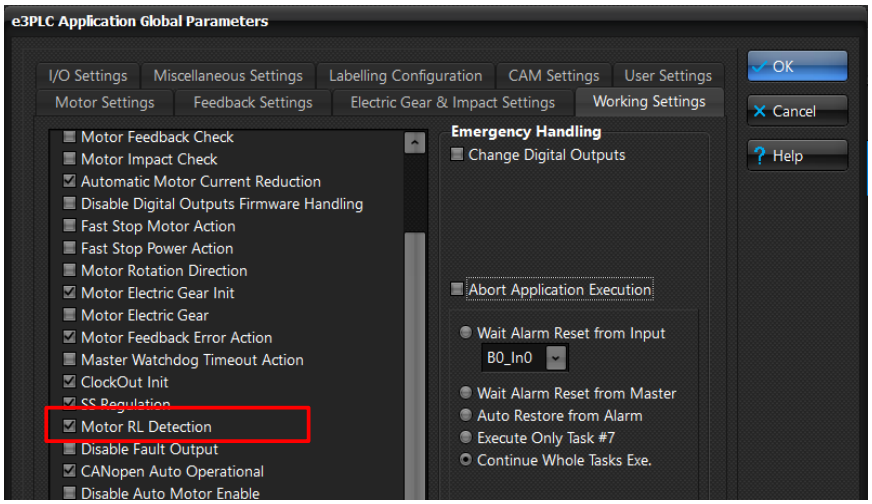
In the next picture, are showed the minimal setting necessary for the moving of the stepper motor in open loop. For the description of each parameters, see the relative objects explanation.

For the startup setting of motor parameters and profile parameters, push on *Global Parameters* button and open the folder Motor_Settings.

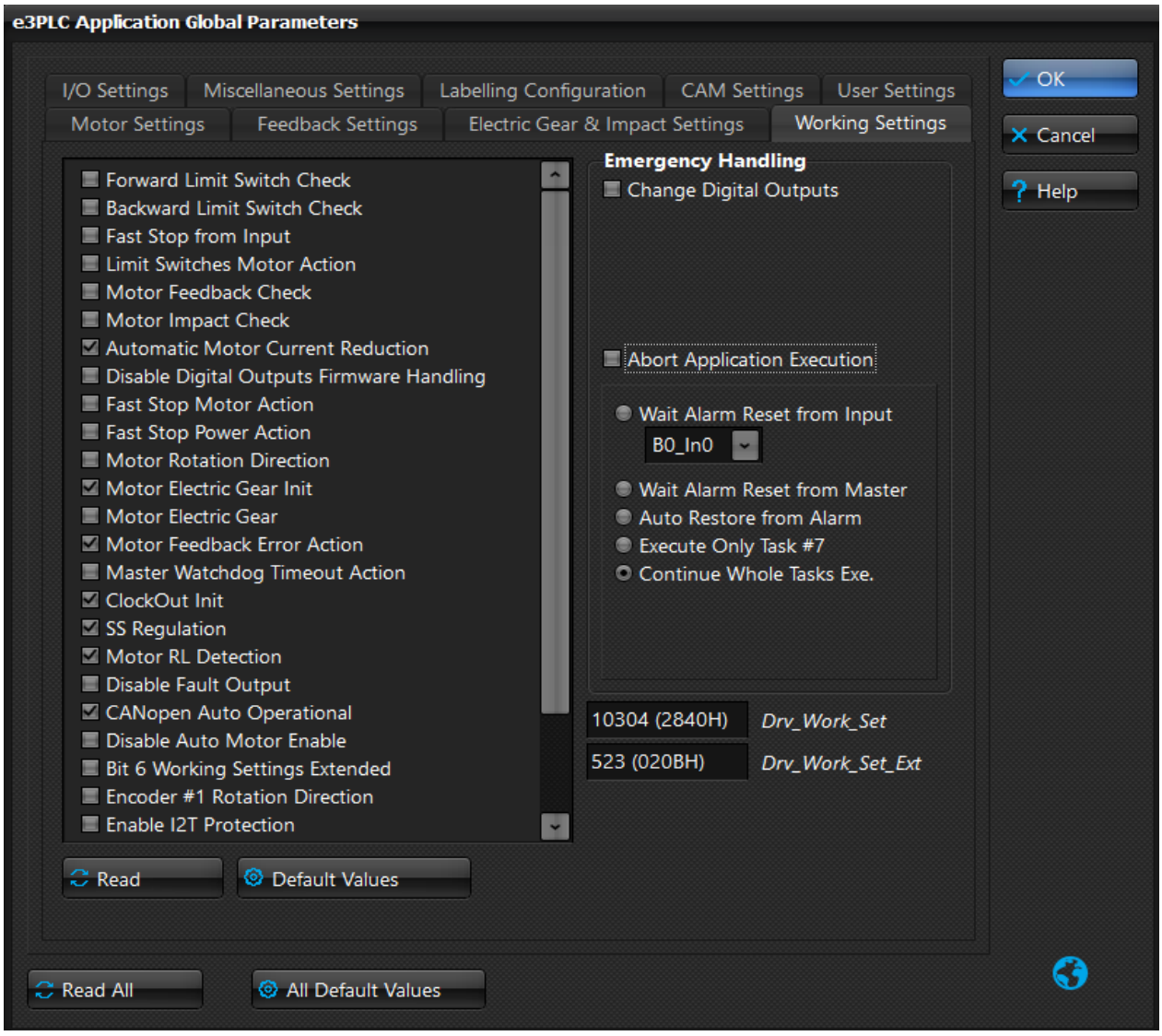




About the parameters Motor_R and Motor_L, it's possible omitted the value if it's enabled the automatic motor parameters detection. (bit *Drive_Working_Settings.Motor_RL_Detetection* =1).



In the *Global_Parameters/Working_Settings folder* is done the settings for the activation of some functionalities



8.2 Closed Loop Modality

The Closed Loop modality is working if the bit *Drive_Working_Settings.Motor Feedback_Check =1.*
In this modality, the stepper motor works as a Servo motor, and the drive supply only the current required at the motor shaft.
For this modality it's necessary a motor with installed on the rear shaft the incremental encoder.
In closed loop the motor can be controlled in position, velocity and torque.

In closed loop is necessary the setting of other parameters regarding the encoder resolution and is necessary the tuning of the GAIN objects that are dependend by load to move and so by application

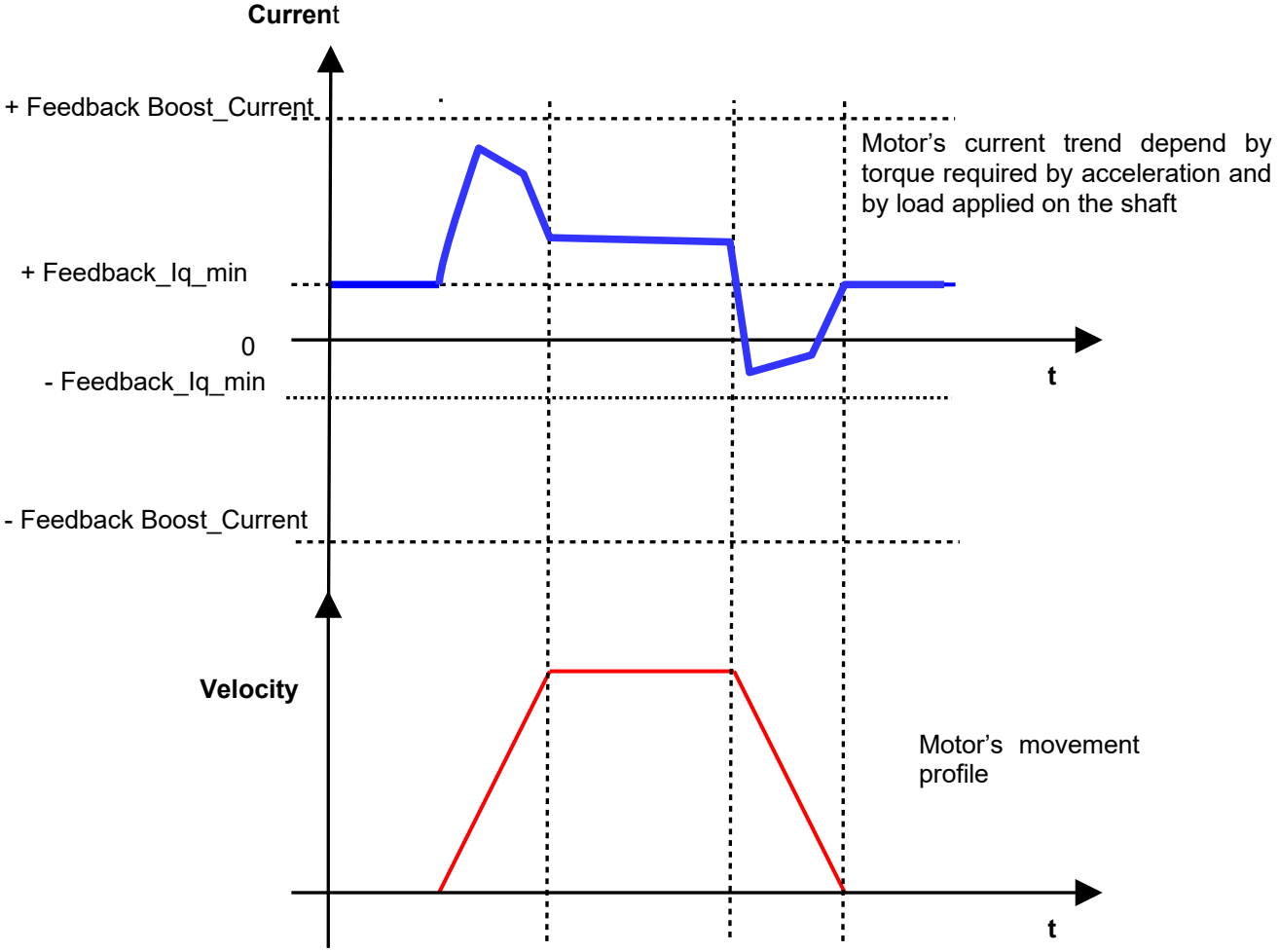
Note :

The 'Closed loop' modality (Feedback feature) and 'Motor_Stall_detection' feature cannot be both active at the same time.Try to enable 'Feedback' feature with 'Motor Stall detection' feature already enabled or vice versa will issue an 'Feedback_Error' alarm (bit5 of Error_Register object and bit14 of Feedback_Status object are set to 1). See §9.4 for more details about 'Motor Stall detection' feature.

Below the list of Objects necessary for the configuration of the drive to works in Closed Loop.

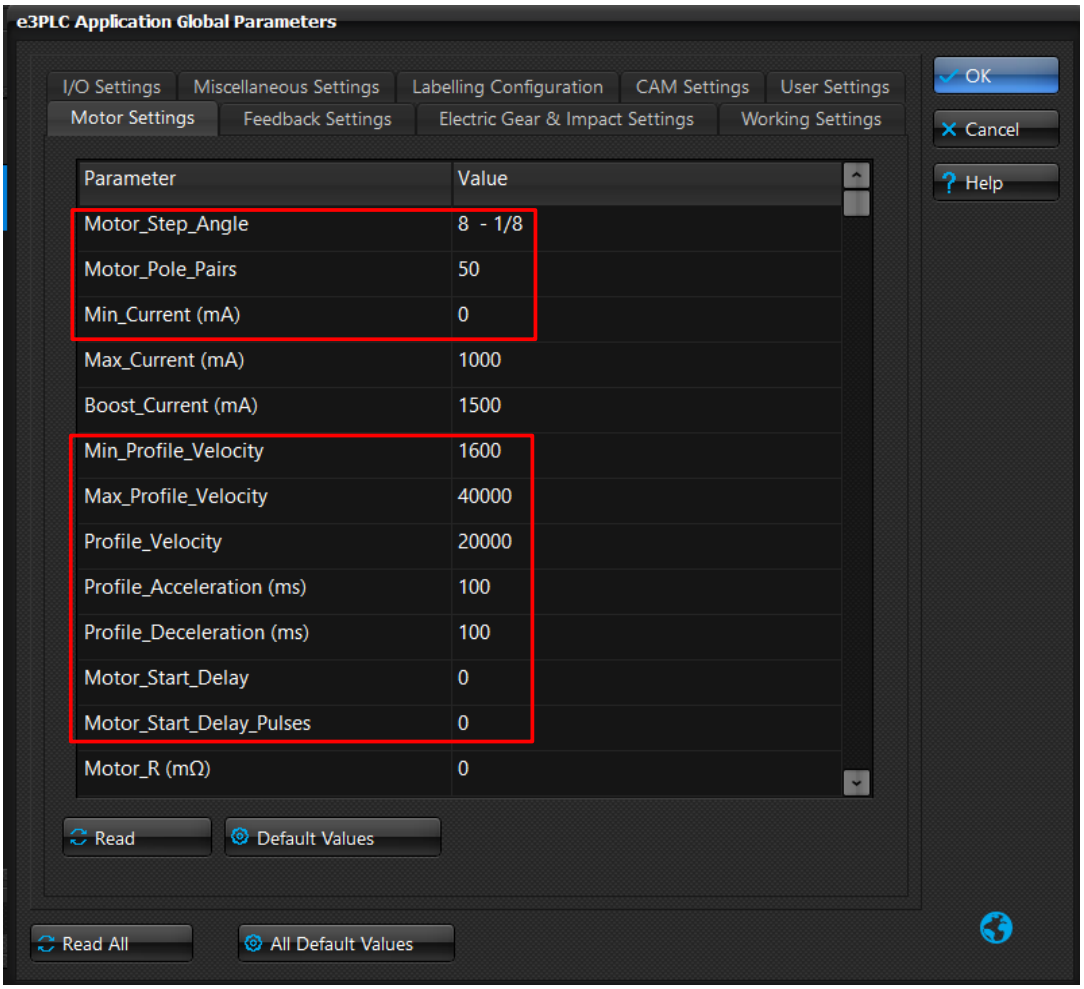
Object name	Note
Motor Poles	Mandatory
Motor_Step_Angle	Mandatory
Min_Current	To set to keep the motor in torque when the closed loop is disabled for example at switch-on or when happen the Feedback Error Alarm
Max_Current	Not used for closed loop, can be keep to 0
Boost_Current	Not used for closed loop, can be keep to 0
Nominal_Current	Mandatory
Min_Profile_Velocity	Mandatory
Max_Profile_Velocity	Mandatory
Profile_Velocity	Mandatory
Motor_R	The values can be omitted if it's enabled the automatic motor parameters detection. (bit *Drive_Working_Settings.Motor_RL_Detetection* =1).
Motor_L	
Motor_Start_Delay	Optional set it to 0. To define only if necessary delay the start of the motor
Motor_Start_Delay_Pulses	Optional set it to 0. To define only if necessary delay the start of the motor respect the master encoder rotation
Drive_Working_Settings	This Object permit the settings of some working features and also to enable the Close Loop.
Drive_Working_Settings_Extended	This Object permit the settings of some working features
Feedback_Source_PPR	Mandatory
Feedback_Calibration_Current	Mandatory.
Feedback_Calibration_Speed	Mandatory. Generally a default value of 5 rpm works well
Feedback_Settings	Mandatory
Feedback_Limit_Speed	Mandatory. Default value is set to 3000 rpm
Feedback_Boost_Current	Mandatory
Feedback_Position_Error_Limit	Has to be defined in Mode 0,1 and 2
Feedback_Velocity_Error_Limit	Has to be defined in Mode 3 and 4
Feedback_Encoder_Filter_Time	Mandatory
Feedback_Current_Filter_Time	Mandatory
Feedback_Iq_min	Mandatory
Feedback_Kp	Mandatory .. GAIN to tuning depending by application
Feedback_Kv	Mandatory.. GAIN to tuning depending by application
Feedback_Ki	Mandatory .. GAIN to tuning depending by application
Feedback_Ki_Limit	Mandatory ..GAIN to tuning depending by application
Feedback_Kalfas	Mandatory
Feedback_Kffw_Acc	Mandatory. In the first testing can be keep to 0
Feedback_Kffw_Dec	Mandatory. In the first testing can be keep to 0
Feedback_Kfbw_Acc	Mandatory. In the first testing can be keep to 0
Feedback_Kfbw_Dec	Mandatory. In the first testing can be keep to 0

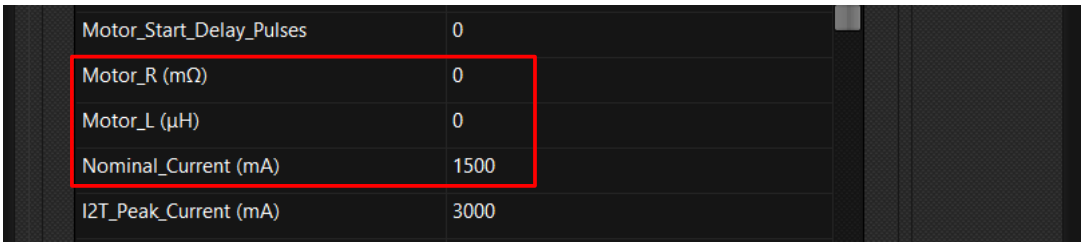
Diagram of the motor's current reference in Close Loop



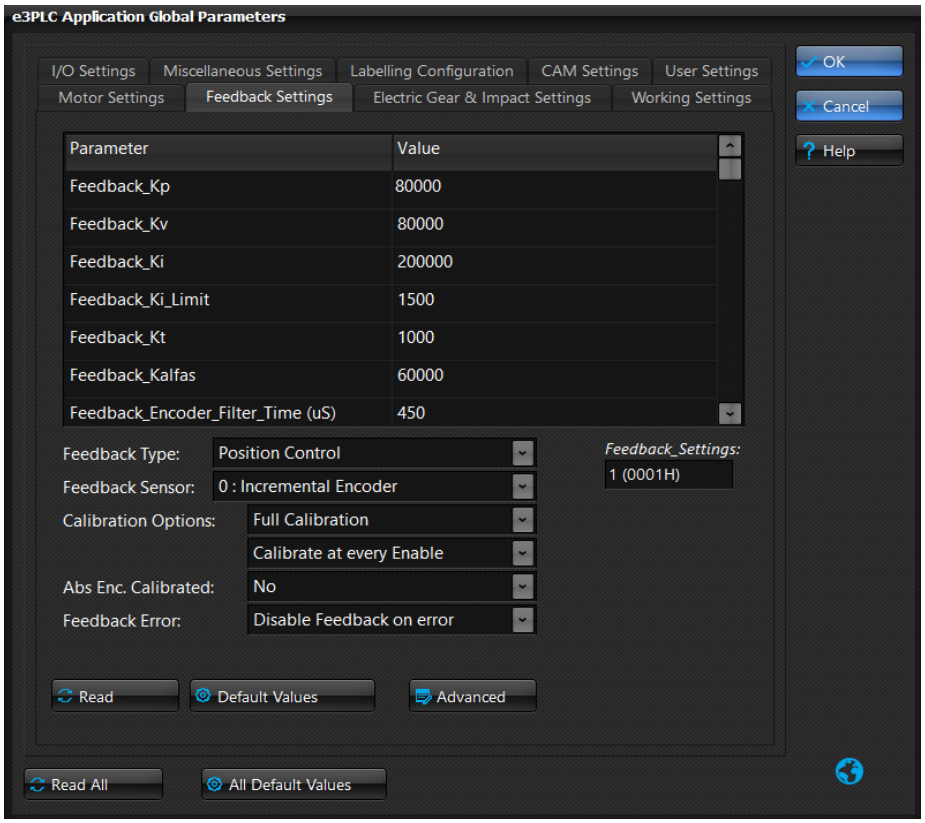
8.2.1 Close Loop Global Parameters Settings

In the next picture, are showed the settings necessary for the moving of the stepper motor in close loop.
For the description of each parameters, see the relative objects explanation.
Some parameters are to set in the *Global Parameters/*Motor_Settings folder.

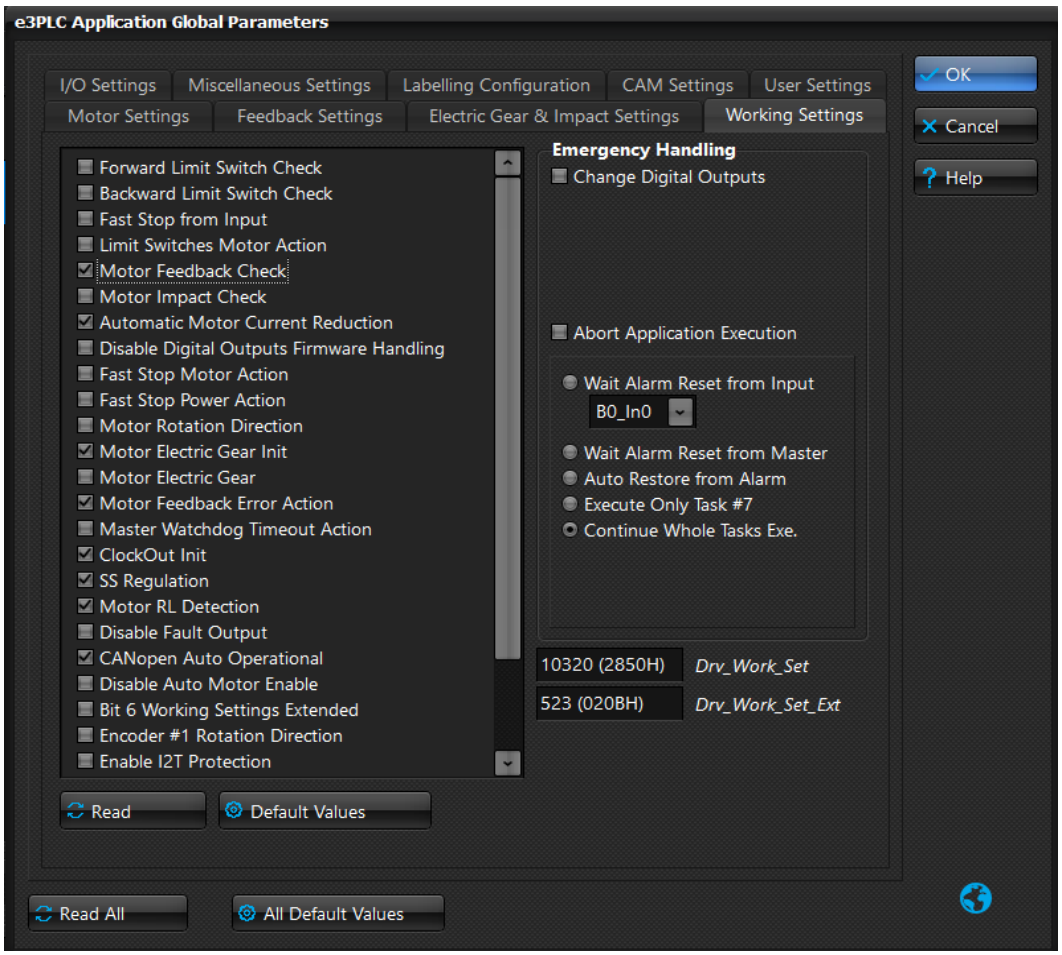




And the other parameters are to set in the *Global Parameters/Feedback_Settings* folder.



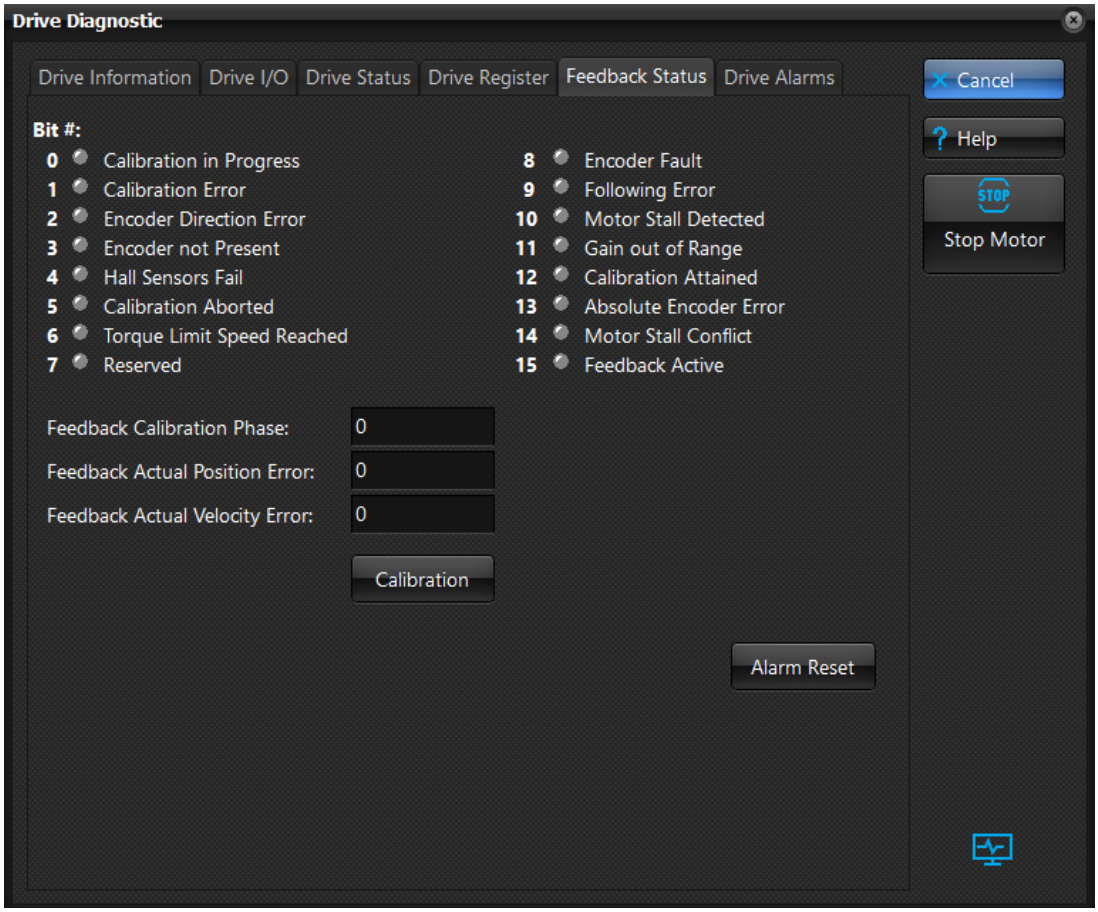
In the *Global Parameters/Working_Settings* folder is done the settings for the activation of some functionalities



8.2.2 Closed Loop Calibration

Each time the closed loop is came enabled, the calibration procedure is executed. The scope of the procedure is to align the stator position with the motor shaft position. During this calibration procedure, it's necessary that the motor shaft is free to moves. During the calibration procedure, the motor will do a little movement at very low speed.

In the Drive_Diagnostic/Feedback_Status windows is showed the Feedback's status
If the led Feedback_Active is green means that the system is working in closed loop otherwise some other red leds will inform about the problem happen. See the object *Feedback_Status* for details.



The Calibration procedure it's necessary to align the encoder mounted on motor rear shaft at the stator position.
Are available two Calibration Procedure:

- Feedback_Calibration_Strong
- Feedback_Calibration Lite

Feedback_Calibration_Strong

In this modality a little movement of the motor shaft is done. The motor shaft has to be free to move.
During the procedure, the system recognize if the motor is on mechanic limit and try to move in the opposte direction. After each switch-on of the drive, this procedure has to be executed for the activation of the Closed loop.
The bit *Feedback_Status*.Strong_Calibration_Done is set at the end of sequence executed with success.
The bit *Feedback_Status*.Active is set at the end of sequence executed with success.

Feedback_Calibration_Lite

In this modality is only done the alignment between the stator position and the rotor position without the motor shaft is moved.

This procedure can be done only if it is done at least once the Feedback_Calibration_Strong procedure from the switch-on of the drive.
The bit *Feedback_Status*.Active is set at the end of sequence executed with success.

The bit *Feedback_Status*.Strong_Calibration_Done give info about the execution of Feedback_Calibration_Strong procedure.

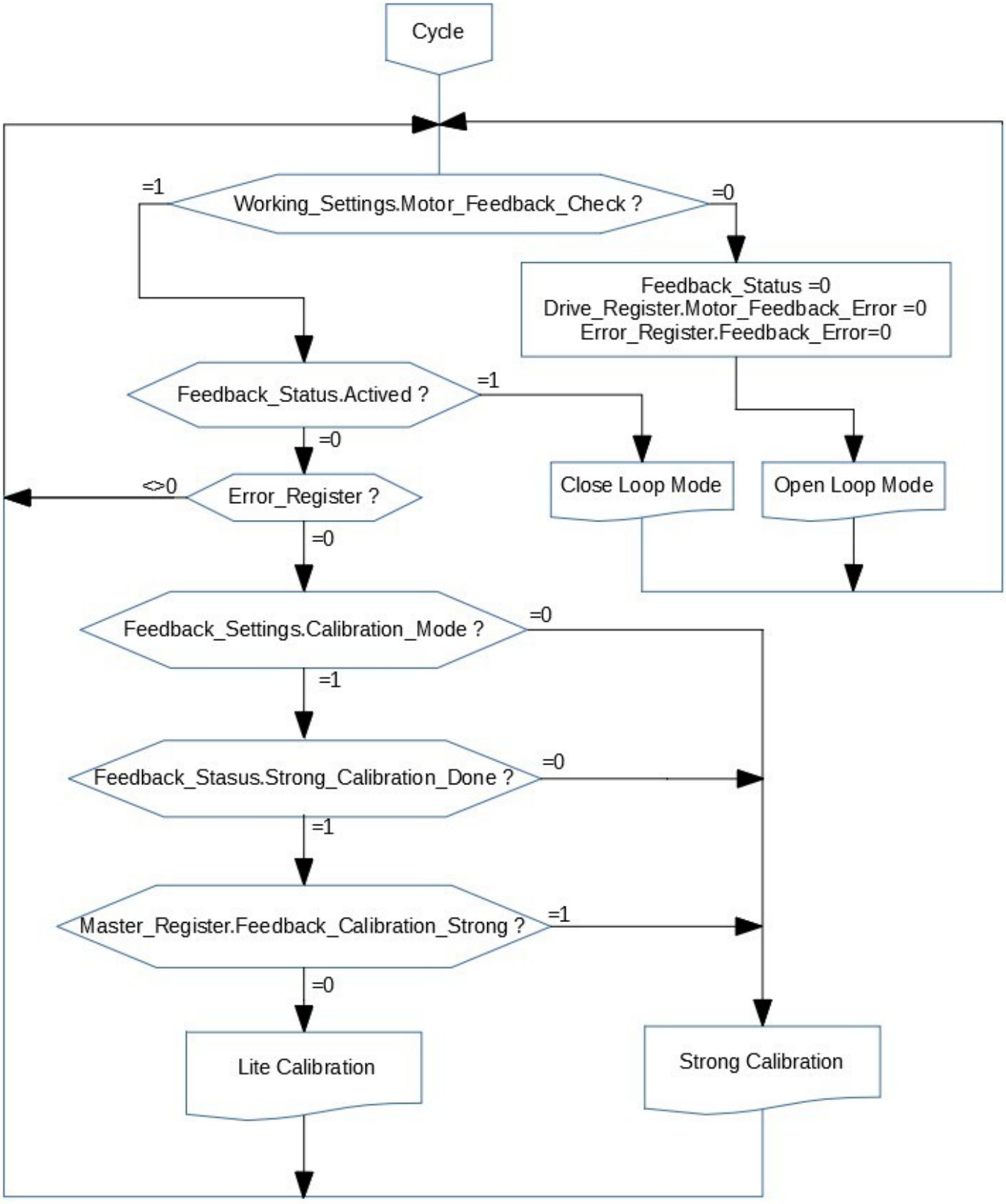
Some considerations:

The Calibration procedure is executed:
- Each time the Close loop became enabled (bit *Drive_Working_Settings*.Motor_Feedback_Check that switch from 0 to 1) if there are not alarm (*Error_Register*=0).
- Each time a Reset_Alarm is done and the bit *Error_Register*.Feedback_Error =1

The bit *Feedback_Status*.Strong_Calibration_Done it's cleared if *Error_Register*.Motor_Feedback_Error is set together with one of these alarms:
- *Feedback_Status*.Encoder Not Present
- *Feedback_Status*.Encoder Fault

so if the *Feedback_Status*.Strong_Calibration_Done is 0, the next time that will be necessary enable the close loop, the *Feedback_Status*.Strong procedure will be done.

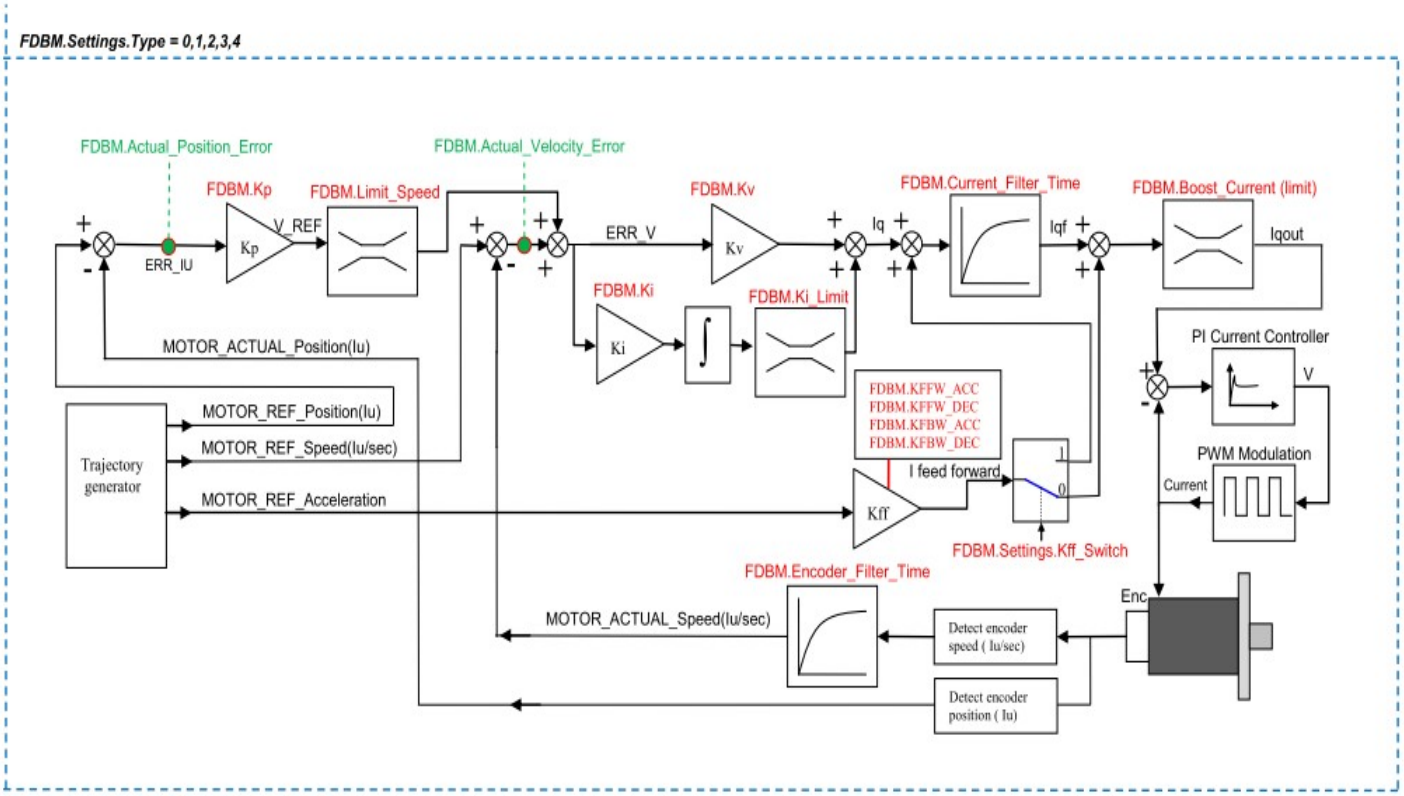
8.2.3 Closed Loop Calibration Diagram



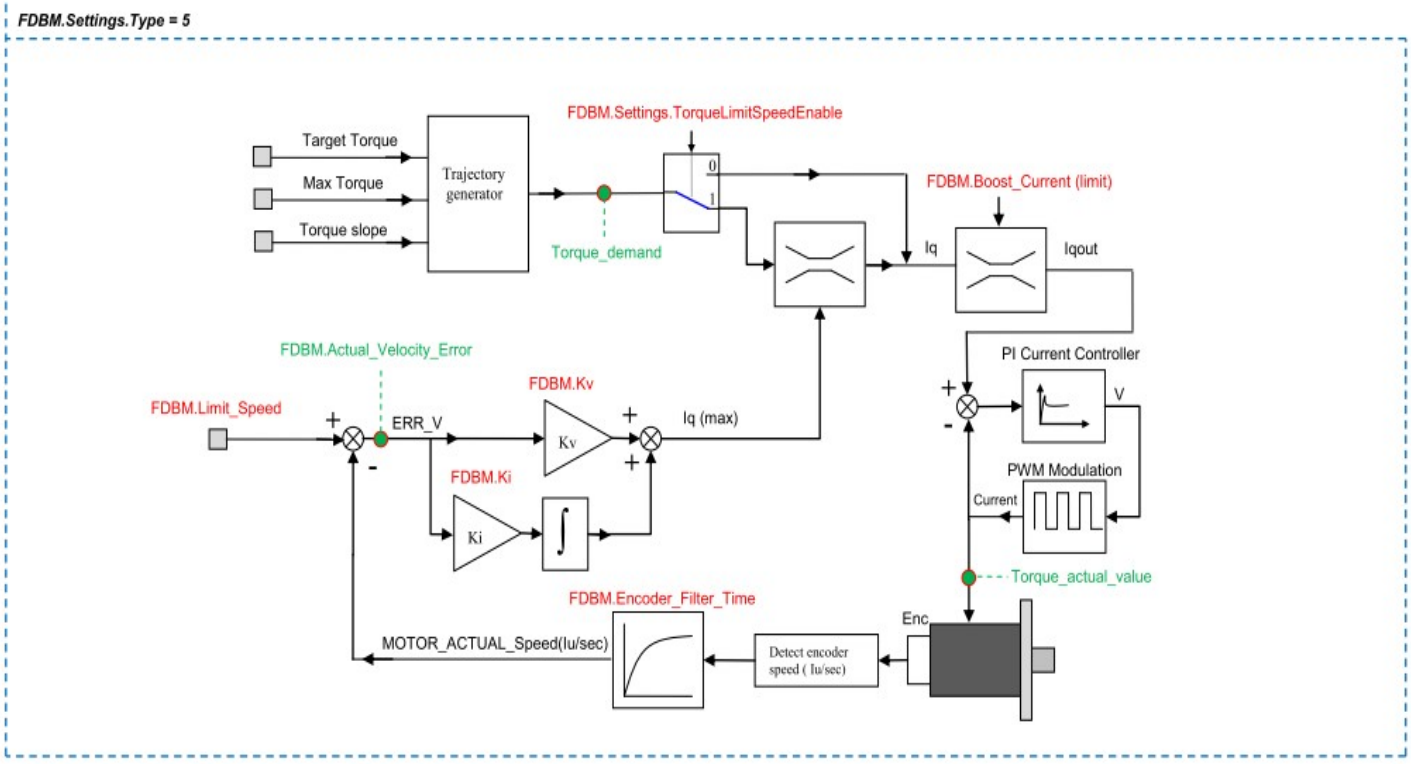
8.2.4 GAIN tuning

When the system is used in closed loop, it's necessary the tuning of the GAIN that are depending by type of load applied on the motor shaft.
For the tuning of the GAIN is a good way use the integrated Motor feedback tuning windows.

FDBM.Settings.Type = 0,1,2,3,4



FDBM.Settings.Type = 5



Below the list of Objects that can need to be tuned

Object name	Note
Feedback_Iq_min	
Feedback_Kp	Mandatory .. GAIN to tuning depending by application
Feedback_Kv	Mandatory.. GAIN to tuning depending by application
Feedback_Ki	Mandatory .. GAIN to tuning depending by application
Feedback_Ki_Limit	Mandatory ..GAIN to tuning depending by application
Feedback_Kalfas	
Feedback_Kffw_Acc	Mandatory. In the first testing can be keep to 0
Feedback_Kffw_Dec	Mandatory. In the first testing can be keep to 0
Feedback_Kfbw_Acc	Mandatory. In the first testing can be keep to 0
Feedback_Kfbw_Dec	Mandatory. In the first testing can be keep to 0
Feedback_Encoder_Filter_Time	
Feedback_Current_Filter_Time	

A good way for the tuning where is possible, is to have the motor that can run forward and backward.
One method for the tuning is:

1. Set all the feedforward GAIN to 0 (Feedback_Kffw_Acc/Dec=0 and Kfbw_Acc/Dec = 0
2. Set Feedback_Kp=0 and Feedback_Ki=0 and start the tuning of Feedback_Kv to check how the motor respond in velocity(check the velocity Error)
3. Then introduce the Feedback_Kp to check the position error
4. Then introduce the Feedback_Ki to reduce the following at constant speed
5. If necessary work on the feed forward parameter (Feedback_Kfxx) to decrease the position or velocity error during the acceleration/deceleration ramp.

8.2.5 Feedback_Type Modality

With the field Type of the object *Feedback_Settings*, can be setting some Close loop work's modality(from 0 to 5).

The modality 0,1,2 are used when is necessary keep under control the *Feedback_Actual_Position_Error*.

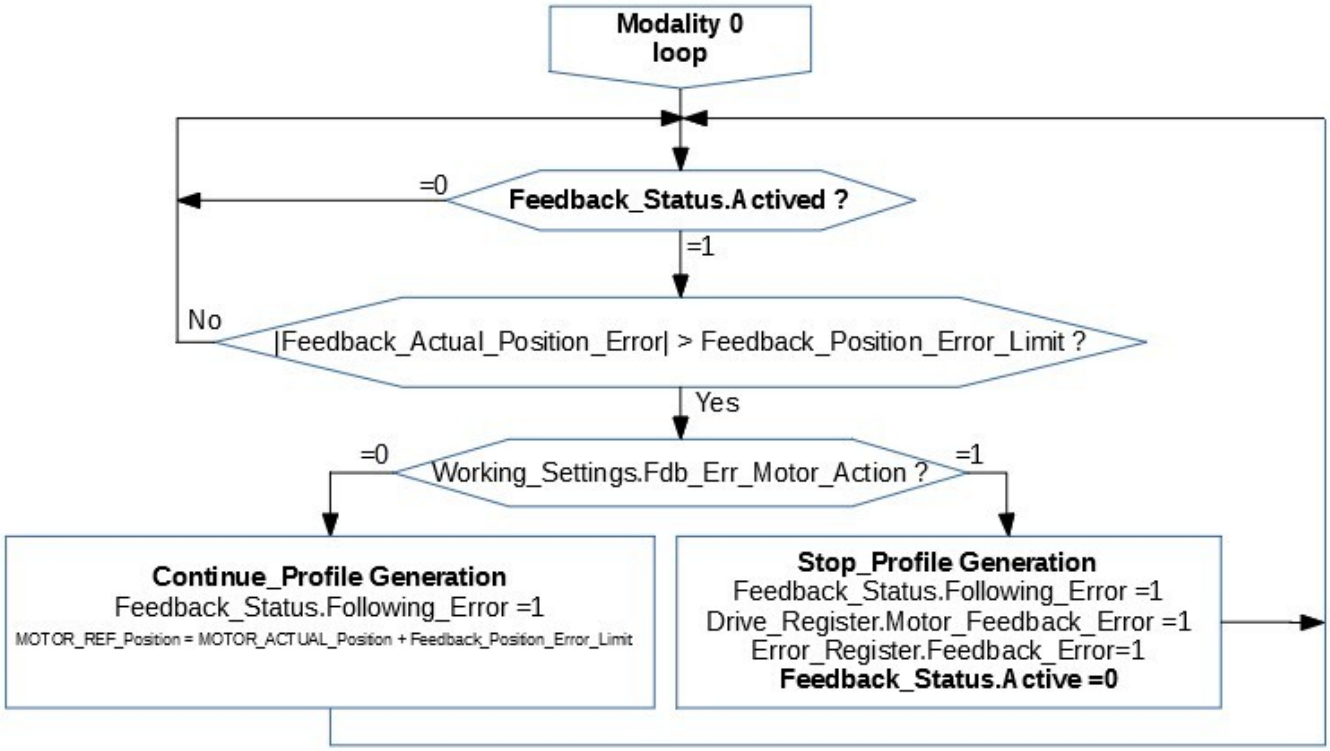
The modality 3,4 are used when is necessary keep under control the *Feedback_Actual_Velocity_Error*. When these modality 3 and 4 are used means that the position error is not important so the GAIN *Feedback_Kp* can be keep to 0.

Modality 0

This modality, in also called "Compatible" modality because is compatible with the firmware version lower then V2.xx.

In this modality is controlled the *Feedback_Actual_Position_Error* depending by bit *Drive_Working_Settings.Fdb_Err_Mot_Action.*

The following diagram show how the modality works.

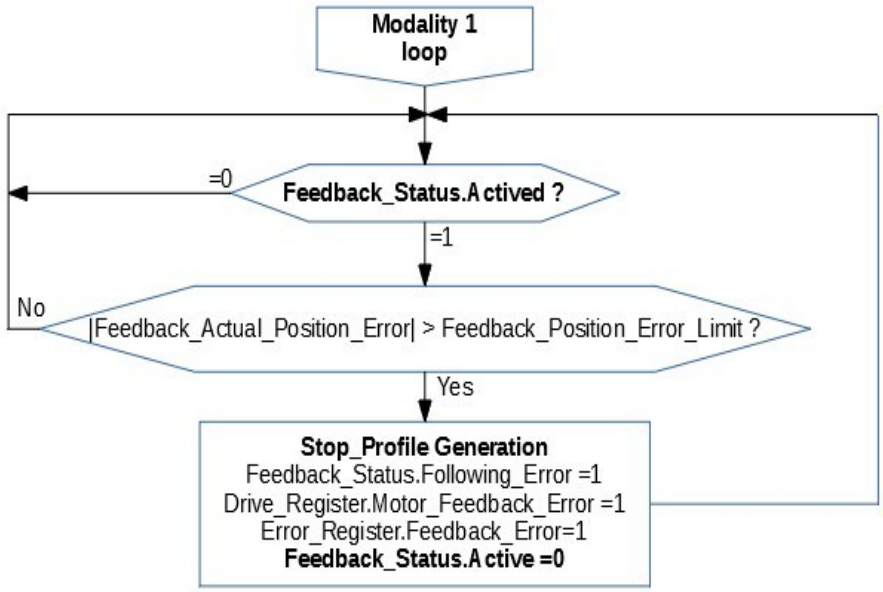


The *Drive_Register.Motor_Feedback_Error*, can go to 1 also if least one of the bit (1,3,5,8) of the *Feedback_Status* object is 1.

When the bit *Drive_Register*.Motor_Feedback_Error is ON, the Close loop is disabled and at the motor is supplied the current defined with the object *Min_Current*.

Modality 1

In this modality is controlled the *Feedback_Actual_Position_Error* and when it became out of limit, the profile generation is stopped(the motor stops) and the alarm is issued.

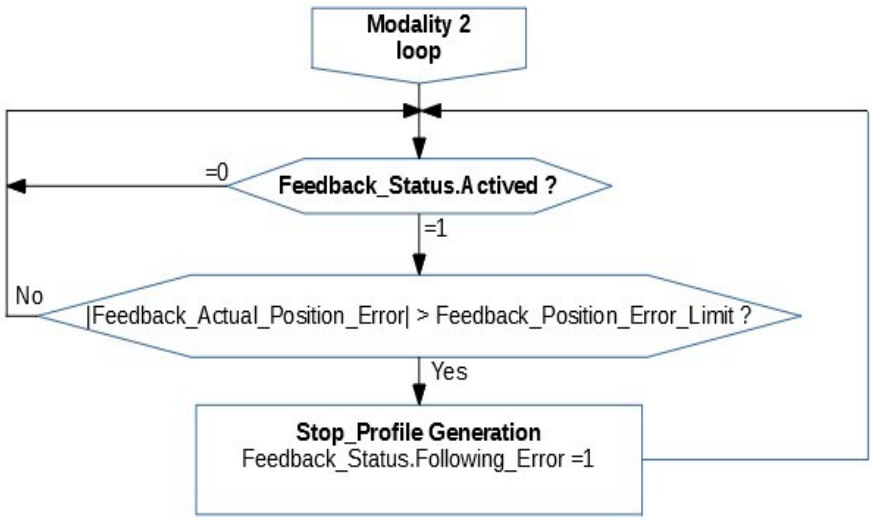


The *Drive_Register.Motor_Feedback_Error*, can go to 1 also if least one of the bit (1,3,5,8) of the *Feedback_Status* object is 1.

When the bit *Drive_Register.Motor_Feedback_Error* is ON, the Close loop is disabled and at the motor is supplied the current defined with the object *Min_Current*.

Modality 2

In this modality is controlled the *Feedback_Actual_Position_Error* and when it became out of limit, the profile generation is stopped(the motor stops) but the Close loop remains active.

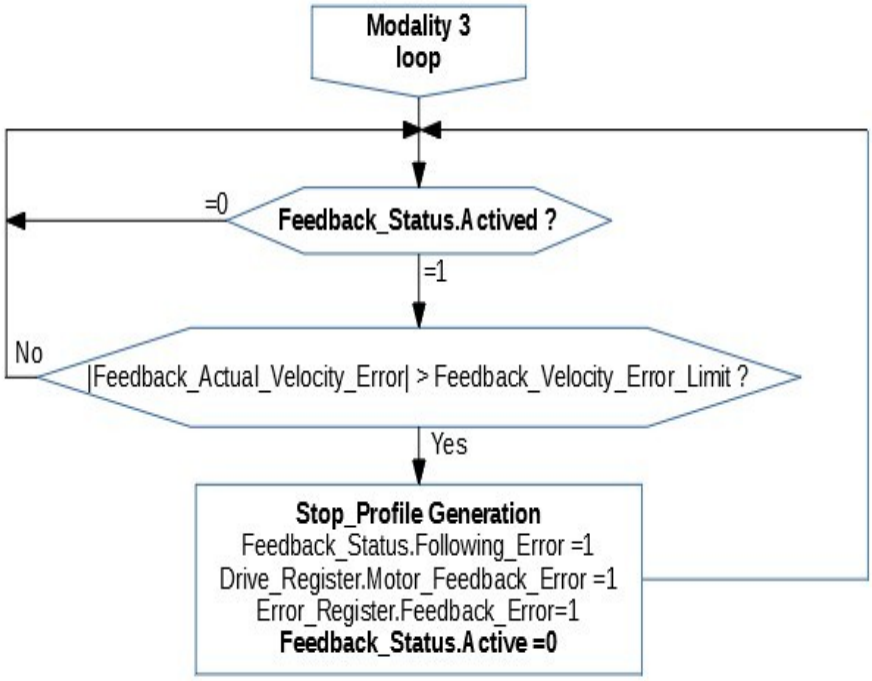


The *Drive_Register.Motor_Feedback_Error*, can go to 1 if least one of the bit(1,3,5,8) of the *Feedback_Status* object is 1.

When the bit *Drive_Register.Motor_Feedback_Error* is ON, the Close loop is disabled and at the motor is supplied the current defined with the object *Min_Current*.

Modality 3

In this modality is controlled the *Feedback_Actual_Velocity_Error* and when it became out of limit, the profile generation is stopped(the motor stops) and the alarm is issued.

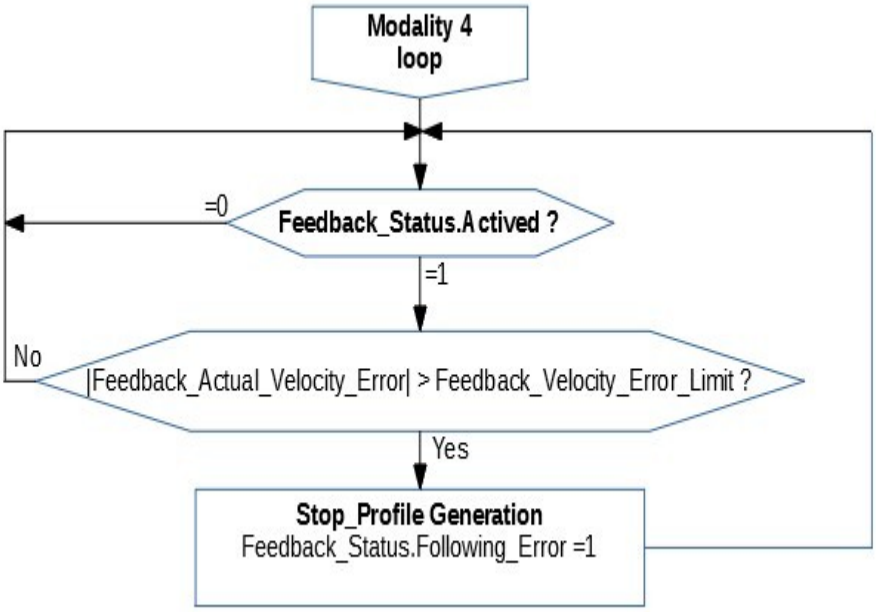


The *Drive_Register.Motor_Feedback_Error*, can go to 1 if least one of the bit(1,3,5,8) of the *Feedback_Status* object is 1.

When the bit *Drive_Register.Motor_Feedback_Error* is ON, the Close loop is disabled and at the motor is supplied the current defined with the object *Min_Current*.

Modality 4

In this modality is controlled the *Feedback_Actual_Velocity_Error* and when it became out of limit, the profile generation is stopped(the motor stops) but the Close loop remains active.

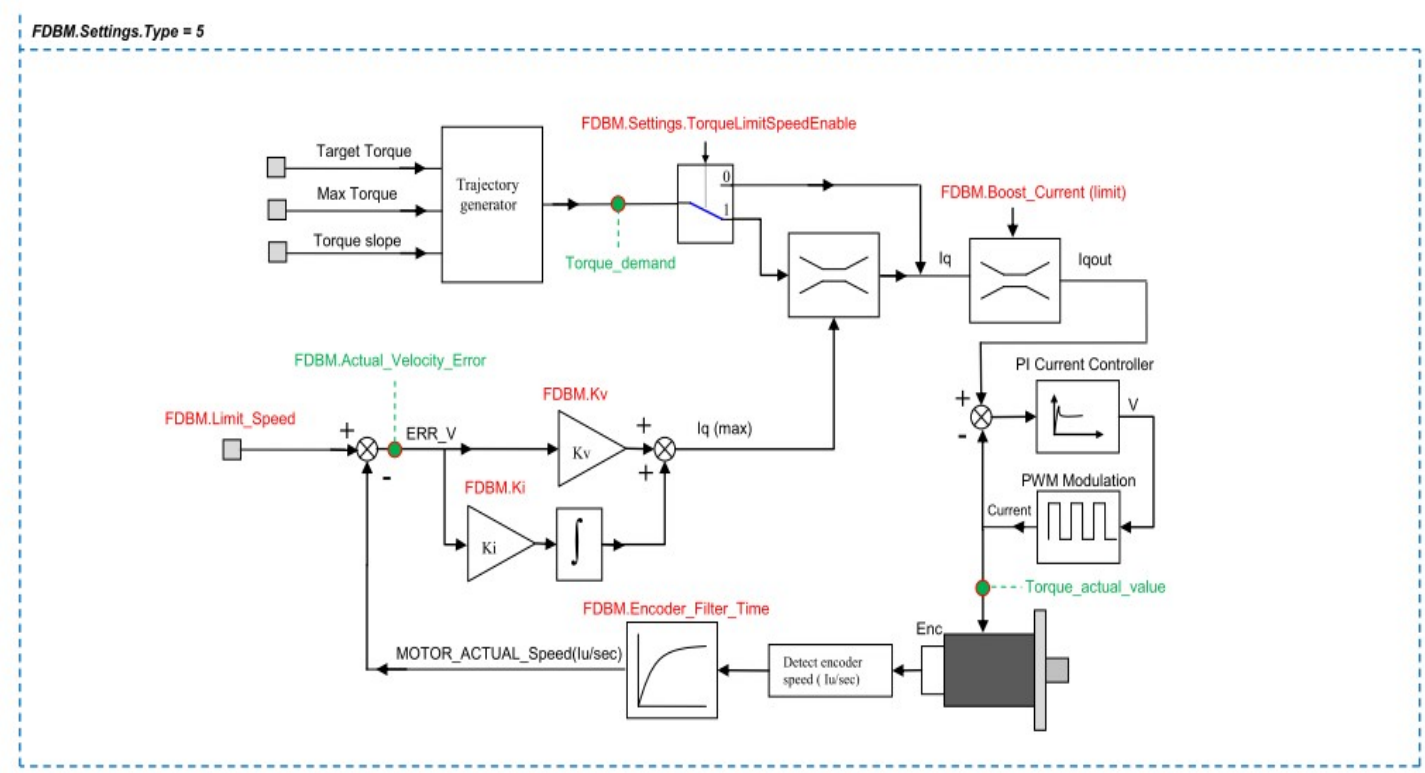


The *Drive_Register.Motor_Feedback_Error*, can go to 1 if least one of the bit(1,3,5,8) of the *Feedback_Status* object is 1.

When the bit *Drive_Register.Motor_Feedback_Error* is ON, the Close loop is disabled and at the motor is supplied the current defined with the object *Min_Current*.

Modality 5

In this mode, the torque value (*Target_torque)* is preset as a set value and reached via a ramp function (*Torque_slope*) and trajectory generator.



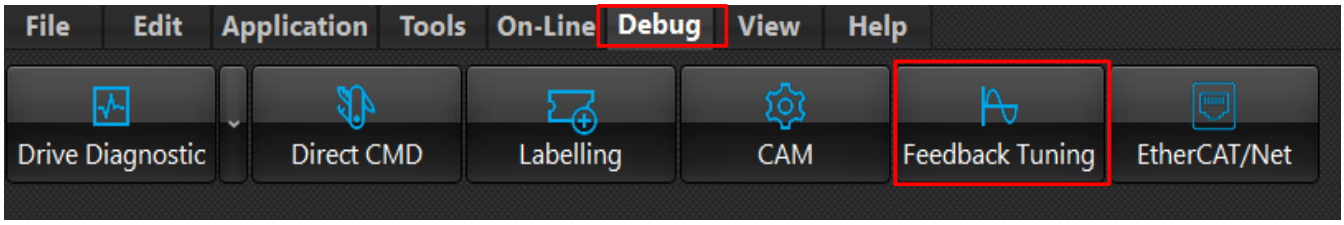
Related objects :

Object name	Note
Target_torque	
Max_Torque	
Torque_demand	
Torque_actual_value	
Torque_slope	
Velocity_Actual_Value	
Velocity_demand_value	
Feedback_Ki	
Feedback_Kv	
Feedback_Limit_Speed	
Feedback_Boost_Current	
Feedback_Settings	
Feedback_Encoder_Filter_Time	
Feedback_Status	

Note: This modality is available only with firmware version V02r82 or superior.

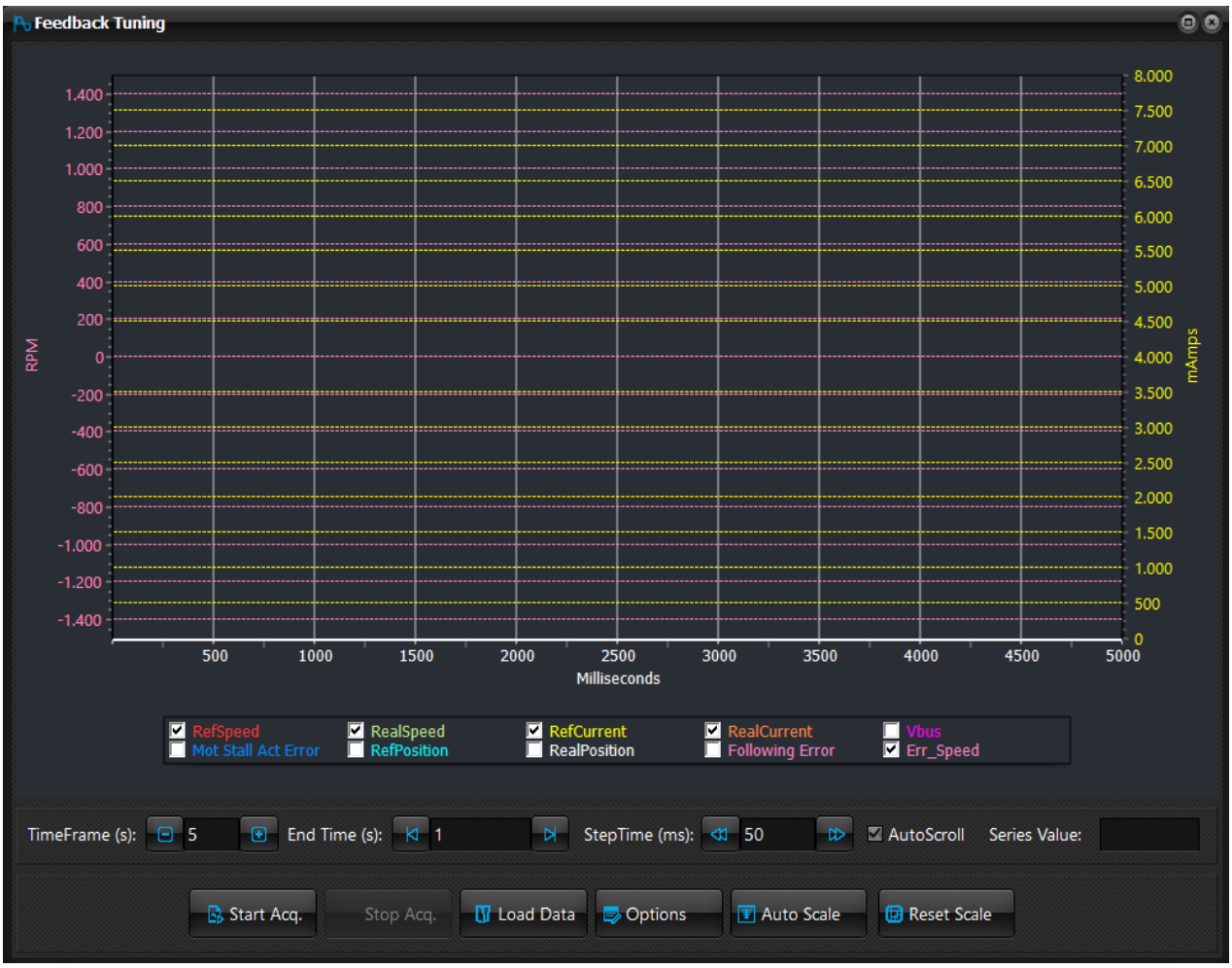
8.2.6 Scope Monitor

The integrated Motor Feedback Tuning Windows is used to watch the motion profile and for the tuning of the GAIN for Close loop.



In the Scope are showed 8 traces:

Trace Name	Unit	Decription	
Ref_Speed	rpm	Ref Speed Profile	
Real_Speed	rpm	Real motor profile	
Ref_Position	IU	Ref Position	
Real_Position	IU	Real motor position	
Following_Error	IU	Displacement between Motor_Ref_Position and Motor_Real_Position	
Ref_Current	A	Current supplied at the motor	
Real_Current	A	Motor Irms current	
Vbus	V	Vbus	
Mot_Stall_Act_Error	rad	Motor_Stall_Actual_Err_Angle	
Err_Speed			


Feedback Tuning
RPM
mAmps
Milliseconds
RefSpeed
RealSpeed
RefCurrent
RealCurrent
Vbus
Mot Stall Act Error
RefPosition
RealPosition
Following Error
Err_Speed
TimeFrame (s):
End Time (s):
StepTime (ms):
AutoScroll
Series Value:
Start Acq.
Stop Acq.
Load Data
Options
Auto Scale
Reset Scale

8.3 Types of motor movement

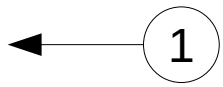
Once the movement parameters defined in §*8.0* have been set, motor movements can be carried out. Titanio series drives allow for a wide range of types of movements, using the ***MOVE*** instruction which will be explained further below. In order to better understand the motor control logic the following concepts must be borne in mind:

- The drive features an internal counter (*Position_Actual_Value* object) which is updated every time the motor makes a step.
- The Target position is the position that the motor must reach when a finite number of steps is launched. The Target position is updated every time a finite number of steps is launched.

8.3.1 Basic movements

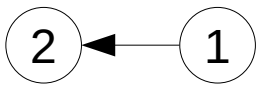
These motor movements can be directly carried out by the system without having to set other parameters than those defined in § 8.0 and without connecting any external device (such as encoders).
Here follows a description of the basic movements available:

Free running movements



The motor moves in free running forward or backward reaching the velocity set using the *Profile_Velocity* object with the acceleration ramp previously defined. The motor will stop only after the **STOP** command has been received.

Relative Step Movements



The motor moves by an **x** number of motor steps (angular shift of a step depends on the step angle set) forward or backward, trying to reach the movement velocity set using the *Profile_Velocity* object with the acceleration ramp previously defined. Movement velocity will only be reached if the steps to be carried out are equal to or greater than the sum of the acceleration and deceleration ramp steps. The motor will stop only when the number of steps to be carried out is equal to or lower than the steps of the deceleration ramp. However, the motor's movement can be stopped at any moment, using the command **STOP**. Every time that a step movement is launched, the system will update the Target position; once movement is over the motor position (*Position_Actual_Value* object) will be equal to the Target position.

If a step movement is launched while the motor is at a standstill, the Target position is set equal to the current position of the motor plus the steps still to be carried out.
If a step movement is re-launched, while the motor is already running, the Target position is updated with the following procedure:

1. **The motor is moving in free running forward and a forward step movement is launched.**
   The steps to be carried out are summed to the current position of the motor and stored into the Target position. The motor continues to run forward until the Target position is reached.

2. **The motor is moving in free running forward and a backward step movement is launched.**
   The steps to be carried out are deducted from the current position of the motor and stored into the Target position. The motor stops with a ramp, changes direction and continues to run until the Target position is reached.

3. **The motor is moving in free running backward and a forward step movement is launched.**
   The steps to be carried out are summed to the current position of the motor and stored into the Target position. The motor stops with a ramp, changes direction and continues to run until the Target position is reached.

4. **The motor is moving in free running backward and a backward step movement is launched.**
   The steps to be carried out are deducted from the current position of the motor and stored into the Target position. The motor continues to run until the Target position is reached.

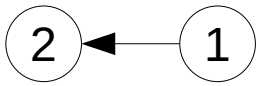
5. **The motor is moving in steps forward and a forward step movement is launched.**
   The Target position is updated by adding the steps still to be carried out. The motor continues to move forward until the Target position is reached.

6. **The motor is moving in steps forward and a backward step movement is launched.**
   The Target position is updated by deducting the steps still to be carried out.
   If the difference between the new Target position and the current motor position is greater than the steps required to slow down, the motor continues to move forward until the target position is reached.
   If the difference between the new Target position and the current motor position is lower than the steps required to slow down, the motor slows down, changes direction and moves until the Target position is reached.

7. **The motor is moving in steps backward and a forward step movement is launched**
   The Target position is updated by adding the steps still to be carried out.
   If the difference between the new Target position and the current motor position is negative and the absolute value is greater than the steps required to slow down, the motor continues to move backward until the target position is reached.
   If the difference between the new Target position and the current motor position is either positive or negative, but the absolute value is lower than the steps required to slow down, the motor slows down, changes direction and moves until the Target position is reached.

8. **The motor is moving in steps backward and a backward step movement is launched**
   The Target position is updated by deducting the steps still to be carried out. The motor continues to move forward until the Target position is reached.

Movements to the Target position



The system sets the Target position equal to **x** : the motor will therefore reach the Target position moving forward or backward depending on whether the motor's current position (*Position_Actual_Value* object) is lower than or greater than the Target position. The motor will try to reach the movement velocity set using the object *Profile_Velocity* object with the acceleration ramp defined. The movement velocity will be reached only if the steps to be carried out are equal to or greater than the sum of the steps of the acceleration and deceleration ramps. The motor will start to stop only when the number of steps still to be carried out is equal to or lower than the number of steps in the preset deceleration ramp. The current movement can always be stopped using the command **STOP**.

If a movement towards a Target position is re-launched, while the motor is already running, the system will set the new value for the Target position and the motor can behave in one of the following ways:

1. **The motor is moving forward and the new Target position is greater than the motor's current position.**
   The motor continues to move forward until the Target position is reached.

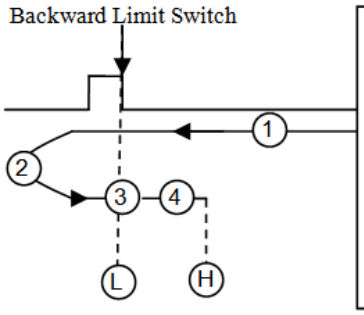
2. **The motor is moving forward and the new Target position is lower than the motor's current position.**
   The motor slows down, changes direction and moves until the Target position is reached.

3. **The motor is moving backward and the new Target position is greater than the motor's current position.**
   The motor slows down, changes direction and moves until the Target position is reached.

4. **The motor is moving backward and the new Target position is lower than the motor's current position.**
   The motor continues to move backward until the Target position is reached.

8.3.2 Homing Movements

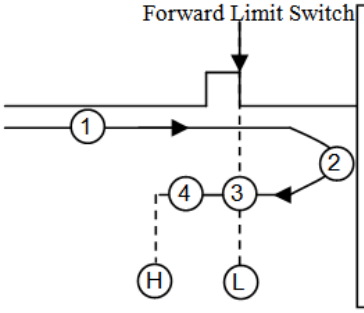
CMD Command 11



1: Move backward at speed defined by MOV command until Back Limit switch busy(Edge Position)
2: Return to the Edge Position at speed Homig_Speed_Out
3: Move forward at very slow speed until Back Limit switch free. (L_Pos)
4: Move to Homing_Offset[1] position at speed Homing_Speed_Out
5: Set Position_Actual Value = Homing_Preset_Position. (H_Pos)
6: End of Homing sequence, the Homing_Status object return the state of Homing procedure

About the working level of Back Limit_Switch see Drive_Inputs_Settings and Drive_Inputs_Level object. If the procedure is not completed within Homing_Overrun[1], the procedure is interrupted and in the Homing_Status object is setted the bit Limit_Switch_Not_Found

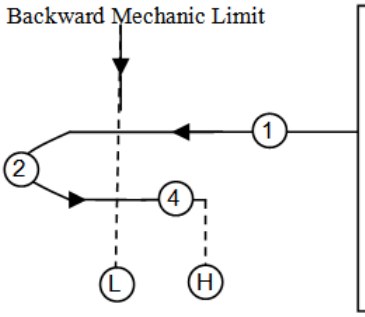
CMD Command 10



1: Move forward at speed defined by MOV command until Forward Limit switch busy(Edge Position)
2: Return to the Edge Position at speed Homig_Speed_Out
3: Move backward at very slow speed until For Limit switch free. (L_Pos)
4: Move to Homing_Offset[0] position at speed Homing_Speed_Out
5: Set Position_Actual Value = Homing_Preset_Position. (H_Pos)
6: End of Homing sequence, the Homing_Status object return the state of Homing procedure

About the working level of Forward Limit_Switch see Drive_Inputs_Settings and Drive_Inputs_Level object. If the procedure is not completed within Homing_Overrun [0], the procedure is interrupted and

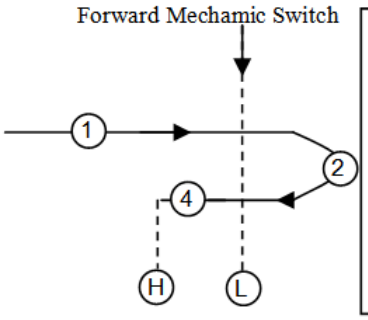
CMD Command 17



1: Move backward at speed defined by MOV command until Feedback_Position_Error < -1000 with torque limited by parameter Homing_Torque_Current_Limit.
2: Move forward at speed Homing_Speed_Out until Feedback_Position_Error became 0. (L_Pos)
4: Move to Homing_Offset[1] position at speed Homing_Speed_Out
5: Set Position_Actual Value = Homing_Preset_Position. (H_Pos)
6: End of Homing sequence, the Homing_Status object return the state of Homing procedure

If the procedure is not completed within Homing_Overrun [1], the procedure is interrupted and in the Homing_Status object is setted the bit Limit_Switch_Not_Found

CMD Command 16



1: Move backward at speed defined by MOV command until Feedback_Position_Error > 1000 with torque limited by parameter Homing_Torque_Current_Limit.

2: Move backward at Homing_Speed_Out, until Feedback_Position_Error became 0. (L_Pos)
4: Move to Homing_Offset[0] position at speed Homing_Speed_Out
5: Set Position_Actual Value = Homing_Preset_Position. (H_Pos)
6: End of Homing sequence, the Homing_Status object return the state of Homing procedure

If the procedure is not completed within Homing_Overrun [0], the procedure is interrupted and in the Homing_Status object is setted the bit Limit_Switch_Not_Found

Attention: For the Homing on Mechanic Position (16, 17), the speed has to be set low value and also the *Homing_Torque_Current_Limit* has to be set at low value to avoid the broke of mechanics and dangerous situations.

8.3.3 Movements with Trigger

The movements with start trigger are equivalent in functionality to those without trigger. The only difference is that the movements with trigger are not executed immediately but only when the trigger (a digital input assigned to this function, see *Driver_Inputs_Setting* object and B Appendix) is detected. While waiting for the trigger, the drive set the bit *Start_Trigger_Armed* (**MOVE** commands) or *Stop_Trigger_Armed* (**STOP** commands) of *Drive_Register* object.

8.3.4 Movements with SYNC

The movements with SYNC are equivalent in functionality to those without SYNC. The only difference is that the movements with SYNC are not executed immediately but only when SYNC object is write to 1. This feature is used to synchronize multi axes movements. Usually the command to write the SYNC object is sent as broadcast message (see §*10.0* and §*11.8*).
While waiting for the SYNC, the drive set the bit *SYNC_Armed* of *Drive_Register* object.

9.0 Drive Software Features

In this chapter the drives software features are described.

9.1 Impact Feature

The impact check consists in comparing the theoretical position of the motor (*Position_Actual_Value* object) with the position of the incremental encoder mounted on the same axis direction as the motor.
Before enabling the check (*Impact_Motor_Check* bit of *Drive_Working_Settings* object) you need to the define:

- ➔ The maximum displacement allowed: *Impact_Max_Displacement* object.
- ➔ The conversion factor encoder/motor steps: *Impact_Factor* object.
- ➔ The Encoder number: *Impact_Source* object.

Then it is possible to enable the check.
The Impact check is performed every 1-2 ms with following the formula:

Impact_Actual_Displacement = Position_Actual_Value - Encoder_Actual_Value[Impact_Source] * Impact_Factor

If Impact_Actual_Displacement > Impact_Max_Displacement then Impact Detected!!

When the impact is detected the drive will stop the motor immediately, will open the fault digital output and will set the *Motor_Impacted* bit of *Drive_Register* object.
No motor movement is executed until the master will set the *Master_Alarm_Reset* bit of *Master_Register* object. Then the drive will set:

Position_Actual_Value = Encoder_Actual_Value[Impact_Source] * Impact_Factor

If the *Impact_Source* is the BiSS Absolute Encoder, if the impact check is enabled at startup the *Position_Actual_Value* object is set equal to *BiSS_Encoder_Actual_Value* * *Impact_Factor* to avoid an impact error at switch-on.

See also *A Appendix Multiplexed I/O Allocations*

9.2 Electric Gear Feature

The electric gear feature allows to follow the position and/or speed of an external device (typically an incremental encoder or a clock). Before enabling the electric gear feature (*Motor_Gear* bit of *Drive_Working_Settings* object) you need to the define:

- ➔ The ratio between the encoder and the motor: *Motor_Gear_Ratio* object.
- ➔ The electric gear type: *Motor_Gear_Type* object.
- ➔ The electric gear k: *Motor_Gear_Kp* object.

Then it is possible to enable the electric gear feature.

Once the electric gear feature is enabled each time a movement is issued (**MOVE** Commands) the drive will follows the encoder speed/position reference. If the motor movement is in progress but no input clocks are detected the *Motor_Standby* bit of *Drive_Register* object is set. If the input frequency is higher than the *Max_Profile_Velocity* the motor speed is limited to *Max_Profile_Velocity*.

See also *A Appendix Multiplexed I/O Allocations*

9.3 Clockout Feature

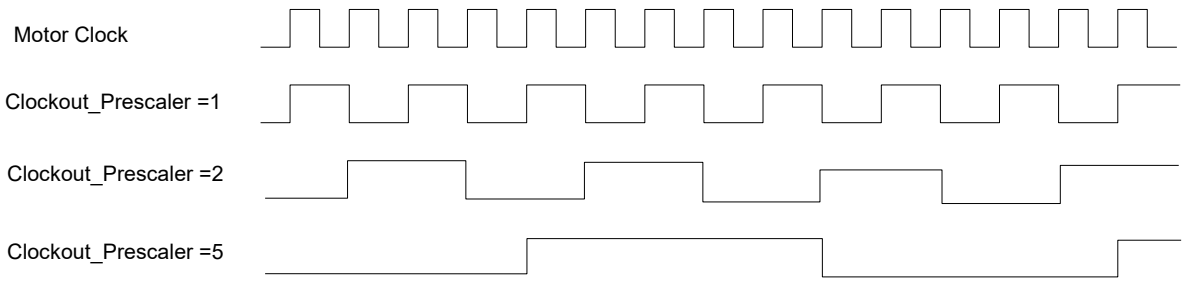
The Clockout feature allows the generation of a programmed clock depending on the motor position or replication of Incremental Encoder Input signals (Enc#0) on digital outputs (B0_OUT0 and B0_OUT1).

Objects list regarding ClockOut functionality are :

- *Clockout_Prescaler*
- *Drive_Working_Settings_Extended.Clockout_Init*

Clockout modes :

➔ The clock is generated on output B0_OUT1 while the motor is moving.
Clockout output is toggled every Clockout_Prescaler motor microsteps. The Max Clockout Frequency is limited to 10Khz so the max Motor Clock Frequency has to be less or equal to (10Kz * Clockout_Prescaler *2).



The Motor Clock Frequency depend on the motor speed and *Motor_Step_Angle* object so for example if the motor runs at the speed of 200 rpm we have :

Motor_Clock_Frequency = 666.66 Hz if Motor_Step_Angle = 1
Motor_Clock_Frequency = 1333.33 Hz if Motor_Step_Angle = 2
Motor_Clock_Frequency = 2666.66 Hz if Motor_Step_Angle = 4

Motor_Clock_Frequency = 85333.33 Hz if Motor_Step_Angle = 128

See also the means of *Working_Setting_Extended.Clockout_Init* object bit.

If *Drive_Working_Settings_Extended.Clockout_Init* bit is set(1), each time a new motor movement is invoked, the clockout generator is reset so the first clockout pulse will be out after Clockout_Prescaler Motor clocks.
If *Drive_Working_Settings_Extended.Clockout_Init* bit is set(0), each time a new motor movement is invoked, the clockout generator continue from the last Motor Position.

➔ The clock is generated on outputs B0_OUT0 and B0_OUT1 in quadrature mode while the motor is moving. The *Clockout_Prescaler object defines clockout resolution (incs/rev).*

➔ The Incremental encoder digital input signals (Enc#0) are replicated on B0_OUT0 and B0_OUT1 digital outputs. This function is not available on all Titanio drives. Refer to specific hardware manual of the used drive.

➔ The B0_OUT1 digital output is ON when motor is moving and OFF when motor is stopped.

Note:

The Clcokout feature is available only for Drives with firmware V02R27 or superior.

9.4 Motor Stall detection

This feature allows to recognize the Motor stall without using an external sensor (for example incremental or absolute encoder).

Related Objects :

- *Motor_Stall_Filter_Time*
- *Motor_Stall_Actual_Err_Angle*
- *Motor_Stall_Max_Err_Angle*
- *Drive_Working_Settings_Extended*
- *Feedback_Status*
- *Drive_Register_Extended*
- *Error_Register*

To enable 'Motor Stall' feature set bit13 of *Drive_Working_Settings_Extended* object.
If activation has been successful then bit6 of *Drive_Working_Settings_Extended* object will be set to 1.

The 'Feedback' feature (bit4 of *Drive_Working_Settings* object) and the 'Motor Stall detection' feature (bit13 of *Drive_Working_Settings_Extended* object) cannot be both active at the same time. Try to enable 'Feedback' feature with 'Motor Stall detection' feature already enabled or vice versa will issue an 'Feedback_Error' alarm (bit5 of *Error_Register* object and bit14 of *Feedback_Status* object are set to 1).

The motor Stall detection consists in comparing the theoretical angle and estimated angle of motor rotor position.

If | *Motor_Stall_Actual_Err_Angle* | > *Motor_Stall_Max_Err_Angle* then 'Motor Stall detected' !!

When the motor stall is detected :

- the drive will stop the motor immediately
- will open the fault digital output
- *Min_Current* object value is applied to the motor
- bit5 (*Feedback_Error*) of *Error_Register* object is set to 1
- bit10 (*Motor_Stall_detected*) of *Feedback_Status* object is set to 1

No motor movement is executed until the master will set the *Master_Alarm_Reset* bit of *Master_Register* object.

Notes:

- The 'Motor Stall detection' feature is available only with firmware version V02r74 or superior.
- In the 'Motor Feedback Tuning' window a dedicated trace shows 'Motor_Stall_Actual_Err_Angle' object value.

9.5 Brake Control

By mean of *Brake_Control_Settings* object is possible to enable/disable (bit0) the Automatic Brake Control, define the digital output used for the Brake (bit4÷bit7 and bit3) and the details of the Brake control handling (bit1 and bit2).

The following descriptions are related to Automatic Brake Control enabled (bit0 = 1 of *Brake_Control_Settings* object).

When the drive is in Emergency condition then the Brake is close (active) and the settings of the bit1 and bit2 of *Brake_Control_Settings* object are not considered. When an Emergency occurs the Brake is immediately closed (actived) and the motor current is switched off.

When the drive is not in the emergency situation the state close or open of the Brake depends from settings of the bit1 and bit2 of *Brake_Control_Settings* object :

bit2	bit1	bit0	Normal condition (no emergency active)
0	0	1	The Brake is open (released).
0	1	1	The procedure described below can be used when is required an automatic handling of the closing and opening of the Brake during the stop and movement of the motor. The Brake is closed (actived) when the motor is at standstill and open (released) when motor is running. The following automatic sequence is performed from the drive when the Brake is closed : - after the motor stop the time defined by *2C01.0H* is allowed to elapse. - the Brake is closed (actived). - the time defined by *2C02.0H* is allowed to elapse. - the motor current is switched off. The following automatic sequence is performed from the drive when the Brake is open : - the motor current is switched on. - the time defined by *2C03.0H* is allowed to elapse. - the Brake is open (released). - the time defined by *2C04.0H* is allowed to elapse. - the drive can perform motor movements. Note : This type of Automatic Brake Handling can be used when Electric Gear feature is disabled. With Electric Gear enabled is suggested to use the Brake handling as defined by settings of the bit2 = 1 and bit1 = 0 because the slave drive does not know when it will receive the first step to move from a Master drive and therefore could apply the automatic sequence to open the brake (described above) simultaneously with the reception of the subsequent movement steps, risking a block of the motor movement.
1	X	1	The state of Brake can be changed by *B0_Digital_Outputs* object (if bit3 = 0 of *Brake_Control_Settings* object) or *B1_Digital_Outputs* object (if bit3 = 1 of *Brake_Control_Settings* object). This mode allows a manual handling of the sequence concerning open/close of the Brake (*2C01.0H*, *2C02.0H*, *2C03.0H*, *2C04.0H* objects are not considered in this case). The sequence can be handled from an external Master (by mean of fieldbus) or inside EEplc user program and have to consider the timing of the Brake and switch off and switch on of the motor current. To switch off the motor current : - for *Open Loop Modality* the objects *Min_Current* , *Max_Current* , *Boost_Current* must be set to 0 value. - for *Closed Loop Modality* the objects *Feedback_Boost_Current* must be set to 0 value. To switch on the motor current restore the values previously used for the objects *Min_Current* , *Max_Current* , *Boost_Current* , *Feedback_Boost_Current* before they were forced to 0 by mean of switch off motor current.

The Brake Control is available with firmware V03r20 or superior.

Related objects :

Object name	Note
Brake_Control_Settings	
Brake_Control_Time1_Close_Brake	
Brake_Control_Time2_Close_Brake	
Brake_Control_Time1_Open_Brake	
Brake_Control_Time2_Open_Brake	
B0_Digital_Outputs	
B1_Digital_Outputs	

9.6 Feedback Sensor Calibration mode

By mean of this mode is possible the calibration of the Feedback Sensor used for *Closed Loop Modality*.
The Feedback Sensor Calibration mode is available only with firmware version V03r18 or superior.

The type of Sensor to be calibrated is defined with the *Feedback_Settings* object (bit8÷bit11).

At the end of successful calibration, the calibrated values are stored in NVRAM.
Every time that *Closed Loop Modality* will be actived, the calibrated values will be used and no additional calibration will be needed .

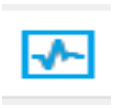
During calibration procedure the motor shaft must be load-free and free to turn in any direction.

The calibration procedure is actived by the objects *Direct_Command_Parameter_1*, *Direct_Command_Parameter_2*, *Direct_Command_CMD* .

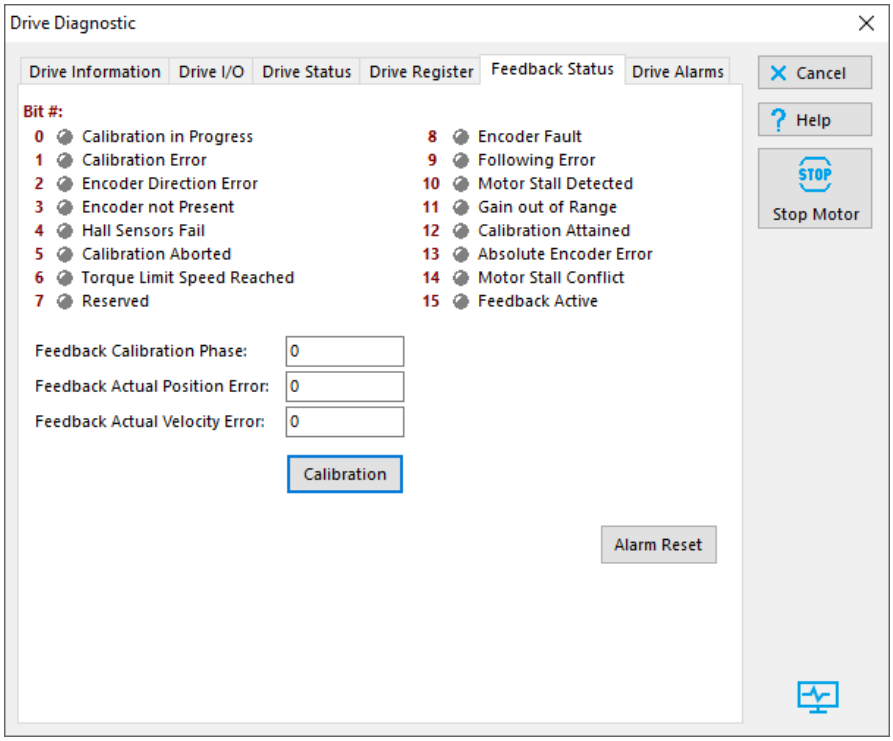
The ‘e3PLC STUDIO’ Tool can be also used for Feedback Sensor Calibration :

- go in the ONLINE condition by clicking on ‘Go Online’ button

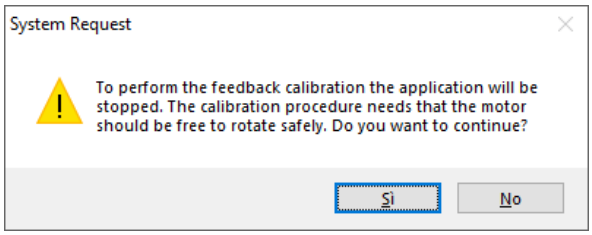
- Open the ‘Drive Diagnostic Window’ clicking on ‘Drive Diagnostic’ button



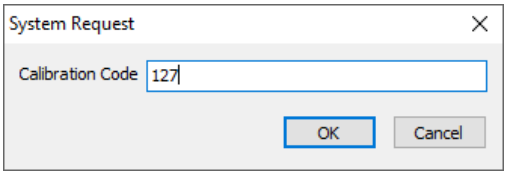
- In the ‘Feedback Status’ sub-window click on ‘Calibration’ button



confirm to continue in the calibration procedure



and then insert the value 127 to active the procedure



Before starting the calibration procedure must be defined the values of parameters concerning the type of sensor that will be calibrated .

9.6.1 Multi-Turn Absolute Encoder BiSS

Before starting the calibration procedure the following parameters must be defined :

- *Feedback_Calibration_Speed*
- *Feedback_Calibration_Current*
- *BiSS_Encoder_Config*
- *Nominal_Current*
- *Feedback_Source_PPR*
- *Motor_Pole_Pairs*
- *Motor_Step_Angle*
- *Motor_Resolution*
- *Drive_Working_Settings_Extended*
- *Motor_R* and *Motor_L* (*if bit9=0 of Drive_Working_Settings_Extended* object)

The type of Feedback Sensor has to be defined with value 6 in the (bit8÷bit11) of the *Feedback_Settings* object.
The bit12 of *Feedback_Settings* object must be set (1) to active the Full Feedback Calibration procedure.
The *Feedback_Source_PPR* object must be set to value 65536.

The calibration procedure is carried out on with the following sequence :

- *Direct_Command_Parameter_1* = **127**
- *Direct_Command_Parameter_2* = **0**
- *Direct_Command_CMD* = **1**
- *Direct_Command_Parameter_2* = **1**
- *Direct_Command_CMD* = **1**

The values of the objects *Direct_Command_Parameter_1* and *Direct_Command_Parameter_2* must be configured before to write the *Direct_Command_CMD* object.

During the procedure the bit0 (Calibration in progress) of the *Feedback_Status object* is set (1). At the end of the procedure, if successfully procedure, the bit12 (Calibration attained) and bit15 (Feedback active) of the *Feedback_Status* object are set (1). If an error occurred the bit1 (Calibration error) of the *Feedback_Status* object is set(1) and the procedure is not considered successful.

The procedure can be interrupted with the following sequence :

- *Direct_Command_Parameter_1* = **127**
- *Direct_Command_Parameter_2* = **1**
- *Direct_Command_CMD* = **1**
- *Direct_Command_Parameter_2* = **0**
- *Direct_Command_CMD* = **1**

The values of the objects *Direct_Command_Parameter_1* and *Direct_Command_Parameter_2* must be configured before to write the *Direct_Command_CMD* object.

The calibrated values that have been calibrated and stored in NVRAM can be used afterwards only if the bit14 of the *Feedback_Settings* is set (1). Otherwise if bit14 = 0 then the Sensor Calibration procedure is performed every time when the *Closed Loop Modality* is actived.

Related objects :

Object name	Note
Feedback_Settings	
Feedback_Status	
Feedback_Calibration_Current	
Feedback_Calibration_Speed	
Feedback_Source_PPR	
BiSS_Encoder_Config	
Nominal_Current	
Motor_Pole_Pairs	
Motor_Step_Angle	
Motor_Resolution	
Drive_Working_Settings_Extended	
Motor_R	
Motor_L	

9.6.2 Single-Turn Magnetic Encoder

Before starting the calibration procedure the following parameters must be defined :

- *Feedback_Calibration_Speed*
- *Feedback_Calibration_Current*
- *Nominal_Current*
- *Feedback_Source_PPR*
- *Motor_Pole_Pairs*
- *Motor_Step_Angle*
- *Motor_Resolution*
- *Drive_Working_Settings_Extended*
- *Motor_R* and *Motor_L* *(if bit9=0 of* *Drive_Working_Settings_Extended* object)

The type of Feedback Sensor has to be defined with value 8 in the (bit8÷bit11) of the *Feedback_Settings* object.
The bit12 of *Feedback_Settings* object must be set (1) to active the Full Feedback Calibration procedure.

The calibration procedure is carried out on with the following sequence :

- *Direct_Command_Parameter_1* = **127**
- *Direct_Command_Parameter_2* = **0**
- *Direct_Command_CMD* = **1**
- *Direct_Command_Parameter_2* = **1**
- *Direct_Command_CMD* = **1**

The values of the objects *Direct_Command_Parameter_1* and *Direct_Command_Parameter_2* must be configured before to write the *Direct_Command_CMD* object.

During the procedure the bit0 (Calibration in progress) of the *Feedback_Status* *object* is set (1). At the end of the procedure, if successfully procedure, the bit12 (Calibration attained) and bit15 (Feedback active) of the *Feedback_Status* object are set (1). If an error occurred the bit1 (Calibration error) of the *Feedback_Status* object is set (1) and the procedure is not considered successful.

The procedure can be interrupted with the following sequence :

- *Direct_Command_Parameter_1* = **127**
- *Direct_Command_Parameter_2* = **1**
- *Direct_Command_CMD* = **1**
- *Direct_Command_Parameter_2* = **0**
- *Direct_Command_CMD* = **1**

The values of the objects *Direct_Command_Parameter_1* and *Direct_Command_Parameter_2* must be configured before to write the *Direct_Command_CMD* object.

Related objects :

Object name	Note
Feedback_Settings	
Feedback_Status	
Feedback_Calibration_Current	
Feedback_Calibration_Speed	
Feedback_Source_PPR	
Nominal_Current	
Motor_Pole_Pairs	
Motor_Step_Angle	
Motor_Resolution	
Drive_Working_Settings_Extended	
Motor_R	
Motor_L	

9.6.3 Hall Sensors

Before starting the calibration procedure the following parameters must be defined :

- *Feedback_Calibration_Speed*
- *Motor_Pole_Pairs*
- *Motor_Step_Angle*
- *Motor_Resolution*
- *Nominal_Current*
- *Min_Current*
- *Max_Current*
- *Boost_Current*
- *Drive_Working_Settings_Extended*
- *Motor_R* and *Motor_L* (*if bit9=0 of Drive_Working_Settings_Extended* object)

The type of Feedback Sensor has to be defined with value 1 (Hall Sensors) or 2 (Hall Sensors+Incremental Encoder) in the (bit8÷bit11) of the *Feedback_Settings* object.

The calibration procedure is carried out on with the following sequence :

- *Direct_Command_Parameter_1* = **127**
- *Direct_Command_Parameter_2* = **0**
- *Direct_Command_CMD* = **1**
- *Direct_Command_Parameter_2* = **1**
- *Direct_Command_CMD* = **1**

The values of the objects *Direct_Command_Parameter_1* and *Direct_Command_Parameter_2* must be configured before to write the *Direct_Command_CMD* object.

During the procedure the bit0 (Calibration in progress) of the *Feedback_Status* *object* is set (1). At the end of the procedure, if successfully procedure, the bit12 (Calibration attained) and bit15 (Feedback active) of the *Feedback_Status* object are set (1). If an error occurred the bit1 (Calibration error) of the *Feedback_Status* object is set (1) and the procedure is not considered successful.

The procedure can be interrupted with the following sequence :

- *Direct_Command_Parameter_1* = **127**
- *Direct_Command_Parameter_2* = **1**
- *Direct_Command_CMD* = **1**
- *Direct_Command_Parameter_2* = **0**
- *Direct_Command_CMD* = **1**

The values of the objects *Direct_Command_Parameter_1* and *Direct_Command_Parameter_2* must be configured before to write the *Direct_Command_CMD* object.

At the end of the succesfully procedure the Hall Sensors sequence detected (*290A.0H*) is stored in the *Hall_Sensors_Sequence_Settings* object.

Related objects :

Object name	Note
Feedback_Settings	
Feedback_Status	
Feedback_Calibration_Current	
Feedback_Calibration_Speed	
Nominal_Current	
Hall_Sensors_Status	
Hall_Sensors_Position	
Hall_Sensors_Sequence_Settings	
Hall_Sensors_Sequence_Detected	
Motor_Pole_Pairs	
Motor_Step_Angle	
Motor_Resolution	
Drive_Working_Settings_Extended	
Motor_R	
Motor_L	
Min_Current	
Max_Current	
Boost_Current	

9.7 Braking Resistor Function

When load is accelerated electrical energy is converted into mechanical energy. During deceleration the conversion is reversed. If during the braking too much energy is generated an external resistor (or internal if the driver is provided with), known as braking resistor, is used to dissipate the excess energy preventing that the driver's overvoltage protection wil shut down the driver.

To drive the external resistor the driver must be equipped with a dedicated internal circuit, refer to the hardware manual of a specific drive to check if an external braking resitor can be used.

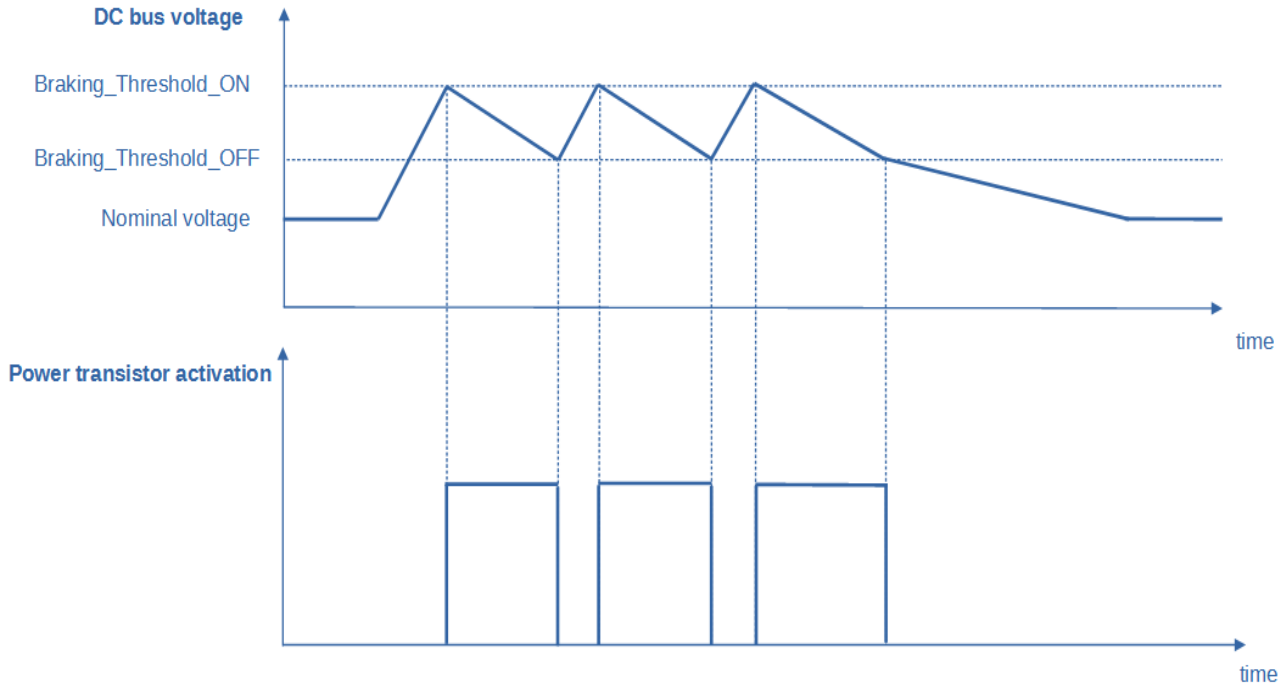
Operation of braking resitor function is quite simple: if during a braking the internal voltage goes beyond the "*Braking_Threshold_ON*", a power transistor is turned on and the external braking resistor starts to shunt energy until the internal voltage returns below the "*Braking_Threshold_OFF*".

The driver protects the braking circuit against short circuit, and uses I2T peak current/time algorithms to protect both the braking resistor and internal transistor.

To make the protection work properly one must set the following parameters:

- *Braking_Resistor_Value* : ohmic value of the resistor expressed in ohms.
- *Braking_Resistor_Power* : rated power of the resistor expressed in watts
- *Braking_Resistor_Overload_Time* : maximum time the braking resistor can withstand at the power peak expressed in tenths of milliseconds

The peak power can be estimated as (Braking_Threshold_ON^2) / Braking_Resistor_Value.



10.0 MODBUS Protocol

MODBUS RTU (Remote Terminal Unit) Protocol is a messaging structure used to establish master–slave communications between intelligent devices. When a MODBUS master sends a message to a MODBUS slave, the message contains the address of the slave, the function, the data and a checksum. The slave's response message contains fields confirming the master's request, any data requested and an error–checking field.

A typical MODBUS RTU frame consists of the following fields:

ADDRESS	FUNCTION	DATA	CHECKSUM

The **address field** of a message contains 8 bits. Valid slave addresses are in the range of 0– 247 decimal. The individual slave devices are set in the range of 1 – 247 decimal (address 0 is the broadcast to all slaves address). The master specifies a slave by placing the slave address in the address field of the message. When the slave responds, it places its own address in the address field to identify to the master which slave is responding.
The **function code field** of a message contains 8 bits. Valid function codes are in the range of 1 – 255 decimal. The function code instructs the slave what kind of action to take.
The **data field** contains additional information that the slave uses to execute the action defined by the function code. This can include internal addresses, quantity of items to be handled, etc.
The data field of a response from a slave to a master contains the data requested if no error occurs.
The **checksum field** is used for error checking.
The checksum field consists of two bytes, creating a 16 bit binary value. The CRC is calculated in the transmitting device and is recalculated and compared by the receiving device.
The slave device checks the entire message frame during receipt.
For details on how the checksum is computed refers to chapter **'CRC Generation'** on *PI-MBUS-300 Modbus Reference Guide*.
Note: Even though the Modbus protocol allow frames up to 255 bytes long, the Titanio drives can receive Modbus frame no longer than 128 bytes.

10.1 MODBUS Protocol Parameters

MODBUS RTU Specifications	
Connection Port Type	RS485
Protocol	MODBUS RTU / TCP
Device Id Number	1 to 127 (1 on Modbus TCP)
Baud Rate Supported	1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
Modbus TCP Ports	502, 64738, 64739
Start Bits	1
Data Bits	8
Stop Bits	1
Parity	None

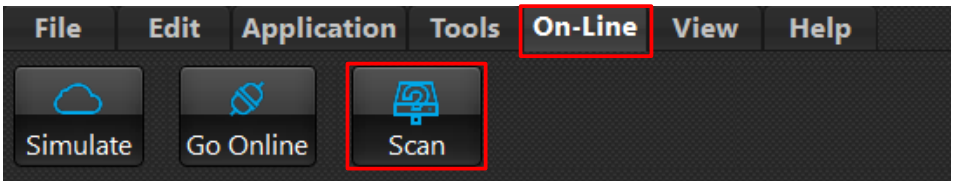
10.1.1 Baud Rate & Node Id Selection on Drives with dips-switches and rotoswitches

For drives fitted with dips-switches and rotoswitches, look at hardware short manual for details on settings.

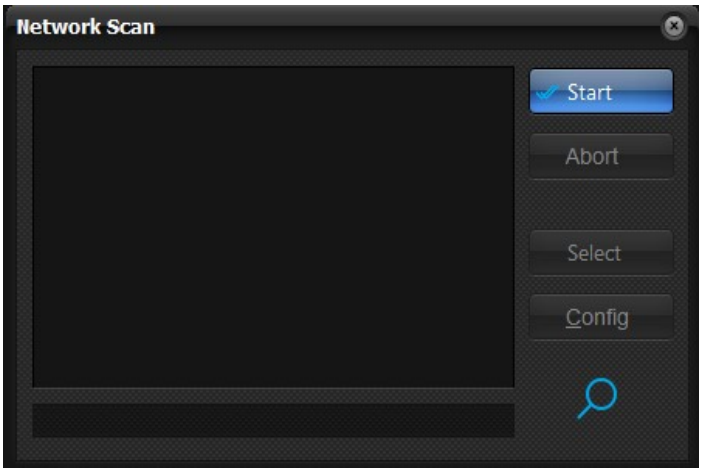
10.1.2 Baud Rate & Node Id Selection on Drives without dip-switches and rotoswitches

The drives without dips-switches and rotoswitches must be configured by means of e3PLC Studio Software. The drives factory settings is baud rate = 57600 and NodeId = 1. Follows the steps to perform to change the default settings:

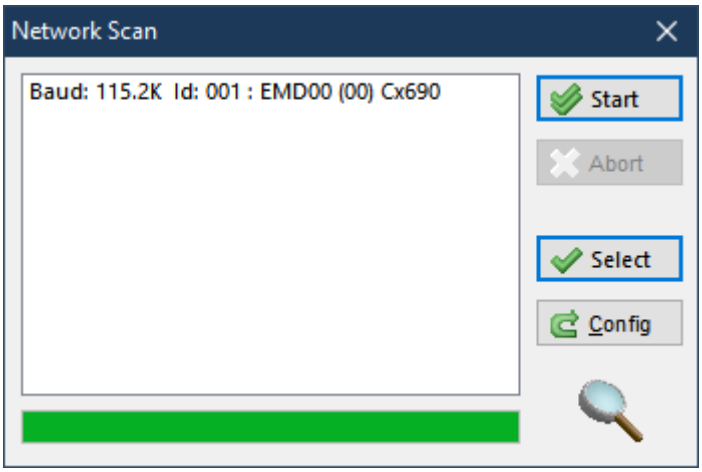
1. Open e3PLC Studio Software.
2. From the main window select the *On-line → Scan.*



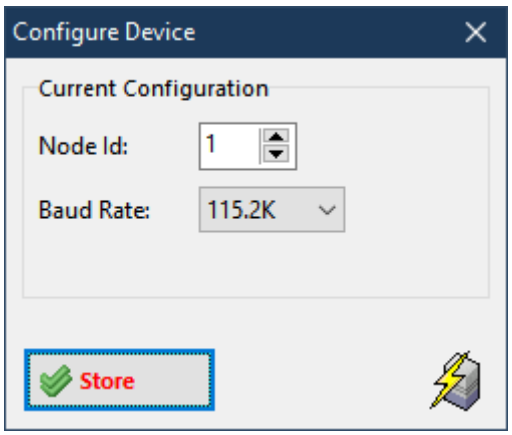
The following window will appear:



3. Press on ***"Start"*** button.
4. If the drive has been detected its system code will be added to the list box.



5. Select the device and press on ***"Config"*** button. The following window will appear:



6. Change the Node Id and Baud Rate settings as desired and press on ***"Store"*** button.
7. The new settings will be used at the next drive's switch on.

Instead of using the e3PLC Studio Software, the Baud Rate & Node Id can be changed directly writing on the registers (40B8 & 40B9) according to the following instructions:

1. Write the register 40B8 (Node Id) keeping the high byte equal to 0xAA and setting in the low byte the new Id. (example to set the Node Id = 5 write 0xAA05)
2. Write the register 40B9 (Baud Rate) keeping the high byte equal to 0x55 and setting in the low byte the new baud rate according to the following table (example to set the baud rate = 57600 write 0x5501):

Value	Baud Rate
0	115200
1	57600
2	38400
3	19200

3. After having written the new value it is necessary to wait about 1 second to permit to the system to store the new values in NVRAM.
4. The new BaudRate & NodeId will be effective at the next drive switch on.

10.2 MODBUS RTU Function Codes

The following MODBUS RTU functions are supported by Titanio family drives:

Function Code	Description
03	Read Holding Registers (Read n words)
06	Preset Single Register (Write 1 word)
16	Preset Multiple Registers (Write n words)

10.2.1 MODBUS RTU Function Code : 03

The function Code **03** (Read n words) is the primary command to acquire drive's parameters data. This function implemented on Titanio family drives supports reading of more than one drive's parameter at a time regardless of its length (one or more data words). The broadcast is not supported.

Function Code **03** format :

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Slave Address	**Function**	**Register Address HI**	**Register Address LO**	**# of Words HI**	**# of Words LO**	**CRC LO**	**CRC HI**
xx	03	xx	xx	00	xx	xx	xx

10.2.2 MODBUS RTU Function Code : 06

The function Code **06** (write 1 word) can be used to set drive's one word long parameters. This function implemented on Titanio family drives supports writing of one drive's parameter at a time. The broadcast is supported.

Function Code **06** format :

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Slave Address	**Function**	**Register Address HI**	**Register Address LO**	**Data HI**	**Data LO**	**CRC LO**	**CRC HI**
xx	06	xx	xx	xx	xx	xx	xx

10.2.3 MODBUS RTU Function Code : 16

The function Code **16** (write n words) can be used to set all drive's parameters. This function implemented on Titanio family drives supports writing of more than one drive's parameter at a time regardless of its length (one or more data words). The broadcast is supported.

Function Code **16** format :

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Slave Address	**Function**	**Starting Address HI**	**Starting Address LO**	**# of Words HI**	**# of Words LO**	**Byte Count**	**Data 0 HI**
xx	10	xx	xx	00	xx	xx	xx
Byte 8	**Byte 9**	**Byte 10**	**….**	**Byte n**	**Byte n+1**	**Byte n+2**	**Byte n+3**
Data 0 LO	**Data 1 HI**	**Data 1 LO**	**….**	**Data n HI**	**Data n LO**	**CRC LO**	**CRC HI**
xx	xx	xx	…..	xx	xx	xx	xx

10.2.4 MODBUS RTU Function Code : 23

The function Code **23** (read/write n words) can be used to set and get all drive's parameters. This function implemented on Titanio family drives supports writing and reading of more than one drive's parameter at a time regardless of its length (one or more data words). The broadcast is not supported.

Function Code **23** format :

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Slave Address	Function	Starting Read Address HI	Starting Read Address LO	# of Words Read HI	# of Words Read LO	Starting Write Address HI	Starting Write Address LO
xx	17	xx	xx	00	xx	xx	xx
Byte 8	**Byte 9**	**Byte 10**	**Byte 11**	**Byte 12**	**Byte n**	**Byte n+1**	**Byte n+2**
# of Words Write HI	# of Words Write LO	# of Write Data Bytes	Write Data 0 HI	Write Data 0 LO	Write Data n HI	Write Data n LO	CRC LO
00	xx	xx	xx	xx	xx	xx	xx
Byte n+3							
CRC HI							
xx							

10.3 MODBUS Error Codes

The Titanio family drives supports the following Exception Codes described on chapter **'Exception Responses'** on *PI-MBUS-300 Modbus Reference Guide*:

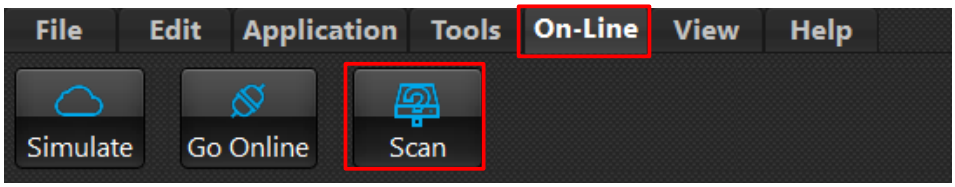
Code	Name	Meaning
01	**ILLEGAL FUNCTION**	The function code received is not supported.
02	**ILLEGAL DATA ADDRESS**	The data address received does not exist
03	**ILLEGAL DATA VALUE**	The data value received is not allowable for the data address specified

10.4 MODBUS TCP

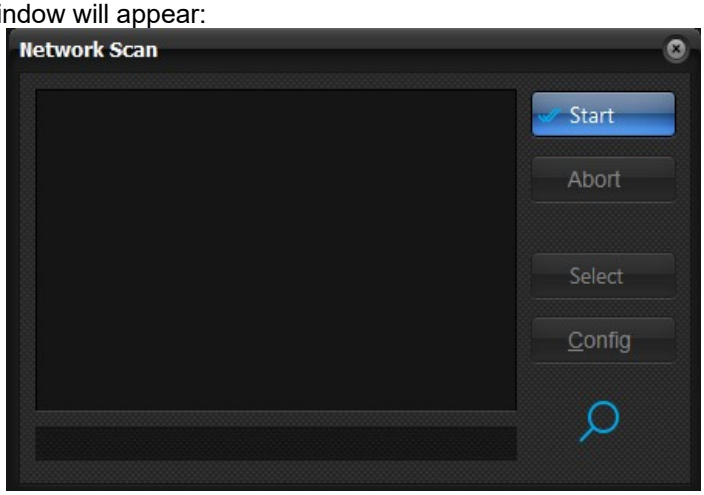
The Titanio drives implement the Modbus TCP protocol for the Ethernet interface. This protocol was implemented to allow communications with HMI or PLC devices which manage this communications protocol. The TCP ports available are: 502 (Standard Modbus TCP Port), 64738 and 64739. The same functions as Modbus RTU are available. The registers available are the same ones used for Modbus RTU.

The drive's factory default IP Settings is 90.0.0.254. Follows the steps to perform to change the default settings:

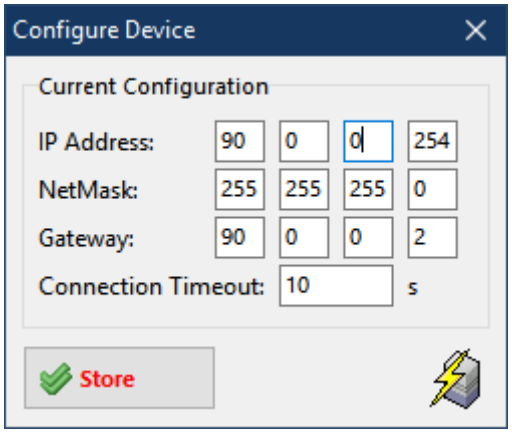
1. Open e3PLC Studio Software.
2. From the main window select the *On-line → Scan.*



The following window will appear:



3. Press on "Start" button.
4. If the drive has been detected its system code will be added to the list box.
5. Select the device and press on "Config" button. The following window will appear:



Change the IP Settings as desired and press on "Store" button.

Set Connection Timeout to a value different than 0 to enable the check of dead socket connections (if no read or write operations are detected within the specified timeout value, the socket will be automatically closed).

6. The new settings will be effective at the next drive switch on.

11.0 CANopen Protocol

The CANopen protocol is one of the most common CAN protocols. Since 1995 the CANopen specification is handed over to CAN in Automation (CiA) international users and manufacturers group. The CANopen Device Specification version 4.01 has been accepted by the European standardization authorities as EN 50325-4.
The main concept of CANopen is based on use of an object dictionary (basically device's variables, parameters, etc.). This dictionary gathers data related to the communication and the application. To access to these objects two methods are used: SDO & PDO that are explained further in this manual.

11.1 CANopen Protocol Parameters

CANopen Specifications	
CANopen Functionality	Slave
Device Id Number	1 to 127
Baud Rate Supported (Kbits)	1000,500,250,125
NMT	Slave
Server SDOs	1 (Standard)
Client SDOs	No
Receive PDOs	2
Transmit PDOs	2
PDO Mapping	Static
Emergency Telegram	Yes
Nodeguarding	No
Heartbeat	Yes

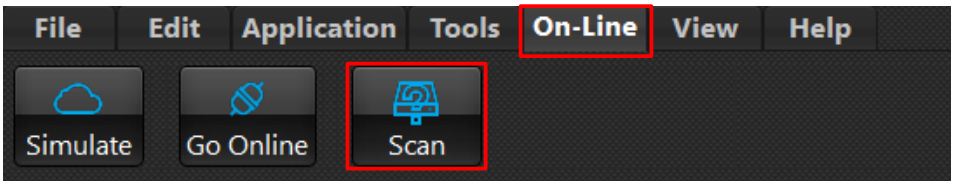
11.1.1 Baud Rate & Node Id Selection on drives with dip-switches and rotoswitches

For drives fitted with dips-switches and rotoswitches, look at hardware short manual for details on settings.

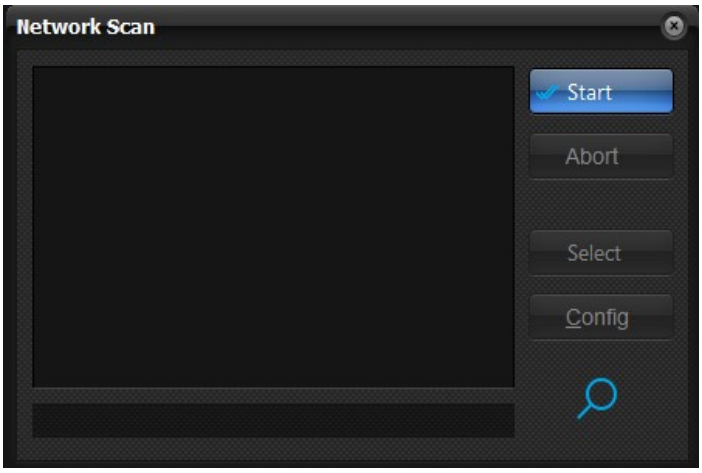
11.1.2 Baud Rate & Node Id Selection on drives without dips-switches and rotoswitches

The Titanio drives without dip-switches and rotoswitches can be configured by means of Titanio eePLC Studio Software Tool. The drives factory settings is baud rate = 500 Kbit and NodeId = 1. Follows the steps to perform to change the default settings:

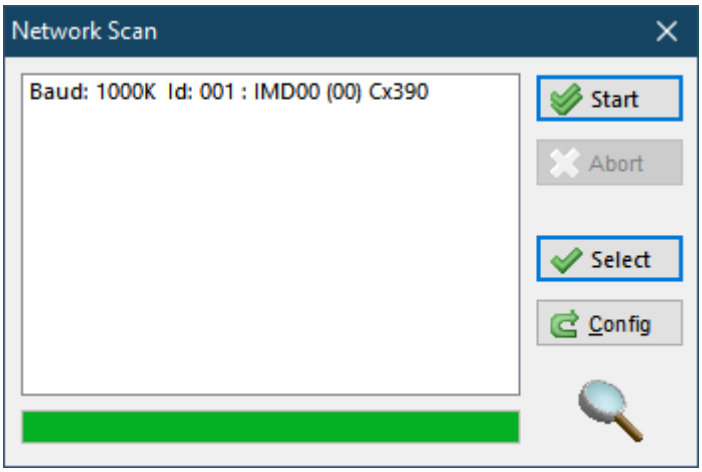
1. Open e3PLC Studio Software Tool.
2. From the main window select the *On-line → Scan.*



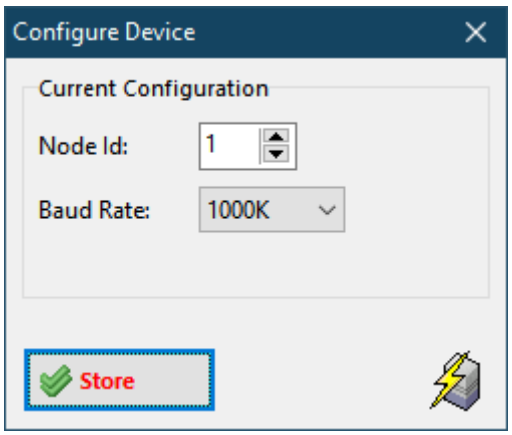
The following window will appear:



3. Press on ***"Start"*** button.
4. If the drive has been detected its system code will be added to the list box.



5. Select the device and press on ***"Config"*** button. The following window will appear:



6. Change the Node Id and Baud Rate settings as desired and press on ***"Store"*** button.
7. The new settings will be used at the next drive's switch on.

Instead of using the Titanio eePLC Studio Software Tool, the Baud Rate & Node Id can be changed directly writing on the objects (4000.7 & 4000.8) according to the following instructions:

1. Write using SDO service 2 bytes in the object 4000.7 (Node Id) keeping the high byte equal to 0xAA and setting in the low byte the new Id. (example to set the Node Id = 5 write 0xAA05)
2. Write using SDO service 2 bytes in the object 4000.8 (Baud Rate) keeping the high byte equal to 0x55 and setting in the low byte the new baud rate according to the following table (example to set the baud rate = 500K write 0x5501):

Value	Baud Rate
0	1M
1	500K
2	250K
3	125K

3. After having written the new value it is necessary to wait about 1 second to permit to the system to store the new values in NVRAM.
4. The new BaudRate & NodeId will be effective at the next drive switch on.

11.2 CANopen SDO (Service Data Object)

Service Data Objects are used to establish a peer to peer connection between two CANopen devices. This kind of connection is based on a Client/Server based mechanism.
The SDO server is the device that is serving the object dictionary to which the access is required.
The SDO client is the device that wants to access the object dictionary of a specific device.
The SDO service is based on two CAN messages with different identifiers. One message is used by the SDO client and the second message is used by the SDO server.
There are two different methods for SDO download/upload:

➔ **Expedited SDO transfer:**
- For objects no longer than 4 bytes.

➔ **Segmented SDO transfer:**
- For objects longer than 4 bytes.

Request (Client → Server)

Cob_ID	B0	B1	B2	B3	B4	B5	B6	B7
600H+NodeId	ccs/cntrl	Object Index		SubIndex	Data (Optional)			

Response (Client ← Server)

Cob_ID	B0	B1	B2	B3	B4	B5	B6	B7
580H+NodeId	scs/cntrl	Object Index		SubIndex	Data (Optional)			

Examples:

SDO - Expedited protocol download (write an object 4 bytes long):

Request (Client → Server)

Cob_ID	B0	B1	B2	B3	B4	B5	B6	B7
600H+NodeId	22H	Index		SubIndex	Value			

Response (Client ← Server)

Cob_ID	B0	B1	B2	B3	B4	B5	B6	B7
580H+NodeId	60H	Reserved						

SDO - Expedited protocol upload (read an object 4 bytes long):

Request (Client → Server)

Cob_ID	B0	B1	B2	B3	B4	B5	B6	B7
600H+NodeId	40H	Index		SubIndex	Reserved			

Response (Client ← Server)

Cob_ID	B0	B1	B2	B3	B4	B5	B6	B7
580H+NodeId	43H	Index		SubIndex	Value			

11.3 CANopen PDO (Process Data Object)

Process Data Objects are used to transmit any process data for the process control.
The PDOs are transmitted in broadcast and without any confirmation back to the transmitting device.
In the standard version of Titanio drive's firmware 2 Transmit PDOs and 3 Receive PDOs are available.

PDO Mapping:

Receive PDO #1

Cob_Id	B0	B1	B2	B3	B4	B5	B6	B7
200H + NodeId	2300.2H CANopen_RX_PDO1_Data[0] DWORD				2300.3H CANopen_RX_PDO1_Data[1] DWORD			

Receive PDO #2

Cob_Id	B0	B1	B2	B3	B4	B5	B6	B7
300H + NodeId	2300.4H CANopen_RX_PDO2_Data[0] DWORD				2300.5H CANopen_RX_PDO2_Data[1] DWORD			

Receive PDO #3

Cob_Id	B0	B1	B2	B3	B4	B5	B6	B7
300H	2300.6H CANopen_RX_PDO3_Data[0] DWORD				2300.7H CANopen_RX_PDO3_Data[1] DWORD			

Transmit PDO #1

Cob_Id	B0	B1	B2	B3	B4	B5	B6	B7
180H + NodeId	4000.3H		4000.2H		6063.0H			

Transmit PDO #2

Cob_Id	B0	B1	B2	B3	B4	B5	B6	B7
280H + NodeId	2301.3H CANopen_TX_PDO2_Data[0] DWORD				2301.4H CANopen_TX_PDO2_Data[1] DWORD			

All PDOs can be asynchronous or synchronous (See Transmission Types objects).
The receive PDOs are handled as soon as possible after their receipt if set as asynchronous.
For Transmit PDOs has been implemented subindex 5 (event timer) of 18xxH objects as described in the standard CiA DS 301 V4.01 that permits to specify also a transmission frequency for asynchronous PDOs.

11.4 CANopen SYNC (Synchronization Message)

The SYNC message has been implemented in drive's firmware. Anyway the Cob-Id is fixed to 80h (CiA DS301 default value) and cannot be change by means of object 1005.0H (Cob-Id SYNC). The SYNC message is useful when it is necessary to retrieve PDOs from the drive only when requested by the master (SYNC producer) or to makes PDOs sent by the master processed at the same time by the drives.

Cob_ID
80H

11.5 CANopen Heartbeat

The Titanio family drives implement the heartbeat protocol as defined in CiA DS 301 V4.01. This permits to the Master to check the drive working condition. It is possible to change the frequency of heartbeat transmission with the object 1017.0H (Producer Heartbeat Time). At switch-on the drive send the Boot-up message that is an heartbeat message with Status = 0;

Cob_ID	B0
700H+NodeId	Status

The Titanio family drives support only Pre-Operational (127) and Operational (5) states.

11.6 CANopen Emergency Telegram

The Titanio drives send an Emergency Telegram every time a fault (software or hardware) is detected.
The Titanio drives send also an Emergency Telegram at switch on without any data bytes (only Cob_Id).

Cob_ID	B0	B1	B2	B3	B4	B5	B6	B7
80H+NodeId	Error Code		Error Register	Manufacturer Specific (4000.1H)				

The field ‘Manufacturer Specific’ (Error_Code object) can be one of the following:

Value	Description
15H	Thermal Protection
16H	Voltage Protection
17H	Current Protection
1AH	Watchdog Occurred
20H	Missing Setup
28H	Missing Calibration
33H	Open Phase
4AH	Software Trap
59H	E^2PROM Failure
5AH	Motor Thermal Protection
5BH	Current Regulation Out Of Range
5CH	ADC Offset Out Of Range
5DH	eePLC Program
5EH	Expired eePLC Software Trial
60H	eePLC Software Protection
61H	Unavailable Feature
62H	EEprom Write Overrun

11.7 CANopen Boot Up / NMT Protocols

At switch-on the Titanio drives can be in Pre-Operational or Operational state (see *Drive_Working_Settings_Extended* object), this means that PDOs are disabled. At switch-on the drive send the Boot-up message that is an heartbeat message with Status = 0;

Cob_ID	B0
700H+NodeId	0

The Master have to send the NMT frame with command Start Node.

Start Node Command

Cob_ID	B0	B1
00H	1	NodeId

Enter Pre-Operational State Command

Cob_ID	B0	B1
00H	128	NodeId

If NodeId = 0 all devices connected to CAN network will execute the command.

It is supported also the NMT - Reset Node Protocol to reset the drive:

Reset Node Command

Cob_ID	B0	B1
00H	129	NodeId

11.8 EVER Motor SYNC Message

The EVER Titanio family drives implement also a custom SYNC message to synchronize only motor start/stop on multiaxes systems. See §*8.3.4* for more details on movement with SYNC. When the Titanio drive receives the EVER Motor_SYNC message it sets the *Motor_SYNC* object (2280.0H) object to 1.

EVER Motor_SYNC

Cob_ID
180H

11.9 CANopen Objects Dictionary

The following tables show the CANopen objects that are not accessible as eePLC Objects but only through the CAN interface.

CiA Draft Standard 301 (V4.01):

Index (hex)	SubIndex (hex)	Name	Type	Attr.	Description	Default Value
1000	0	Device Type	Unsigned32	ro	Device Type	00000000H
1001	0	Error Register	Unsigned8	ro	Error Register (only Bit#0 used)	00H
1005	0	COB-ID SYNC	Unsigned32	ro	Cob-Id SYNC	80H
1010	1	Save Parameters	Unsigned32	rw	Save all Parameters in nv memory	0
1011	1	Restore all Parameters	Unsigned32	rw	Restore all default Parameters	0
1014	0	COB_ID Emergency	Unsigned32	ro	Cob-Id EMCY	80H + NodeId
1017	0	Producer_Heartbeat_Time	Unsigned16	rw	Producer Heartbeat Time	500
1018	1	Vendor ID	Unsigned32	ro	Vendor - ID	4BH
1018	2	Product Code	Unsigned32	ro	Drive Hardware Code	NA
1018	3	Revision Number	Unsigned32	ro	Drive Hardware Revision Number	NA
1018	4	Serial_Number	Unsigned32	ro	Drive Serial Number	NA
1200	0 ÷ 2	1st Server SDO Parameters	SDO Par.	ro	Server SDO #1 Parameters	----
1400	1	RX_PDO1_Cob_Id	PDO CommPar	rw	Cob Id RX PDO 1	200H + NodeId
1400	2	RX_PDO1_Tx_Type	PDO CommPar	ro	Transmission Type Rx PDO #1	254
1401	1	RX_PDO2_Cob_Id	PDO CommPar	rw	Cob Id RX PDO 2	300H + NodeId
1401	2	RX_PDO2_Tx_Type	PDO CommPar	ro	Transmission Type Rx PDO #2	254
1600÷1601	0 ÷ 4	Receive PDOs (1 ÷ 2) mapping	PDO Mapping	ro	Mapping RX PDOs	----
1800	1	TX_PDO1_Cob_Id	PDO CommPar	rw	Cob Id TX PDO 1	180H + NodeId
1800	2	TX_PDO1_Tx_Type	PDO CommPar	ro	Transmission Type Tx PDO #1	254
1800	5	TX_PDO1_Event_Timer	PDO CommPar	rw	Timer Tx PDO #1	100
1801	1	TX_PDO2_Cob_Id	PDO CommPar	rw	Cob Id TX PDO 2	280H + NodeId
1801	2	TX_PDO2_Tx_Type	PDO CommPar	ro	Transmission Type Tx PDO #2	254
1801	5	TX_PDO2_Event_Timer	PDO CommPar	rw	Timer Tx PDO #2	100
1A00÷1A01	0 ÷ 4	Transmit PDOs (1 ÷ 2) mapping	PDO Mapping	ro	Mapping TX PDOs	----

12.0 EtherCAT Protocol

The drives equipped with EtherCAT fieldbus and e3PLC programmability have a configuration code of C690. They do not need to be configured as NodeId and BaudRate. The protocol supported is: CoE (CANopen over EtherCAT). The EVER drives supports different types of synchronization: Free Run, Synchronous with SM Event, Distributed Clocks. The services EMCY (§*11.6*) and Diagnostics are supported too.

12.1 LEDS

Near the EtherCAT Connector A there is a RUN Led that can be in one of the following situations:

State	Slave Condition	Communication
Off	Init	After switch-on the EtherCAT slave is in the *Init* state. No SDO or PDO communication is possible.
Blinking	Pre-Operational	In *Pre-Operational* state SDO communication is possible, but not PDO communication.
Single Flash	Safe-Operational	In *Safe-Operational* state SDO and PDO communication is possible, although the slave keeps its outputs (RPDO) in a safe state, while the input (TPDO) data are updated cyclically.
On	Operational	In the *Operational* state the slave copies the output data of the masters to its outputs (RPDO). PDO and SDO communication is possible.

Some drive models (SW5) have also an ERROR Led that can be in one of the following situations:

State	Ethercat State
Off	No Error
Blinking	Invalid Configuration
Single Flash	Unsolicited State Change
Double Flash	Application Watchdog Timeout
Flickering	Booting Error
On	PDI Watchdog Timeout

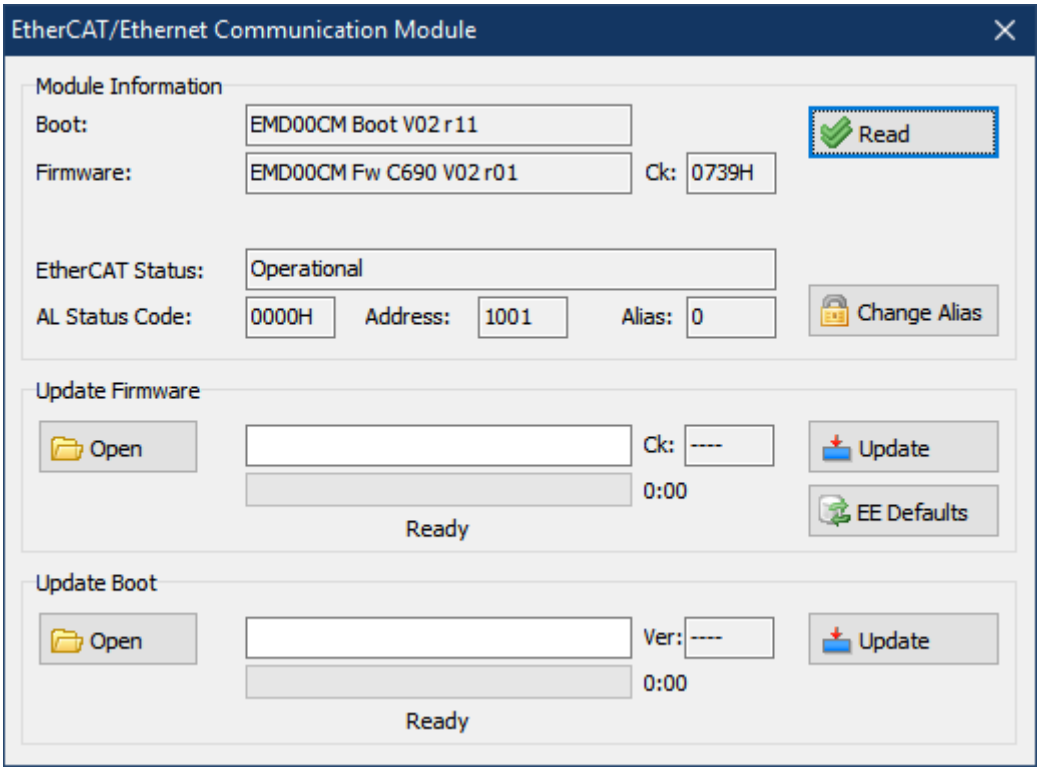
12.2 PDO Mapping

The PDO mapping for EtherCAT devices is fixed and contains the same objects of the CANopen configuration.

Direction	Description
RPDO	*CANopen_RX_PDO1_Data[0]* (2300.2H), *CANopen_RX_PDO1_Data[1]* (2300.3H), *CANopen_RX_PDO2_Data[0]* (2300.4H), *CANopen_RX_PDO2_Data[1]* (2300.5H)
TPDO	*Drive_Register* (4000.3H), *Error_Register* (4000.2H), *Position_Actual_Value* (6063.0H), *CANopen_TX_PDO2_Data[0]* (2301.3H), *CANopen_TX_PDO2_Data[1]* (2301.4H)

12.3 Station Alias Setting

For PLCs that use the Station Alias to identify the various devices on EtherCAT network (for example OMRON® PLCs) the Station Alias can be set either using the PLC functionality or using the eePLC Studio IDE. To set the Station Alias from the eePLC Studio IDE it is necessary to enter in ONLINE condition (see §*3.0*) and select the menu *'Debug/EtherCAT/Net Comm.Module'*.



Pressing the *'Change Alias'* button it is possible to select a different station Alias. The new setting will be active at the next drive switch-on.

12.4 EtherCAT Slave Information (ESI)

The EtherCAT Slave Information file (XML file) is available for configuring the EtherCAT master. It can be found on the CD-ROM supplied with the drive software kit under the \XML directory.

A Appendix – Multiplexed IO allocations

The following table shows the functions assigned to each digital input.

Titanio Series Drives							
Input	Encoder Quadrature[3] Counter_Config[x]=0	Up/down Counter[3] Counter_Config[x]=2	Up Counter[3] Counter_Config[x]=1	Start Trigger (4)(8)	Stop Trigger (5)(8)	Electric Gear Source (6)	Impact Source (7)
B0_In0				√	√		
B0_In1				√	√		
B0_In2	Phase B Encoder #1	Encoder #1(dir)		√	√	√	√
B0_In3	Phase A Encoder #1	Encoder #1 (clock)	Encoder #1(clock)	√	√		
B0_In4							
B0_In5	Zero Encoder #0			√	√		
B0_In6	Phase A Encoder #0			√	√		√
B0_In7	Phase B Encoder #0			√	√		
B1_In0 (1)							
B1_In1 (1)							
B1_In2 (1)							
B1_In3 (1)							
B1_In4 (1)							
B1_In5 (1)							
B1_In6 (1)							
B1_In7 (1)							

Follows the hardware functions assigned to each digital output.

	Titanio Series Drives
Output	**Function [2]**
B0_Out0	Fault (Off) / Drive Ok (On)[10]
B0_Out1	Ready (On) - Busy (Off) / Clockout[9]
B1_Out0 ([1])	//
B1_Out1 ([1])	//
B1_Out2 ([1])	//
B1_Out3 ([1])	//
B1_Out4 ([1])	//
B1_Out5 ([1])	//
B1_Out6 ([1])	//
B1_Out7 ([1])	//

([1]) Available only on drives with expansion.
([2]) See bit7 of *Drive_Working_Settings*.
([3]) See *Counter_Config* object for more details on encoder uses.
([4]) See *Drive_Inputs_Setting* and *Direct_Command_CMD* objects and *§8.3.3 Movements with Trigger*.
([5]) See *Drive_Inputs_Setting* and *Direct_Command_CMD* objects.
([6]) See *§9.2 Electric Gear Features*.
([7]) See *§9.1 Impact Feature*.
([8]) Enabling the Start/Stop Trigger function(from *Direct_Command_CMD* objects) set on the selected input a digital filter that limit the maximun input frequency to 5 Khz so pay attention to not use the same input also for counter functionality. Once the digital filter is set it will remain set until system switch off.
([9]) The clockout function has the priority respect to the ready-busy function. So if the Clockout is enabled to use B0_Out1 then this digital output is used as Clockout and not as Ready-Busy or user handling. See *§9.3 Clockout Feature*.
([10]) The clockout function has the priority respect to the 'Fault/Drive Ok' function. So if the Clockout is enabled to use B0_Out0 then this digital output is used as Clockout and not as 'Fault/Drive Ok' or user handling. See *§9.3 Clockout Feature*.

Clockout_Prescaler	Drive_Working_Setting.Disable_Digital_Outputs_FW_Handling	B0_Out1 driven by
0	0	Ready-Busy
0	1	B0_Digital_Outputs
> 0	0	Clockout
> 0	1	Clockout

Clockout_Prescaler	Drive_Working_Setting_Extended.Disable_Fault_Output	B0_Out0driven by
0	0	Fault-Drive Ok
0	1	B0_Digital_Outputs
> 0	0	Clockout
> 0	1	Clockout

Examples:

Reading the position of an incremental encoder:

- Connect the two encoder phases to digital inputs B0_IN2 and B0_IN3.
- Configure the hardware counter (object *Counter_Config[1]* = 0)
- The *Encoder_Actual_Value[1]* object, returns the position of the encoder connected to digital inputs B0_IN2 and B0_IN3 while the *Encoder_Frequency[1]* object returns the encoder pulses frequency.

Reading the pulses from an external clock source

- Connect the clock source to digital input B0_IN3.
- Configure the hardware counter (object *Counter_Config[1]* = 1)
- The *Encoder_Actual_Value[1]* object, returns the pulses number generated by external clock source while the *Encoder_Frequency[1]* object returns the pulses frequency.

Reading the pulses from an external clock & direction source

- Connect the clock source to digital input B0_IN3, and the direction signal to B0_IN2.
- Configure the hardware counter (object *Counter_Config[1]* = 2)
- The *Encoder_Actual_Value[1]* object, returns the pulses number generated by external clock source (positive or negative depending on the direction signal) while the *Encoder_Frequency[1]* object returns the pulses frequency.

B Appendix – Display Status

The 7 segments drive (only for drives fitted with it) display can have the following status:

Display Status	Description
L	Drive in boot mode. A new firmware should be downloaded to drive.
U	Firmware update in progress. Do not power off the drive until the update process is completed!
I	Initialization phase. Should last few seconds. While in this condition the drive is not fully operational.
S	Fixed Character = Drive ok and operational Blinking Character = (Master Enable off)
S+1	Warning : Power supply near limit
S+3	Warning : Temperature near limit
S+7	Warning : EEprom near Write Overrun
S+8	Warning : EEprom near End of Life
E+3	Error: Expired eePLC Software Trial
F+0	Error : Watchdog
F+1	Error : Internal software error
F+2	Error : Missing Calibration
F+4	Error : Eeprom fail
F+6	Error : eePLC Application error (end of program execution, division by zero, etc.)
F+7	Error : EEprom Write Overrun
F+U	Error: Feature Unavailable (the application tried to use a feature (for instance the CAM Module) that is not available in the current drive configuration
P+0	Protection: Motor is in open phase condition
P+1	Voltage protection
P+2	Current protection
P+3	Thermal protection
P+5	Missing Torque Enable
P+6	Motor Current Regulation is out of range (*)

P+7	eePLC User Protection (generated by setting bit #0 of *eePLC_User_Settings*)
P9	Feedback Error

(*) Verify Motor Currents correctness

C Appendix – Analog Inputs

Some Titanio drive models have Analog Inputs according to the drive version. There are some differences in the behavior of the Analog Inputs between the boards. The following table shows these differences:

Connection Type	*Boards*		
	IMD08	**IMD02 IMD04**	**IMDxx ISDxx**
Differential[(1)]	-10V ÷ 10V	Not Applicable	-10V ÷ 10V
Potentiometer with internal reference[(2)]	0V ÷ 3.3V	0V ÷ 5V	0V ÷ 5V
Potentiometer with external reference	Not Applicable	Not Applicable	0V ÷ 10V

(1) IMD08: (JMP600 – position 1)
(2) IMD08: (JMP600 – position 2)

Please refers to *'Installation user manual'* for details about the connection of Analog Inputs.

Drive Objects Index

This appendix shows all Titanio-Platino-Vanadio Series Drives objects and commands sorted by name to facilitate their identification in this manual.