iPOS360x HX-CAN

TECHNOSOFT

Intelligent Servo Drive for Step, DC, Brushless DC and AC Motors

Intelligent Servo Drive

Technical Reference

TECHNOSOFT

iPOS360x HX-CAN

Technical Reference

Technosoft S.A.
Avenue des Alpes 20
CH-2000 NEUCHATEL
Switzerland
Tel.: +41 (0) 32 732 5500
Fax: +41 (0) 32 732 5504
e-mail: contact@technosoftmotion.com
http://www.technosoftmotion.com/

Read This First

About This Manual

This book is a technical reference manual for the **iPOS360x HX-CAN** family of intelligent servo drives, including the following products:

iPOS3602 HX-CAN (p/n P028.001.E501)
iPOS3604 HX-CAN (p/n P028.002.E501)

In order to operate the **iPOS360x** drives, you need to pass through 3 steps:

- ❑ **Step 1 Hardware installation**
- ❑ **Step 2 Drive setup** using Technosoft **EasySetUp** software for drive commissioning
- ❑ **Step 3 Motion programming** using one of the options:

  - ❑ A **CANopen master**[1]
  - ❑ The drive's **built-in motion controller** executing a Technosoft Motion Language (**TML**) program developed using Technosoft **EasyMotion Studio** software
  - ❑ A **TML_LIB motion library for PCs** (Windows or Linux)
  - ❑ A **TML_LIB motion library for PLCs**
  - ❑ A **distributed control** approach which combines the above options, like for example a host calling motion functions programmed on the drives in TML

This manual covers **Step 1** in detail. It describes the **iPOS360x HX-CAN** hardware including the technical data, the connectors and the wiring diagrams needed for installation. The manual also presents an overview of the following steps, and includes the scaling factors between the real SI units and the drive internal units. For detailed information regarding the next steps, refer to the related documentation.

Notational Conventions

This document uses the following conventions:

- • **iPOS360x**– all products described in this manual
- • **IU units** – Internal units of the drive
- • **SI units** – International standard units (meter for length, seconds for time, etc.)
- • **TML** – Technosoft Motion Language
- • **TMLCAN** – Technosoft protocol for exchanging TML commands via CAN-bus

---

[1] when iPOS360x is set in CANopen mode

Related Documentation

Help of the EasySetUp software – describes how to use **EasySetUp** to quickly setup any Technosoft drive for your application using only 2 dialogues. The output of EasySetUp is a set of setup data that can be downloaded into the drive EEPROM or saved on a PC file. At power-on, the drive is initialized with the setup data read from its EEPROM. With EasySetUp it is also possible to retrieve the complete setup information from a drive previously programmed. **EasySetUp can be downloaded free of charge from Technosoft web page**

***iPOS CANopen Programming* (part no. P091.063.iPOS.UM.xxxx)** – explains how to program the iPOS family of intelligent drives using **CANopen** protocol and describes the associated object dictionary for **CiA 301 v.4.2** application layer and communication profile, **CiA WD 305 v.2.2.13** layer settings services and protocols and **CiA DSP 402 v3.0** device profile for drives and motion control now included in IEC 61800-7-1 Annex A, IEC 61800-7-201 and IEC 61800-7-301 standards

***Motion Programming using EasyMotion Studio* (part no. P091.034.ESM.UM.xxxx)** – describes how to use the EasyMotion Studio to create motion programs using in Technosoft Motion Language (TML). EasyMotion Studio platform includes **EasySetUp** for the drive/motor setup, and a **Motion Wizard** for the motion programming. The Motion Wizard provides a simple, graphical way of creating motion programs and automatically generates all the TML instructions. *With EasyMotion Studio you can fully benefit from a key advantage of Technosoft drives – their capability to execute complex motions without requiring an external motion controller, thanks to their built-in motion controller.* **A demo version of EasyMotion Studio (with EasySetUp part fully functional) can be downloaded free of charge from Technosoft web page**

***TML_LIB v2.0* (part no. P091.040.v20.UM.xxxx)** – explains how to program in **C, C++,C#, Visual Basic or Delphi Pascal** a motion application for the Technosoft intelligent drives using TML_LIB v2.0 motion control library for PCs. The TML_lib includes ready-to-run examples that can be executed on **Windows** or **Linux** (x86 and x64).

***TML_LIB_LabVIEW v2.0* (part no. P091.040.LABVIEW.v20.UM.xxxx)** – explains how to program in **LabVIEW** a motion application for the Technosoft intelligent drives using TML_LIB_Labview v2.0 motion control library for PCs. The TML_Lib_LabVIEW includes over 40 ready-to-run examples.

***TML_LIB_S7* (part no. P091.040.S7.UM.xxxx)** – explains how to program in a PLC **Siemens series S7-300 or S7-400** a motion application for the Technosoft intelligent drives using TML_LIB_S7 motion control library. The TML_LIB_S7 library is **IEC61131-3 compatible**.

***TML_LIB_CJ1* (part no. P091.040.CJ1.UM.xxxx)** – explains how to program in a PLC **Omron series CJ1** a motion application for the Technosoft intelligent drives using TML_LIB_CJ1 motion control library for PLCs. The TML_LIB_CJ1 library is **IEC61131-3 compatible**.

***TML_LIB_X20* (part no. P091.040.X20.UM.xxxx)** – explains how to program in a PLC **B&R series X20** a motion application for the Technosoft intelligent drives using TML_LIB_X20 motion control library for PLCs. The TML_LIB_X20 library is **IEC61131-3 compatible**.

***TechnoCAN* (part no. P091.063.TechnoCAN.UM.xxxx)** – presents TechnoCAN protocol – an extension of the CANopen communication profile used for TML commands

If you Need Assistance …

If you want to …	Contact Technosoft at …
Visit Technosoft online	World Wide Web: http://www.technosoftmotion.com/
Receive general information or assistance (see Note)	World Wide Web: http://www.technosoftmotion.com/ Email: contact@technosoftmotion.com
Ask questions about product operation or report suspected problems (see Note)	Fax: (41) 32 732 55 04 Email: hotline@technosoftmotion.com
Make suggestions about, or report errors in documentation (see Note)	Mail: Technosoft SA Avenue des Alpes 20 CH-2000 Neuchatel Switzerland

Contents

1. Safety information

Read carefully the information presented in this chapter before carrying out the drive installation and setup! It is imperative to implement the safety instructions listed hereunder.

This information is intended to protect you, the drive and the accompanying equipment during the product operation. Incorrect handling of the drive can lead to personal injury or material damage.

Only qualified personnel may install, set up, operate and maintain the drive. A "qualified person" has the knowledge and authorization to perform tasks such as transporting, assembling, installing, commissioning and operating drives.

The following safety symbols are used in this manual:

	WARNING!	*SIGNALS A DANGER THAT MIGHT CAUSE BODILY INJURY TO THE OPERATOR. MAY INCLUDE INSTRUCTIONS TO PREVENT THIS SITUATION*
	CAUTION!	*SIGNALS A DANGER FOR THE DRIVE, WHICH MIGHT DAMAGE THE PRODUCT OR OTHER EQUIPMENT. MAY INCLUDE INSTRUCTIONS TO AVOID THIS SITUATION*

1.1. Warnings

	WARNING!	*TO AVOID ELECTRIC ARCING AND HAZARDS, NEVER PLUG / UNPLUG THE iPOS360x HX-CAN FROM IT'S SOCKET WHILE THE POWER SUPPLIES ARE ON!*
	WARNING!	**THE DRIVE MAY HAVE HOT SURFACES DURING OPERATION.**
	WARNING!	**DURING DRIVE OPERATION, THE CONTROLLED MOTOR WILL MOVE. KEEP AWAY FROM ALL MOVING PARTS TO AVOID INJURY**

1.2. Cautions

	CAUTION!	**THE POWER SUPPLIES CONNECTED TO THE DRIVE MUST COMPLY WITH THE PARAMETERS SPECIFIED IN THIS DOCUMENT**
	CAUTION!	**TROUBLESHOOTING AND SERVICING ARE PERMITTED ONLY FOR PERSONNEL AUTHORISED BY TECHNOSOFT**

2. Product Overview

2.1. Introduction

The **iPOS360x HX-CAN** is a family of fully digital intelligent servo drives, based on the latest DSP technology and they offer unprecedented drive performance combined with an embedded motion controller.

Suitable for control of brushless DC, brushless AC (vector control), DC brushed motors and step motors, the iPOS360x HX-CAN drives accept as position feedback incremental encoders (quadrature or sine/cosine) and linear Halls signals.

All drives perform position, speed or torque control and work in single, multi-axis or stand-alone configurations. Thanks to the embedded motion controller, the iPOS360x HX-CAN drives combine controller, drive and PLC functionality in a single compact unit and are capable to execute complex motions without requiring intervention of an external motion controller. Using the high-level Technosoft Motion Language (**TML**) the following operations can be executed directly at drive level:

- ❑ Setting various motion modes (profiles, PVT, PT, electronic gearing[1] or camming[1], etc.)
- ❑ Changing the motion modes and/or the motion parameters
- ❑ Executing homing sequences
- ❑ Controlling the program flow through:

    - ▪ Conditional jumps and calls of TML functions

    - ▪ TML interrupts generated on pre-defined or programmable conditions (protections triggered, transitions on limit switch or capture inputs, etc.)
    - ▪ Waits for programmed events to occur
- ❑ Handling of digital I/O and analogue input signals
- ❑ Executing arithmetic and logic operations
- ❑ Performing data transfers between axes
- ❑ Controlling motion of an axis from another one via motion commands sent between axes
- ❑ Sending commands to a group of axes (multicast). This includes the possibility to start simultaneously motion sequences on all the axes from the group
- ❑ Synchronizing all the axes from a network

By implementing motion sequences directly at drive level you can really distribute the intelligence between the master and the drives in complex multi-axis applications, reducing both the development time and the overall communication requirements. For example, instead of trying to command each movement of an axis, you can program the drives using TML to execute complex motion tasks and inform the master when these tasks are done. Thus, for each axis control the master job may be reduced at: calling TML functions stored in the drive EEPROM and waiting for a message, which confirms the TML functions execution completion.

All iPOS360x drives are equipped with a serial RS232 and a CAN 2.0B interface and can be set (hardware, via a jumper) to operate in 2 modes:

- ❑ **CANopen**
- ❑ **TMLCAN**

When **CANopen** mode is selected, the iPOS360x conforms to **CiA 301 v4.2** application layer and communication profile, **CiA WD 305 v2.2.13** and **CiA DSP 402 v3.0** device profile for drives and motion control, now included in IEC 61800-7-1 Annex A, IEC 61800-7-201 and IEC 61800-7-301 standards. In this mode, the iPOS360x may be controlled via a CANopen master. As a bonus, iPOS360x offers a CANopen master the

---

[1] Available if the master axis sends its position via a communication channel

option to call motion sequences, written in TML and stored in the drive EEPROM, using manufacturer specific objects (see for details par. 5.3).

When **TMLCAN** mode is selected, the iPOS360x behaves as standard Technosoft intelligent drive and conforms to Technosoft protocol for exchanging TML commands via CAN-bus. When TMLCAN protocol is used, it is not mandatory to have a master. Any iPOS360x can be set to operate standalone, and may play the role of a master to coordinate both: network communication/ synchronization and the motion application via TML commands sent directly to the other drives.

When higher level coordination is needed, apart from a CANopen master, the iPOS360x drives can also be controlled via a PC or a PLC using one of the **TML_LIB** motion libraries.

For iPOS360x commissioning **EasySetUp** or **EasyMotion Studio** PC applications may be used.

EasySetUp is a subset of EasyMotion Studio, including only the drive setup part. The output of EasySetUp is a set of setup data that can be downloaded into the drive EEPROM or saved on a PC file. At power-on, the drive is initialized with the setup data read from its EEPROM. With EasySetUp it is also possible to retrieve the complete setup information from a drive previously programmed. EasySetUp shall be used for drive setup in all cases where the motion commands are sent exclusively from a master. Hence neither the iPOS360x TML programming capability nor the drive camming mode are used. **EasySetUp can be downloaded free of charge from Technosoft web page.**

EasyMotion Studio platform includes EasySetUp for the drive setup, and a **Motion Wizard** for the motion programming. The Motion Wizard provides a simple, graphical way of creating motion programs and automatically generates all the TML instructions. *With EasyMotion Studio you can fully benefit from a key advantage of Technosoft drives – their capability to execute complex motions without requiring an external motion controller, thanks to their built-in motion controller*s. EasyMotion Studio, shall be used to program motion sequences in TML. This is the iPOS360x typical operation mode when TMLCAN protocol is selected. EasyMotion Studio shall also be used with CANopen protocol, if the user wants to call TML functions stored in the drive EEPROM or to use the camming mode. With camming mode, EasyMotion Studio offers the possibility to quickly download and test a cam profile and also to create a **.sw** file (see par. 5.2.4) with the cam data. The .sw file can be afterwards stored in a master and downloaded to the drive, wherever needed. **A demo version of EasyMotion Studio (with EasySetUp part fully functional) can be downloaded free of charge from Technosoft web page.**

2.2. Key Features

- Fully digital servo drive suitable for the control of rotary or linear brushless, DC brush, and step motors
- Very compact design
- Sinusoidal (FOC) or trapezoidal (Hall-based) control of brushless motors
- Open or closed-loop control of 2 and 3-phase steppers
- Various modes of operation, including: torque, speed or position control; position or speed profiles, external analogue reference or sent via a communication bus
- Technosoft Motion Language (TML) instruction set for the definition and execution of motion sequences
- Standalone operation with stored motion sequences
- Various motion programming modes:
    - Position profiles with trapezoidal or S-curve speed shape
    - Position, Velocity, Time (PVT) 3rd order interpolation
    - Position, Time (PT) 1st order interpolation
    - Electronic gearing and camming[1]
    - 35 Homing modes
- Single-ended and RS-422 differential encoder interface
- Single-ended, open collector Digital Hall sensor interface

---

[1] Available if the master axis sends its position via a communication channel

- Linear Hall sensor interface
- Analogue Sin/Cos encoder interface (differential 1V$_{pp}$)
- 1 analogue input: 12-bit, 0-5V, Reference or Feedback or general-purpose
- 5 digital inputs: 5-36V, PNP or NPN software selectable: Enable, Limit switch +, Limit switch −, 2 general-purpose
- 4 digital outputs : 5-36V, 0.5A, NPN open-collector: Ready, Error, 2 general-purpose
- RS-232 serial interface (up to 115200 bps)
- CAN-bus 2.0B up to 1Mbit/s with hardware axis ID selection[1]
- Two operation modes selectable by jumper:
  - **CANopen** – conforming with **CiA 301 v4.2, CiA WD 305 v2.2.13** and **CiA DSP 402 v3.0**
  - **TMLCAN** – intelligent drive conforming with Technosoft protocol for exchanging TML commands via CAN-bus
- 2 K × 16 internal SRAM memory for data acquisition
- 4K × 16 E^{2}ROM to store TML programs and data
- PWM switching frequency up to 100kHz
- Motor supply: 9-36V
- Logic supply: 9-36V.Separate supply is optional
- Output current:
  - iPOS3602 HX-CAN: 2A[2] continuous; 3.2A peak
  - iPOS3604 HX-CAN: 4A[3] continuous; 10A peak
- Operating ambient temperature: 0-40°C (over 40°C with derating)
- Protections:
  - Short-circuit between motor phases
  - Short-circuit from motor phases to ground
  - Over-voltage
  - Under-voltage
  - Over-current
  - Communication error
  - Control error

---

[1] 15 hardware addresses in CANopen mode or 16 in TMLCAN mode
[2] 2A cont. with DC, step and BLDC motors (trapezoidal), 2A amplitude (1.41A$_{RMS}$) for PMSM (sinusoidal)
[3] 4A cont. with DC, step and BLDC motors (trapezoidal), 4A amplitude (2.82A$_{RMS}$) for PMSM (sinusoidal)

2.3. Supported Motor-Sensor Configurations

iPOS360x HX-CAN supports the following configurations:

1. Position, speed or torque control of a **brushless AC rotary motor** with an **incremental quadrature encoder** on its shaft. The brushless motor is vector controlled like a permanent magnet synchronous motor. It works with **sinusoidal** voltages and currents. Scaling factors take into account the transmission ratio between motor and load (rotary or linear). Therefore, the motion commands (for position, speed and acceleration) expressed in SI units (or derivatives) refer to the load[1], while the same commands, expressed in IU units, refer to the motor.
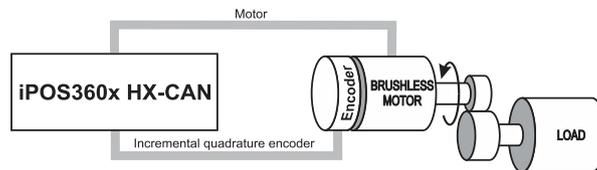
Figure 2.1. *Brushless AC rotary motor. Position / speed / torque control.*
Quadrature encoder on motor.

2. Position, speed or torque control of a **brushless AC rotary motor** with an **incremental sine/cosine encoder** on its shaft. The brushless motor is vector controlled like a permanent magnet synchronous motor. It works with **sinusoidal** voltages and currents. Scaling factors take into account the transmission ratio between motor and load (rotary or linear). Therefore, the motion commands (for position, speed and acceleration) expressed in SI units (or derivatives) refer to the load[1], while the same commands, expressed in IU units, refer to the motor.
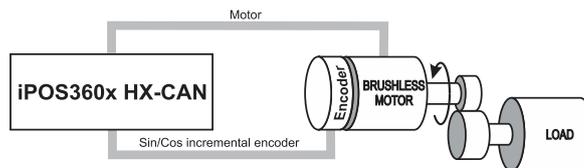
Figure 2.2. *Brushless AC rotary motor. Position/speed/torque control.*
Sine/cosine incremental encoder on motor

3. Position, speed or torque control of a **brushless AC linear motor** with an **incremental sine/cosine encoder**. The brushless motor is vector controlled like a permanent magnet synchronous motor. It works with **sinusoidal** voltages and currents. Scaling factors take into account the transmission ratio between motor and load (rotary or linear). Therefore, the motion commands (for position, speed and acceleration) expressed in SI units (or derivatives) refer to the load, while the same commands, expressed in IU units, refer to the motor.
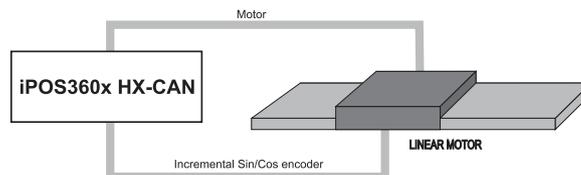
Figure 2.3. *Brushless AC linear motor. Position/speed/torque control.*
Sine/cosine incremental encoder on motor

4. Position, speed or torque control of a **brushless DC rotary motor** with **digital Hall sensors** and an **incremental quadrature encoder** on its shaft. The brushless motor is controlled using Hall sensors for commutation. It works with rectangular currents and **trapezoidal** BEMF voltages. Scaling factors take into account the transmission ratio between motor and load (rotary or linear). Therefore, the motion commands (for position, speed and acceleration) expressed in SI units (or derivatives) refer to the load, while the same commands, expressed in IU units, refer to the motor.

---

[1] Motion commands can be referred to the motor by setting in EasySetUp a rotary to rotary transmission with ratio 1:1
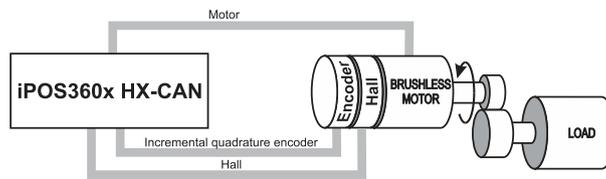
Figure 2.4. *Brushless DC rotary motor. Position / speed / torque control.*
Hall sensors and quadrature encoder on motor.

5. Position, speed or torque control of a **brushless AC linear motor** with an **incremental quadrature linear encoder** on the track. The brushless motor is vector controlled like a permanent magnet synchronous motor. It works with **sinusoidal** voltages and currents. Scaling factors take into account the transmission ratio between motor and load (linear or rotary). Therefore, the motion commands (for position, speed and acceleration) expressed in SI units (or derivatives) refer to the load[1], while the same commands, expressed in IU units, refer to the motor.
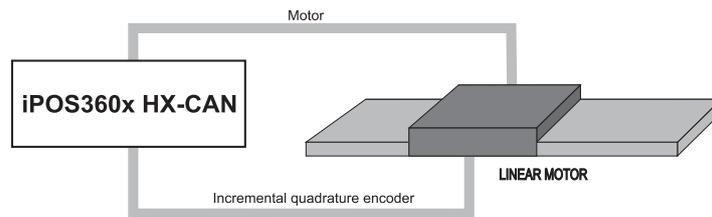
Figure 2.5. *Brushless AC linear motor with incremental quadrature linear encoder.*
Position / speed / torque control.

6. Position, speed or torque control of a **brushless AC rotary motor** with **linear Hall signals**. The brushless motor is vector controlled like a permanent magnet synchronous motor. It works with **sinusoidal** voltages and currents. Scaling factors take into account the transmission ratio between motor and load (rotary or linear). Therefore, the motion commands (for position, speed and acceleration) expressed in SI units (or derivatives) refer to the load[2], while the same commands, expressed in IU units, refer to the motor.
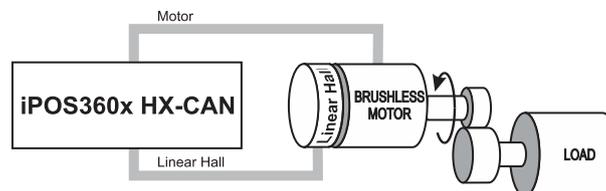
Figure 2.6. *Brushless AC rotary motor with linear Hall signals.*
Position / speed / torque control.

7. Position, speed or torque control of a **brushless AC linear motor** with **linear Hall signals**. The brushless motor is vector controlled like a permanent magnet synchronous motor. It works with **sinusoidal** voltages and currents. Scaling factors take into account the transmission ratio between motor and load (rotary or linear). Therefore, the motion commands (for position, speed and acceleration) expressed in SI units (or derivatives) refer to the load, while the same commands, expressed in IU units, refer to the motor.

---

[1] Motion commands can be referred to the motor by setting in EasySetUp a linear to linear transmission with ratio 1:1
[2] Motion commands can be referred to the motor by setting in EasySetUp a rotary to rotary transmission with ratio 1:1
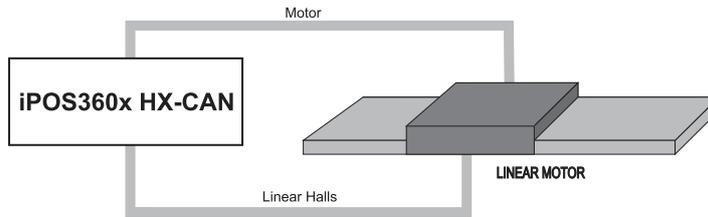
Figure 2.7. *Brushless AC linear motor with linear Hall signals.*
Position / speed / torque control.

8. Position, speed or torque control of a **DC brushed rotary motor** with an **incremental quadrature encoder** on its shaft. Scaling factors take into account the transmission ratio between motor and load (rotary or linear). Therefore, the motion commands (for position, speed and acceleration) expressed in SI units (or derivatives) refer to the load[1], while the same commands, expressed in IU units, refer to the motor.
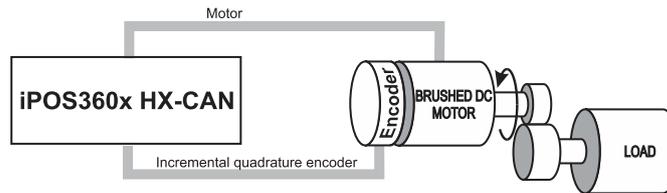
Figure 2.8. *DC brushed rotary motor. Position / speed / torque control.*
Quadrature encoder on motor.

9. Load position control using an **incremental quadrature encoder** on load, combined with speed control of a **DC brushed rotary motor** having a **tachometer** on its shaft. The motion commands (for position, speed and acceleration) in both SI and IU units refer to the load.
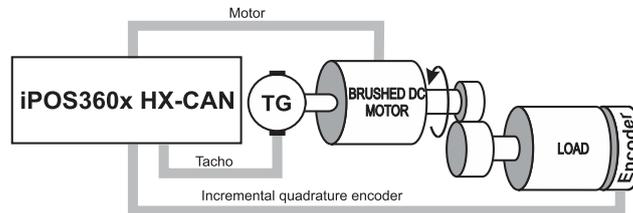
Figure 2.9. *DC brushed rotary motor. Position / speed / torque control.*
Quadrature encoder on load plus tachometer on motor.

10. Speed or torque control of a **DC brushed rotary motor** with a **tachometer** on its shaft. Scaling factors take into account the transmission ratio between motor and load (rotary or linear). Therefore, the motion commands (for speed and acceleration) expressed in SI units (or derivatives) refer to the load[21], while the same commands, expressed in IU units, refer to the motor.
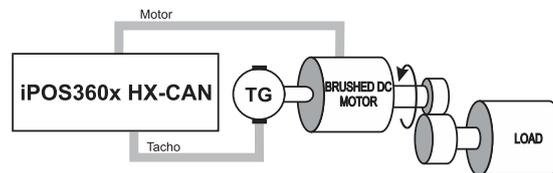

Figure 2.10. *DC brushed rotary motor. Speed/torque control.*
Tachometer on motor.

---

[1] Motion commands can be referred to the motor by setting in EasySetUp a rotary to rotary transmission with ratio 1:1

[2] Motion commands can be referred to the motor by setting in EasySetUp a rotary to rotary transmission with ratio 1:1

---

11. Open-loop control of a 2 or 3-phase **step motor** in position or speed. Scaling factors take into account the transmission ratio between motor and load (rotary or linear). Therefore, the motion commands (for position, speed and acceleration) expressed in SI units (or derivatives) refer to the load, while the same commands, expressed in IU units, refer to the motor.
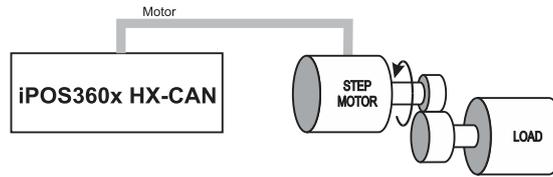
Figure 2.11. *No position or speed feedback.*
Open-loop control: motor position or speed.

12. Closed-loop control of **load position using an encoder on load**, combined with open-loop control of **a 2 or 3-phase step motor** in speed, with speed reference provided by the position controller. The motion commands in both SI and IU units refer to the load.
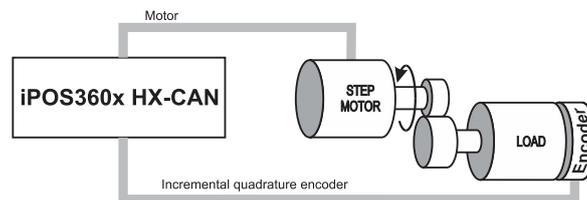
Figure 2.12. *Encoder on load.*
Closed-loop control: load position, open-loop control: motor speed.

13. Closed-loop control of **a 2-phase step motor** in position, speed or torque. Scaling factors take into account the transmission ratio between motor and load (rotary or linear). Therefore, the motion commands expressed in SI units (or derivatives) refer to the load[1], while the same commands, expressed in IU units refer to the motor.
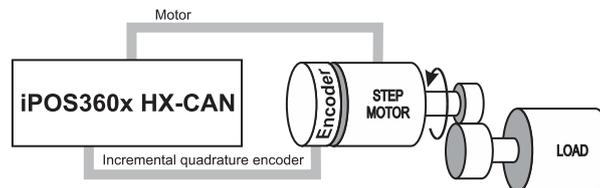
Figure 2.13. *Encoder on motor shaft.*
Closed-loop control: motor position, speed or torque.

---

[1] Motion commands can be referred to the motor by setting in EasySetUp a rotary to rotary transmission with ratio 1:1

2.4. iPOS360x HX-CAN Drive Dimensions
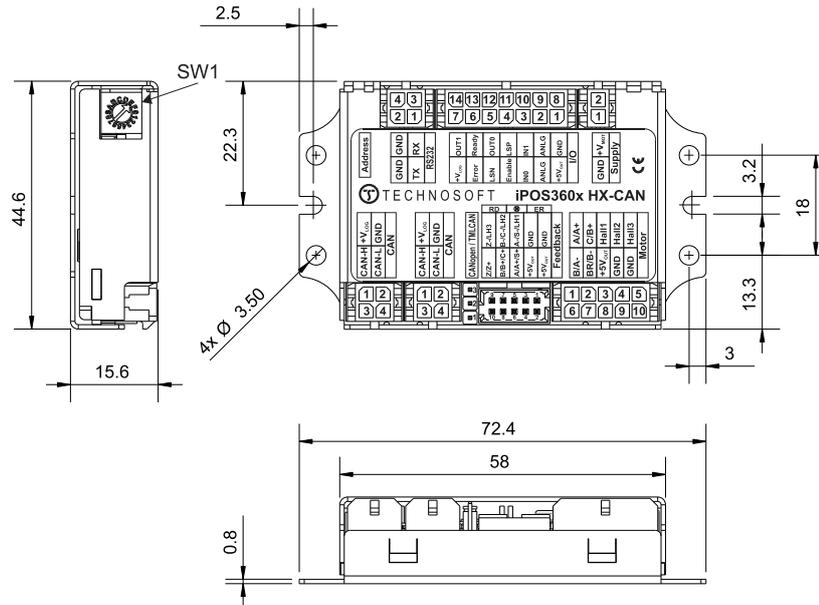
Figure 2.14. *iPOS360x HX-CAN drive dimensions*

All dimensions are in mm. The drawings are not to scale.

2.5. Identification Labels

Identification label
with
Technosoft part number

Identification label
with
Technosoft serial number
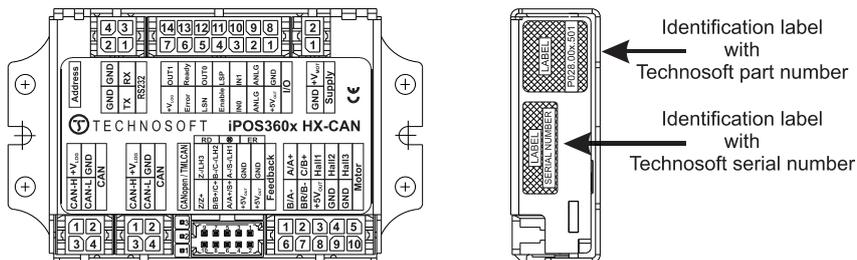
Figure 2.15. *iPOS360x HX-CAN Identification Labels*

2.6. Electrical Specifications

All parameters measured under the following conditions (unless otherwise specified):

T_{amb} = 0…40°C, V_{LOG} = 24 V_{DC}; V_{MOT} = 36V_{DC}; Supplies start-up / shutdown sequence: -*any*-
Load current (sinusoidal amplitude / continuous BLDC, DC, stepper) = 4A iPOS3604; 2A iPOS3602;

2.6.1. Operating Conditions

		Min.	Typ.	Max.	Units
Ambient temperature[1]		0		+40	ºC
Ambient humidity	Non-condensing	0		90	%Rh
Altitude / pressure[2]	Altitude (referenced to sea level)	-0.1	0 ÷ 2.5	[2]	Km
	Ambient Pressure	0 [2]	0.75 ÷ 1	10.0	atm

2.6.2. Storage Conditions

		Min.	Typ.	Max.	Units
Ambient temperature		-40		+85	ºC
Ambient humidity	Non-condensing	0		100	%Rh
Ambient Pressure		0		10.0	atm

2.6.3. Mechanical Mounting

Airflow			natural convection[3], closed box		

2.6.4. Environmental Characteristics

		Min.	Typ.	Max.	Units
Size (Length x Width x Height)	Without mating connector		72.4 x 44.6 x 15.6		mm
			~2.85 x 1.76 x 0.61		inch
Weight	Without mating connectors		48		g
Power dissipation	Idle (no load)		1		W
	Operating		3	5	
Efficiency			98		%
Cleaning agents	Dry cleaning is recommended		Only Water- or Alcohol- based		
Protection degree	According to IEC60529, UL508		IP20		-

2.6.5. Logic Supply Input (+V_{LOG})

			Min.	Typ.	Max.	Units
Supply voltage	Nominal values		9		36	V_{DC}
	Absolute maximum values, drive operating but outside guaranteed parameters		5.9		39	V_{DC}
	Absolute maximum values, surge (duration ≤ 10ms) [†]		0		+45	V
Supply current	No Load on Digital Outputs	+V_{LOG} = 9V		125	300	mA
		+V_{LOG} = 12V		80	200	
		+V_{LOG} = 24V		50	125	
		+V_{LOG} = 40V		40	100	

---

[1] Operating temperature can be extended up to **+65ºC** with reduced current and power ratings. See *Figure 2.16* and *Figure 2.21*
[2] iPOS360x can be operated in vacuum (no altitude restriction), but at altitudes over 2,500m, current and power rating are reduced due to thermal dissipation efficiency. See *Figure 2.17* and *Figure 2.22*
[3] It is recommended to mount the iPOS3604 HX-CAN on a metallic support using the provided mounting holes, for better reliability and reduced de-rating due to heat dissipation

2.6.6. Motor Supply Input (+V$_{MOT}$)

			Min.	Typ.	Max.	Units
Supply voltage	Nominal values		9		36	V$_{DC}$
	Absolute maximum values, drive operating but outside guaranteed parameters		8.5		40	V$_{DC}$
	Absolute maximum values, surge (duration $\leq$ 10ms) [†]		0		+42	V
Supply current	Idle			1	5	mA
	Operating	iPOS3604	-10	±4	+10	A
		iPOS3602	-3.2	±2	+3.2	
	Absolute maximum value, short-circuit condition (duration $\leq$ 10ms) [†]	iPOS3604			15	A
		iPOS3602			5	

2.6.7. Motor Outputs (A/A+, B/A-, C/B+, BR/B-)

			Min.	Typ.	Max.	Units
Nominal output current, continuous	for DC brushed, steppers and BLDC motors with Hall-based trapezoidal control	iPOS3604			4	A
		iPOS3602			2	
	for PMSM motors with FOC sinusoidal control (sinusoidal amplitude value)	iPOS3604			4	
		iPOS3602			2	
	for PMSM motors with FOC sinusoidal control (sinusoidal effective value)	iPOS3604			2.82	
		iPOS3602			1.41	
Motor output current, peak	maximum 2.5s	iPOS3604	-10		+10	A
	maximum 24s	iPOS3602	-3.2		+3.2	A
Short-circuit protection threshold	measurement range	iPOS3604		±13	±15	A
		iPOS3602		±4.3	±5	
Short-circuit protection delay			5	10		µS
On-state voltage drop	Nominal output current; including typical mating connector contact resistance			±0.3	±0.5	V
Off-state leakage current				±0.5	±1	mA
Motor inductance (phase-to-phase)	Recommended value, for ripple ±5% of measurement range; +V$_{MOT}$ = 36 V	F$_{PWM}$ = 20 kHz	250			µH
		F$_{PWM}$ = 40 kHz	120			
		F$_{PWM}$ = 60 kHz	100			
		F$_{PWM}$ = 80 kHz	60			
		F$_{PWM}$ = 100 kHz	45			
	Absolute minimum value, limited by short-circuit protection; +V$_{MOT}$ = 36 V	F$_{PWM}$ = 20 kHz	75			µH
		F$_{PWM}$ = 40 kHz	25			
		F$_{PWM}$ = 60 kHz	20			
		F$_{PWM}$ = 80 kHz	10			
		F$_{PWM}$ = 100 kHz	5			
Motor electrical time-constant (L/R)	Recommended value, for ±5% current measurement error due to ripple	F$_{PWM}$ = 20 kHz	250			µs
		F$_{PWM}$ = 40 kHz	125			
		F$_{PWM}$ = 60 kHz	100			
		F$_{PWM}$ = 80 kHz	63			
		F$_{PWM}$ = 100 kHz	50			
Current measurement accuracy	FS = Full Scale			±4	±8	%FS

2.6.8. Digital Inputs (IN0, IN1, IN2/LSP, IN3/LSN, IN4/Enable)[1]

		Min.	Typ.	Max.	Units
Mode compliance			PNP		
Default state	Input floating (wiring disconnected)		Logic LOW		
Input voltage	Logic "LOW"		0	3.2	V
	Logic "HIGH"	4	24	36	
	Floating voltage (not connected)		0		
	Absolute maximum, continuous	-10		+39	
	Absolute maximum, surge (duration ≤ 1s) †	-20		+40	
Input current	Logic "LOW"; Pulled to GND		0	0	mA
	Logic "HIGH"; pulled to +5V		2	3	
	Logic "HIGH"; pulled to +24V		10	15	
Input frequency		0		150	kHz
Minimum pulse width		3.3			µs
ESD protection	Human body model	±5			kV

		Min.	Typ.	Max.	Units
Mode compliance			CMOS / Open-collector / NPN / 24V outputs		
Default state	Input floating (wiring disconnected)		Logic HIGH		
Input voltage	Logic "LOW"		0	3.2	V
	Logic "HIGH"	4	5÷24		
	Floating voltage (not connected)		+Vlog		
	Absolute maximum, continuous	-10		+30	
	Absolute maximum, surge (duration ≤ 1s) †	-20		+40	
Input current	Logic "LOW"; +Vlog=24V		10	15	mA
	Logic "HIGH"; Pulled to +Vlog		0	0	
Input frequency		0		150	kHz
Minimum pulse width		3.3			µs
ESD protection	Human body model	±5			kV

---

[1] The digital inputs are software selectable as PNP or NPN

2.6.9. Digital Outputs (OUT0, OUT1, OUT2/Error, OUT3/ Ready)

		Min.	Typ.	Max.	Units
Mode compliance	All outputs (OUT0, OUT1, OUT2/Error, OUT3/Ready)	TTL / CMOS / Open-collector / NPN 24V			
	Ready, Error	Same as above + LVTTL (3.3V)			
Default state	Not supplied (+V_{LOG} floating or to GND)	High-Z (floating)			
	Immediately after power-up — OUT0, OUT1	Logic "HIGH"			
	Immediately after power-up — OUT2/Error, OUT3/ Ready	Logic "LOW"			
	Normal operation — OUT0, OUT1, OUT2/Error	Logic "HIGH"			
	Normal operation — OUT3/Ready	Logic "LOW"			
Output voltage	Logic "LOW"; output current = 0.5A		0.2	0.8	V
	Logic "HIGH"; output current = 0, no load — OUT2/Error, OUT3/ Ready	2.9	3	3.3	V
	Logic "HIGH"; output current = 0, no load — OUT0, OUT1	4	4.5	5	V
	Logic "HIGH", external load to +V_{LOG}		V_{LOG}		
	Absolute maximum, continuous	-0.5		V_{LOG}+0.5	
	Absolute maximum, surge (duration $\leq$ 1s) [†]	-1		V_{LOG}+1	
Output current	Logic "LOW", sink current, continuous			0.5	A
	Logic "LOW", sink current, pulse ≤ 5 sec.			1	A
	Logic "HIGH", source current; external load to GND; V_{OUT} >= 2.0V — OUT2/Error, OUT3/ Ready			2	mA
	Logic "HIGH", source current; external load to GND; V_{OUT} >= 2.0V — OUT0, OUT1			4	mA
	Logic "HIGH", leakage current; external load to +V_{LOG}; V_{OUT} = V_{LOG} max = 40V		0.1	0.2	mA
Minimum pulse width		2			µs
ESD protection	Human body model	±15			kV

2.6.10. Digital Hall Inputs (Hall1, Hall2, Hall3)

		Min.	Typ.	Max.	Units
Mode compliance		TTL / CMOS / Open-collector			
Default state	Input floating (wiring disconnected)	Logic HIGH			
Input voltage	Logic "LOW"		0	0.8	V
	Logic "HIGH"	2	5		V
	Floating voltage (not connected)		4.4		V
	Absolute maximum, surge (duration $\leq$ 1s) [†]	-10		+15	V

		Min.	Typ.	Max.	Units
Input current	Logic "LOW"; Pull to GND			1.2	mA
	Logic "HIGH"; Internal 4.7KΩ pull-up to +5	0	0	0	mA
Minimum pulse width		2			µs
ESD protection	Human body model	±5			kV

2.6.11. Encoder Inputs (A/A+, A-, B/B+, B-, Z/Z+, Z-)

		Min.	Typ.	Max.	Units
Single-ended mode compliance	Leave negative inputs disconnected	TTL / CMOS / Open-collector			
Input voltage, single-ended mode A/A+, B/B+	Logic "LOW"			1.6	V
	Logic "HIGH"	1.8			V
	Floating voltage (not connected)		4.5		V

		Min.	Typ.	Max.	Units
Input voltage, single-ended mode Z/Z+	Logic "LOW"			1.2	V
	Logic "HIGH"	1.4			
	Floating voltage (not connected)		4.7		
Input current, single-ended mode A/A+, B/B+, Z/Z+	Logic "LOW"; Pull to GND		2.5	3	mA
	Logic "HIGH"; Internal 2.2KΩ pull-up to +5	0	0	0	
Differential mode compliance	For full RS422 compliance, see [1]		TIA/EIA-422-A		
Input voltage, differential mode	Hysteresis	±0.06	±0.1	±0.2	V
	Common-mode range (A+ to GND, etc.)	-7		+7	
Input impedance, differential	A+ to A-, B+ to B-	4.2	4.7		kΩ
	Z+ to Z-	6.1	7.2		
Input frequency	Single-ended mode, Open-collector / NPN	0		500	kHz
	Differential mode, or Single-ended driven by push-pull (TTL / CMOS)	0		10	MHz
Minimum pulse width	Single-ended mode, Open-collector / NPN	1			µs
	Differential mode, or Single-ended driven by push-pull (TTL / CMOS)	50			nS
Input voltage, any pin to GND	Absolute maximum values, continuous	-7		+7	V
	Absolute maximum, surge (duration ≤ 1s) [†]	-11		+14	
ESD protection	Human body model	±1			kV

2.6.12. Linear Hall Inputs (LH1, LH2, LH3)

		Min.	Typ.	Max.	Units
Input voltage	Operational range	0	0.5÷4.5	4.9	V
	Absolute maximum values, continuous	-7		+7	
	Absolute maximum, surge (duration ≤ 1s) [†]	-11		+14	
Input current	Input voltage 0…+5V	-1	±0.9	+1	mA
Interpolation Resolution	Depending on software settings			11	bits
Frequency		0		1	kHz
ESD protection	Human body model	±1			kV

2.6.13. Sin-Cos Encoder Inputs (Sin+, Sin-, Cos+, Cos-)

		Min.	Typ.	Max.	Units
Input voltage, differential	Sin+ to Sin-, Cos+ to Cos-		1	1.25	V$_{PP}$
Input voltage, any pin to GND	Operational range	-1	2.5	4	V
	Absolute maximum values, continuous	-7		+7	
	Absolute maximum, surge (duration ≤ 1s) [†]	-11		+14	
Input impedance	Differential, Sin+ to Sin-, Cos+ to Cos- [2]	4.2	4.7		kΩ
	Common-mode, to GND		2.2		kΩ
Interpolation Resolution	Depending on software settings			11	bits
Frequency	Sin-Cos interpolation	0		450	kHz
	Quadrature, no interpolation	0		10	MHz
ESD protection	Human body model	±1			kV

---

[1] For full RS-422 compliance, 120Ω termination resistors must be connected across the differential pairs, as close as possible to the drive input pins. See *Figure 3.15. Differential incremental encoder connection*

[2] For many applications, an 120Ω termination resistor should be connected across SIN+ to SIN-, and across COS+ to COS-. Please consult the feedback device datasheet for confirmation.

2.6.14. Analog 0…5V Input (ANLG)

		Min.	Typ.	Max.	Units
Input voltage	Operational range	0		5	V
	Absolute maximum values, continuous	-12		+18	
	Absolute maximum, surge (duration ≤ 1s) [†]			±36	
Input impedance	To GND		26.5		kΩ
Resolution			12		bits
Integral linearity				±2	bits
Offset error			±2	±10	bits
Gain error			±1%	±3%	% FS[1]
Bandwidth (-3dB)	Depending on software settings	0		1	kHz
ESD protection	Human body model	±5			kV

2.6.15. RS-232

		Min.	Typ.	Max.	Units
Standards compliance			TIA/EIA-232-C		
Bit rate	Depending on software settings	9600		115200	Baud
Short-circuit protection	232TX short to GND		Guaranteed		
ESD protection	Human body model	±2			kV

2.6.16. CAN-Bus

		Min.	Typ.	Max.	Units
Standards compliance			ISO11898, CiA 301v4.2, CiA WD 305 v2.2.13, CiA DSP402v3.0		
Bit rate	Depending on software settings	125		1000	Kbps
Bus length	1Mbps			25	m
	500Kbps			100	
	≤ 250Kbps			250	
Number of CAN nodes/drives				125	-
Termination resistor	Between CAN-Hi, CAN-Lo		none on-board		
Node addressing	Hardware: by Hex switch (SW1)		1 ÷ 15 & LSS non-configured (CANopen); 1-15 & 255 (TMLCAN)		
	Software		1 ÷ 127 (CANopen); 1- 255 (TMLCAN)		
ESD protection	Human body model	±15			kV

2.6.17. Supply Output (+5V)

		Min.	Typ.	Max.	Units
+5V output voltage	Current sourced = 250mA	4.8	5	5.2	V
+5V output current		250	350		mA
Short-circuit protection			NOT protected		
Over-voltage protection			NOT protected		
ESD protection	Human body model	±1			kV

[†] Stresses beyond values listed under "absolute maximum ratings" may cause permanent damage to the device. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

---

[1] "FS" stands for "Full Scale"
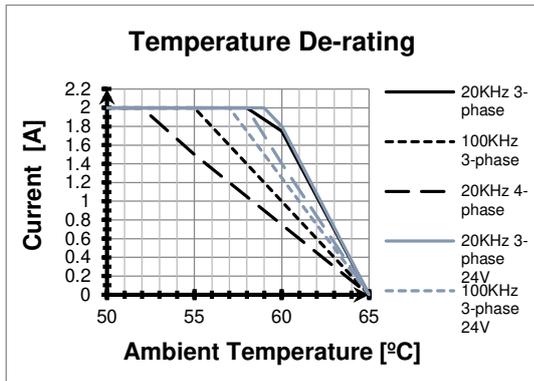
Figure 2.16 *iPOS3602 HX-CAN De-rating with ambient temperature*
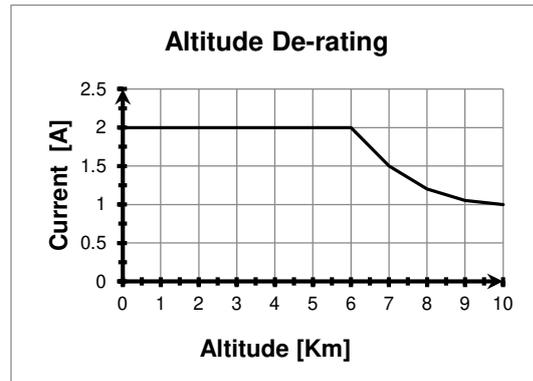
Figure 2.17 *iPOS3602 HX-CAN De-rating with altitude*
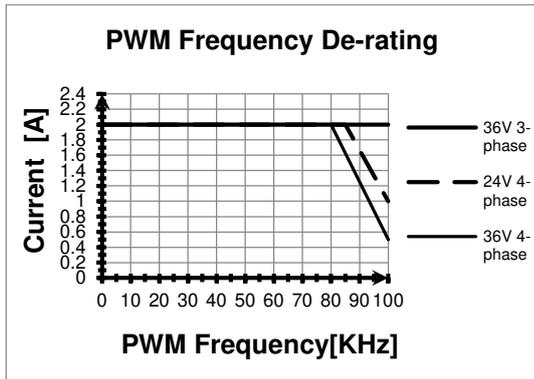
Figure 2.18 *iPOS3602 HX-CAN Current De-rating with PWM frequency*
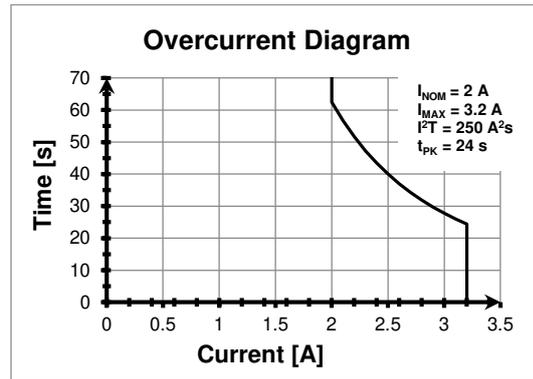
Figure 2.19 *iPOS3602 HX-CAN Over-current diagram*
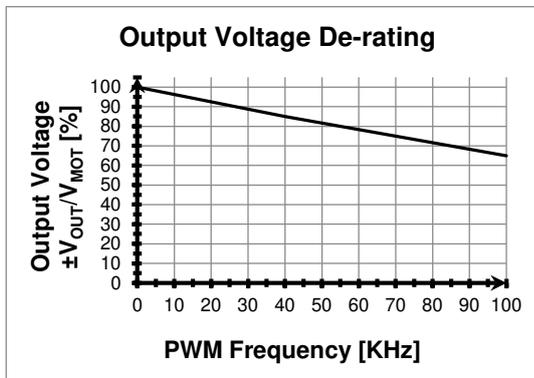
Figure 2.20. *iPOS360x HX-CAN Output Voltage De-rating with PWM frequency*[1]
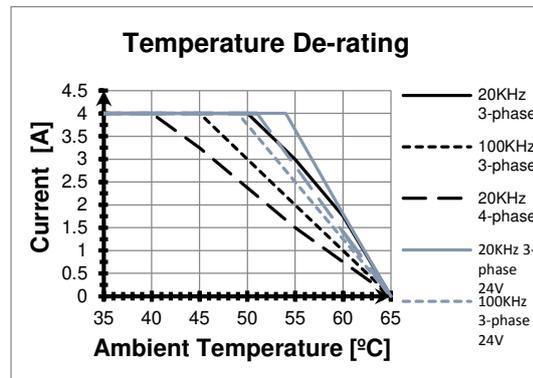
Figure 2.21. *iPOS3604 HX-CAN De-rating with ambient temperature*

---
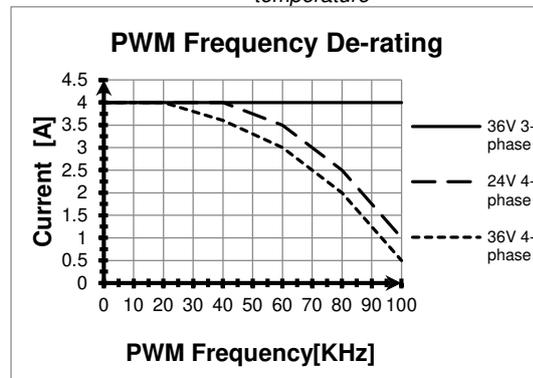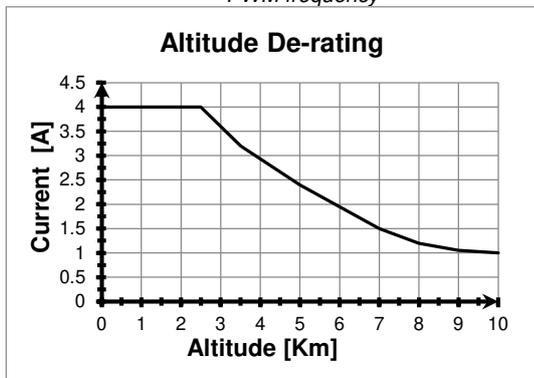
[1] V_{OUT} – the output voltage, V_{MOT} – the motor supply voltage

Figure 2.23. iPOS3604 HX-CAN Current De-rating with PWM frequency

Overcurrent Diagram

I_{NOM} = 4 A
I_{MAX} = 10 A
I^2T = 250 A^2s
t_{PK} = 2.5 s

Figure 2.24. iPOS3604 HX-CAN Over-current diagram

⚠	CAUTION!	*FOR PWM FREQUENCIES LESS THAN 20 KHZ, CORRELATE THE PWM FREQUENCY WITH THE MOTOR PARAMETERS IN ORDER TO AVOID POSSIBLE MOTOR DAMAGE.*

3. Step 1. Hardware Installation

3.1. Mechanical Mounting

The iPOS360x HX-CAN drive is intended to be mounted horizontally on a metallic support using the provided mounting holes and the recommended mating connectors, as specified in chapter 3.2.

For thermal calculations, each iPOS360x drive can be assumed to generate 1 Watt at idle, and up to 5 Watts (= 17 BTU/hour) worst case while driving a motor and using all digital outputs.

3.1.1. Horizontal Mounting

Figure 3.1 shows the recommended spacing to assure proper airflow by natural convection, *in the worst case* – closed box done from a plastic (non-metallic) material with no ventilation openings.

Whenever possible, ventilation openings shall be foreseen.

Remark: *In case of using a metallic box, with ventilation openings, all spacing values may be reduced substantially. With proper ventilation, keeping the air surrounding the iPOS360x HX-CAN inside the limits indicated in Figure 2.16 and Figure 2.21 the spacing values may be reduced down to the mechanical tolerance limits of Figure 3.1.*
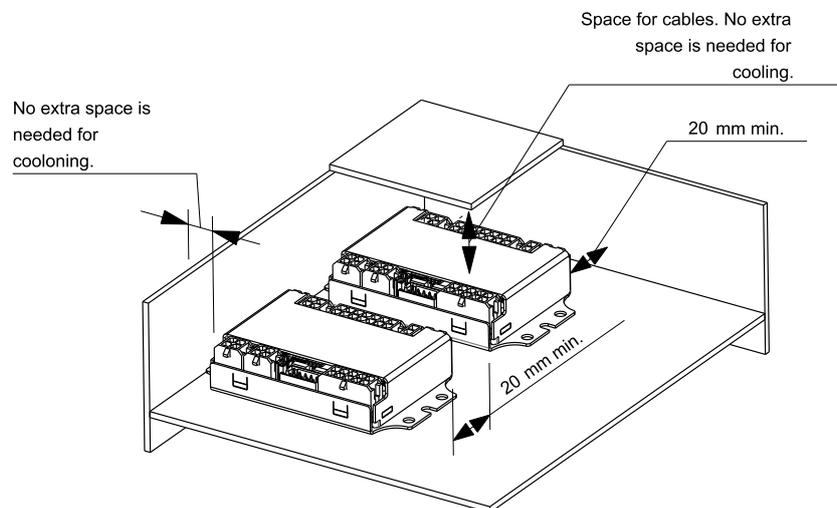
Figure 3.1 Recommended spacing for horizontal mounting, worst case: non-metallic, closed box

3.2. Mating Connectors

Connector	Description	Manufacturer	Part Number	Wire Gauge	Insulation Diameter
J1	MICROFIT RECEPTACLE HOUSING, 2x1 WAY	MOLEX	43025-0200	AWG 20..24	1.85 mm max.
J5,J6,J7	MICROFIT RECEPTACLE HOUSING, 2x2 WAY	MOLEX	43025-0400	AWG 20..24	1.85 mm max.
J2	MICROFIT RECEPTACLE HOUSING, 2x5 WAY	MOLEX	43025-1000	AWG 20..24	1.85 mm max.
J4	MICROFIT RECEPTACLE HOUSING, 2x7 WAY	MOLEX	43025-1400	AWG 20..24	1.85 mm max.
J1,J2,J4,J5, J6,J7	CRIMP PIN, MICROFIT, 5A	MOLEX	43030-0007	AWG 20..24	1.85 mm max.
J3	MILLIGRID RECEPTACLE HOUSING, 2x5 WAY	MOLEX	51110-1056	AWG 24..30	1.4 mm max.
J3	CRIMP PIN, MILLIGRID	MOLEX	50394-8400	AWG 24..30	1.4 mm max.

3.3. Connectors and Connection Diagrams

3.3.1. Connector Layout
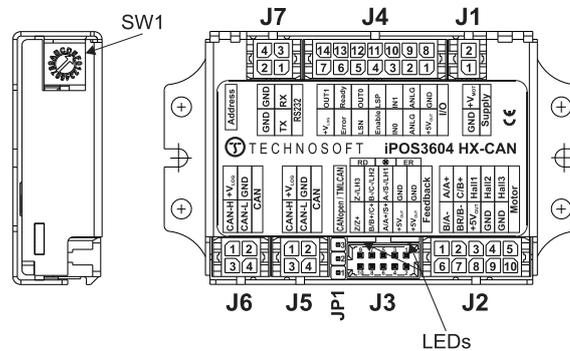
Figure 3.2. *iPOS360x HX-CAN drive connectors*

3.3.2. **J1 Power supply input connector pinout**

	Pin	Name	Type	Description
Connector description				
	Pin	**Name**	**Type**	**Description**
J1	**1**	**GND**	-	Negative return (ground) of the power supply
	2	**+V$_{MOT}$**	**I**	Positive terminal of the motor supply: 9 to 36V$_{DC}$. Feeds the positive terminal of the logic supply if J4 pin 7 not connected separately

3.3.3. **J2 Motor output and digital hall signals connector pinout**

	Pin	Name	Type	Description
Connector description				
	Pin	**Name**	**Type**	**Description**
J2	**1**	**A/A+**	**O**	**Phase A** for 3-ph motors, A+ for 2-ph steppers, Motor+ for DC brush motors
	2	**C/B+**	**O**	**Phase C** for 3-ph motors, B+ for 2-ph steppers
	3	**Hall 1**	**I**	Digital input Hall 1 sensor
	4	**Hall 2**	**I**	Digital input Hall 2 sensor
	5	**Hall 3**	**I**	Digital input Hall 3 sensor
	6	**B/A-**	**O**	**Phase B** for 3-ph motors, A- for 2-ph steppers, Motor- for DC brush motors
	7	**BR/B-**	**O**	**Brake resistor** / **Phase B-** for step motors
	8	**+5V$_{OUT}$**	**O**	5V output supply - internally generated
	9	**GND**	**-**	Negative return (ground) of the motor supply
	10	**GND**	**-**	Negative return (ground) of the motor supply

3.3.4. **J3 Feedback connector pinout**

	Pin	Name	Type	Description
Connector description				
	Pin	**Name**	**Type**	**Description**
J3	**1**	**GND**	**-**	Return ground for sensors supply
	2	**+5V$_{OUT}$**	**O**	5V output supply for I/O usage
	3	**GND**	**-**	Return ground for sensors supply
	4	**+5V$_{OUT}$**	**O**	5V output supply for I/O usage
	5	**A- /Sin-/LH1**	**I**	Incr. encoder A- diff. input, or analogue encoder Sin- diff. input, or linear Hall 1 input
	6	**A/A+/Sin+**	**I**	Incr. encoder A single-ended, or A+ diff. input, or analogue encoder Sin+ diff. input
	7	**B-/Cos-/LH2**	**I**	Incr. encoder B- diff. input, or analogue encoder Cos- diff. input, or linear Hall 2 input
	8	**B/B+/Cos+**	**I**	Incr. encoder B single-ended, or B+ diff. input, or analogue encoder Cos+ diff. input
	9	**Z- /LH3**	**I**	Incr. encoder Z- diff. input, or linear Hall 3 input
	10	**Z/ Z+**	**I**	Incr. encoder Z (index) single-ended, or Z+ diff. input

3.3.5. J4 Digital, analog I/O and logic supply connector pinout

	Pin	Name	Type	Description
	Connector description			
	Pin	**Name**	**Type**	**Description**
J4	**1**	**+5V$_{OUT}$**	**O**	5V output supply for I/O usage
	2	ANLG	I	Analogue input, 12-bit, 0-5V. Used to read an analog position, speed or torque reference, or an analogue position or speed feedback or as general purpose analogue input
	3	**IN0**	**I**	5-36V general-purpose digital PNP/NPN input
	4	IN4/Enable	I	5-36V digital PNP/NPN input. Drive enable input
	5	**IN3/LSN**	**I**	5-36V digital PNP/NPN input. Negative limit switch input
	6	OUT2/Error	O	5-36V 0.5A, drive Error output, active low, NPN open-collector/TTL pull-up. Also drives the red LED
	7	**+V$_{LOG}$**	**I**	Positive terminal of the logic supply: 9 to 36V$_{DC}$ / If not connected, the logic supply is automatically routed from J1 pin 2[1]
	8	GND	-	Return ground for I/O pins
	9	ANLG	I	Analogue input, 12-bit, 0-5V. Used to read an analog position, speed or torque reference, or an analogue position or speed feedback or as general purpose analogue input
	10	IN1	I	5-36V general-purpose digital PNP/NPN input
	11	**IN2/LSP**	**I**	5-36V digital PNP/NPN input. Positive limit switch input
	12	OUT0	O	5-36V 0.5A, general-purpose digital output, NPN open-collector/TTL pull-up
	13	**OUT3/Ready**	**O**	5-36V 0.5A, drive Ready output, active low, NPN open-collector/TTL pull-up. Also drives the green LED.
	14	OUT1	O	5-36V 0.5A, general-purpose digital output, NPN open-collector/TTL pull-up

3.3.6. J5, J6 CAN connectors pinout

	Pin	Name	Type	Description
	Connector description			
	Pin	**Name**	**Type**	**Description**
J5, J6	**1**	**+V$_{LOG}$**	**O**	Positive terminal of the logic supply: 9 to 36V$_{DC}$
	2	GND	-	Return ground for CAN-Bus
	3	**Can-Hi**	**I/O**	CAN-Bus positive line (dominant high)
	4	Can-Lo	I/O	CAN-Bus negative line (dominant low)

3.3.7. J7 RS232 connector pinout

	Pin	Name	Type	Description
	Connector description			
	Pin	**Name**	**Type**	**Description**
J7	**1**	**232TX**	O	RS-232 Data Transmission
	2	GND	-	Return ground for RS-232 pins
	3	**232RX**	I	RS-232 Data Reception
	4	GND	-	Return ground for RS-232 pins

---

[1] In case +V$_{LOG}$ (J4 pin7) is not connected, the digital outputs and inputs will not be operational.

3.3.8. 24V Digital I/O Connection

3.3.8.1 PNP inputs
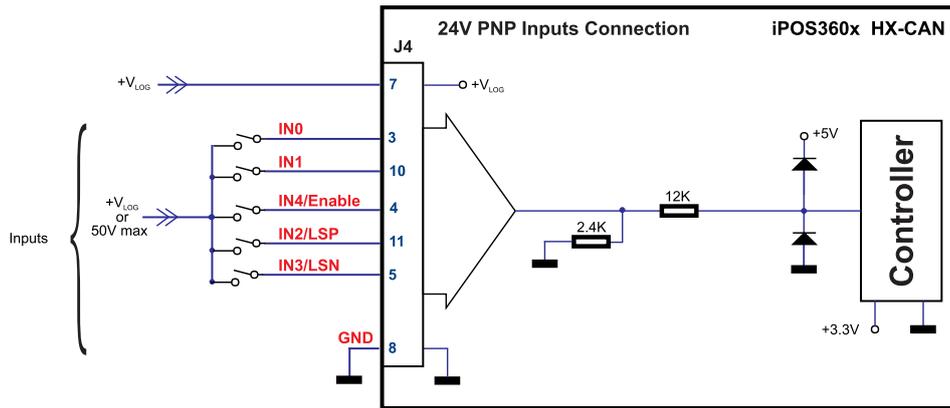
Figure 3.3. *24V Digital PNP Inputs connection*

Remarks:
1. *If V_{log} is not supplied, the digital inputs will not work.*
2. *The inputs are selectable as PNP/ NPN by software.*
3. *The inputs are compatible with PNP type outputs (input must receive a positive voltage value (5-36V) to change its default state)*

3.3.8.2 NPN inputs
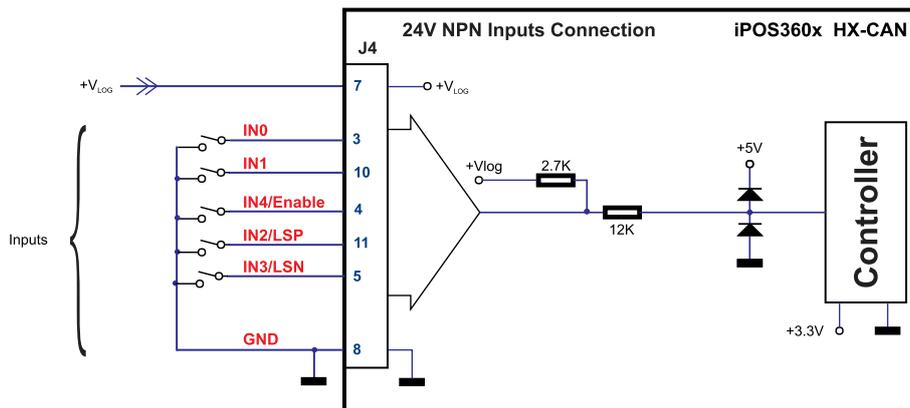
Figure 3.4. *24V Digital NPN Inputs connection*

Remarks:
1. *If V_{log} is not supplied, the digital inputs will not work.*
2. *The inputs are selectable as PNP/ NPN by software.*
3. *The inputs are compatible with NPN type outputs (input must be pulled to GND to change its default state)*

3.3.8.3 NPN outputs
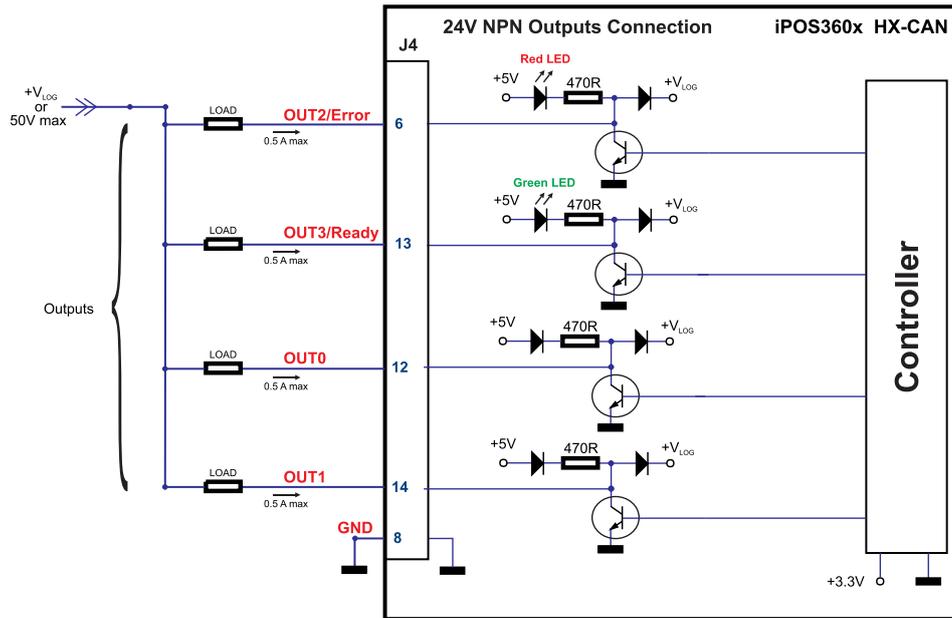
Figure 3.5. 24V Digital NPN Inputs connection

Remarks:
1. The outputs are compatible with NPN type inputs (load is tied to common $+V_{LOG}$, output pulls to GND when active and is floating when inactive)
2. The outputs are NOT compatible with PNP inputs! Even if it seems to work, proper operation is NOT guaranteed!

3.3.9. 5V Digital NPN I/O Connection
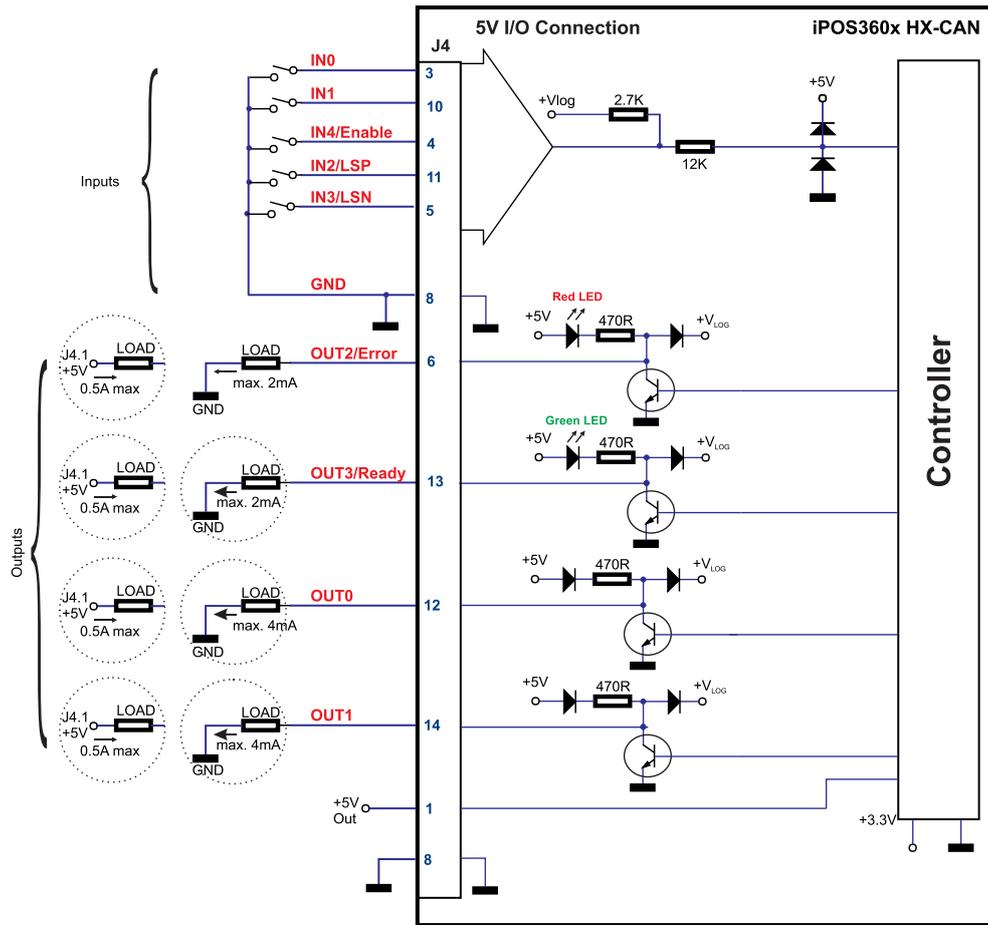
Figure 3.6. *5V Digital I/O connection*

Remarks:
1. *The inputs are selectable as PNP/ NPN by software. For this connection they are selected as NPN.*
2. *The inputs are compatible with TTL(5V), LVTTL(3.3V), CMOS and open collector outputs*
3. *The outputs are compatible with TTL (5V) inputs*
4. *The output loads can be individually and independently connected to +5V or to GND.*

3.3.10. **Analog Input Connection**

3.3.10.1 0-5V Input Range
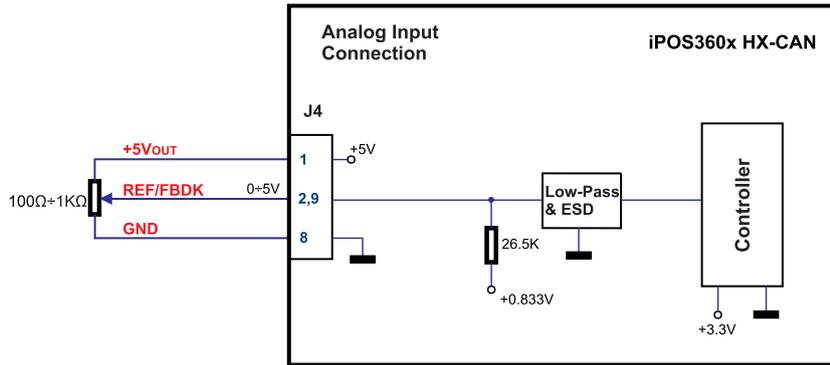
Figure 3.7. *Analog input connection*

Remark: *Default input range for analog input is 0÷5 V for ANLG. For a +/-10 V range, see Figure 3.8.*

3.3.10.2 +/- 10V to 0-5V Input Range Adapter
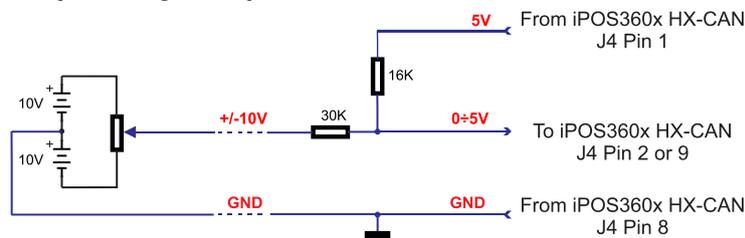
Figure 3.8. *+/-10V to 0-5V adapter*

Remark: *When this +/-10V range adapter is in place, in order to adjust for component non-ideal values, one must modify the following parameters within SETUP.CFG file: RFOFFSET=1616=0x0650 (instead of default 624=0x0270), and RFGAIN=34448=0x8690 (instead of default 34387=0x8653). With this range adapter in place, the input impedance will be 40Kohm, and floating input voltage will be +3.43V.*

3.3.10.3 Recommendation for wiring

a) If the analogue signal source is single-ended, use a 2-wire twisted shielded cable as follows: 1st wire connects the live signal to the drive input; 2nd wire connects the source ground to the drive ground; shield will be connected to the drive ground terminal.
b) If the analogue signal source is differential and the signal source ground is isolated from the drive GND, use a 2-wire twisted shielded cable as follows: 1st wire connects the source plus (positive, in-phase) to the drive analogue input; 2nd wire connects the source minus (negative, out-of-phase) to the drive ground (GND). Shield is connected only at the drive side, to the drive GND, and is left unconnected at the source side.
c) If the analogue signal source is differential and the signal source ground is common with the drive GND, use a 2-wire shielded cable as follows: 1st wire connects the source plus (positive, in-phase) to the drive analogue input; 2nd wire connects the source ground to the drive ground (GND); shield is connected only at the drive side, to the drive GND, and is left unconnected at the source side. The source minus (negative, out-of-phase) output remains unconnected.

3.3.11. Motor connections

3.3.11.1 Brushless Motor connection
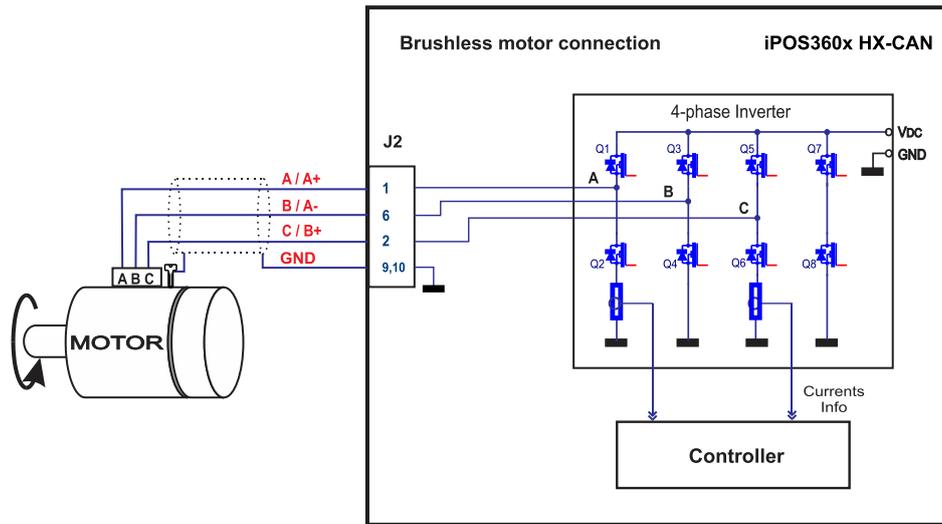
Figure 3.9. *Brushless motor connection*

3.3.11.2 2-phase Step Motor connection
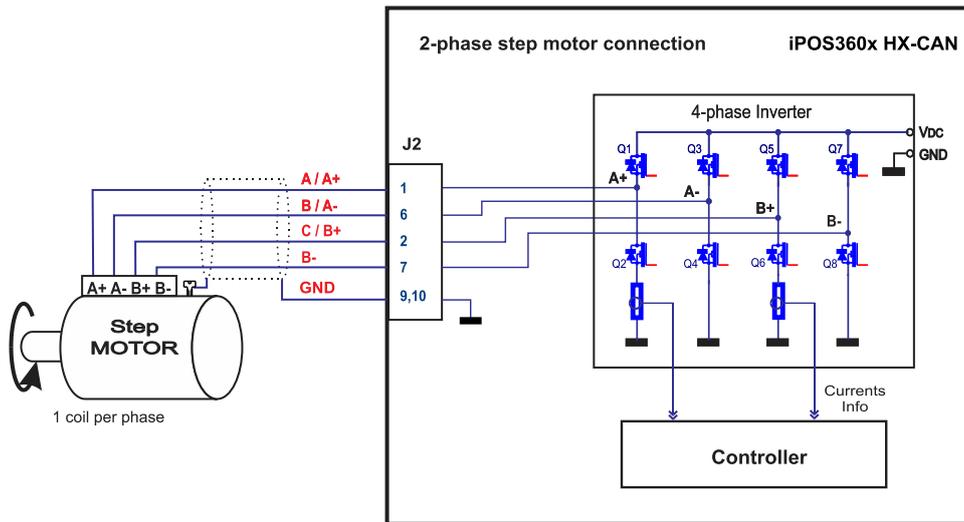
Figure 3.10. *2-phase step motor connection, one coil per phase*
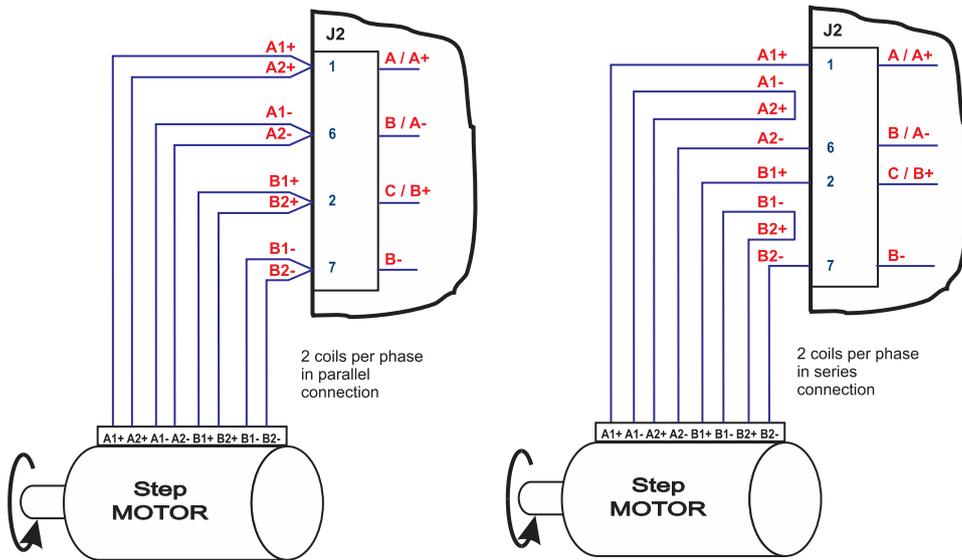
Figure 3.11. *2-phase step motor connection, two coils per phase*

3.3.11.3 3-Phase Step Motor connection
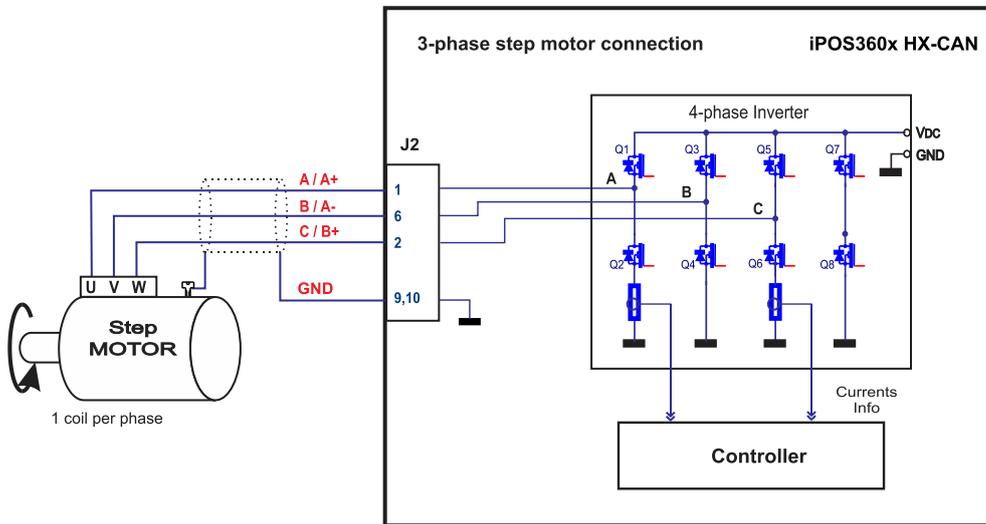
Figure 3.12. *3-phase step motor connection*

3.3.11.4 DC Motor connection
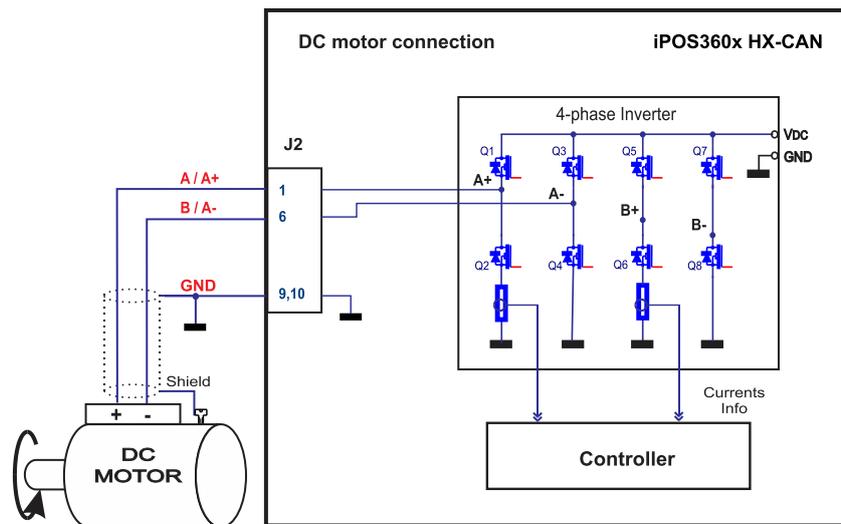
Figure 3.13. *DC Motor connection*

3.3.11.5 Recommendations for motor wiring

a) Avoid running the motor wires in parallel with other wires for a distance longer than 2 meters. If this situation cannot be avoided, use a shielded cable for the motor wires. Connect the cable shield to the iPOS360x GND pin. Leave the other end disconnected.

b) The parasitic capacitance between the motor wires must not bypass 10nF. If very long cables (tens of meters) are used, this condition may not be met. In this case, add series inductors between the iPOS360x outputs and the cable. The inductors must be magnetically shielded (toroidal, for example), and must be rated for the motor surge current. Typically the necessary values are around 100 µH.

c) A good shielding can be obtained if the motor wires are running inside a metallic cable guide.

3.3.12. **Feedback connections**

3.3.12.1 Single-ended Incremental Encoder Connection
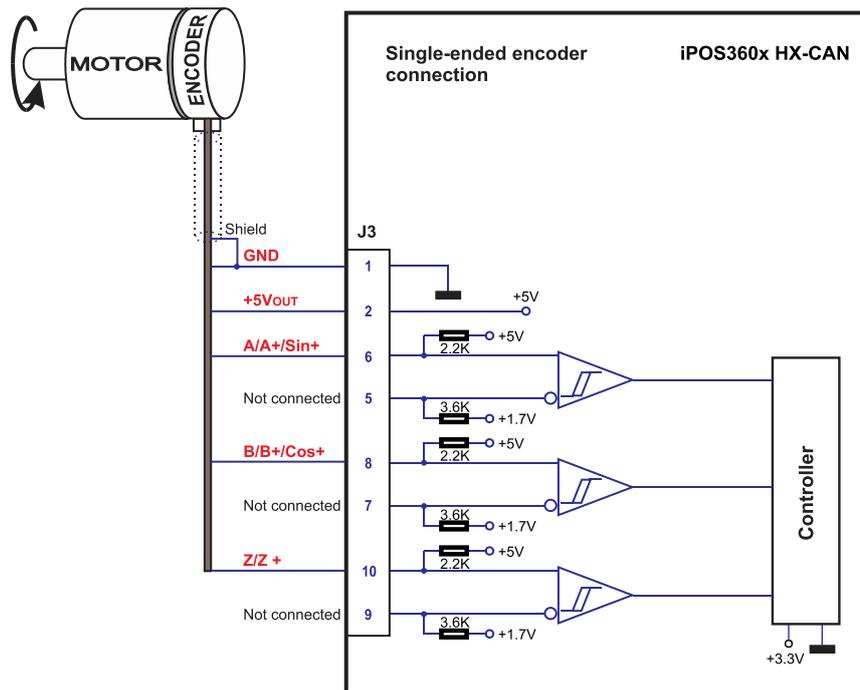
Figure 3.14. *Single-ended incremental encoder connection*

3.3.12.2 Differential Incremental Encoder Connection
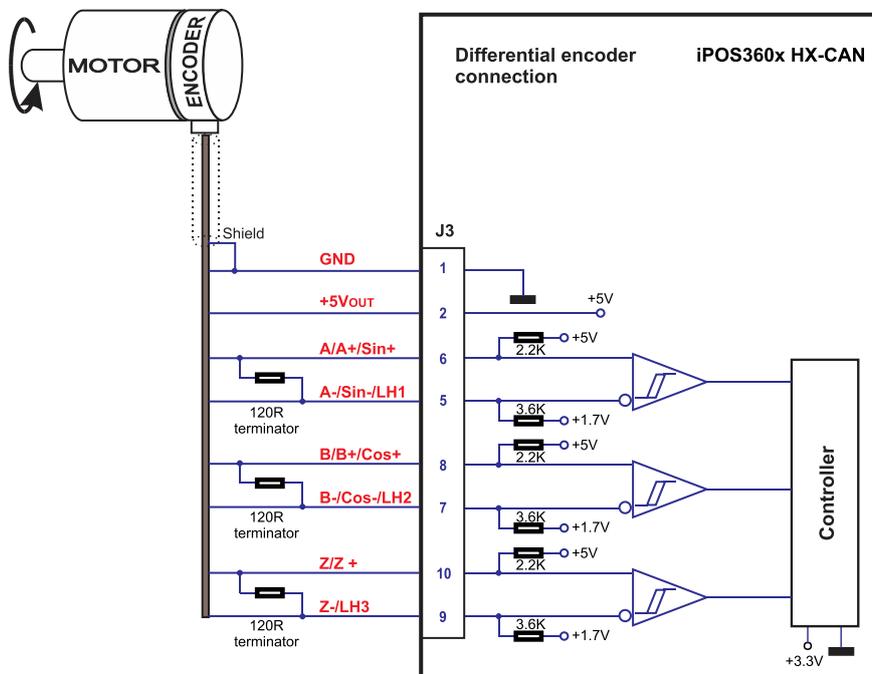
Figure 3.15. *Differential incremental encoder connection*

Remark: *120Ω (0.25W) terminators are required for long encoder cables, or noisy environments.*

3.3.12.3 Digital Hall Connection
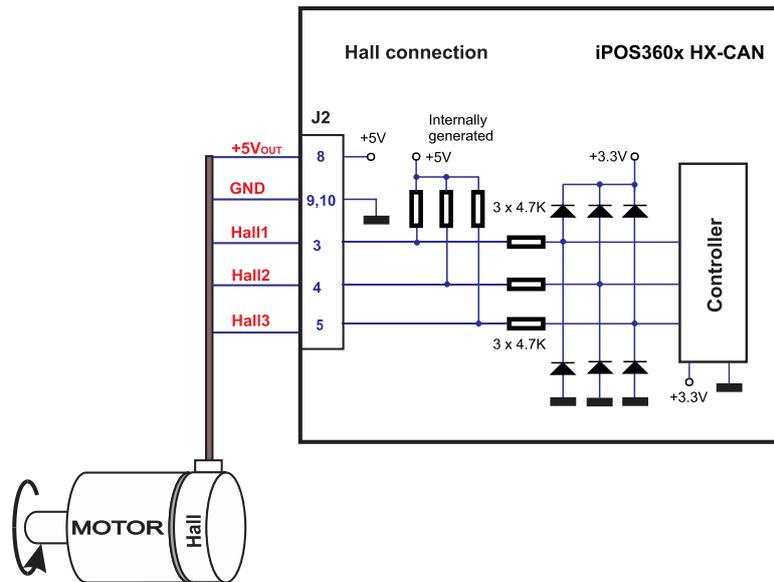
Figure 3.16. *Digital Hall connection*

3.3.12.4 Linear Hall Connection
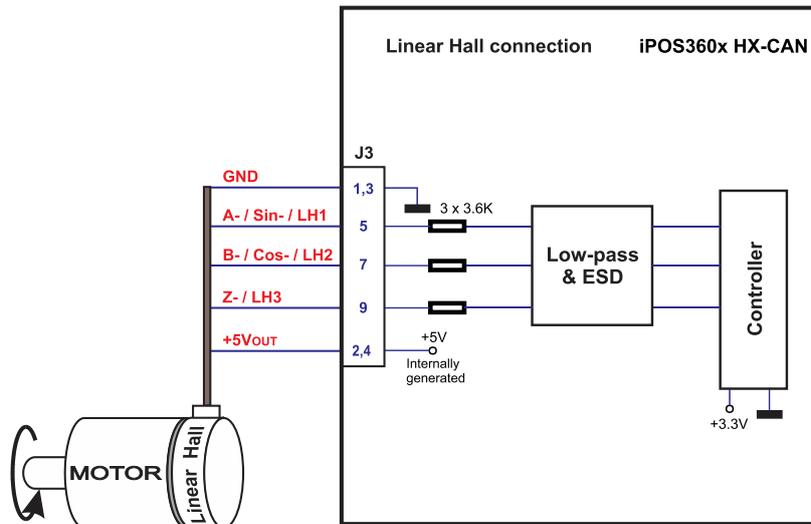
Figure 3.17. *Linear Hall connection*

3.3.12.5 Sine-Cosine Analog Encoder Connection
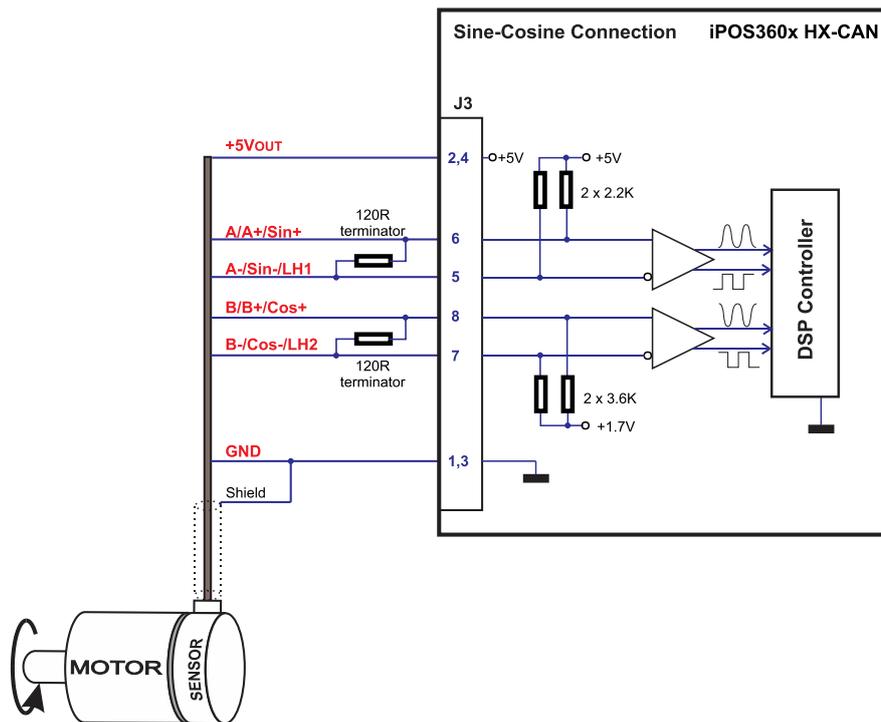
Figure 3.18. Sine-Cosine analogue encoder connection

3.3.12.6 Recommendations for wiring

a) Always connect both positive and negative signals when the position sensor is differential and provides them. Use one twisted pair for each differential group of signals as follows: A+/Sin+ with A-/Sin-/LH1, B+/Cos+ with B-/Cos-/LH2, Z+ with Z-/LH3. Use another twisted pair for the 5V supply and GND.

b) Always use shielded cables to avoid capacitive-coupled noise when using single-ended encoders or Hall sensors with cable lengths over 1 meter. Connect the cable shield to the GND, at only one end. This point could be either the iPOS360x (using the GND pin) or the encoder / motor. Do not connect the shield at both ends.

c) If the iPOS360x 5V supply output is used by another device (like for example an encoder) and the connection cable is longer than 5 meters, add a decoupling capacitor near the supplied device, between the +5V and GND lines. The capacitor value can be 1...10 µF, rated at 6.3V.

3.3.13. Power Supply Connection

3.3.13.1 Supply Connection
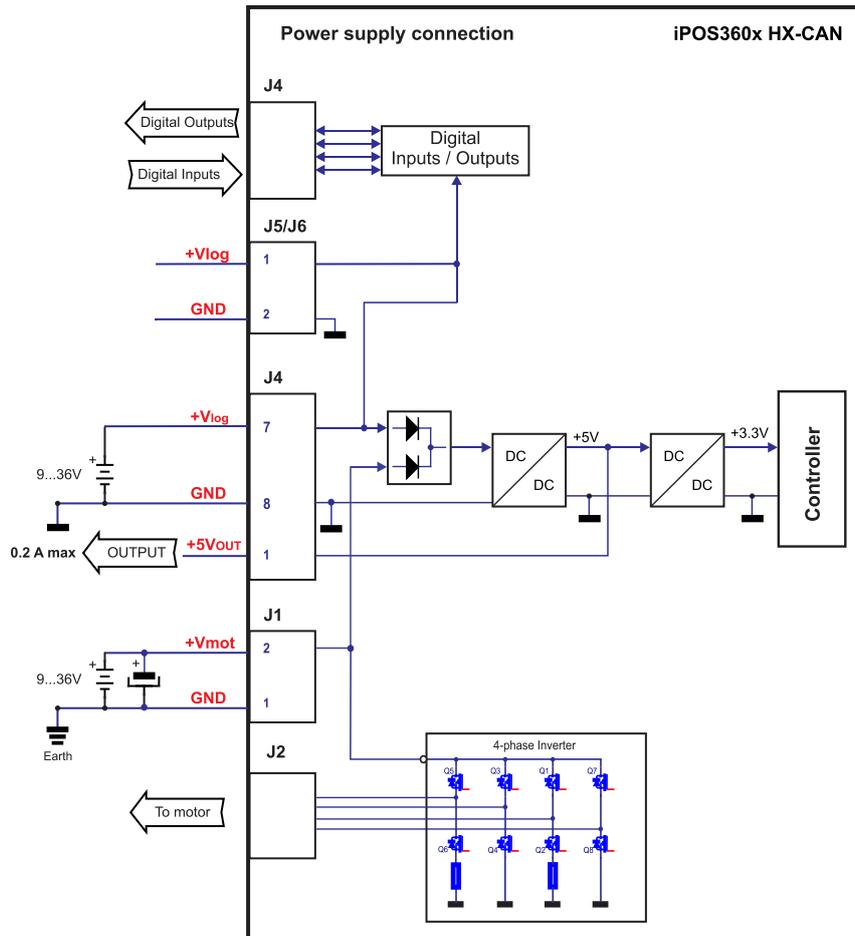
Figure 3.19. *Supply connection*

3.3.13.2 Recommendations for Supply Wiring

The logic power supply can be connected to either J4 pin 7, J5 pin 1 or J6 pin 1. Al three input pins are connected together inside the iPOS4808 HX-CAN.

Use short, thick wires between the iPOS360x and the motor power supply. Connect power supply wires to all the indicated pins. If the wires are longer than 2 meters, use twisted wires for the supply and ground return. For wires longer than 20 meters, add a capacitor of at least 2,200μF (rated at an appropriate voltage) right on the terminals of the iPOS3604, respectively 1,000μF for the iPOS3602.

It is recommended to connect the negative motor supply return (GND) to the Earth protection near the power supply terminals.

3.3.13.3 Recommendations to limit over-voltage during braking

During abrupt motion brakes or reversals the regenerative energy is injected into the motor power supply. This may cause an increase of the motor supply voltage (depending on the power supply characteristics). If the voltage bypasses 39V, the drive over-voltage protection is triggered and the drive power stage is disabled. In order to avoid this situation you have 2 options:

Option 1. Add a capacitor on the motor supply big enough to absorb the overall energy flowing back to the supply. The capacitor must be rated to a voltage equal or bigger than the maximum expected over-voltage and can be sized with the formula:

$$C \geq \frac{2 \times E_M}{U_{MAX}^2 - U_{NOM}^2}$$

where:

U_{MAX} = 39V is the over-voltage protection limit

U_{NOM} is the nominal motor supply voltage

E$_M$ = the overall energy flowing back to the supply in Joules. In case of a rotary motor and load, E$_M$ can be computed with the formula:

$$E_M = \frac{1}{2}\underbrace{(J_M + J_L)\varpi_M^2}_{\text{Kinetic energy}} + \underbrace{(m_M + m_L)g(h_{initial} - h_{final})}_{\text{Potential energy}} - \underbrace{3I_M^2 R_{Ph}t_d}_{\text{Copper losses}} - \underbrace{\frac{t_d\varpi_M}{2}T_F}_{\text{Friction losses}}$$

where:

J_M – total rotor inertia [kgm^2]

J_L – total load inertia as seen at motor shaft after transmission [kgm^2]

ϖ_M – motor angular speed before deceleration [rad/s]

m_M – motor mass [kg] – when motor is moving in a non-horizontal plane

m_L – load mass [kg] – when load is moving in a non-horizontal plane

g – gravitational acceleration i.e. 9.8 [m/s^2]

$h_{initial}$ – initial system altitude [m]

h_{final} – final system altitude [m]

I_M – motor current during deceleration [A$_{RMS}$/phase]

R_{Ph} – motor phase resistance [Ω]

t_d – time to decelerate [s]

T_F – total friction torque as seen at motor shaft [Nm] – includes load and transmission

In case of a linear motor and load, the motor inertia J_M and the load inertia J_L will be replaced by the motor mass and the load mass measured in [kg], the angular speed ϖ_M will become linear speed measured in [m/s] and the friction torque T_F will become friction force measured in [N].

Option 2. Connect a brake resistor R$_{BR}$ between phase BR/B- (pins A25-A28) and ground (pins A23+A24+B23+B24), and activate the software option of dynamic braking (see below).

This option is not available when the drive is used with a step motor.

The dynamic braking option can be found in the Drive Setup dialogue within EasyMotion / EasySetup. The braking will occur when DC bus voltage increases over U$_{BRAKE}$. This parameter (U$_{BRAKE}$) should be adjusted depending on the nominal motor supply. Optimally (from a braking point of view), U$_{BRAKE}$ should be a few volts above the maximum nominal supply voltage. This setting will activate the brake resistor earlier, before reaching dangerous voltages – when the over-voltage protection will stop the drive. Of course, U$_{BRAKE}$ must always be less than U$_{MAX}$ – the over-voltage protection threshold.

Remark: *This option can be combined with an external capacitor whose value is not enough to absorb the entire regenerative energy E$_M$ but can help reducing the brake resistor size.*

Brake resistor selection

The brake resistor value must be chosen to respect the following conditions:

1. to limit the maximum current below the drive peak current I$_{PEAK}$ = 10A

$$R_{BR} > \frac{U_{MAX}}{I_{PEAK}}$$

2. to sustain the required *braking power*:

$$P_{BR} = \frac{E_M - \frac{1}{2}C(U_{MAX}^2 - U_{brake}^2)}{t_d}$$

where C is the capacitance on the motor supply (external), i.e:

$$R_{BR} < \frac{U_{BRAKE}^2}{2 \times P_{BR}}$$

3. to limit the average current below the drive nominal current I$_{NOM}$=4A

$$R_{BR} > \frac{P_{BR} \times t_d}{t_{CYCLE} \times I_{NOM}^2}$$

where t$_{CYCLE}$ is the time interval between 2 brakes in case of repetitive moves.

4. to be rated for an average power $P_{AV} = \dfrac{P_{BR} \times t_d}{t_{CYCLE}}$ and a peak power $P_{PEAK} = \dfrac{U_{MAX}^2}{R_{BR}}$

Remarks:

1. *If* $\dfrac{U_{MAX}}{I_{PEAK}} > \dfrac{U_{BRAKE}^2}{2 \times P_{BR}}$ *the braking power P_{BR} must be reduced by increasing either t_d – the time to decelerate or C – the external capacitor on the motor supply*

2. *If* $\dfrac{P_{BR} \times t_d}{t_{CYCLE} \times I_{NOM}^2} > \dfrac{U_{BRAKE}^2}{2 \times P_{BR}}$ *either the braking power must be reduced (see Remark 1) or t_{CYCLE} – the time interval between braking cycles must be increased*

⚠	**WARNING!**	***THE BRAKE RESISTOR MAY HAVE HOT SURFACES DURING OPERATION.***

3.3.14. Serial RS-232 connection

3.3.14.1 Serial RS-232 connection
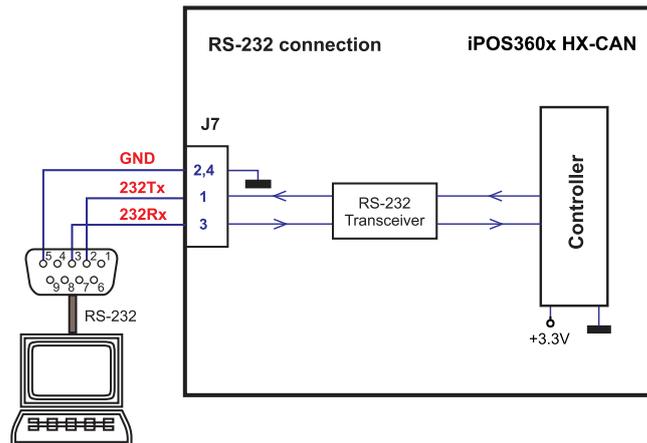
Figure 3.20. *Serial RS-232 connection*

3.3.14.2 Recommendation for wiring

a) If you build the serial cable, you can use a 3-wire shielded cable with shield connected to BOTH ends. Do not use the shield as GND. The ground wire (pin 2 or 4 of J7) must be included inside the shield, like the 232Rx and 232Tx signals
b) Always power-off all the iPOS360x HX-CAN supplies before inserting/removing the RS-232 serial connector
c) Do not rely on an earthed PC to provide the iPOS360x GND connection! The drive must be earthed through a separate circuit. Most communication problems are caused by the lack of such connection

3.3.15. CAN-bus connection

3.3.15.1 CAN connection
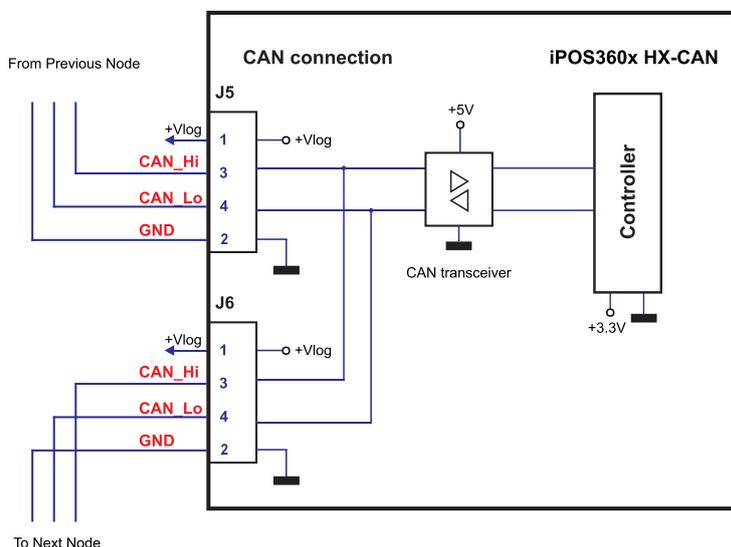
Figure 3.21. *CAN connection*

Remarks:
1. *The CAN network requires a 120-Ohm terminator. This is not included in the drive. shows how to connect it on your network*
2. *CAN signals are not insulated from other iPOS360x circuits.*

3.3.15.2 Recommendation for wiring

a) Build CAN network using cables with twisted wires (2 wires/pair), with CAN-Hi twisted together with CAN-Lo. It is recommended but not mandatory to use a shielded cable. If so, connect the shield to GND. The cable impedance must be 105 ... 135 ohms (120 ohms typical) and a capacitance below 30pF/meter.

b) The 120Ω termination resistors must be rated at 0.2W minimum. Do not use winded resistors, which are inductive.
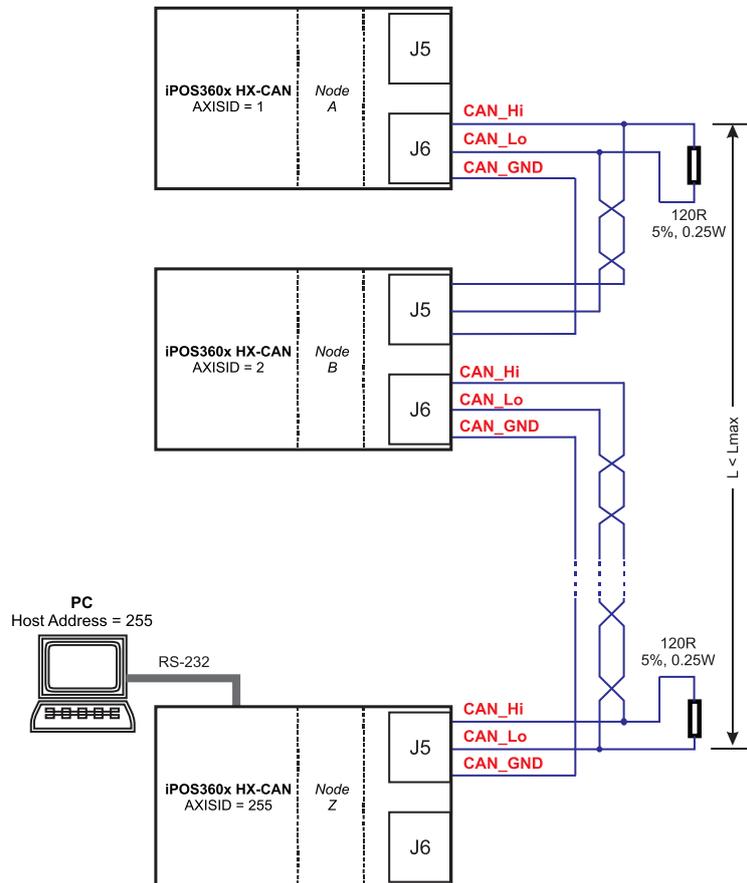
Figure 3.22. Multiple-Axis CAN network

Remarks:

1. *The axis IDs in* **Figure 3.22**, *are valid for TMLCAN mode. For CANopen mode, the highest axis ID a drive can have is 127.*

2. *Lmax is the bus length defined in paragraph* **2.6.16**.

3.3.16. Disabling Autorun Mode

When an iPOS360x HX-CAN is set in TMLCAN operation mode, by default after power-on it enters automatically in *Autorun* mode. In this mode, if the drive has in its local EEPROM a valid TML application (motion program), this is automatically executed as soon as the motor supply V$_{MOT}$ is turned on.

In order to disable *Autorun* mode, there are 2 methods:

a) Software - by writing value 0x0001 in first EEPROM location at address 0x4000

b) Hardware – by temporary connecting all digital Hall inputs to GND, during the power-on for about 1 second, until the green LED is turned on, as shown in *Figure 3.23*. This option is particularly useful when it is not possible to communicate with the drive.

After the drive is set in *non-Autorun/slave* mode using 2nd method, the 1st method may be used to invalidate the TML application from the EEPROM. On next power on, in absence of a valid TML application, the drive enters in the *non-Autorun/slave* mode independently of the digital Hall inputs status.
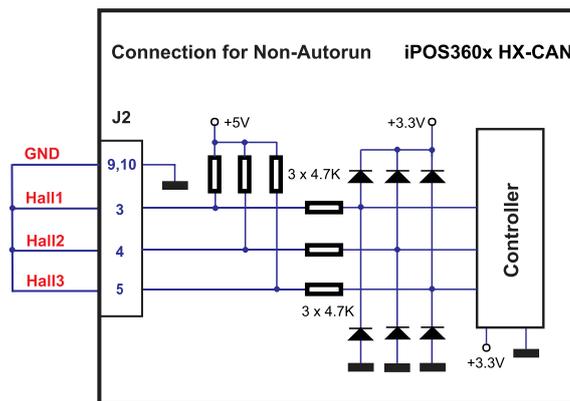
Figure 3.23. *Temporary connection during power-on to disable **Autorun** mode*

3.4. Operation Mode and Axis ID Selection

3.4.1. Selection of the Operation Mode

On iPOS360x HX-CAN, the selection of the operation mode CANopen or TMLCAN is done by setting the JP1 jumper:
- CANopen mode, JP1 = 1-2
- TMLCAN mode, JP1 = 2-3

3.4.2. Selection of the Axis ID

The Hardware Axis ID selection is done through the hex switch SW1. It contains numbers from 0x0 to 0xF. Depending on SW1 position, the axis ID will be:

SW1 position	AxisID in TMLCAN mode	AxisID in CANopen mode
0x0	255	LSS non-configured state
0x1	1	1
0x2	2	2
0x3	3	3
0x4	4	4
0x5	5	5
0x6	6	6
0x7	7	7
0x8	8	8
0x9	9	9
0xA	10	10
0xB	11	11
0xC	12	12
0xD	13	13
0xE	14	14
0xF	15	15

Note: LSS "non-configured" state, is a state in which the drive does not have assigned an active Axis ID while connected to the CAN network. In this mode the Axis ID for RS232 communication is 255. The Axis ID can be configured via a LSS master using CiA-305 protocol, which can set and save a new unique value. While the drive has a non-configured Axis ID, it cannot communicate with other drives in the network. The green (Ready) led will flash at 1 second time intervals when in this mode.

4. Step 2. Drive Setup

4.1. Installing EasySetUp

EasySetUp is a PC software platform for the setup of the Technosoft drives. It can be downloaded **free of charge** from Technosoft web page. EasySetUp comes with an **Update via Internet tool** through which you can check if your software version is up-to-date, and when necessary download and install the latest updates.

EasySetUp can be installed independently or together with **EasyMotion Studio** platform for motion programming using TML. You will need EasyMotion Studio only if you plan to use the advanced features presented in Section 5.3. A **demo version of EasyMotion Studio** including the **fully functional version of EasySetUp** can be downloaded free of charge from Technosoft web page.

On request, EasySetUp can be provided on a CD too. In this case, after installation, use the update via internet tool to check for the latest updates. Once you have started the installation package, follow its indications.

4.2. Getting Started with EasySetUp

Using EasySetUp you can quickly setup a drive for your application. The drive can be connected with your PC in one of the following ways:

1. Via an RS232 link, directly connected to the PC, or via an USB to RS232 adapter or using Technosoft Ethernet to RS232 adapter, function of your PC communication interfaces;
2. Via a CAN-bus link, directly connected to the PC through a PC-CAN interface, or using Technosoft Ethernet to CAN adapter
3. Via another drive from the same CAN-bus network, which is connected to the PC via one of the above options from point 1.

The output of EasySetUp is a set of *setup data*, which can be downloaded into the drive EEPROM or saved on your PC for later use.

EasySetUp includes a set of evaluation tools like the Data Logger, the Control Panel and the Command Interpreter which help you to quickly measure, check and analyze your drive commissioning.

EasySetUp works with **setup** data. A **setup** contains all the information needed to configure and parameterize a Technosoft drive. This information is preserved in the drive EEPROM in the *setup table*. The setup table is copied at power-on into the RAM memory of the drive and is used during runtime. With EasySetUp it is also possible to retrieve the complete setup information from a drive previously programmed.

Note that with EasySetUp you do only your drive/motor commissioning. For motion programming you have the following options:

- Use a **CANopen** master to control the iPOS360x as a standard CANopen drive
- Use **EasyMotion Studio** to create and download a **TML** program into the drive/motor memory
- Use one of the **TML_LIB** motion libraries to control the drives/motors from your host/master. If your host is a **PC**, TML_LIB offers a collection of high level motion functions which can be called from applications written in C/C++, Visual Basic, Delphi Pascal or LabVIEW. If your host is a **PLC**, TML_LIB offers a collection of function blocks for motion programming, which are **IEC61131-3 compatible** and can be integrated in your PLC program.
- **Implement** on your master the TML commands you need to send to the drives/motors using one of the supported communication channels. The implementation must be done according with Technosoft communication protocols.
- **Combine** TML programming at drive level with one of the other options (see Section 5.3)

4.2.1. Establish communication

EasySetUp starts with an empty window from where you can create a **New** setup, **Open** a previously created setup which was saved on your PC, or **Upload** the setup from the drive/motor.
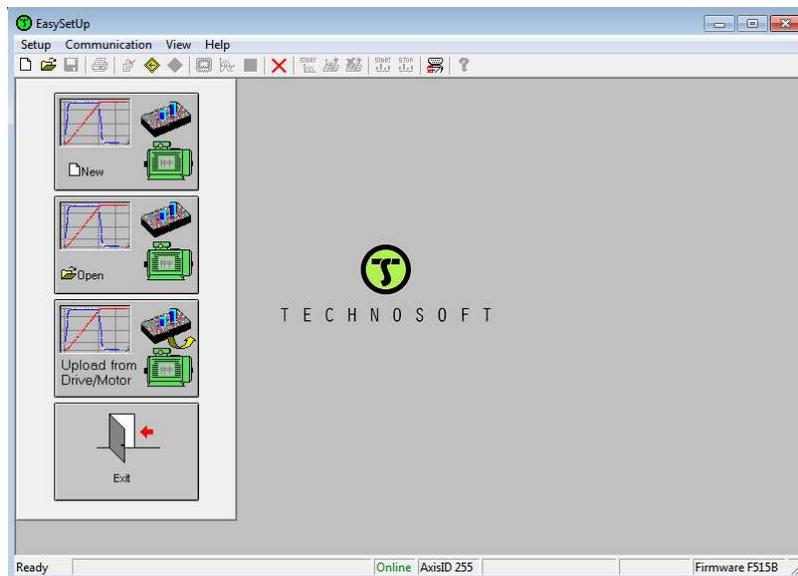
Before selecting one of the above options, you need to establish the communication with the drive you want to commission. Use menu command **Communication | Setup** to check/change your PC communication settings. Press the **Help** button of the dialogue opened. Here you can find detailed information about how to setup your drive and do the connections. Power on the drive, then close the Communication | Setup dialogue with OK. If the communication is established, EasySetUp displays in the status bar (the bottom line) the text "**Online**" plus the axis ID of your drive/motor and its firmware version. Otherwise the text displayed is "**Offline**" and a communication error message tells you the error type. In this case, return to the Communication | Setup dialogue, press the Help button and check troubleshoots

*Remark: When first started, EasySetUp tries to communicate via RS-232 and COM1 with a drive having axis ID=255 (default communication settings). If the drive has a different axis ID and you don't know it, select in the Communication | Setup dialogue at "Axis ID of drive/motor connected to PC" the option **Autodetected**. If this drive is part of a CANbus network, use the menu command **Communication | Scan Network***

4.2.2. Setup drive/motor
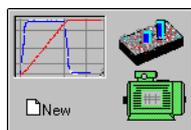
Press **New** button and select your drive type.
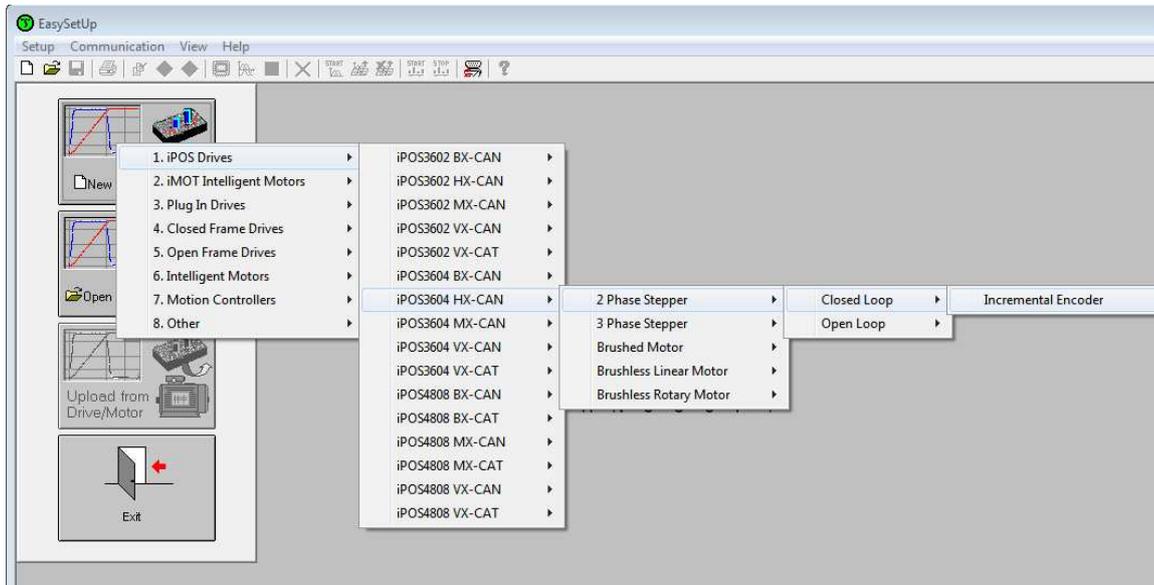
The selection continues with the motor technology (for example: brushless or brushed) and type of feedback device (for example: Incremental encoder, Linear Halls).

The selection opens 2 setup dialogues: for **Motor Setup** and for **Drive setup** through which you can configure and parameterize a Technosoft drive, plus several predefined control panels customized for the product selected.
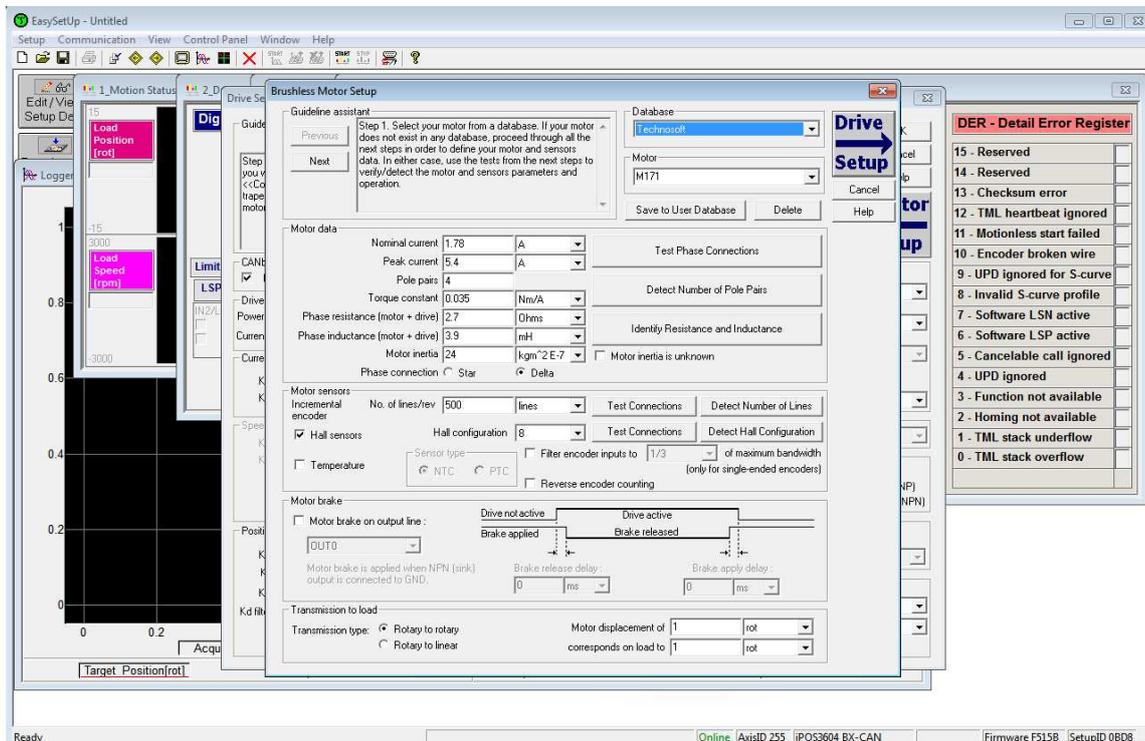
In the **Motor setup** dialogue you can introduce the data of your motor and the associated sensors. Data introduction is accompanied by a series of tests having as goal to check the connections to the drive and/or to determine or validate a part of the motor and sensors parameters. In the **Drive setup** dialogue you can configure and parameterize the drive for your application. In each dialogue you will find a **Guideline Assistant**, which will guide you through the whole process of introducing and/or checking your data. Close the Drive setup dialogue with **OK** to keep all the changes regarding the motor and the drive setup.

4.2.3. **Selecting NPN/PNP inputs type in Setup**
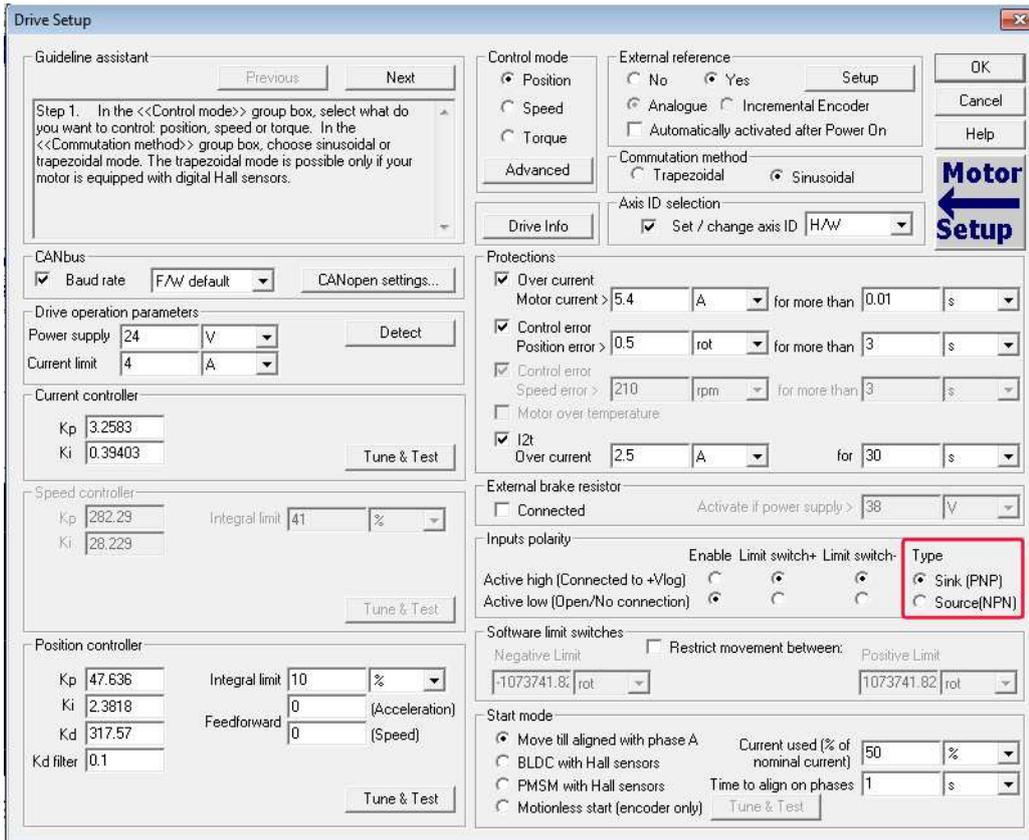
In Drive Setup, choose the inputs type PNP or NPN.
Remark: the inputs can be used, only when $+V_{log}$ from J4, J5 or J6 is connected to the power supply.

4.2.4. **Download setup data to drive/motor**

Press the **Download to Drive/Motor** button to download your setup data in the drive/motor EEPROM memory in the *setup table*. From now on, at each power-on, the setup data is copied into the
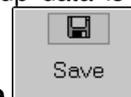
drive/motor RAM memory which is used during runtime. It is also possible to **Save** the setup data on your PC and use it in other applications.

To summarize, you can define or change the setup data in the following ways:

- create a new setup data by going through the motor and drive dialogues
- use setup data previously saved in the PC
- upload setup data from a drive/motor EEPROM memory

4.2.5. **Evaluate drive/motor behavior (optional)**

You can use the **Data Logger** or the **Control Panel** evaluation tools to quickly measure and analyze your application behavior. In case of errors like protections triggered, use the Drive Status control panel to find the cause.

4.3. Changing the drive Axis ID
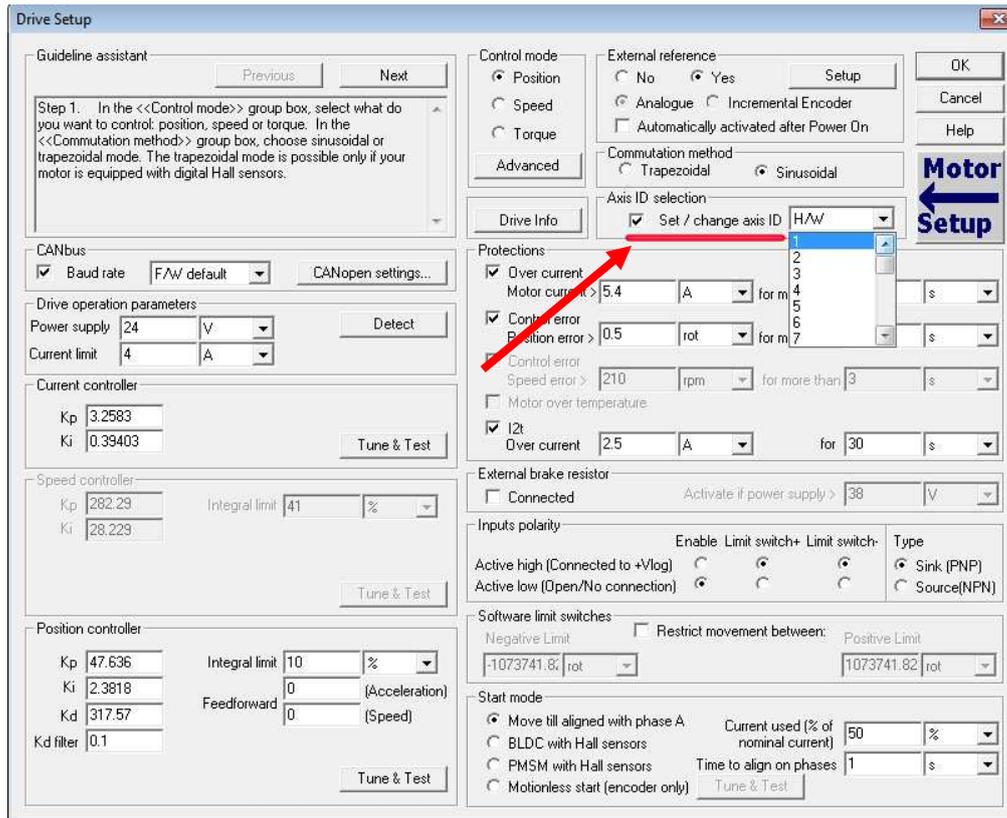
The axis ID of an iPOS360x drive can be set in 3 ways:

- Hardware (H/W) – according with AxisID hex switch SW1 par. 3.4.2 .
- Software (via Setup)– any value between 1 and 255, stored in the setup table. If the drive is in CANopen mode, a Node ID value above 127 is automatically converted into 255 and the drive is set with CAN communication in "non-configured" mode waiting for a CANopen master to configure it using CiA-305 protocol. A "non-configured" drive answers only to CiA-305 commands. All other CANopen commands are ignored and transmission of all other CANopen messages (including boot-up) is disabled. The green (Ready) led will flash at 1 second time intervals when in this mode. In absence of a CANopen master, you can get out a drive from "non-configured" mode, by setting another axis ID between 1 and 127, from above dialogue using a serial link between the drive and the PC.
- Software (via CANopen master) – using CiA-305 protocol

The axis ID is initialized at power on, using the following algorithm:

a) If a valid setup table exists, and this setup table was created with the *Axis ID Selection* checkbox checked in the Drive Setup dialogue (see above) – with the value read from the setup table. This value can be an axis number 1 to 255 or can indicate that axis ID will be set according with the AxisID hex switch. If the drive is set in CANopen mode and the Axis ID is over 127 it is converted into 255 and the drive enters in CAN communication "LSS non-configured" mode

b) If a valid the setup table exists, and this was created with the *Axis ID Selection* checkbox unchecked in the Drive Setup dialogue (see above) – with the last value set either from a valid setup table or by a CANopen master via CiA-305 protocol. This value can be an axis number 1 to 255 for TMLCAN, 1 to 127 for CANopen, or can indicate that axis ID will be set according with the AxisID hex switch

c) If the setup table is invalid, with the last value set either from a valid setup table or by a CANopen master via CiA-305 protocol. This value can be an axis number 1 to 255 for TMLCAN, 1 to 127 for CANopen, or can indicate that axis ID will be set according with the AxisID hex switch

d) If the setup table is invalid, there is no previous axis ID set from a valid setup table or by a CANopen master, according with the AxisID hex switch

Remark: *If you don't know the axis ID set in a drive, you can find it in the following way:*

a) Connect the drive via a serial RS232 link to a PC where EasySetUp or EasyMotion Studio are installed
b) With the drive powered, open EasySetUp or EasyMotion Studio and check the status bar. If communication with the drive is established, the status bar displays **Online** in green and nearby the drive's Axis ID. If the status bar displays **Offline** in red, execute menu command "Communication|Setup…" and in the dialogue opened select at "Channel Type" **RS232** and at "Axis ID of drive/motor connected to PC" the option **Autodetected**. After closing the dialogue with OK, communication with the drive shall be established and the status bar shall display the drive's Axis ID
c) If the access to the drive with the unknown Axis ID is difficult, but this drive is connected via CANbus with other Technosoft drives having an easier access, connect your PC serially to one of the other drives. Use EasySetUp or EasyMotion Studio menu command **Communication | Scan Network** to find the axis IDs of all the Technosoft drives present in the network.

4.4. Setting CANbus rate

The iPOS360x drives accept the following CAN rates: 125Kbps, 250 Kbps, 500kbps and 1Mbps. Using the Drive Setup dialogue you can choose the initial CAN rate after power on. This information is stored in the setup table The CAN rate is initialized using the following algorithm:
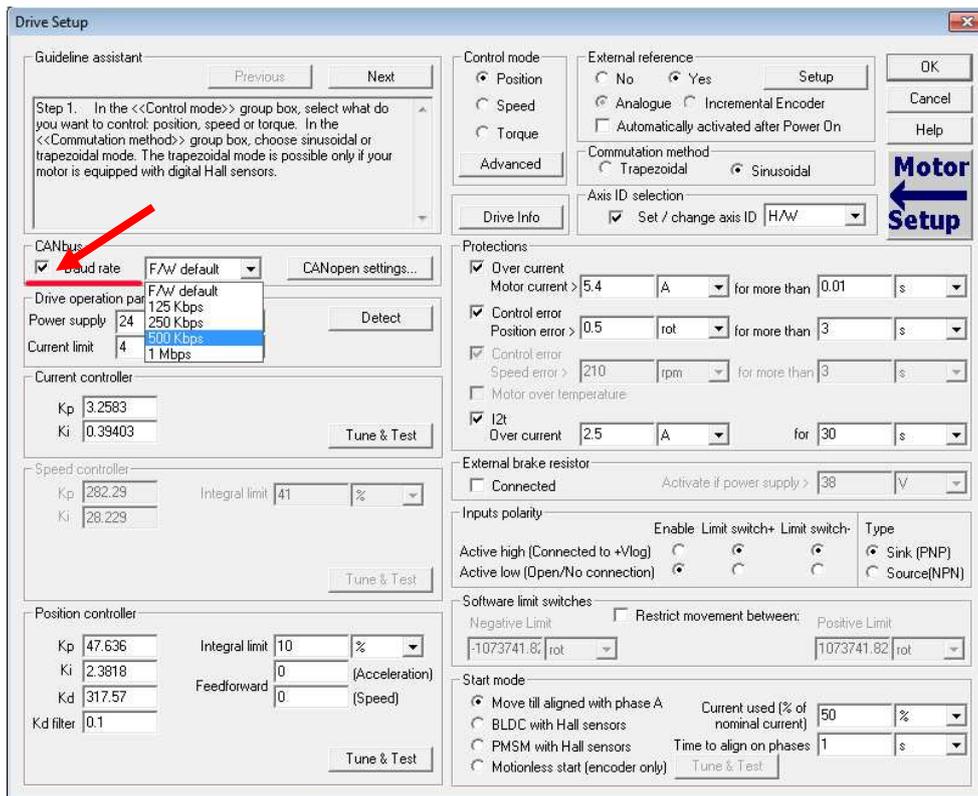
a) If a valid setup table exists, and this setup table was created with the *Set baud rate* checkbox <u>checked</u> in the Drive Setup dialogue (see above) – with the value read from the setup table. This value can be one of the above 4 values or the firmware default (F/W default) which is 500kbs
b) If a valid setup table exists, and this setup table was created with the *Set baud rate* checkbox <u>unchecked</u> in the Drive Setup dialogue (see above) – with the last value set either from a valid setup table or by a CANopen master via CiA-305 protocol
c) If the setup table is invalid, with the last value set either from a valid setup table or by a CANopen master via CiA-305 protocol.
d) If the setup table is invalid, there is no previous CAN rate set from a valid setup table or by a CANopen master, with f/w default value which is 500kbs

4.5. Creating an Image File with the Setup Data

Once you have validated your setup, you can create with the menu command **Setup | Create EEPROM Programmer File** a software file (with extension **.sw**) which contains all the setup data to write in the EEPROM of your drive.

A software file is a text file that can be read with any text editor. It contains blocks of data separated by an empty row. Each block of data starts with the block start address, followed by data values to place in ascending order at consecutive addresses: first data – to write at start address, second data – to write at start address + 1, etc. All the data are hexadecimal 16- bit values (maximum 4 hexadecimal digits). Each raw contains a single data value. When less then 4 hexadecimal digits are shown, the value must be right justified. For example 92 represent 0x0092.

The **.sw** file can be programmed into a drive:

- from a CANopen master, using the communication objects for writing data into the drive EEPROM
- from a host PC or PLC, using the TML_LIB functions for writing data into the drive EEPROM
- using the EEPROM Programmer tool, which comes with EasySetUp but may also be installed separately. The EEPROM Programmer was specifically designed for repetitive fast and easy programming of **.sw** files into the Technosoft drives during production.

5. Step 3. Motion Programming

5.1. Using a CANopen Master (for iPOS360x CANopen execution)

The iPOS360x drive conforms to **CiA 301 v.4.2** application layer and communication profile, **CiA WD 305 v.2.2.13** layer settings services and protocols and **CiA DSP 402 v3.0** device profile for drives and motion control the now included in IEC 61800-7-1 Annex A, IEC 61800-7-201 and IEC 61800-7-301 standards. For details see *iPOS CANopen Programming* **(part no. P091.063.iPOS.UM.xxxx)**.

5.1.1. CiA-301 Application Layer and Communication Profile Overview

The iPOS360x drive accepts the following basic services and types of communication objects of the CANopen communication profile CiA301 v4.2:

- **Service Data Object (SDO)**

Service Data Objects (SDOs) are used by CANopen master to access any object from the drive's Object Dictionary. Both expedited and segmented SDO transfers are supported. SDO transfers are confirmed services. The SDOs are typically used for drive configuration after power-on, for PDOs mapping and for infrequent low priority communication between the CANopen master and the drives.

- **Process Data Object (PDO)**

Process Data Objects (PDO) are used for high priority, real-time data transfers between CANopen master and the drives. The PDOs are unconfirmed services which are performed with no protocol overhead. Transmit PDOs are used to send data from the drive, and receive PDOs are used to receive on to the drive. The iPOS360x accepts 4 transmit PDOs and 4 receive PDOs. The contents of the PDOs can be set according with the application needs using the dynamic PDO-mapping. This operation can be done during the drive configuration phase using SDOs.

- **Synchronization Object (SYNC)**

The SYNC message provides the basic network clock, as the SYNC producer broadcasts the synchronization object periodically. The service is unconfirmed. The iPOS360x supports both SYNC consumer and producer.

- **Time Stamp Object (TIME)**

The Time Stamp Object is supported by the iPOS360x device.

- **Emergency Object (EMCY)**

Emergency objects are triggered by the occurrence of a drive internal error situation. An emergency object is transmitted only once per 'error event'. As long as no new errors occur, the drive will not transmit further emergency objects.

- **Network Management Objects (NMT)**

The Network Management is node oriented and follows a master-slave structure. NMT objects are used for executing NMT services. Through NMT services the drive can be initialized, started, monitored, reset or stopped. The iPOS360x is a NMT slave in a CANopen network.

- **Module Control Services** – through these unconfirmed services, the NMT master controls the state of the drive. The following services are implemented: Start Remote Node, Stop Remote Node, Enter Pre-Operational, Reset Node, Reset Communication
- **Error Control Services** – through these services the NMT master detects failures in a CAN-based network. Both error control services defined by DS301 v4.02 are supported by the iPOS360x: Node Guarding (including Life Guarding) and Heartbeat

- **Bootup Service** - through this service, the drive indicates that it has been properly initialized and is ready to receive commands from a master

5.1.2. CiA-305 Layer Setting Services (LSS) and Protocols Overview

When used in a CANopen network, the iPOS360x drives accept node-ID and CAN bus bit timing settings according with CiA 305 protocol. This allows a CANopen master supporting CiA WD 305 to configure each iPOS360x from the network with the desired node-ID and CAN bus bit timing. CiA-305 protocol allows connecting non-configured drives to a CANopen network and performing the drives configuration on-the-fly via the CANopen master.

5.1.3. CiA-402 and Manufacturer Specific Device Profile Overview

The iPOS360x supports the following CiA 402 modes of operation:

- **Profile position and velocity modes**
- **Homing mode**
- **Interpolated position mode**

Additional to these modes, there are also several manufacturer specific modes defined:

- **External reference modes (position, speed or torque)**
- **Electronic gearing and camming position mode**

5.1.4. TechnoCAN Extension

In order to take full advantage of the powerful Technosoft Motion Language (TML) built into the iPOS360x, Technosoft has developed an extension to CANopen, called TechnoCAN through which TML commands can be exchanged with the drives. Thanks to TechnoCAN you can inspect or reprogram any of the Technosoft drives from a CANopen network using EastSetup or EasyMotion Studio and an RS-232 link between your PC and any of the drives.

TechnoCAN uses only identifiers outside of the range used by the default by the CANopen predefined connection set (as defined by CiA 301). Thus, TechnoCAN protocol and CANopen protocol can co-exist and communicate simultaneously on the same physical CAN bus, without disturbing each other.

5.1.5. Checking Setup Data Consistency

During the configuration phase, a CANopen master can quickly verify using the checksum objects and a reference **.sw** file (see 4.5 and 5.2.4 for details) whether the non-volatile EEPROM memory of an iPOS360x drive contains the right information. If the checksum reported by the drive doesn't match with that computed from the **.sw** file, the CANopen master can download the entire **.sw** file into the drive EEPROM using the communication objects for writing data into the drive EEPROM.

5.2. Using the built-in Motion Controller and TML

One of the key advantages of the Technosoft drives is their capability to execute complex motions without requiring an external motion controller. This is possible because Technosoft drives offer in a single compact package both a state of art digital drive and a powerful motion controller.

5.2.1. Technosoft Motion Language Overview

Programming motion directly on a Technosoft drive requires creating and downloading a TML (Technosoft Motion Language) program into the drive memory. The TML allows you to:

- Set various motion modes (profiles, PVT, PT, electronic gearing or camming[1], etc.)
- Change the motion modes and/or the motion parameters

---

[1] Optional for iPOS360x CANopen execution

- ▪ Execute homing sequences[1]
- ▪ Control the program flow through:

  - • Conditional jumps and calls of TML functions
  - • TML interrupts generated on pre-defined or programmable conditions (protections triggered, transitions on limit switch or capture inputs, etc.)
  - • Waits for programmed events to occur

- ▪ Handle digital I/O and analogue input signals
- ▪ Execute arithmetic and logic operations
- ▪ Perform data transfers between axes
- ▪ Control motion of an axis from another one via motion commands sent between axes
- ▪ Send commands to a group of axes (multicast). This includes the possibility to start simultaneously motion sequences on all the axes from the group
- ▪ Synchronize all the axes from a network

In order to program a motion using TML you need EasyMotion Studio software platform.

5.2.2. Installing EasyMotion Studio

EasyMotion Studio is an integrated development environment for the setup and motion programming of Technosoft intelligent drives. It comes with an ***Update via Internet tool*** through which you can check if your software version is up-to-date, and when necessary download and install the latest updates.

A **demo version of EasyMotion Studio** including the **fully functional version of EasySetUp** can be downloaded free of charge from Technosoft web page.

EasyMotion Studio is delivered on a CD. Once you have started the installation package, follow its indications. After installation, use the update via internet tool to check for the latest updates. Alternately, you can first install the demo version and then purchase a license. By introducing the license serial number in the menu command **Help | Enter registration info…**, you can transform the demo version into a fully functional version.

5.2.3. Getting Started with EasyMotion Studio

Using EasyMotion Studio you can quickly do the setup and the motion programming of a Technosoft a drive according with your application needs. The drive can be connected with your PC in one of the following ways:

1. Via an RS232 link, directly connected to the PC, or via an USB to RS232 adapter or using Technosoft Ethernet to RS232 adapter, function of your PC communication interfaces;
2. Via a CAN-bus link, directly connected to the PC through a PC-CAN interface, or using Technosoft Ethernet to CAN adapter
3. Via another drive from the same CAN-bus network, which is connected to the PC via one of the above options from point 1.

The output of the EasyMotion Studio is a set of setup data and a motion program, which can be downloaded to the drive/motor EEPROM or saved on your PC for later use.

EasyMotion Studio includes a set of evaluation tools like the Data Logger, the Control Panel and the Command Interpreter which help you to quickly develop, test, measure and analyze your motion application.

EasyMotion Studio works with **projects**. A project contains one or several **Applications**.

Each application describes a motion system for one axis. It has 2 components: the **Setup** data and the **Motion** program and an associated axis number: an integer value between 1 and 255. An application may be used either to describe:

---

[1] The customization of the homing routines is available only for iPOS360x CAN execution

1. One axis in a multiple-axis system
2. An alternate configuration (set of parameters) for the same axis.

In the first case, each application has a different axis number corresponding to the axis ID of the drives/motors from the network. All data exchanges are done with the drive/motor having the same address as the selected application. In the second case, all the applications have the same axis number.

The setup component contains all the information needed to configure and parameterize a Technosoft drive. This information is preserved in the drive/motor EEPROM in the *setup table*. The setup table is copied at power-on into the RAM memory of the drive/motor and is used during runtime.

The motion component contains the motion sequences to do. These are described via a TML (Technosoft Motion Language) program, which is executed by the drives/motors built-in motion controller.

5.2.3.1   Create a new project

EasyMotion Studio starts with an empty window from where you can create a new project or open a previously created one.
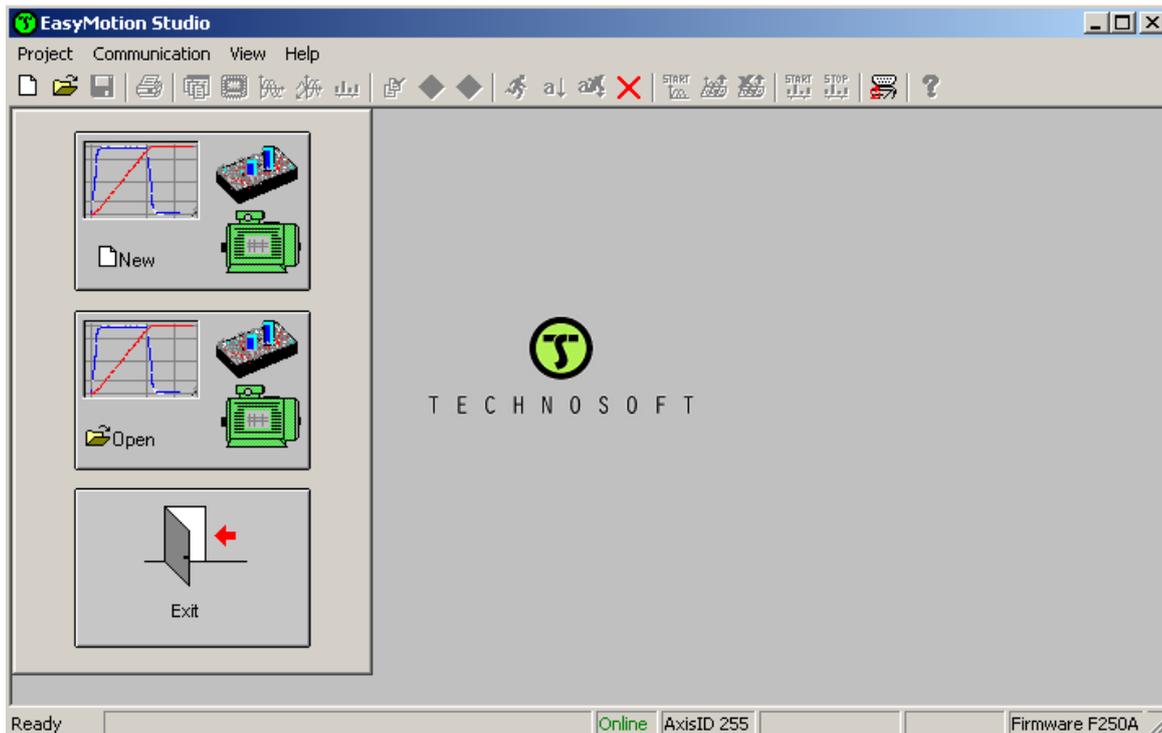
When you start a new project, EasyMotion Studio automatically creates a first application. Additional applications can be added later. You can duplicate an application or insert one defined in another project.
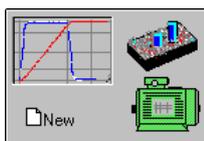
Press **New** button ⬚⬚⬚⬚ to open the "New Project" dialogue. Set the axis number for your first application equal with your drive/motor axis ID. The initial value proposed is 255 which is the default axis ID of the drives. Press **New** button and select your drive type. Depending on the product chosen, the selection may continue with the motor technology (for example: brushless or brushed) and the type of feedback device (for example: incremental encoder).
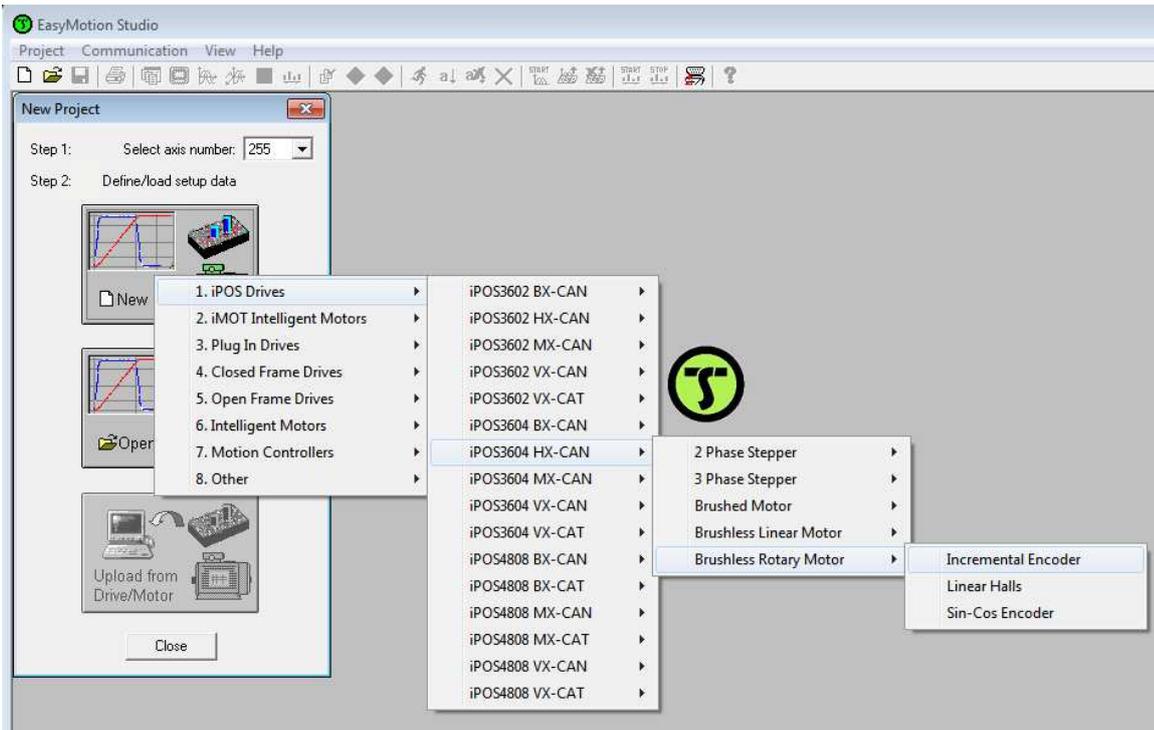
Click on your selection. EasyMotion Studio opens the Project window where on the left side you can see the structure of a project. At beginning both the new project and its first application are named "Untitled". The application has 2 components: **S** Setup and **M** Motion (program).
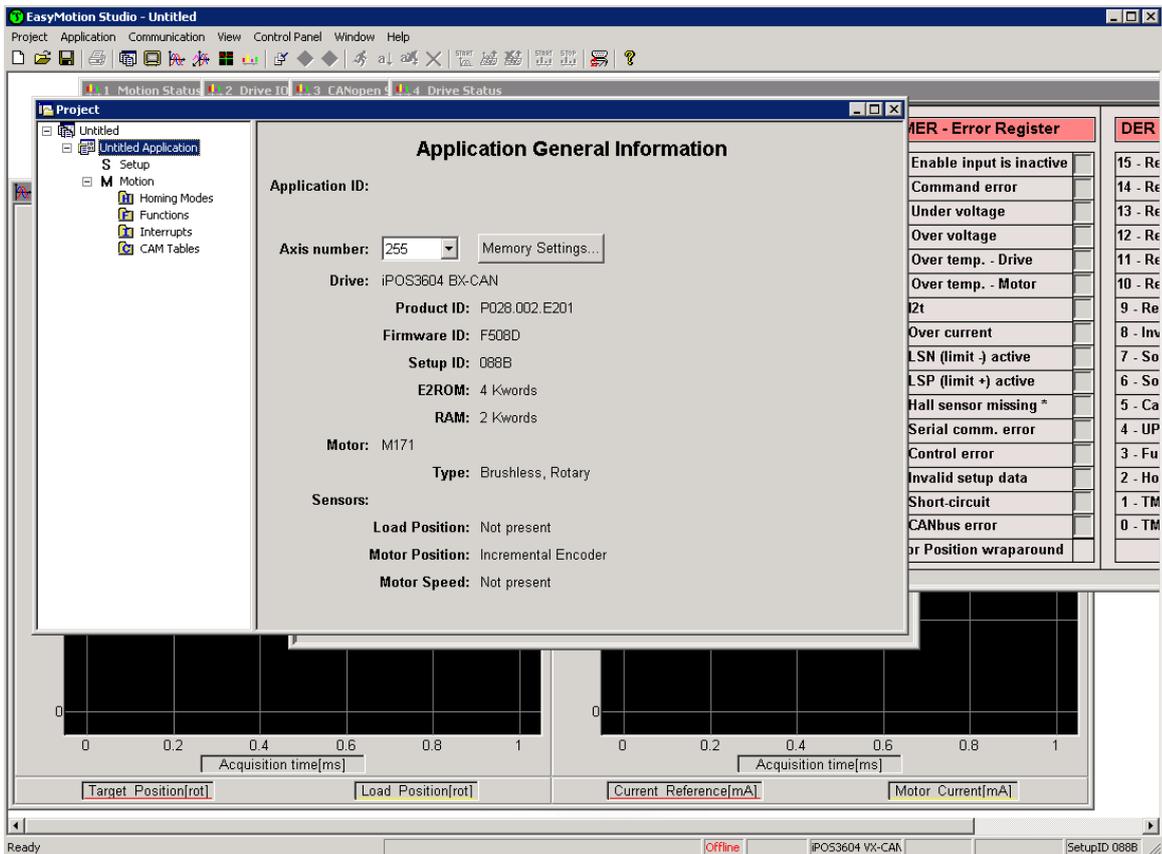
5.2.3.2   Step 2 Establish communication

If you have a drive/motor connected with your PC, now its time to check the communication. Use menu command **Communication | Setup** to check/change your PC communication settings. Press the **Help** button

of the dialogue opened. Here you can find detailed information about how to setup your drive/motor and the connections. Power on the drive, then close the Communication | Setup dialogue with OK. If the communication is established, EasyMotion Studio displays in the status bar (the bottom line) the text "**Online**" plus the axis ID of your drive/motor and its firmware version. Otherwise the text displayed is "**Offline**" and a communication error message tells you the error type. In this case, return to the Communication | Setup dialogue, press the Help button and check troubleshoots.

*Remark: When first started, EasyMotion Studio tries to communicate via RS-232 and COM1 with a drive having axis ID=255 (default communication settings).).   If the drive has a different axis ID and you don't know it, select in the Communication | Setup dialogue at "Axis ID of drive/motor connected to PC" the option **Autodetected.** If this drive is part of a CANbus network and the PC is serially connected with another drive, use the menu command **Communication | Scan Network***

5.2.3.3   Setup drive/motor

In the project window left side, select "**S** Setup", to access the setup data for your application.
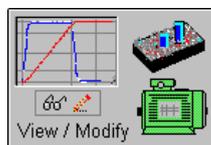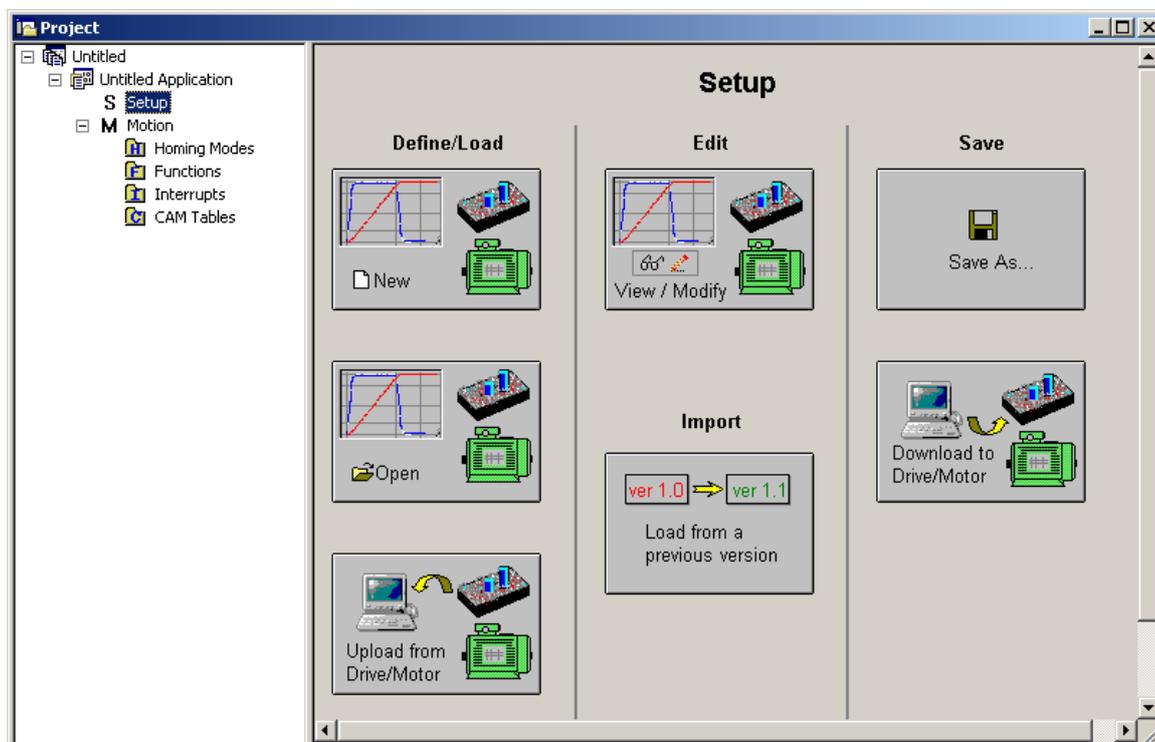
Press **View/Modify** button ⬛. This opens 2 setup dialogues: for **Motor Setup** and for **Drive Setup** (same like on EasySetUp) through which you can configure and parameterize a Technosoft drive. In the **Motor setup** dialogue you can introduce the data of your motor and the associated sensors. Data introduction is accompanied by a series of tests having as goal to check the connections to the drive and/or to determine or validate a part of the motor and sensors parameters. In the **Drive setup** dialogue you can configure and parameterize the drive for your application. In each dialogue you will find a **Guideline Assistant**, which will guide you through the whole process of introducing and/or checking your data.

Press the **Download to Drive/Motor** button to download your setup data in the drive/motor EEPROM memory in the *setup table*. From now on, at each power-on, the setup data is copied into the drive/motor RAM memory which is used during runtime. It is also possible to save the setup data on your PC and use it in other applications. Note that you can upload the complete setup data from a drive/motor.

To summarize, you can define or change the setup data of an application in the following ways:

- create a new setup data by going through the motor and drive dialogues

- use setup data previously saved in the PC

- upload setup data from a drive/motor EEPROM memory

5.2.3.4 Program motion

In the project window left side, select "**M** Motion", for motion programming. This automatically activates the **Motion Wizard**.
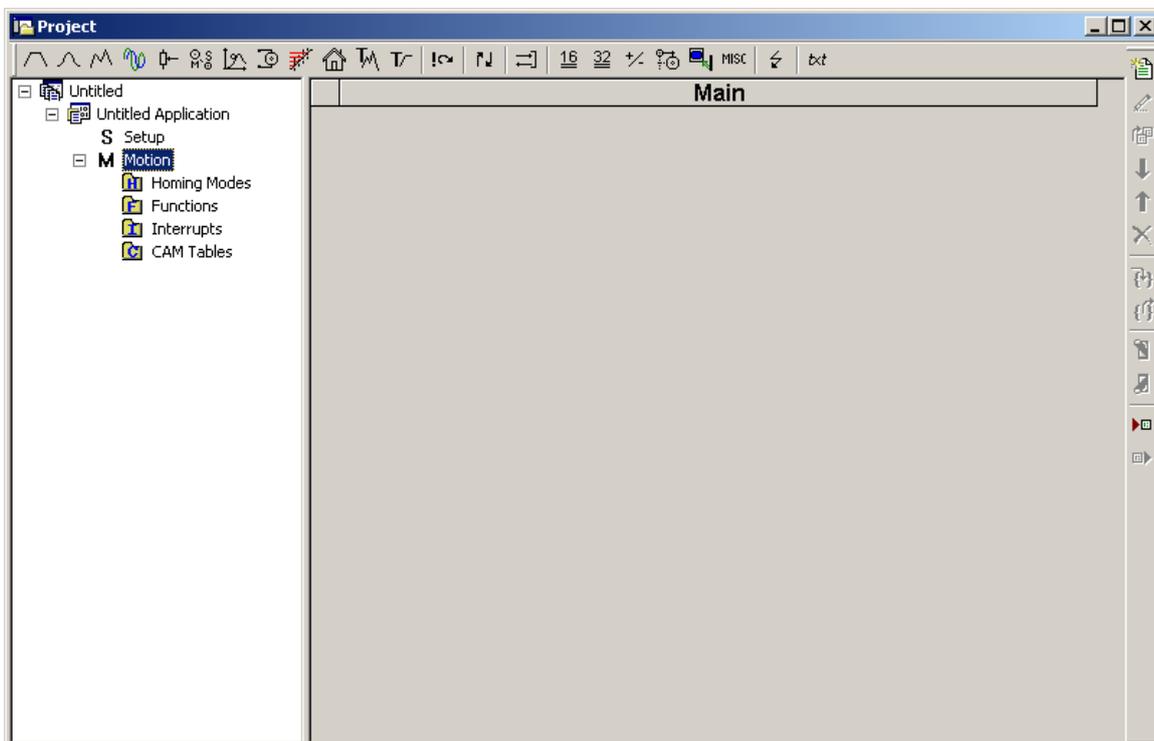
The Motion Wizard offers you the possibility to program all the motion sequences using high level graphical dialogues which automatically generate the corresponding TML instructions. Therefore with Motion Wizard you can develop motion programs using almost all the TML instructions without needing to learn them. A TML program includes a main section, followed by the subroutines used: functions, interrupt service routines and homing procedures[1]. The TML program may also include cam tables used for electronic camming applications[2].

When activated, Motion Wizard adds a set of toolbar buttons in the project window just below the title. Each button opens a programming dialogue. When a programming dialogue is closed, the associated TML

---

[1] The customization of the interrupt service routines and homing routines is available only for iPOS360x CAN execution
[2] Optional for iPOS360x CANopen execution

instructions are automatically generated. Note that, the TML instructions generated are not a simple text included in a file, but a motion object. Therefore with Motion Wizard you define your motion program as a collection of motion objects.

The major advantage of encapsulating programming instructions in motion objects is that you can very easily manipulate them. For example, you can:

- Save and reuse a complete motion program or parts of it in other applications
- Add, delete, move, copy, insert, enable or disable one or more motion objects
- Group several motion objects and work with bigger objects that perform more complex functions

As a starting point, push for example the leftmost Motion Wizard button – Trapezoidal profiles, and set a position or speed profile. Then press the **Run** button. At this point the following operations are done automatically:

- A TML program is created by inserting your motion objects into a predefined template
- The TML program is compiled and downloaded to the drive/motor
- The TML program execution is started

For learning how to send TML commands from your host/master, using one of the communication channels and protocols supported by the drives use menu command **Application | Binary Code Viewer…** Using this tool, you can get the exact contents of the messages to send and of those expected to be received as answers.

5.2.3.5 Evaluate motion application performances

EasyMotion Studio includes a set of evaluation tools like the **Data Logger**, the **Control Panel** and the **Command Interpreter** which help you to quickly measure and analyze your motion application.

5.2.4. Creating an Image File with the Setup Data and the TML Program

Once you have validated your application, you can create with the menu command **Application | Create EEPROM Programmer File** a software file (with extension **.sw**) which contains all the data to write in the EEPROM of your drive. This includes both the setup data and the motion program. For details regarding the **.sw** file format and how it can be programmed into a drive, see paragraph 4.5

5.3. Combining CANopen /or other host with TML

Due to its embedded motion controller, an iPOS360x offers many programming solutions that may simplify a lot the task of a CANopen master. This paragraph overviews a set of advanced programming features which arise when combining TML programming at drive level with CANopen master control. A detailed description of these advanced programming features is included in the **CANopen Programming (part no. P091.063.UM.xxxx)** manual. All features presented below require usage of EasyMotion Studio as TML programming tool

Remark: *If you don't use the advanced features presented below you don't need EasyMotion Studio. In this case the iPOS360x is treated like a standard CANopen drive, whose setup is done using EasySetUp.*

5.3.1. Using TML Functions to Split Motion between Master and Drives

With Technosoft intelligent drives you can really distribute the intelligence between a CANopen master and the drives in complex multi-axis applications. Instead of trying to command each step of an axis movement, you can program the drives using TML to execute complex tasks and inform the master when these are done. Thus for each axis, the master task may be reduced at: calling TML functions (with possibility to abort their execution) stored in the drives EEPROM and waiting for a message, which confirms the finalization of the TML functions execution.

5.3.2. Executing TML programs

The distributed control concept can go one step further. You may prepare and download into a drive a complete TML program including functions, homing procedures[1], etc. The TML program execution can be started by simply writing a value in a dedicated object,

5.3.3. Loading Automatically Cam Tables Defined in EasyMotion Studio

The iPOS360x offers others motion modes like[2]: electronic gearing, electronic camming, external modes with analogue or digital reference etc. When electronic camming is used, the cam tables can be loaded in the following ways:

a) The master downloads the cam points into the drive active RAM memory after each power on;
b) The cam points are stored in the drive EEPROM and the master commands their copy into the active RAM memory
c) The cam points are stored in the drive EEPROM and during the drive initialization (transition to Ready to Switch ON status) are automatically copied from EEPROM to the active RAM

For the last 2 options the cam table(s) are defined in EasyMotion Studio and are included in the information stored in the EEPROM together with the setup data and the TML programs/functions.
Remark: The cam tables are included in the **.sw** file generated with EasyMotion Studio. Therefore, the drives can check the cam presence in the drive EEPROM using the same procedure as for testing of the setup data.

5.3.4. Customizing the Homing Procedures

The iPOS360x supports all homing modes defined in CiA402 device profile, plus 4 custom based on hard stop. If needed, any of these homing modes can be customized. In order to do this you need to select the Homing Modes from your EasyMotion Studio application and in the right side to set as "User defined" one of the Homing procedures. Following this operation the selected procedure will occur under Homing Modes in a subtree, with the name *HomeX* where X is the number of the selected homing.
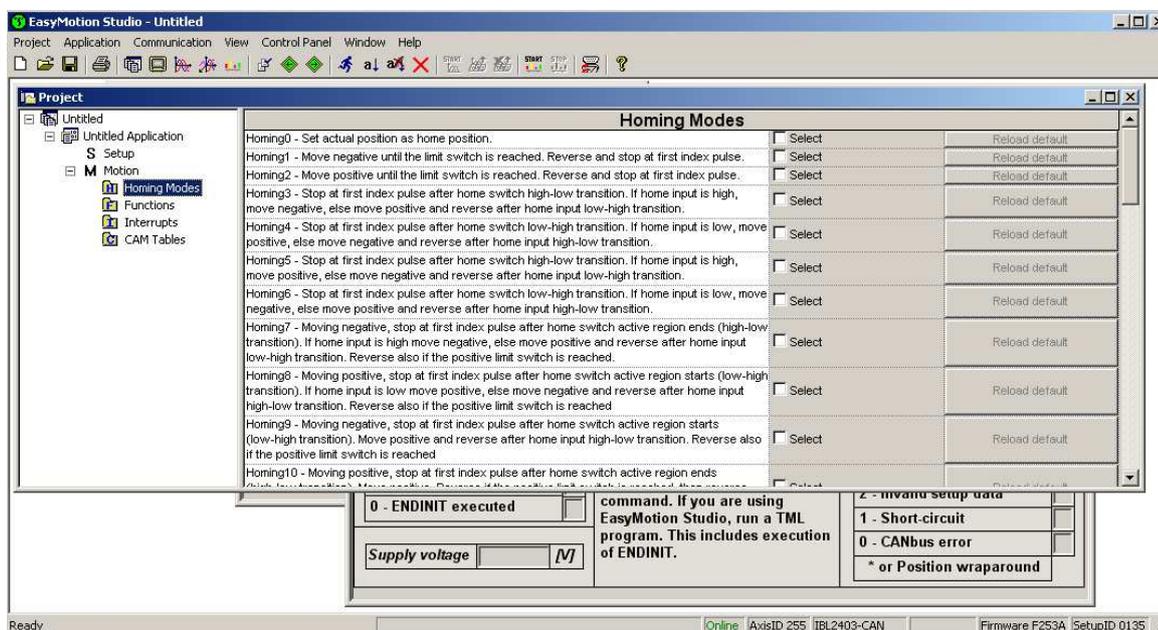
---

[1] The customization of the interrupt service routines and homing routines is available only for iPOS360x CAN executions
[2] Optional for the iPOS360x CANopen execution

If you click on the *HomeX* procedure, on the right side you'll see the TML function implementing it. The homing routine can be customized according to your application needs. It's calling name and method remain unchanged.

5.3.5. Customizing the Drive Reaction to Fault Conditions

Similarly to the homing modes, the default service routines for the TML interrupts can be customized according to your application needs. However, as most of these routines handle the drive reaction to fault conditions, it is mandatory to keep the existent functionality while adding your application needs, in order to preserve the correct protection level of the drive. The procedure for modifying the TML interrupts is similar with that for the homing modes.

5.4. Using Motion Libraries for PC-based Systems

A **TML Library for PC** is a collection of high-level functions allowing you to control from a PC a network of Technosoft intelligent drives. It is an ideal tool for quick implementation on PCs of motion control applications with Technosoft products.

With the TML Motion Library functions you can: communicate with a drive / motor via any of its supported channels (RS-232, CAN-bus, etc.), send motion commands, get automatically or on request information about drive / motor status, check and modify its setup parameters, read inputs and set outputs, etc.

The TML Motion Library can work under a **Windows** or **Linux** operating system. Implemented as a .dll/.so, it can be included in an application developed in **C/C++/C#, Visual Basic, Delphi Pascal** or **Labview**.

Using a TML Motion Library for PC, you can focus on the main aspects of your application, while the motion programming part can be reduced to calling the appropriate functions and getting the confirmation when the task was done.

All Technosoft's TML Motion Libraries for PCs are provided with EasySetUp.

5.5. Using Motion Libraries for PLC-based Systems

A **TML Motion Library for PLC** is a collection of high-level functions and function blocks allowing you to control from a PLC the Technosoft intelligent drives. The motion control function blocks are developed in accordance with the **PLC IEC61131-3 standard** and represent an ideal tool for quick implementation on PLCs of motion control applications with Technosoft products.

With the TML Motion Library functions you can: communicate with a drive/motor via any of its supported channels, send motion commands, get automatically or on request information about drive/motor status, check and modify its setup parameters, read inputs and set outputs, etc. Depending on the PLC type, the communication is done either directly with the CPU unit, or via a CANbus or RS-232 communication module.

Using a TML Motion Library for PLC, you can focus on the main aspects of your PLC application, while the motion programming part can be reduced to calling the appropriate functions and monitoring the confirmations that the task was done.

All these blocks have been designed using the guidelines described in the PLC standards, so they can be used on any development platform that is **IEC 61136 compliant.**

All Technosoft's TML Motion Libraries for PLC are provided with EasySetUp.

6. Scaling factors

Technosoft drives work with parameters and variables represented in the drive internal units (IU). These correspond to various signal types: position, speed, current, voltage, etc. Each type of signal has its own internal representation in IU and a specific scaling factor. This chapter presents the drive internal units and their relation with the international standard units (SI).

In order to easily identify them, each internal unit has been named after its associated signal. For example the **position units** are the internal units for position, the **speed units** are the internal units for speed, etc.

6.1. Position units

6.1.1. Brushless / DC brushed motor with quadrature encoder on motor

The internal position units are encoder counts. The correspondence with the load **position in SI units**[1] is:

$$Load\_Position[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times Tr} \times Motor\_Position[IU]$$

where:

No_encoder_lines – is the rotary encoder number of lines per revolution

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

6.1.2. Brushless motor with linear Hall signals

The internal position units are counts. The motor is rotary. The resolution i.e. number of counts per revolution is programmable as a power of 2 between 512 and 8192. By default it is set at 2048 counts per turn. The correspondence with the load position in SI units is:

For rotary motors: $$Load\_Position[SI] = \frac{2 \times \pi}{resolution \times Tr} \times Motor\_Position[IU]$$

For linear motors: $$Load\_Position[SI] = \frac{Pole\_Pitch}{Tr} \times Motor\_Position[IU]$$

where:

resolution – is the motor position resolution

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

Pole_Pitch – is the magnetic pole pitch NN (distance expressed in [m])

6.1.3. DC brushed motor with quadrature encoder on load and tacho on motor

The internal position units are encoder counts. The motor is rotary and the transmission is rotary-to-rotary. The correspondence with the load position in SI units is:

---

[1]SI units for position are: [rad] for a rotary movement, [m] for a linear movement

$$Load\_Position[rad] = \frac{2 \times \pi}{4 \times No\_encoder\_lines} \times Load\_Position[IU]$$

where:

No_encoder_lines – is the encoder number of lines per revolution

6.1.4. **Step motor open-loop control. No feedback device**

The internal position units are motor µsteps. The correspondence with the load **position in SI units** is:

$$Load\_Position[SI] = \frac{2 \times \pi}{No\_\mu steps \times No\_steps \times Tr} \times Motor\_Position[IU]$$

where:

No_steps – is the number of motor steps per revolution

No_µsteps – is the number of microsteps per step. You can read/change this value in the "Drive Setup" dialogue from EasySetUp.

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

Step motor closed-loop control. Incremental encoder on motor

The internal position units are motor encoder counts. The correspondence with the load **position in SI units** [1] is:

$$Load\_Position[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times Tr} \times Motor\_Position[IU]$$

where:

No_encoder_lines – is the motor encoder number of lines per revolution

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

6.1.5. **Step motor open-loop control. Incremental encoder on load**

The internal position units are load encoder counts. The transmission is rotary-to-rotary. The correspondence with the load position in SI units is:

$$Load\_Position[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines} \times Load\_Position[IU]$$

where:

No_encoder_lines – is the rotary encoder number of lines per revolution

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

---

[1] SI units for position are [rad] for a rotary movement , [m] for a linear movement

6.1.6. Brushless motor with sine/cosine encoder on motor

The internal position units are interpolated encoder counts. The correspondence with the load position in SI units is:

For rotary motors:

$$Load\_Position[SI] = \frac{2 \times \pi}{4 \times Enc\_periods \times Interpolation \times Tr} \times Motor\_Position[IU]$$

For linear motors:

$$Load\_Position[SI] = \frac{Encoder\_accuracy}{Interpolation \times Tr} \times Motor\_Position[IU]$$

where:

Enc_periods – is the rotary encoder number of sine/cosine periods or lines per revolution

Interpolation – is the interpolation level inside an encoder period. Its a number power of 2 between 1 an 256. 1 means no interpolation

Encoder_accuracy – is the linear encoder accuracy in [m] for one sine/cosine period

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

6.2. Speed units

The internal speed units are internal position units / (slow loop sampling period) i.e. the position variation over one slow loop sampling period

6.2.1. Brushless / DC brushed motor with quadrature encoder on motor

The internal speed units are encoder counts / (slow loop sampling period). The correspondence with the load **speed in SI units** is:

$$Load\_Speed[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times Tr \times T} \times Motor\_Speed[IU]$$

where:

No_encoder_lines – is the rotary encoder number of lines per revolution

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.2.2. Brushless motor with linear Hall signals

The internal speed units are counts / (slow loop sampling period). The motor is rotary. The position resolution i.e. number of counts per revolution is programmable as a power of 2 between 512 and 8192. By default it is set at 2048 counts per turn. The correspondence with the load speed in SI units is:

For rotary motors: $$Load\_Speed[SI] = \frac{2 \times \pi}{resolution \times Tr \times T} \times Motor\_Speed[IU]$$

For linear motors: $$Load\_Speed[SI] = \frac{Pole\_Pitch}{resolution \times Tr \times T} \times Motor\_Speed[IU]$$

where:

resolution – is the motor position resolution

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

Pole_Pitch – is the magnetic pole pitch NN (distance expressed in [m])

6.2.3. DC brushed motor with quadrature encoder on load and tacho on motor

The internal speed units are encoder counts / (slow loop sampling period). The motor is rotary and the transmission is rotary-to-rotary. The correspondence with the load speed in SI units is:

$$Load\_Speed[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times T} \times Load\_Speed[IU]$$

where:

No_encoder_lines – is the encoder number of lines per revolution

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.2.4. DC brushed motor with tacho on motor

When only a tachometer is mounted on the motor shaft, the internal speed units are A/D converter bits. The correspondence with the load **speed in SI units**[1] is:

$$Load\_Speed[SI] = \frac{Analogue\_Input\_Range}{4096 \times Tacho\_gain \times Tr} \times Motor\_Speed[IU]$$

where:

Analogue_Input_Range – is the range of the drive analogue input for feedback, expressed in [V]. You can read this value in the "Drive Info" dialogue, which can be opened from the "Drive Setup"

Tacho_gain – is the tachometer gain expressed in [V/rad/s]

6.2.5. Step motor open-loop control. No feedback device

The internal speed units are motor µsteps / (slow loop sampling period). The correspondence with the load **speed in SI units** is:

$$Load\_Speed[SI] = \frac{2 \times \pi}{No\_\mu steps \times No\_steps \times Tr \times T} \times Motor\_Speed[IU]$$

where:

---

[1] SI units for speed are [rad/s] for a rotary movement, [m/s] for a linear movement

No_steps – is the number of motor steps per revolution

No_µsteps – is the number of microsteps per step. You can read/change this value in the "Drive Setup" dialogue from EasySetUp.

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

Step motor open-loop control. Incremental encoder on load

The internal speed units are load encoder counts / (slow loop sampling period). The transmission is rotary-to-rotary. The correspondence with the load speed in SI units is:

$$Load\_Speed[rad/s] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times T} \times Load\_Speed[IU]$$

where:

No_encoder_lines – is the rotary encoder number of lines per revolution

Tr – transmission ratio between the motor displacement in [rad] and load displacement in [rad] or [m]

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup".

6.2.6. **Step motor closed-loop control. Incremental encoder on motor**

The internal speed units are motor encoder counts / (slow loop sampling period). The correspondence with the load **speed in SI units**[1] is:

$$Load\_Speed[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times Tr \times T} \times Motor\_Speed[IU]$$

where:

No_encoder_lines – is the motor encoder number of lines per revolution

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup".

6.2.7. **Brushless motor with sine/cosine encoder on motor**

The internal speed units are interpolated encoder counts / (slow loop sampling period). The correspondence with the load speed in SI units is:

For rotary motors:

$$Load\_Speed[SI] = \frac{2 \times \pi}{4 \times Enc\_periods \times Interpolation \times Tr \times T} \times Motor\_Speed[IU]$$

For linear motors:

---

[1] SI units for speed are [rad/s] for a rotary movement , [m/s] for a linear movement

$$Load\_Speed[SI] = \frac{Encoder\_accuracy}{Interpolation \times Tr \times T} \times Motor\_Speed[IU]$$

where:

  Enc_periods – is the rotary encoder number of sine/cosine periods or lines per revolution
  Encoder_accuracy – is the linear encoder accuracy in [m] for one sine/cosine period
  Interpolation – is the interpolation level inside an encoder period. Its a number power of 2 between 1 an 256. 1 means no interpolation
  Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units
  T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.3. Acceleration units

The internal acceleration units are internal position units / (slow loop sampling period)^2 i.e. the speed variation over one slow loop sampling period.

6.3.1. Brushless / DC brushed motor with quadrature encoder on motor

The internal acceleration units are encoder counts / (slow loop sampling period)^2. The correspondence with the load **acceleration in SI units** is:

$$Load\_Acceleration[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times Tr \times T^2} \times Motor\_Acceleration[IU]$$

where:

No_encoder_lines – is the rotary encoder number of lines per revolution

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.3.2. Brushless motor with linear Hall signals

The internal acceleration units are counts / (slow loop sampling period)^2. The motor is rotary. The position resolution i.e. number of counts per revolution is programmable as a power of 2 between 512 and 8192. By default it is set at 2048 counts per turn. The correspondence with the load **acceleration in SI units**[1] is:

For rotary motors:

$$Load\_Acceleration[SI] = \frac{2 \times \pi}{resolution \times Tr \times T^2} \times Motor\_Acceleration[IU]$$

For linear motors:

$$Load\_Acceleration[SI] = \frac{Pole\_Pitch}{resolution \times Tr \times T^2} \times Motor\_Acceleration[IU]$$

---

[1] SI units for acceleration are [rad/s^2] for a rotary movement, [m/s^2] for a linear movement

where:

resolution – is the motor position resolution

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

Pole_Pitch – is the magnetic pole pitch NN (distance expressed in [m])

6.3.3. DC brushed motor with quadrature encoder on load and tacho on motor

The internal acceleration units are encoder counts / (slow loop sampling period)[2]. The motor is rotary and the transmission is rotary-to-rotary. The correspondence with the load acceleration in SI units is:

$$Load\_Acceleration[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times T^2} \times Load\_Acceleration[IU]$$

where:

No_encoder_lines – is the encoder number of lines per revolution

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.3.4. Step motor open-loop control. No feedback device

The internal acceleration units are motor μsteps / (slow loop sampling period)[2]. The correspondence with the load **acceleration in SI units** is:

$$Load\_Acceleration[SI] = \frac{2 \times \pi}{No\_\mu steps \times No\_steps \times Tr \times T^2} \times Motor\_Acceleration[IU]$$

where:

No_steps – is the number of motor steps per revolution

No_μsteps – is the number of microsteps per step. You can read/change this value in the "Drive Setup" dialogue from EasySetUp.

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.3.5. Step motor open-loop control. Incremental encoder on load

The internal acceleration units are load encoder counts / (slow loop sampling period)[2]. The correspondence with the load acceleration in SI units is:

    For rotary-to-rotary transmission:

$$Load\_Acceleration[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times T^2} \times Load\_Acceleration[IU]$$

For rotary-to-linear transmission:

$$Load\_Acceleration[m/s^2] = \frac{Encoder\_accuracy}{T^2} \times Load\_Acceleration[IU]$$

where:

No_encoder_lines – is the rotary encoder number of lines per revolution

Encoder_accuracy – is the linear encoder accuracy i.e. distance in [m] between 2 pulses

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup".

6.3.6. Step motor closed-loop control. Incremental encoder on motor

The internal acceleration units are motor encoder counts / (slow loop sampling period)[2]. The transmission is rotary-to-rotary. The correspondence with the load **acceleration in SI units** is:

$$Load\_Acceleration[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times Tr \times T^2} \times Motor\_Acceleration[IU]$$

where:

No_encoder_lines – is the motor encoder number of lines per revolution

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.3.7. Brushless motor with sine/cosine encoder on motor

The internal acceleration units are interpolated encoder counts / (slow loop sampling period)[2]. The correspondence with the load **acceleration in SI units**[1] is:

For rotary motors:

$$Load\_Acceleration[SI] = \frac{2 \times \pi}{4 \times Enc\_periods \times Interpolation \times Tr \times T^2} \times Motor\_Acceleration[IU]$$

For linear motors:

$$Load\_Acceleration[SI] = \frac{Encoder\_accuracy}{Interpolation \times Tr \times T^2} \times Motor\_Acceleration[IU]$$

where:
Enc_periods – is the rotary encoder number of sine/cosine periods or lines per revolution
Encoder_accuracy – is the linear encoder accuracy in [m] for one sine/cosine period

---

[1] SI units for acceleration are [rad/s^2] for a rotary movement, [m/s^2] for a linear movement

Interpolation – is the interpolation level inside an encoder period. Its a number power of 2 between 1 an 256. 1 means no interpolation

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.4. Jerk units

The internal jerk units are internal position units / (slow loop sampling period)[3] i.e. the acceleration variation over one slow loop sampling period.

6.4.1. Brushless / DC brushed motor with quadrature encoder on motor

The internal jerk units are encoder counts / (slow loop sampling period)[3]. The correspondence with the load **jerk in SI units**[1] is:

$$Load\_Jerk[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times Tr \times T^3} \times Motor\_Jerk[IU]$$

where:

No_encoder_lines – is the rotary encoder number of lines per revolution

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.4.2. Brushless motor with linear Hall signals

The internal jerk units are counts / (slow loop sampling period)[3]. The motor is rotary. The position resolution i.e. number of counts per revolution is programmable as a power of 2 between 512 and 8192. By default it is set at 2048 counts per turn. The correspondence with the load acceleration in SI units is:

For rotary motors: $Load\_Jerk[SI] = \dfrac{2 \times \pi}{resolution \times Tr \times T^3} \times Motor\_Jerk[IU]$

For linear motors: $Load\_Jerk[SI] = \dfrac{Pole\_Pitch}{resolution \times Tr \times T^3} \times Motor\_Jerk[IU]$

where:

resolution – is the motor position resolution

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

Pole_Pitch – is the magnetic pole pitch NN (distance expressed in [m])

---

[1] SI units for jerk are [rad/s[3]] for a rotary movement, [m/s[3]] for a linear movement

6.4.3. DC brushed motor with quadrature encoder on load and tacho on motor

The internal jerk units are encoder counts / (slow loop sampling period)[3]. The motor is rotary and the transmission is rotary-to-rotary. The correspondence with the load jerk in SI units is:

$$Load\_Jerk[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times T^3} \times Load\_Jerk[IU]$$

where:

No_encoder_lines – is the encoder number of lines per revolution

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.4.4. Step motor open-loop control. No feedback device

The internal jerk units are motor µsteps / (slow loop sampling period)[3]. The correspondence with the load **jerk in SI units**[1] is:

$$Load\_Jerk[SI] = \frac{2 \times \pi}{No\_\mu steps \times No\_steps \times Tr \times T^3} \times Motor\_Jerk[IU]$$

where:

No_steps – is the number of motor steps per revolution

No_µsteps – is the number of microsteps per step. You can read/change this value in the "Drive Setup" dialogue from EasySetUp.

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.4.5. Step motor open-loop control. Incremental encoder on load

The internal jerk units are load encoder counts / (slow loop sampling period)[3]. The transmission is rotary-to-rotary. The correspondence with the load jerk in SI units is:

$$Load\_Jerk[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times T^3} \times Load\_Jerk[IU]$$

where:

No_encoder_lines – is the rotary encoder number of lines per revolution

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup".

6.4.6. Step motor closed-loop control. Incremental encoder on motor

---

[1] SI units for jerk are [rad/s[3]] for a rotary movement, [m/s[3]] for a linear movement

The internal jerk units are motor encoder counts / (slow loop sampling period)[3]. The correspondence with the load jerk in SI units is:

$$Load\_Jerk[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times Tr \times T^3} \times Motor\_Jerk[IU]$$

where:

No_encoder_lines – is the motor encoder number of lines per revolution

Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup".

6.4.7. Brushless motor with sine/cosine encoder on motor

The internal jerk units are interpolated encoder counts / (slow loop sampling period)[3]. The correspondence with the load jerk in SI units is:

For rotary motors: $Load\_Jerk[SI] = \dfrac{2 \times \pi}{4 \times Enc\_periods \times Interpolation \times Tr \times T^3} \times Motor\_Jerk[IU]$

For linear motors: $Load\_Jerk[SI] = \dfrac{Encoder\_accuracy}{Interpolation \times Tr \times T^3} \times Motor\_Jerk[IU]$

where:
Enc_periods – is the rotary encoder number of sine/cosine periods or lines per revolution
Encoder_accuracy – is the linear encoder accuracy in [m] for one sine/cosine period
Interpolation – is the interpolation level inside an encoder period. Its a number power of 2 between 1 an 256. 1 means no interpolation
Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units
T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.5. Current units

The internal current units refer to the motor phase currents. The correspondence with the motor currents in [A] is:

$$Current[A] = \frac{2 \times Ipeak}{65520} \times Current[IU]$$

where   Ipeak – is the drive peak current expressed in [A]. You can read this value in the "Drive Info" dialogue, which can be opened from the "Drive Setup".

6.6. Voltage command units

The internal voltage command units refer to the voltages applied on the motor. The significance of the voltage commands as well as the scaling factors, depend on the motor type and control method used.

In case of **brushless motors** driven in **sinusoidal** mode, a field oriented vector control is performed. The voltage command is the amplitude of the sinusoidal phase voltages. In this case, the correspondence with the motor phase voltages in SI units i.e. [V] is:

$$\text{Voltage command[V]} = \frac{1.1 \times \text{Vdc}}{65534} \times \text{Voltage command[IU]}$$

where   Vdc – is the drive power supply voltage expressed in [V].

In case of **brushless** motors driven in **trapezoidal** mode, the voltage command is the voltage to apply between 2 of the motor phases, according with Hall signals values. In this case, the correspondence with the voltage applied in SI units i.e. [V] is:

$$\text{Voltage command[V]} = \frac{\text{Vdc}}{32767} \times \text{Voltage command[IU]}$$

This correspondence is akso available for **DC brushed** motors which have the voltage command internal units as the brushless motors driven in trapezoidal mode.

6.7. Voltage measurement units

The internal voltage measurement units refer to the drive V_{MOT} supply voltage. The correspondence with the supply voltage in [V] is:

$$\text{Voltage\_measured[V]} = \frac{\text{VdcMaxMeasurable}}{65520} \times \text{Voltage\_measured[IU]}$$

where   VdcMaxMeasurable – is the maximum measurable DC voltage expressed in [V]. You can read this value in the "Drive Info" dialogue, which can be opened from the "Drive Setup".

Remark: the voltage measurement units occur in the scaling of the over voltage and under voltage protections and the supply voltage measurement

6.8. Time units

The internal time units are expressed in slow loop sampling periods. The correspondence with the time in [s] is:

$$\text{Time[s]} = \text{T} \times \text{Time[IU]}$$

where   T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup". For example, if T = 1ms, one second = 1000 IU.

6.9. Master position units

When the master position is sent via a communication channel, the master position units depend on the type of position sensor present on the master axis.

6.10. Master speed units

The master speed is computed in internal units (IU) as master position units / slow loop sampling period i.e. the master position variation over one position/speed loop sampling period.

6.11. Motor position units

6.11.1. Brushless / DC brushed motor with quadrature encoder on motor

The internal motor position units are encoder counts. The correspondence with the motor **position in SI units**[1] is:

$$Motor\_Position[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines} \times Motor\_Position[IU]$$

where:

No_encoder_lines – is the rotary encoder number of lines per revolution

6.11.2. Brushless motor with linear Hall signals

The internal motor position units are counts. The motor is rotary. The resolution i.e. number of counts per revolution is programmable as a power of 2 between 512 and 8192. By default it is set at 2048 counts per turn. The correspondence with the motor position in SI units is:

For rotary motors:
$$Motor\_Position[SI] = \frac{2 \times \pi}{resolution} \times Motor\_Position[IU]$$

For linear motors:
$$Motor\_Position[SI] = \frac{Pole\_Pitch}{resolution} \times Motor\_Position[IU]$$

where:

resolution – is the motor position resolution

Pole_Pitch – is the magnetic pole pitch NN (distance expressed in [m])

6.11.3. DC brushed motor with quadrature encoder on load and tacho on motor

The motor position is not computed.

6.11.4. Step motor open-loop control. No feedback device

The internal motor position units are motor μsteps. The correspondence with the motor **position in SI units**[1] is:

$$Motor\_Position[SI] = \frac{2 \times \pi}{No\_\mu steps \times No\_steps} \times Motor\_Position[IU]$$

where:

No_steps – is the number of motor steps per revolution

No_μsteps – is the number of microsteps per step. You can read/change this value in the "Drive Setup" dialogue from EasySetUp.

---

[1]SI units for motor position are: [rad] for a rotary motor, [m] for a linear motor

6.11.5. Step motor open-loop control. Incremental encoder on load

In open-loop control configurations with incremental encoder on load, the motor position is not computed.

6.11.6. Step motor closed-loop control. Incremental encoder on motor

The internal motor position units are motor encoder counts. The correspondence with the motor position in SI units is:

$$Motor\_Position[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines} \times Motor\_Position[IU]$$

where:

No_encoder_lines – is the motor encoder number of lines per revolution

6.11.7. Brushless motor with sine/cosine encoder on motor

The internal motor position units are interpolated encoder counts. The correspondence with the motor position in SI units is:

For rotary motors:

$$Motor\_Position[SI] = \frac{2 \times \pi}{4 \times Enc\_periods \times Interpolation} \times Motor\_Position[IU]$$

For linear motors:

$$Motor\_Position[SI] = \frac{Encoder\_accuracy}{Interpolation} \times Motor\_Position[IU]$$

where:
Enc_periods – is the rotary encoder number of sine/cosine periods or lines per revolution
Interpolation – is the interpolation level inside an encoder period. Its a number power of 2 between 1
an 256. 1 means no interpolation
Encoder_accuracy – is the linear encoder accuracy in [m] for one sine/cosine period

6.12. Motor speed units

6.12.1. Brushless / DC brushed motor with quadrature encoder on motor

The internal motor speed units are encoder counts / (slow loop sampling period). The correspondence with the motor **speed in SI units** is:

For rotary motors: $$Motor\_Speed[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times T} \times Motor\_Speed[IU]$$

where:

No_encoder_lines – is the rotary encoder number of lines per revolution

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.12.2. Brushless motor with linear Hall signals

The internal motor speed units are counts / (slow loop sampling period). The motor is rotary. The position resolution i.e. number of counts per revolution is programmable as a power of 2 between 512 and 8192. By default it is set at 2048 counts per turn. The correspondence with the motor speed in SI units is:

For rotary motors:
$$Motor\_Speed[SI] = \frac{2 \times \pi}{resolution \times T} \times Motor\_Speed[IU]$$

For linear motors:
$$Motor\_Speed[SI] = \frac{Pole\_Pitch}{resolution \times T} \times Motor\_Speed[IU]$$

where:

resolution – is the motor position resolution

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

Pole_Pitch – is the magnetic pole pitch NN (distance expressed in [m])

6.12.3. DC brushed motor with quadrature encoder on load and tacho on motor

The internal motor speed units are A/D converter bits. The correspondence with the motor **speed in SI units**[1] is:

$$Motor\_Speed[SI] = \frac{Analogue\_Input\_Range}{4096 \times Tacho\_gain} \times Motor\_Speed[IU]$$

where:

Analogue_Input_Range – is the range of the drive analogue input for feedback, expressed in [V]. You can read this value in the "Drive Info" dialogue, which can be opened from the "Drive Setup"

Tacho_gain – is the tachometer gain expressed in [V/rad/s]

6.12.4. DC brushed motor with tacho on motor

The internal motor speed units are A/D converter bits. The correspondence with the motor speed in SI units is:

$$Motor\_Speed[SI] = \frac{Analogue\_Input\_Range}{4096 \times Tacho\_gain} \times Motor\_Speed[IU]$$

where:

Analogue_Input_Range – is the range of the drive analogue input for feedback, expressed in [V]. You can read this value in the "Drive Info" dialogue, which can be opened from the "Drive Setup"

Tacho_gain – is the tachometer gain expressed in [V/rad/s]

6.12.5. Step motor open-loop control. No feedback device or incremental encoder on load

---

[1] SI units for motor speed are [rad/s] for a rotary motor, [m/s] for a linear motor

The internal motor speed units are motor μsteps / (slow loop sampling period). The correspondence with the motor **speed in SI units** is:

$$Motor\_Speed[SI] = \frac{2 \times \pi}{No\_\mu steps \times No\_steps \times T} \times Motor\_Speed[IU]$$

where:

No_steps – is the number of motor steps per revolution

No_μsteps – is the number of microsteps per step. You can read/change this value in the "Drive Setup" dialogue from EasySetUp.

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"

6.12.6. **Step motor closed-loop control. Incremental encoder on motor**

The internal motor speed units are motor encoder counts / (slow loop sampling period). The correspondence with the load speed in SI units is:

$$Motor\_Speed[SI] = \frac{2 \times \pi}{4 \times No\_encoder\_lines \times T} \times Motor\_Speed[IU]$$

where:

No_encoder_lines – is the motor encoder number of lines per revolution

T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup".

6.12.7. **Brushless motor with sine/cosine encoder on motor**

The internal motor speed units are interpolated encoder counts / (slow loop sampling period). The correspondence with the motor speed in SI units is:

For rotary motors:

$$Motor\_Speed[SI] = \frac{2 \times \pi}{4 \times Enc\_periods \times Interpolation \times T} \times Motor\_Speed[IU]$$

For linear motors:

$$Motor\_Speed[SI] = \frac{Encoder\_accuracy}{Interpolation \times T} \times Motor\_Speed[IU]$$

where:
Enc_periods – is the rotary encoder number of sine/cosine periods or lines per revolution
Encoder_accuracy – is the linear encoder accuracy in [m] for one sine/cosine period
Interpolation – is the interpolation level inside an encoder period. Its a number power of 2 between 1 an 256. 1 means no interpolation
Tr – transmission ratio between the motor displacement in SI units and load displacement in SI units
T – is the slow loop sampling period expressed in [s]. You can read this value in the "Advanced" dialogue, which can be opened from the "Drive Setup"
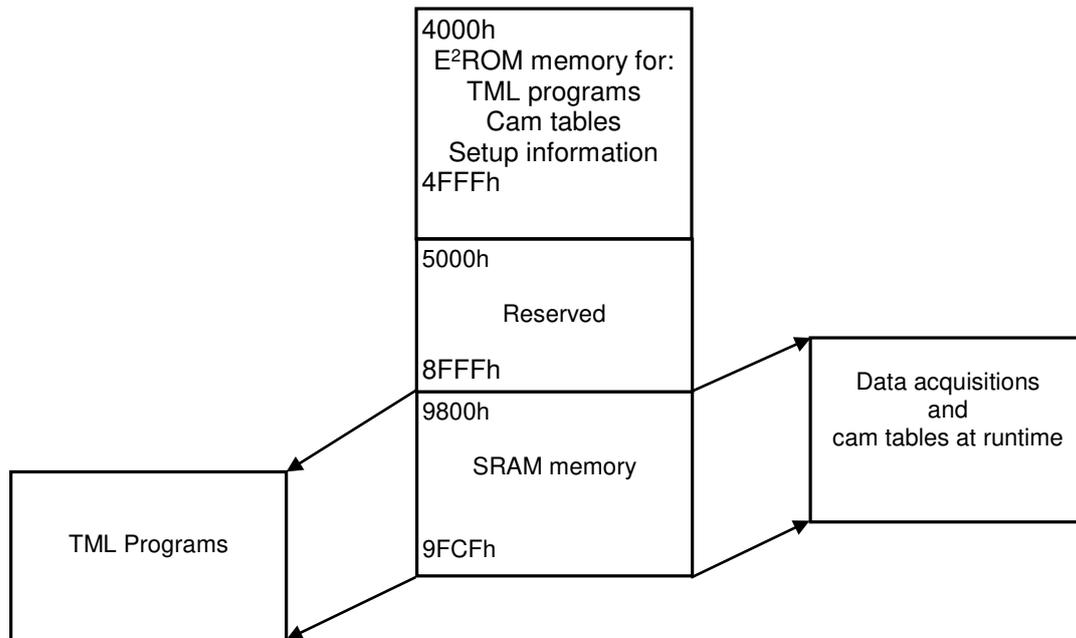
7. Memory Map

iPOS360x HX-CAN has 2 types of memory available for user applications: 2K$\times$16 SRAM and 4K$\times$16 serial E^{2}ROM.

The SRAM memory is mapped in the address range: 9800h to 9FFFh. It can be used to download and run a TML program, to save real-time data acquisitions and to keep the cam tables during run-time.

The E^{2}ROM is mapped in the address range: 4000h to 4FFFh. It is used to keep in a non-volatile memory the TML programs, the cam tables and the drive setup information.

Remark: EasyMotion Studio handles automatically the memory allocation for each motion application. The memory map can be accessed and modified from the main folder of each application

Figure 7.1. *iPOS360x HX-CAN Memory Map*