CNi engineering	**User Manual** **SM137 & SM140 motors**	**H5834D0007ING** **03** **11/05/2007**

Serial Communications Protocol

Publication information

Update List			
Revision	**Added**	**Deleted**	**Changed**
00	First Version		
01			Sect. 3.10
02	General Revision		Sect. 4.4 - 4.5 e 4.15
03	General Revision		

Manual written by the CNI Engineering Technical Publications Office
H5834D0007ING-03_Cop.fm

PUBLICATION ISSUED BY:

Documentation office

Via Carpanelli, 24
40011 Anzola dell'Emilia (Bo) Italy
Tel. +39 051 6508911
Fax +39 051 6508912
Info@cnicnc.com
www.cnicnc.com

Registered offices

Via dell'Artigianato, 1
48011 Alfonsine (Ra) Italy
Tel. +39 0544 84277
Fax +39 0544 80635
P.I. e C.F. 02248390391

Document Code: **H5834D0007ING**

Document revision: **03**

Document edition: **11/05/2007**

Manual written by the CNI Engineering Technical Publications Office

H5834D0007ING-03_Col.fm

INDEX

1 Installation

Make the settings specified in the relevant User Manual – Technical Specifications and Connections to ensure correct installation of SM137 and SM140 motors.

2 Serial communication

The minimum size for a transmittable data packet is 10 bits. These comprise:

- 1 start bit
- 8 data bits
- 1 stop bit

Start	Datum								Stop
0	Lsb							Msb	1

Packet format is as follows:

1 start bit, 8 data bit, no parity, 1 stop bit

2.1 Control characters

SM137 and SM140 motors send Commands and data requests in packets made up of a number of characters. Each packet starts with the control character STX=0x02 (Start of Transmission) and ends with the control character ETX=0x1B (End of Transmission). The control characters STX, ETX and ESC must be converted into sequences of two characters before they can be transmitted as data. These sequences are listed in the following table:

Character to transmit		Characters transmitted		
STX	(0x02)	ESC	(0x1B)	0xFF ⊕ STX (0xFD)
ETX	(0x03)	ESC	(0x1B)	0xFF ⊕ ETX (0xFC)
ESC	(0x1B)	ESC	(0x1B)	0xFF ⊕ ESC (0xE4

The symbol ⊕ represents the logic operation XOR.

2.2 Packet checksum (8 bit CRC)

Each packet also contains a checksum character (CRC), transmitted before the ETX control character.

STX	Datum1	…	DatumN	CRC	ETX

The CRC character is calculated as follows: :

$$CRC = 0xFF \oplus Datum1 \oplus Datum2 \oplus L \oplus DatumN$$

The ⊕ symbol represents the logic operation XOR.

N.B.: The CRC checksum character is calculated on the basis of the Command's data bytes before any STX, ETX, or ESC characters contained in the Command are converted. This is because the packet control characters must also mask the CRC.

2.3 Packet length

The maximum number of data bytes that can be transmitted is 68. This number does not include the control characters in the packet.

2.4 Node address

The second byte of each Command identifies the target node address. The node address is set on the motor's configuration DIP-switch.

2.5 Communication timing

LCommunications with the motor take place over a 2-wire serial line. This type of connection can be used to connect more than one device. Only one connected device can transmit at a time, but all other devices can receive simultaneously.
Devices switch the communication line to a low impedance state to transmit, and switch it to a high impedance state to receive.
The communications architecture is hierarchic. This means that on any one line there must be one master device (such as a numeric control unit, personal computer, etc.), and one or more slave devices (the motors).
The master device determines which slave device it wishes to dialogue with.
Figure 1 illustrates the timing with which the master and slave devices switch the line between high and low impedance states in order to receive and transmit.

Figure 1 Timing of serial line high/low impedance switching

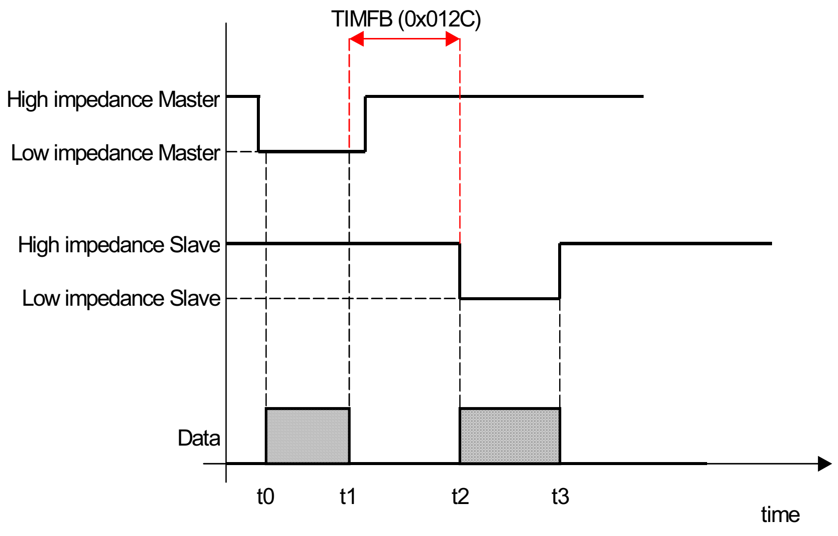


Figura 1 shows that:

- the interval (t2-t1) must be greater than or equal to the time set in the `TIMFB` (`0x12C`) parameter
- if the master device does not recommence transmission within a certain time (`t3 + TIMEOUTFB (0x12D)`), the slave device enters alarm state unless the value of the `TIMEOUTFB` parameter has been set to 0
- the LED of the motor engaged in the communication remains lit from t0 to t3

3 Commands

Packets sent in response to commands always contain 4 bits called Status Bits. Appendix 11 lists the meanings of these bits.

3.1 Firmware versions

Many of the commands described in the following sections are specific to certain firmware versions or motor revisions. To distinguish a revision C SM137 motor from a revision B SM137 motor, check:

- the order code
- the firmware version: if this is lower than 110, the motor is revision B; if it is 110 or higher, it is a revision C motor.

SM140 motors have only been sold with firmware versions of 110 or higher.

3.2 Reset alarm

N.B.: This command can only be used on SM140 motors and on SM137 motors from revision C on. With SM137 motors prior to revision C, use the No Regulation command instead (Section 3.3).

This command causes the motor to exit an alarm state. .

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdReset (0x9c)	CmdReset (0x9c)	
05	0x00	0x00	
06	0x00	0x00	
07	0x00	0x00	
08	CRC	CRC	
09x	ETX (0x03)	ETX (0x03)	

3.3 No Regulation

This command causes the motor to exit regulation mode. When not in regulation mode, the motor makes no attempt to maintain the position set in the command.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdNoReg (0x20)	CmdNoReg (0x20)	
05	0x00	0x00	
06	0x00	0x00	
07	0x00	0x00	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions:

- the command has no effect when the motor is:
  - performing an electrical reset
  - calibrating the current sensor offsets
- the command is not accepted if:
  - bit6 of parameter `BIT_A` (see Appendix 7) is set to 1 and the motor is in alarm state

3.4 Regulation

N.B. This command can only be used on SM140 motors and on SM137 motors from revision C on. With SM137 motors prior to revision C, use the Regulation with wait command instead (Section 5.1).

This command causes the motor to enter regulation mode. When in regulation mode, the motor attempts to maintain the position set in the command, resisting external loads.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdReg (0xcc)	CmdReg (0xcc)	
05	0x00	0x00	
06	0x00	0x00	
07	0x00	0x00	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions :

The command only has any effect if the motor is not in regulation mode.

Notes

The motor actually enters regulation mode only after the MASKBITCOMANDO_SMMASKBITCOMANDO_SM (see Appendix 9) assumes the value 1.

- if the motor is not in regulation mode, the MASKBITCOMANDO_SM bit will always assume the value 0 in any response to this command

3.5 Hold

N.B.: This command can only be used on SM140 motors and on SM137 motors from revision C on. With SM137 motors prior to revision C, use the Hold with wait command instead (Section 5.2).

This command stops the motor, halting the current movement according to a suitable deceleration ramp.

N.B.: The motor is only declared stopped (AXSTOP state) when its theoretical speed reaches 0 and its real speed is below 12 rpm absolute.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x08	Status Bit
04	CmdHold (0xbc)	CmdHold (0xbc)	
05	0x00	0x00	
06	0x00	0x00	
07	0x00	0x00	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions :

The command only has any effect if the motor is performing a movement.

Notes

- the motor is only really stopped when the MASKBITCOMANDO_SM bit assumes the value 1 (see Appendix 9)
- the MASKBITCOMANDO_SM bit will always assume the value 0 in any response to this command before the motor stops.

3.6 Manual position assign (Manual reset)

This command assigns a specific value Q to the motor position. This procedure is frequently referred to as a “manual reset”.
On execution of the command, the motor assumes the position Q.

Manual resets do not wait for the motor to reach the encoder’s zero notch, but have immediate effect.

The motor’s reset state becomes 1 (AZZMAN parameter) after a manual reset.

Reset value Q is a 32 bit number with a sign, made up as follows:

bit 31-24	bit 23-16	bit 15-8	bit 7-0
Q3	Q2	Q1	Q0

<table>
<tr><th>Byte</th><th>Command</th><th colspan="2">Response</th></tr>
<tr><td>01</td><td>STX (0x02)</td><td colspan="2">STX (0x02)</td></tr>
<tr><td>02</td><td>Node</td><td colspan="2">Node</td></tr>
<tr><td>03</td><td>CmdMazz (0x01)</td><td>0x0</td><td>Status Bit</td></tr>
<tr><td>04</td><td>Q1</td><td colspan="2">CmdMazz (0x01)</td></tr>
<tr><td>05</td><td>Q0</td><td colspan="2">Q0</td></tr>
<tr><td>06</td><td>Q3</td><td colspan="2">Q3</td></tr>
<tr><td>07</td><td>Q2</td><td colspan="2">Q2</td></tr>
<tr><td>08</td><td>CRC</td><td colspan="2">CRC</td></tr>
<tr><td>09</td><td>ETX (0x03)</td><td colspan="2">ETX (0x03)</td></tr>
</table>

Conditions :

The command only has any effect if the motor is in alarm state, whether it is in regulation mode or not.

Reset position calculation

Reset position Q must be expressed as an encoder count as shown in the following example.

Example: To calculate components Q0, Q1, Q2, and Q3 if:

- reset position Q is 6766.8176 mm
- the motor’s mechanical step is 0.32 mm/rev
- the encoder step is 800 counts/rev

$$Q = 6780,8465[mm] = \frac{6780,8465}{0,32} \times 800 = 16952116,25[cnt] = 0 \times 0102AB34[cnt]$$

Q3	Q2	Q1	Q0
0x01	0x02	0xAB	0x34

3.7 Automatic position assign (Automatic reset)

This command assigns the value contained in the ORIG_AZZ parameter as motor position. This procedure is frequently referred to as an “automatic reset”. The value contained in the ORIG_AZZ parameter becomes the current motor position as soon as the encoder’s zero notch is encountered.
An automatic reset can be performed in various ways depending on the value contained in the TIPOAZZ parameter.

TIPOAZZ = 0

The motor must be performing a manual movement (jog) to be able to perform this type of automatic reset.
The reset position is assigned to the motor’s actual position the first time the motor encounters the encoder’s zero notch after the automatic reset command has been received.
Before the motor encounters the encoder’s zero notch, reset status becomes SEARCHINGTACCA (0x0006) and the MASKBITCOMANDO_SM status bit assumes the value 0.

After the motor encounters the encoder’s zero notch, reset status becomes AZZAUTO (0x0003) and the MASKBITCOMANDO_SM status bit assumes the value 1.

TIPOAZZ = 1

IMPORTANT! This type of automatic reset can only be performed with SM140 motors.

The motor must be in regulation mode (AXSTOP) to perform this type of automatic reset, which only uses the reset microswitch.
The reset is performed in the following phases:

- the moment the motor receives the reset command, it starts to seek the reset microswitch at the speed defined in the VMAXAZZ parameter. The sign in the parameter determines the direction of motor movement.

  Reset status becomes SEARCHINGMICRO (code 0x0002) and the MASKBITCOMANDO_SM status bit assumes the value 0.

- the moment the motor reaches and passes the microswitch, the logic level of the third input goes high. The motor then stops and reverses at the speed defined in VAZZOUTMIC.

  Reset status becomes LEAVINGMICRO (code 0x0004).

- the moment the motor leaves the microswitch, the logic level of the third input goes low. The reset position is now assigned and the motor terminates the reset procedure.

  Reset state becomes AZZAUTO (code 0x0003) and the ASKBITCOMMAND_SM status bit assumes the value 1.

TIPOAZZ = 2

IMPORTANT! This type of automatic reset can only be performed with SM140 motors

The motor must be in regulation mode (AXSTOP) to perform this type of automatic reset, which uses the reset microswitch and the encoder’s zero notch.

The reset is performed in the following phases:

- the moment the motor receives the reset command, it starts to seek the reset microswitch at the speed defined in the VMAXAZZ parameter. The sign in the parameter determines the direction of motor movement.

  Reset status becomes SEARCHINGMICRO (code 0x0002) and the MASKBITCOMANDO_SM status bit assumes the value 0.

- the moment the motor reaches and passes the microswitch, the logic level of the third input goes high. The motor then stops and reverses at the speed defined in VAZZOUTMIC

  Reset status becomes LEAVINGMICRO (code 0x0004).

- the moment the motor leaves the microswitch, the logic level of the third input goes low. The motor now starts seeking the encoder's zero notch, at the same speed.

  Reset status becomes SEARCHINGTACCA (code 0x0006).

- the moment the motor encounters the encoder's zero notch, the reset position is assigned and the motor terminates the reset procedure.

  Reset state becomes AZZAUTO (code 0x0003) and the MASKBITCOMANDO_SM status bit assumes the value 1.

This type of automatic reset is the only one for which the CmdGetDistMicroZero command has any effect. This command returns the distance between the reset microswitch and the motor encoder's zero notch, as detected during the last reset performed.

If an automatic reset is interrupted, reset state becomes NOAZZ (code 0x0000).

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdAzz (0x38)	CmdAzz (0x38)	
05	0x00	0x00	
06	0x00	0x00	
07	0x00	0x00	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

When TIPOAZZ=0, this command only has any effect if the motor is performing a manual movement (jog). For all other types of automatic reset, the command only has any effect provided the motor is in AXSTOP state (regulation mode).

Notes

Response to this command is immediate. The MASKBITCOMANDO_SM (Appendix C) status bit must be monitored to ascertain when the motor has completed the automatic reset, stopped and re-entered regulation mode.
Once the command has been sent, the MASKBITCOMANDO_SM status bit will always be at 0 in responses until the motor completes the reset.
See Appendix 9 for details of the various possible reset states.

3.8 Manual movement at specified speed (Jog)

N.B.: This command can only be used on SM140 motors and on SM137 motors from revision C on.

This command performs a manual movement at a specified speed V. This procedure is frequently referred to as a ‘jog’.
Speed value V is a 16 bit number with a sign, composed as follows:

bit 15-8	bit 7-0
VH	VL

The sign of V determines the direction of motor rotation:

- + determines incremental rotation
- - determines decremental rotation.

The unit of measure for speed V is rpm.

I

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdJogN (0xe0)	CmdJogN (0xe0)	
05	0x00	0x00	
06	VH	VH	
07	VL	VL	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command only has any effect if the motor is in regulation mode or is performing another manual movement at specified speed (jog).

Example: To calculate the values VH and VL needed to perform a motor movement at a speed of 185 mm/min, if the motor has a mechanical step of 0.32 mm/rev.

$$(\angle 185)\left((mm)/(min) \Leftrightarrow \frac{\angle 185}{0,32}\right) \angle 578[rpm] = 0 \times FDBE[rpm]$$

VH	VL
0xFD	0xBE

3.9 Move to specified position (Line)

This command performs a movement to a specified position `Q`, expressed as an encoder count. Position `Q` is a 32 bit number composed as follows:

bit 31-24	bit 23-16	bit 15-8	bit 7-0
Q3	Q2	Q1	Q0

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	CmdTraj (0x02)	0x0	Status Bit
04	Q1	CmdTraj (0x02)	
05	Q0	Q0	
06	Q3	Q3	
07	Q2	Q2	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

- The command only has any effect if the motor is in regulation mode and has been manually or automatically reset
- The variation in value that can be set using the CmdTraj command is `0x3FFFFFF=67108863` encoder counts maximum, in absolute value. If the variation in value is greater, a warning message `0x400B` (`ALMOVTOOLONG`) is given; for further details see the table on page 64.
  Example: if the current value is 1000, the valued movement must be between -67107863 and 67109863

Notes

- response to this command is immediate. The `MASKBITCOMANDO_SM` status bit must be monitored to ascertain when the motor has completed the manual positioning and re-entered regulation mode.
- once the command has been sent, the `MASKBITCOMANDO_SM` status bit will always assume a value of 0 in responses until the motor reaches the target position.
- the motor is assumed to have reached the target position once the real position coincides with the theoretical target position with the precision determined by the relevant parameter settings.

Target position calculations

Position `Q` must be expressed as an encoder count as shown in the following example.

Example: To calculate components Q0, Q1, Q2, and Q3 if position Q is 1150.75 mm, the motor has a mechanical step of 0.32 mm/rev, and the encoder step is 500 counts/rev. .

$$Q = 1150,75[mm] = \frac{1150,75}{0,32} \times 500 = 1798046,875[cnt] = 0 \times 01B6F9E[cnt]$$

Q3	Q2	Q1	Q0
0x01	0x1B	0x6F	0x9E

3.10 Move to specified position at specified speed (Linevel)

N.B.: This command can only be used on SM140 motors and on SM137 motors from revision C on.

This command performs a movement to a specified position `Q`, expressed as an encoder count, at a positive speed V, expressed in rpm.

Position `Q` is a 32 bit number composed as follows:

bit 31-24	bit 23-16	bit 15-8	bit 7-0
Q3	Q2	Q1	Q0

Speed V is a 16 bit number composed as follows:

bit 15-8	bit 7-0
VH	VL

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x00	0x0	Status Bit
04	CmdTrajVel(0xc8)	CmdTrajVel(0xc8)	
05	0x00		
06	Q1	Q1	
07	Q0	Q0	
08	Q3	0x00	
09	Q2	CRC	
10	VH	ETX	
11	VL		
12	CRC		
13	ETX (0x03)		

Conditions

- the command only has any effect provided the motor is in regulation mode or already performing a movement to a specified position at a specified speed
- If there are any kinematic limitations preventing a certain movement from being performed (e.g. the target position is too near the current position for an adequate deceleration ramp to be implemented), the motor returns the `CmdNACK` response

- the motor cannot reverse direction with respect to the previous movement. For example, if the motor is at a position Qi=0.0 mm and movement to a target position Qf0=500.0 mm is commanded, once the motor reaches the intermediate position Qt=400.0 mm , it is not possible to command a new movement to a new target position Qf1=100.0 mm since this would mean reversing the direction of motor rotation. If such a command were issued, the motor would return the `CmdNACK` response
- the motor must be manually or automatically reset before the `CmdTrajVel` command can be used
- The variation in value that can be set using the CmdTraj command is `0x3FFFFFF=67108863` encoder counts maximum, in absolute value. If the variation in value is greater, a `WARNING` message `0x400B` (`ALMOVTOOLONG`) is given; for further details see the table on page 64.
Example: if the current value is 1000, the valued movement must be between -67107863 and 67109863

Notes

- response to this command is immediate. The `MASKBITCOMANDO_SM` status bit must be monitored to ascertain when the motor has completed the manual positioning and re-entered regulation mode
- once the command has been sent, the `MASKBITCOMANDO_SM` status bit will always assume a value of 0 in responses until the motor reaches the target position
- the motor is assumed to have reached the target position once the real position coincides with the theoretical target position with the precision determined by the relevant parameter settings

Target position and speed calculations

The following examples illustrate how to calculate the target position as an encoder count and positioning speed in rpm.

Example: To calculate components Q0, Q1, Q2, and Q3, if:

- target position Q is 1150.75 mm
- the motor’s mechanical step is 0.32 mm/rev
- the encoder step is 500 counts/rev

$$Q = 1150,75[mm] = \frac{1150{,}75}{0,32} \times 500 = 1798046,875[cnt] = 0 \times 01B6F9E[cnt]$$

Q3	Q2	Q1	Q0
0x01	0x1B	0x6F	0x9E

Example: To calculate VH and VL for a speed of 150 mm/min, if the motor has a mechanical step of 0.32 mm/rev.

$$150[mm/mn] \Leftrightarrow \frac{150}{0,32} = 468,75(rpm) = 0 \times 01D4[rpm]$$

VH	VL
0x01	0xD4

3.11 Request motor position, speed and torque

This command interrogates the motor for its:

- real position
- theoretical position
- real and theoretical speed
- real and theoretical torque

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x00	0x0	Status Bit
04	0x00	D1	
05	CRC	D0	
06	ETX	D3	
07		D2	
08		CRC	
09		ETX (0x03)	

Conditions

The command is effective under all conditions.

Notes

- the interpretation of the data returned depends on what command was sent last, as shown in the following table.
- when the motor is powered on, the last command is assumed to be `CmdGetPos`.

Last Command	Description	Value	Unit of measure
CmdGetPos	The motor returns its real position	Qr = D3-D2-D1-D0	encoder count
CmdGetPosT	The motor returns its theoretical position	Qt = D3-D2-D1-D0	encoder count

CmdGetVel	The motor returns its theoretical and real speeds	Vr = D1-D0 Vt = D3-D2	Q15
CmdGetTor	The motor returns its theoretical and real torques	Tr = D1-D0 Tt = D3-D2	Q15

Example 1: If the last command was CmdGetPosT, the motor's mechanical step is 0.32 mm/rev, and the encoder's step is 800 counts/rev, and the motor returns the following theoretical position values,

D3	D2	D1	D0
0x09	0xC1	0x10	0xAB

then the theoretical position in mm is obtained as follows:

$$Q = 0 \times 09C110AB[cnt] = 163647659[cnt] = \frac{163647659}{800} \times 0,32[mm] = 65459,0636[mm]$$

Example 2: If the last command was CmdGetVel, the motor's mechanical step is 0.32 mm/rev, and the encoder's step is 800 counts/rev, and the motor returns the following theoretical and real speed values,

D3	D2	D1	D0
0x09	0xC1	0x10	0xAB

then the theoretical and real speeds are obtained as follows:

$$Vt = 0 \times 09C1[Q15] = 2497[Q15] = \frac{2497}{2^{15}} \times 8000 \times 0,32[mm/min] = 195,0781[mm/min]$$

$$Vt = 0 \times 10AB[Q15] = 4267[Q15] = \frac{4267}{2^{15}} \times 8000 \times 0,32[mm/min] = 333,3594[mm/min]$$

4 Advanced commands

4.1 Motor EMERGENCY

This command places the motor in alarm state. This state is similar to that triggered by the NO REGULATION COMMAND apart from the fact that the motor cannot return directly from this state to regulation mode.

The motor must be taken out of regulation mode first before it can be returned to regulation mode.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdEmerg (0x90)	CmdEmerg (0x90)	
05	0x00	0x00	
06	0x00	0x00	
07	0x00	0x00	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command is effective under all conditions.

4.2 Set OVERRIDE

N.B.: This command can only be used on SM140 motors and on SM137 motors from revision C on.

This command changes motor speed by a given percentage. The parameter transmitted is a percentage of motor speed between 0 and 200%.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdSetOverr (0xd0)	CmdSetOverr (0xd0)	
05	0x00	0x00	
06	OH	High part of override	
07	OL	Low part of override	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

- the command is effective under all conditions. If the motor is already performing a movement, it will ramp up or down to the new speed.

Example: To reduce motor speed by 10%, simply send the motor an override value of 90.

90 [%] 0x005A [%]

OH	OL
0x00	0x5A

4.3 Get OVERRIDE

N.B.: This command can only be used on SM140 motors and on SM137 motors from revision C on.

This command reads the motor's current override setting.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdGetOverr (0xd8)	CmdGetOverr (0xd8)	
05	0x00	0x00	
06	0x00	High part of override	
07	0x00	Low part of override	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command is effective under all conditions.

4.4 Change parameter

N.B.: This command can only be used on SM140 motors and on SM137 motors from revision C on.

This command modifies one of the motor's parameters. See Section 6 for further information on how modifiable parameters are encoded.

It is important to distinguish a 16 bit parameter from a 32 bit parameter.

Transmission of 16 bit parameters

`PS0` and `PS1` identify respectively the low and high part of the value to assign to 16 bit parameter PS, as shown in the following table.

bit 15-8	bit 7-0
PS 1	PS 0

The command takes the following form:

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdChgParN (0xb8)	CmdChgParN (0xb8)	
05	0x00	0x00	
06	High part of parameter code	High part of parameter code	
07	Low part of parameter code	Low part of parameter code	
08	High part of parameter value (PS_1)	CRC	
09	High part of parameter value (PS_0)	ETX (0x03)	
10	CRC		
11	ETX (0x03)		

Transmission of 32 bit parameters

`PL0`, `PL1`, `PL2` and `PL3` identify the bytes containing the value to assign to 32 bit parameter PL, as shown in the following table.

bit 31-24	bit 23-16	bit 15-8	bit 7-0
PL 3	PL 2	PL 1	PL 0

The command takes the following form.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdChgParN (0xb8)	CmdChgParN (0xb8)	
05	0x00	0x00	
06	High part of parameter code	High part of parameter value	
07	Low part of parameter code	Low part of parameter code	
08	PL_1	CRC	
09	PL_0	ETX (0x03)	
10	PL_3		
11	PL_2		
12	CRC		
13	ETX (0x03)		

Simultaneous setting of more than one parameter

The same command can be used to assign values to more than one parameter. For example, to change one 16 bit parameter and two 32 bit parameters, the command should be expressed as shown in the following table:

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	(0xb8)
04	CmdChgParN (0xb8)	CmdChgParN (0xb8)	
05	0x00	0x00	
06	High part of parameter code	High part of parameter code	
07	Low part of parameter code	Low part of parameter code	
08	High part of parameter value (PS_1)	CRC	
09	High part of parameter value (PS_0)	ETX (0x03)	
10	High part of parameter code		
11	Low part of parameter code		
12	$PL1_1$		
13	$PL1_0$		
14	$PL1_3$		
15	$PL1_2$		
16	High part of parameter code		
17	Low part of parameter code		
18	$PL2_1$		
19	$PL2_0$		
20	$PL2_3$		
21	$PL2_2$		
22	CRC		
23	ETX (0x03)		

Conditions

The command only has any effect if the motor is in alarm state, whether it is in regulation mode or not. The number of parameters that can be sent is limited by maximum packet length, which is 68 bytes.

4.5 Get parameter

N.B.: This command can only be used on SM140 motors and on SM137 motors from revision C on.

This command reads one of the motor's parameters.
See Section 6 for further information on how modifiable parameters are encoded.

It is important to distinguish a 16 bit parameter from a 32 bit parameter.

Reading 16 bit parameters

`PS0` and `PS1` identify respectively the low and high part of the value to be read from 16 bit parameter PS, as shown in the following table.

bit 15-8	bit 7-0
PS 1	PS 0

The command takes the following form:

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdGetParN (0xc0)	CmdGetParN (0xc0)	
05	0x00	0x00	
06	High part of parameter code	High part of parameter code	
07	Low part of parameter code	Low part of parameter code	
08	0x00	High part of parameter value (PS_1)	
09	0x00	High part of parameter value (PS_0)	
10	CRC	CRC	
11	ETX (0x03)	ETX (0x03)	

Reading 32 bit parameters

`PL0`, `PL1`, `PL2` and `PL3` identify the bytes containing the value to read from 32 bit parameter PL, as shown in the following table.

bit 31-24	bit 23-16	bit 15-8	bit 7-0
PL 3	PL 2	PL 1	PL 0

The command takes the following form:

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdGetParN (0xb8)	CmdGetParN (0xb8)	
05	Higt part of parameter code	Higt part of parameter code	
06	Low part of parameter code	Low part of parameter code	
	0x00	PL_1	
07	0x00	PL_0	
08	0x00	PL_3	
09	0x00	PL_2	
10	CRC	CRC	
11	ETX (0x03)	ETX (0x03)	

Conditions

The command is effective under all conditions.

4.6 Save parameters

N.B.: This command can only be used on SM140 motors and on SM137 motors from revision C on.

This command saves active motor parameters to flash memory.
When the motor is next powered up, it re-loads saved parameters from flash memory.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdSaveParFl (0xe8)	CmdSaveParFl (0xe8)	
05	0x00	0x00	
06	0x00	0x00	
07	0x00	0x00	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

N.B.: The flash memory can only be written to a limited number of times. This command should therefore not be used too often.

- the command only has any effect if the motor is in alarm state.

4.7 Calibrate current sensor offsets

This command calibrates the offsets for the current sensors. The motor performs this operation automatically on power-up.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdAdcOff (0x28)	CmdAdcOff (0x28)	
05	0x00	0x00	
06	0x00	0x00	
07	0x00	0x00	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions
The command only has any effect if the motor is in alarm state.

Notes
Response to this command is only given at the end of the write operation.

4.8 Electrical reset

N.B.: This command can only be used on SM140 motors and on SM137 motors from revision C on.

This command resets the electrical position of the motor’s rotor.
The motor performs this operation automatically the first time it enters regulation mode.
There are various types of electrical reset. We generally recommend use of type 0 since this is the only type that does not generate motor movement.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdAzzEl (0xc4)	CmdAzzEl (0xc4)	
05	0x00	0x00	
06	0x00	0x00	
07	Type of reset (0x00)	Type of reset (0x00)	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command is only effective if the motor is not in regulation mode.

Notes

- Response to this command is immediate

  The `MASKBITCOMANDO_SM` status bit must be monitored to ascertain when the motor has completed the electrical reset. `MASKBITCPMANDO_SM` only assumes the value 1 when the motor has completed the electrical reset.

4.9 Get reset state

N.B. This command can only be used on SM140 motors and on SM137 motors from firmware version 116 of revision C on.

This command reads the motor's reset state. The motor returns one of the values shown in the following table:

NOAZZ (0x0000):	the motor has not been reset
AZZMAN (0x0001):	the motor has been manually reset
AZZAUTO (0x0003):	the motor has been automatically reset
SEARCHINGMICRO (0x0002):	the motor is performing an automatic reset and is currently seeking the reset microswitch
LEAVINGMICRO (0x0004):	the motor is performing an automatic reset and is currently leaving the reset microswitch
SEARCHINGTACCA (0x0006):	the motor is performing an automatic reset and is currently seeking the encoder's zero notch

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdGetStatAzz (0x64)	CmdGetStatAzz (0x64)	
05	0x00	0x00	
06	0x00	High part of reset state	
07	0x00	Low part of reset state	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command is effective under all conditions.

4.10 Get distance between microswitch and encoder’s zero notch

N.B.: this command can only be used on SM140 motors.

This command reads the measured distance between the encoder’s zero notch and the reset microswitch. The distance, D, is expressed as an encoder count and is returned in 4 bytes:

bit 31-24	bit 23-16	bit 15-8	bit 7-0
D3	D 2	D 1	D 0

Byte	Command	Response
01	STX (0x02)	STX (0x02)
02	Node	Node
03	0x08	0x0 Status Bit
04	CmdGetDistMicroZero (0x5c)	CmdGetDistMicroZero (0x5c)
05	0x00	0x00
06	0x00	D1
07	0x00	D0
08	CRC	D3
09	ETX (0x03)	D2
10		CRC
11		ETX (0x03)

Conditions

- the command is effective under all conditions
- the only time there is any point in using this command is after a type 2 automatic reset (`TIPOAZZ=2`) has been performed to reset the motor using the microswitch and the zero notch

4.11 Get real position

This command forces the motor to return its real position in response to all null commands. The position is expressed as an encoder count.

Byte	Command	Response
01	STX (0x02)	STX (0x02)
02	Node	Node
03	0x08	0x0 Status Bit
04	CmdGetPos (0x68)	CmdGetPos (0x68)
05	0x00	0x00
06	0x00	0x00
07	0x00	0x00
08	CRC	CRC
09	ETX (0x03)	ETX (0x03)

Conditions

The command is effective under all conditions.

4.12 Get theoretical position

IThis command forces the motor to return its theoretical position in response to all null commands. The position is expressed as an encoder count.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdGetPosT (0x98)	CmdGetPosT (0x98)	
05	0x00	0x00	
06	0x00	0x00	
07	0x00	0x00	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

4.13 Get theoretical and real speed

This command forces the motor to return the theoretical and real speeds in response to all null commands. Speeds are expressed in [Q15] notation (see Appendix 10).

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdGetVel (0x70)	CmdGetVel (0x70)	
05	0x00	0x00	
06	0x00	0x00	
07	0x00	0x00	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command is effective under all conditions.

4.14 Get theoretical and real torque

This command forces the motor to return the theoretical and real torques in response to all null commands. Torques are expressed in [Q15] notation (see Appendix 10).

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdGetTor (0x78)	CmdGetTor (0x78)	
05	0x00	0x00	
06	0x00	0x00	
07	0x00	0x00	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command is effective under all conditions.

4.15 Get alarm or warning messages

This command reads any alarm or warning message present in the motor. See Appendix 10 for further information on how messages are encoded.

Once a message has been read using this command, it is deleted from the motor. .

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdGetAlarm (0x60)	CmdGetAlarm (0x60)	
05	0x00	0x00	
06	0x00	High part of message code	
07	0x00	Low part of message code	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command is effective under all conditions.

4.16 Get firmware version

This command reads the version of the firmware currently loaded in the motor

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdGetVer (0x80)	CmdGetVer (0x80)	
05	0x00	0x00	
06	0x00	High part of version	
07	0x00	Low part of version	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command is effective under all conditions.

Notes

The data returned in the `High part of version` and `Low part of version` bytes is in hexadecimal form and must be converted into decimal.

Example: If the `High part of version` and `Low part of version` bytes contain the data `0x00` and `0x67` respectively, the motor's firmware version is `0x0067=103`.

4.17 Get motor and field bus type

N.B.: This command can only be used on SM140 motors and on SM137 motors from revision C on.

This command reads the motor's communications protocol.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdGetType (0xa4)	CmdGetType (0xa4)	
05	0x00	0x00	
06	0x00	High part of protocol type	
07	0x00	Low part of protocol type	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command is effective under all conditions.

Response

Response to this command can be interpreted according to the following table:

Byte	Command	Communications protocol
0x00	SM137	ENET-X
0x01	SM137	RS-485
0x02	SM137	CAN
0x10	SM140	ENET-X
0x11	SM140	RS-485
0x12	SM140	CAN

4.18 Get motor's internal state

This command reads the motor's internal state.
See Appendix 8 for information on how internal states are encoded.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdGetSmStat (0xA8)	CmdGetSmStat (0xA8)	
05	0x00	0x00	
06	0x00	High part of state code	
07	0x00	Low part of state code	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command is effective under all conditions.

4.19 Sample variables

N.B.: This command can only be used on SM140 motors and on SM137 motors from firmware version 116 of revision C on.

This command enables sampling of two internal firmware variables.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdSampleVar (0xb4)	CmdSampleVar (0xb4)	
05	0x00	0x00	
06	Variable 1 code	Variable 1 code	
07	Variable 2 code	Variable 2 code 2	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command is effective under all conditions.
See the relevant appendices for information on how the variables are encoded.

4.20 Sample variables at specified frequency

N.B.: This command can only be used on SM140 motors and on SM137 motors from firmware version 116 of revision C on.

This command enables the sampling of two internal firmware variables at a frequency specified as a parameter.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdSampleMem (0xac)	CmdSampleMem (0xac)	
05	High part of var. 1 address	High part of var. 1 address	
06	Low part of var. 1 address	Low part of var. 1 address	
07	High part of var. 2 address	High part of var. 2 address	
08	Low part of var. 2 address	CRC	
09	High part of sampling frequency	ETX (0x03)	
10	Low part of sampling frequency		
11	CRC		
12	ETX (0x03)		

Conditions

The command is effective under all conditions.

4.21 Get sampled values

This command downloads sampled values from the motor.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdGetSamp (0xA0)	CmdGetSamp (0xA0)	
05	0x00	0x00	
06	0x00	High part of number of samples	
07	0x00	Low part of number of samples	
08	CRC	High part of sample 1 var. 1	
09	ETX (0x03)	Low part of sample 1 var. 1	
10		High part of sample 1 var. 2	
11		Low part of sample 1 var. 2	
12		High part of sample 2 var. 1	
13		Low part of sample 2 var. 1	
14		High part of sample 2 var. 2	
15		Low part of sample 2 var. 2	
...		...	
8+4 N		High part of sample N var. 1	
8+4 N+1		Low part of sample N var. 1	
8+4 N+2		High part of sample N var. 2	
8+4 N+3		Low part of sample N var. 2	
8+4 N+4		0xXX	
8+4 N+5		0xXX	
8+4 N+6		CRC	
8+4 N+7		ETX (0x03)	

Conditions

The command is effective under all conditions.
The maximum value of N is limited by the maximum length of the packet (see Section 2.3 above).

5 Obsolete commands

IMPORTANT! The commands listed below are obsolete. They are implemented on this motor only to ensure backwards compatibility with previous software versions

Their use is NOT recommended!

5.1 Regulation with wait

IThis command causes the motor to enter regulation mode.
In regulation mode, the motor attempts to maintain its position, resisting external loads.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdRegWait (0x18)	CmdRegWait (0x18)	
05	0x00	0x00	
06	0x00	0x00	
07	0x00	0x00	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

This command only has any effect if the motor is not in regulation mode.

Notes

The first time the motor enters regulation mode after power-up, response to this command is delayed by about 0.1 seconds. This is because the motor must perform an electrical reset of its rotor position before it can enter regulation mode.
To avoid this delay use the Regulation command described in Section 3.4 (code `0xcc`) instead.

5.2 Hold with wait

This command stops the motor, halting the current movement according to a suitable deceleration ramp.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdHoldWait (0x50)	CmdHoldWait (0x50)	
05	0x00	0x00	
06	0x00	0x00	
07	0x00	0x00	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command only has any effect if the motor is performing a movement.

Notes

The master only receives a response to this command after the motor's theoretical speed has reached 0. The actual delay depends on the speed of the motor at the time and on the Hold deceleration ramp set in the parameters. To avoid this delay, use the Hold (code `0xbc`) command described in Section 3.5.

5.3 Manual movement at specified speed (jog)

This command performs a manual movement at a specified speed V. This procedure is frequently referred to as a 'jog'.
Speed value V is a 16 bit number with a sign, composed as follows:

bit 15-8	bit 7-0
VH	VL

The sign of V determines the direction of motor rotation:

- + determines incremental rotation
- - determines decremental rotation

The unit of measure for speed V is expressed in [Q15] notation.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdJog (0x40)	CmdJog (0x40)	
05	0x00	0x00	
06	VH	VH	
07	VL	VL	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command only has any effect if the motor is in regulation mode or is performing another manual movement at specified speed (jog).

Notes

The speed to be entered in bytes 06 and 07 of the command must be expressed in [Q15] notation.

Example: To calculate the values VH and VL needed to perform a motor movement at a speed of –185 mm/min, if the motor has a mechanical step of 0.32 mm/rev and motor speed has a base value of 8000 rpm.

$$\angle 185[mm/min] \Leftrightarrow \frac{\angle 185}{0,32 \times 8000} \times 2^{15} = \angle 2368[Q15] = 0xF6C0[Q15]$$

VH	VL
0xF6	0xC0

To transmit a speed in rpm, use the Manual movement at specified speed command described in Section 3.8 instead.

5.4 Change parameter

This command modifies one of the motor's parameters.
It is obsolete and the Change parameter command described in Section 4.4 should be used instead.
See Section 6 for further information on how modifiable parameters are encoded. .

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	CmdChgPar (0x04)	0x0	Status Bit
04	High part of parameter code	CmdChgPar (0x40)	
05	Low part of parameter code	Low part of parameter code	
06	Parte alta valore parametro	High part of parameter value	
07	High part of parameter value	Low part of parameter value	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command only has any effect if the motor is in alarm state, whether it is in regulation mode or not.
To change a parameter using standard units of measure, use the Change parameter (code `0xb8`) command described in Section 4.4 instead.

5.5 Electrical reset with wait

This command resets the electrical position of the motor's rotor.

The motor performs this operation automatically the first time it enters regulation mode. There are various types of electrical reset. We generally recommend use of type 0.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdAzzElWait (0x30)	CmdAzzElWait (0x30)	
05	0x00	0x00	
06	0x00	0x00	
07	Type of reset (0x00)	Type of reset (0x00)	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command is only effective if the motor is not in regulation mode.

Notes

- The motor only sends a response to this command on completion of the electrical reset (after a delay of about 0.1 sec.)
- To avoid this delay, use the Electrical reset (code `0xc4`) command described in Section 4.8 instead

5.6 Sample variables

This command enables sampling of two internal firmware variables.

Byte	Command	Response	
01	STX (0x02)	STX (0x02)	
02	Node	Node	
03	0x08	0x0	Status Bit
04	CmdSample (0x88)	CmdSample (0x88)	
05	0x00	0x00	
06	Variable 1 code	Variable 1 code	
07	Variable 2 code	Variable 2 code	
08	CRC	CRC	
09	ETX (0x03)	ETX (0x03)	

Conditions

The command is effective under all conditions.

6 Parameters

The parameters listed in the following table can be sent using the Change parameter command `CMDCHGPARN (0xb8)`described in Section 4.4.

Code	Description	Name	Size	Unit of measure	Default values SM137 / SM140	Min/Max values SM137 / SM140
0x0100	Proportional gain of current ring	KP_I	16bit	x0.01	30	0 / 32767
					40	0 / 32767
0x0101	Integrative gain of current ring	KI_I	16bit	x0.01	12	0 / 32767
					10	0 / 32767
0x0103	Minimum value for Proportional Integrative current regulator	PIMIN_I	16bit	Volt x0.1	-194	-32767 / 0
					-195	-32767 / 0
0x0104	Maximum value for Proportional Integrative current regulator	PIMAX_I	16bit	Volt x0.1	194	0 / 32767
					195	0 / 32767
0x0105	Proportional gain of speed ring	KP_VEL	16bit	x 0.01	150	0 / 32767
					600	0 / 32767
0x0106	Integrative gain of speed ring	KI_VEL	16bit	x0.01	10	0 / 32767
					50	0 / 32767
0x0107	Percentage feedforward for speed regulator	KFF_VEL	16bit	%	100	0 / 100
					100	0 / 100
0x0108	Minimum value for Proportional Integrative speed regulator. (Limits maximum torque delivered by motor.)	PIMIN_VEL	16bit	Ax0.01	-500	-900 / 0
					-1800	-3500 / 0
0x0109	Maximum value for Proportional Integrative speed regulator. Unit of measure is the Ampere in [Q15] notation. (3) (Limits maximum torque delivered by motor.)	PIMAX_VEL	16bit	Ax0.01	500	0 / 900
					1800	0 / 3500
0x010A	Proportional gain of position ring	KP_POS	16bit	x0.01	500	0 / 32767
					500	0 / 32767
0x010B	Integrative gain of position ring	KI_POS	16bit	x0.01	0	0 / 32767
					0	0 / 32767
0x010C	Percentage feedforward for position regulator	KFF_POS	16bit	%	70	0 / 100
					70	0 / 100

Code	Description	Name	Size	Unit of measure	Default values SM137	Default values SM140	Min/Max values SM137	Min/Max values SM140
0x010D	Minimum value for Proportional Integrative position regulator. (Limits maximum speed reachable by motor.)	PIMIN_POS	16bit	rpm	-4500	-4500	-5000 / 0	-5000 / 0
0x010E	Maximum value for Proportional Integrative position regulator. (Limits maximum speed reachable by motor.)	PIMAX_POS	16bit	rpm	4500	4500	0 / 5000	0 / 5000
0x010F	Maximum permissible position tracking error	MAXERRORP	32bit	Cnt	0	0	0 / (2^31-1)	0 / (2^31-1)
0x0110	Duration of position tracking error after which motor enters alarm state	TIM_MAXERRORP	16bit	msec	0	0	0 / 32000	0 / 32000
0x0111	Maximum permissible speed tracking error	MAXERRORV	16bit	rpm	0	0	0 / 8000	0 / 8000
0x0112	Duration of speed tracking error after which motor enters alarm state	TIM_MAXERRORV	16bit	msec	0	0	0 / 32000	0 / 32000
0x0113	Duration of speed tracking error after which motor enters alarm state	INPOS	16bit	cnt	0	0	0 / 32000	0 / 32000
0x0114	Time within which position must be within tolerance for movement to be declared complete	TIM_INPOS	16bit	msec	0	0	0 / 32000	0 / 32000
0x0115	Torque value that must be exceeded for the motor to enter alarm state	MAXTORQ	16bit	Ax0.01	250	900	0 / 32767	0 / 32767
0x0116	Time for which torque must exceed MAXTORQ for the motor to enter alarm state	TIM_MAXTORQ	16bit	msec	1000	1000	0 / 32000	0 / 32000
0x0117	Maximum speed for positioning movements	VMAXPOS	16bit	rpm	4000	4000	0 / 32767	0 / 32767
0x0118	Speed for automatic reset	VMAXAZZ	16bit	rpm	0	500	-32767 / 32767	-32767 / 32767

Code	Description	Name	Size	Unit of measure	Default values	Min/Max values
					SM137	SM137
					SM140	SM140
0x0119	Acceleration for nonpositioning movements and holds	AMAX	16bit	10000 cnt/sec²	48	0 / 32767
					40	0 / 32767
0x011A	Acceleration for positioning movements	AMAXPOS	16bit	10000 cnt/sec²	32	0 / 32767
					20	0 / 32767
0x011B	Position assigned during automatic reset	ORIG_AZZ	32bit	cnt	0	-(2^31)-1/(2^31-1)
					0	-(2^31)-1/(2^31-1)
0x011C	Lower software limit	LOW_SLIM	32bit	cnt	-32767*2^16	-(2^31)-1 / (2^31)-1
					-32767*2^16	-(2^31)-1 / (2^31)-1
0x011D	Upper software limit	HIGH_SLIM	32bit	cnt	32767*2^16	-(2^31)-1 / (2^31)-1
					32767*2^16	-(2^31)-1 / (2^31)-1
0x011E	Bit by bit meaning (see table 1)	BIT_A	16bit		0	
0x011F	Reserved parameter. DO NOT CHANGE.	ANGELETTRTACCA	16bit		676	0 / 799
					676	0 / 1999
0x0120	First feedforward component for speed.	RESERVED1			600	
					300	
0x0121	Second feedforward component for speed.	RESERVED2			1400	
					600	
0x0122	Third feedforward component for speed.	RESERVED3			2600	
					2200	
0x0123	Fourth feedforward component for speed.	RESERVED4			6	16
0x0124	Reserved parameter	RESERVED5				
0x0125	Reserved parameter	RESERVED6				
0x0126	Reserved parameter	RESERVED7				
0x0127	Reserved parameter	RESERVED8				
0x0128	Reserved parameter	RESERVED9				
0x012A	Reserved parameter	RESERVED11				
0x012B	Reserved parameter	RESERVED11				
0x012C	For Enet-X: bus actuation time. Must be a multiple of 4.For RS485: minimum delay in motor response to a command.	TIMFB	16bit	msec	8 EnetX 50 RS485	1/100
					8 EnetX 50 RS485	1/100
0x012D	Time from reception of message after which motor enters alarm state.	TIMEOUTFB	16bit		8 EnetX 50 RS485	1/100
					8 EnetX 50 RS485	1/100

Code	Description	Name	Size	Unit of measure	Default values SM137 SM140	Min/Max values SM137 SM140
0x012E	For SM140 only. Bit by bit. Shows logic level of digital inputs.	INVDIN	16bit		0	0 / 7
0x012F	For SM140 only. Type of automatic reset. See CmdAzz command. .	TIPOAZZ	16bit		0	0 / 2
0x0130	For SM140 only. Speed for leaving the reset microswitch.	VAZZOUTMIC	16bit	rpm	-125	-32767 / 32767

Table 1 :Meanings of the bits in the BIT_A (0x011E)parameter

Byte	Default	Meaning
0	0	If 1, enables control of software limit switches
1	0	Reserved. Leave at 0
2	0	Reserved. Leave at 0
3	0	If 1, reverses standard direction of motor rotation. Standard positive direction is anti-clockwise, looking at shaft from flange side
4	0	Reserved. Leave at 0
5	0	Reserved. Leave at 0
6	0	**Only for SM140 and SM137 from Rev. C on.** If 1, prevents CMDNOREG from exiting alarm state. In this case, CMDRESET is the only command to quit alarm state
7	0	Reserved. Leave at 0
8	0	**Only for SM140** If 1, enables the negative overrun cam
9	0	**Only for SM140** If 1, enables the positive overrun cam
10	0	**Only for SM140** If 1, forces the motor to enter alarm state when it reaches one of the two overrun cams. If 0, forces the motor to perform an AXSTOP with a suitable deceleration ramp when it reaches one of the two overrun cams
11	0	**Firmware versions 119 and later** Reserved for EnetX. If 1, disables latching between bus and regulation

12	0	**Only for SM140, firmware version 119 and later** If 0, forces the motor to perform controlled braking before leaving regulation mode. When this bit assumes the value 0, after all alarm conditions (except ALOVERCURR and ALOVERPOWER which could damage the drive) and after CMDGOEMERG and CMDNOREG commands, the motor checks that its rotation speed is 0, and if it is not, it performs controlled braking using a deceleration value equal to the value of the AMAX parameter. If inertia is particularly high during this phase, fit the motor with a braking resistance (module P144 from CNI)

Parameters can be sent using the CMDCHGPARN (0xb8) command.

6.1 Regulator structure

6.1.1 Regulator structure

The motor incorporates three regulators connected one to the other:

- the current regulator receives its reference from the speed regulator (torque/current reference) and directly drives PWM.
- the speed regulator receives its reference from the position regulator (speed reference) and supplies the torque reference to the current regulator.
- the position regulator receives its reference from the trajectory generator and supplies the reference to the speed regulator.

Regulators generally have two sections:

- a feedback section that corrects tracking error
- a feedforward section that only processes the reference signal.

The feedback section is the most important. It is the feedback component of the regulator's output that allows the motor to reach its reference. In this motor the feedback section is made up of a proportional section and an integrative section.

The feedback section receives tracking error as input and increases or decreases regulator output accordingly.
If we consider only the proportional component of the position regulator, if tracking error increases (the motor position falls behind with respect to the target position), the proportional feedback component proportionally increases the output of the position regulator.
This output represents the speed reference for the speed regulator.
Thus, if the motor gets left behind, the position regulator increments the speed reference, forcing the speed regulator to increase motor speed to reduce positioning error.
Obviously, if positioning error is 0, the position regulator would request a speed of 0 from the speed regulator and the motor would stop, generating a positioning error. In other words the feedback section of the regulator always requires a tracking error different from 0 to function correctly. This makes it impossible to reach and maintain any reference.
Furthermore, to achieve low error levels, feedback gain levels have to be increased and this makes the controlled system unstable, causing ever greater vibrations as the gain levels increase.
These are the main reasons for justifying the inclusion of a feedforward component. The feedforward section uses the reference to output a value to the regulator that in theory gives a tracking error of 0.
Considering the position regulator again, assuming the feedforward component is set to 100% and the feedback component is disabled (i.e. proportional and integrative gains are both set to 0), if the position reference is given with a speed of 2000 rpm, then the position regulator provides the speed regulator with a reference of 2000 rpm, even if there is no tracking error.
If the feedforward component is set to 50%, the reference given to the speed regulator would be 1000 rpm.

This allows us to obtain low tracking errors even with low gain levels in the feedback section. Also, by reducing feedback gain levels we move further away from the point at which the system becomes unstable and acquire the ability to control axes even under conditions of extreme inertia.
The feedforward section, however, has the defect of necessitating a relatively rigid axis control. If pushed to the limit, this can subject the axis to excessive mechanical stress.
Parameter calibration is designed to find the right compromise between the actions of the two regulation output components.

6.1.2 Calibrating the feedforward component

As a general principle, the innermost regulators should always be calibrated first.
The current regulator, however, is affected almost exclusively by the electrical characteristics of the motor: its default parameter settings should therefore be left unchanged.
You should therefore start with the speed regulator and in particular with its feedforward component.

Parameters requiring calibration

The parameters you need to calibrate are:

- RESERVED1 (0x0120)
- RESERVED2 (0x0121)
- RESERVED3 (0x0122)
- RESERVED4 (0x0123)

Preparation

- during testing, ensure that the axis can move in total safety for the longest possible travel
- disable the feedback components of the position and speed regulators. Also set the proportional and integrative gain levels of both regulators to zero
- maximise the feedforward components by setting the feedforward gains of the position and speed regulators to 100%

Step 1: calibrate the RESERVED1 and RESERVED2 parameters

This step calibrates the feedforward component needed to overcome mechanical friction.

- set RESERVED3 and RESERVED4 to 0
- RESERVED1 is the speed above which the feedforward that counteracts friction cuts in.
  It is expressed in [Q15] notation and is typically set to a value of 200 (on SM137 and SM140 motors this means that that anti-friction feedforward cuts in at speeds above: :

$$\frac{200}{32768} \times 8000 = 48,8rpm$$

- RESERVED2 is the amount of torque needed to overcome mechanical friction and has to be set by trial and error. Jog the axis and increase RESERVED2 until you find that the axis moves with little resistance when pushed by hand in the direction of the jog. Also make sure that the axis stops as soon as you stop pushing it by hand. If it does not, reduce the value of RESERVED2 until it does.

Step 2: calibrate the RESERVED3 parameter

This step calibrates the feedforward component proportional to reference speed.

- using the values for RESERVED1 and RESERVED2 ascertained above, jog the axis at its normal operating speed

- gradually increase (in steps of 100) `RESERVED3` until full motor speed (the speed reached at the end of the acceleration ramp) is equal to or very near to that specified in the jog command

Step 3: calibrate the RESERVED4 parameter

This step calibrates the feedforward component that controls acceleration and deceleration.

- using the values for `RESERVED1`, `RESERVED2` and `RESERVED3` ascertained above, jog the axis at its normal operating speed and acceleration values.
- monitor real and theoretical speed during the ramps, and gradually increase `RESERVED4` (in steps of 2) until the two speeds coincide

Step 4: Testing

Perform a number of positioning movements at normal operating speed. Despite the fact that the feedback components are set to 0, the axis should reach its target positions with reasonable precision.

6.1.3 Calibrating the feedback component of the speed regulator

Provided calibration of the feedforward part has been completed successfully, the default gain levels should prove satisfactory.
On a trial and error basis you can increase or decrease them one at a time to see if performance improves or gets worse.
It is generally best to leave feedforward at 100% and set an integrative gain other than 0, but apart from this, it is up to your own discretion and experience to set the gain levels you find best.
Use the jog command to test motor settings.

6.1.4 Taratura della parte in feedforward del regolatore di posizione

The only value for the position regulator that needs calibrating is the percentage of feedforward.
When doing so, always bear in mind the mechanical stress that this causes.
Disable the feedback component and perform a number of positioning movements to test the settings.
The best value normally ranges between 50% and 100%.

6.1.5 Calibrating the feedback component of the position regulator

Here too the default gain values should prove adequate.
Generally speaking leave integrative gain at 0.
Gradually change proportional gain, fine tuning position feedforward as necessary, to achieve the best possible performance.

7 Low level parameters

IMPORTANT! The parameters described below cannot be saved to flash memory. These low level parameters have been have been maintained only to ensure backwards compatibility of the software. Their use is NOT recommended.

Low level parameters can be transmitted using the `CMDCHGPAR (0x04)`command.

Code	Description	Name	Default value	Min/Max values
			SM137	SM137
			SM140	SM140
0x1000	Mantissa of current ring proportionalintegrative gain, in [ME15] notation	KIQDPIMANT	13762	0 / 32767
			16383	0 / 32767
0x1001	Exponent of current ring proportionalintegrative gain, in [ME15] notation	KIQDPIEXP	0	0 / 6
			0	0 / 6
0x0002	Current ring correction factor gain	KIQDKCOR	9362	0 / 32767
			6552	0 / 32767
0x0003	Not used	KIQDKFF	--	--
0x0004	Minimum value for the output of the roportional-integrative current regulator, in Volts, expressed in [Q15] notation	KIQDLIMMIN	-32767	-32767/0
0x0005	Maximum value for the output of the proportional-integrative current regulator, in Volts, expressed in [Q15] notation	KIQDLIMAX	32767	0 / 32767
			32767	0 / 32767
0x2006	Mantissa of speed ring proportionalintegrative gain, in [ME15] notation	KVELPIMANT	26214	0 / 32767
			26624	0 / 32767
0x2007	Exponent of speed ring proportionalintegrative gain, in [ME15] notation	KVELPIEXP	1	0 / 6
			3	0 / 6
0x0008	Speed ring correction factor gain	KVELKCOR	2047	0 / 32767
			2520	0 / 32767
0x0009	Speed regulator feedforward percentage	KVELKFF	32767	0 / 32767
			32767	0 / 32767
0x000A	Minimum value for the output of the proportional-integrative speed regulator, in Amps, expressed in [Q15] notation. In practice, this parameter limits the maximum torque that can be demanded of the motor in the negative direction of rotation	KVELLIMIN	-14486	-26067 / 0
			-13034	-25342 / 0
0x000B	Maximum value for the output of the proportional-integrative speed regulator, in Amps, expressed in [Q15] notation. In practice, this parameter limits the maximum torque that can be demanded of the motor in the positive direction of rotation	KVELLIMAX	14486	0 / 26067
			13034	0 / 25342

Code	Description	Name	Default value SM137	Default value SM140	Min/Max values SM137	Min/Max values SM140
0x300C	Mantissa of position ring proportionalintegrative gain in [ME15] notation.	KPOSPIMANT	20480	20480	0 / 32767	0 / 32767
0x300D	Exponent of position ring proportionalintegrative gain in [ME15] notation	KPOSPIEXP	3	3	0 / 6	0 / 6
0x000E	Position ring correction factor gain	KPOSKCOR	0	0	0 / 32767	0 / 32767
0x000F	Position regulator feedforward percentage	KPOSKFF	22937	22937	0 / 32767	0 / 32767
0x0010	Minimum value for the output of the proportional-integrative position regulator, in rpm, expressed in [Q15] notation. In practice, this parameter limits the maximum speed that can be demanded of the motor in the negative direction of rotation	KPOSLIMMIN	-18432	-18432	-32767 / 0	-32767 / 0
0x0011	Maximum value for the output of the proportional-integrative position regulator, in rpm, expressed in [Q15] notation. In practice, this parameter limits the maximum speed that can be demanded of the motor in the positive direction of rotation	KPOSLIMAX	18432	18432	0 / 32767	0 / 32767
0x4012	16 least significant bits of the maximum permissible position tracking error, expressed as an encoder count (1)	MAXERRORPL	0	0	-32768 / 32767	-32768 / 32767
0x4013	16 most significant bits of the maximum permissible position tracking error, expressed as an encoder count (1)	MAXERRORPH	0	0	0 / 32767	0 / 32767
0x0014	Acceleration used during non-positioning movements and holds in revs/sec², expressed in [Q15] notation	AMAXHOLD	2089	696	1 / 32767	1 / 32767
0x0015	Acceleration used during positioning movements in revs/sec², expressed in [Q15] notation	AMAXTRAJ	1392	348	1 / 32767	1 / 32767
0x0016	Maximum speed of positioning movements in rpm, expressed in [Q15] notation (1)	VMAXTRAJ	16384	16384	0 / 32767	0 / 32767
0x5017	16 least significant bits of position set during an automatic reset, expressed as an encoder count.	ORIGINEL	0	0	-32768 / 32767	-32768 / 32767

Code	Description	Name	Default value SM137	Default value SM140	Min/Max values SM137	Min/Max values SM140
0x5018	16 most significant bits of position set during an automatic reset, expressed as an encoder count	ORIGINEH	0	0	-32768 / 32767	-32768 / 32767
0x0019	Bit by bit meaning: - bit0=1: enables control of software limit switches - bit1: reserved - bit2: reserved - bit3: toggles direction of motor rotation. The meaning of these bits is described in the table1 on page 40	TESTSLIM	0			
0x601A	16 least significant bits of lower software limit switch, expressed as an encoder count	LOWSLIML	0	0	-32768 / 32767	-32768 / 32767
0x601B	16 most significant bits of lower software limit switch, expressed as an encoder count	LOWSLIMH	0	0	-32768 / 32767	-32768 / 32767
0x601C	16 least significant bits of upper software limit switch, expressed as an encoder count	HIGHSLIML	0	0	-32768 / 32767	-32768 / 32767
0x601D	16 most significant bits of upper software limit switch, expressed as an encoder count	HIGHSLIMH	0	0	-32768 / 32767	-32768 / 32767
0x001E	Maximum permissible speed tracking error, in rpm, expressed in [Q15] notation	MAXERRORV	0	0	0 / 32767	0 / 32767
0x001F	Maximum torque that can be demanded of the motor for more than TIMEOUTMAXTORQUE milliseconds, in Amps, expressed in [Q15] notation	MAXTORQUE	7243	6517	0 / 32767	0 / 32767
0x0020	Reserved parameter	ANGELETTRTACCA	676	676	--	--
0x0021	Reserved parameter	RESERVED1	600	300	0 / 32767	0 / 32767
0x0022	Reserved parameter	RESERVED2	1400	600	0 / 32767	0 / 32767
0x0023	Reserved parameter	RESERVED3	2600	2200	0 / 32767	0 / 32767
0x0024	Reserved parameter	RESERVED4	6	16	0 / 32767	0 / 32767
0x0025	Reserved parameter	RESERVED5				
0x0026	Reserved parameter	RESERVED6				
0x0027	Reserved parameter	RESERVED7				

Code	Description	Name	Default value	Min/Max values
			SM137	SM137
			SM140	SM140
0x0028	Reserved parameter	RESERVED8		
0x0029	Reserved parameter	RESERVED9		
0x002A	Reserved parameter	RESERVED10		
0x002B	Reserved parameter	RESERVED11		
0x002C	Reserved parameter	RESERVED12	0	
0x002D	Time during which torque must exceed MAXTORQUE for the motor to enter alarm state, in [msec]	TIMEOUTMAXTORQUE	1000 1000	0 / 32767 0 / 32767

Notes:

- (1) A null value in this parameter disables the relevant control.

Representation of numbers in [Q15] notation

The representation of a number in [Q15] notation requires the multiplication of a real number by the maximum value that can be expressed by 15 bits, i.e. by 15² and the truncation of the resulting number.
This gives linear correspondence between decimal values and integers.
[Q15] notation is often used to assign values to motor parameters. Given a quantity f and its maximum value, also known as its base value and identified by fbase, the said quantity f is represented in [Q15] notation by the following formula.

$$f_{[Q15]} = \frac{f}{f_{base}} \times 2^{15} \quad con \quad \angle 32767 \leq f_{[Q15]} \leq 32767$$

For example, on the SM137 motor, base current is 11.313 A and absorbed current is I=5A. The value for absorbed current in [Q15] notation is:

$$5[A] \Leftrightarrow \frac{5}{11,313} \times 2^{15} = 14482[Q15]$$

Mantissa-exponent [ME15] notation

For quantities that have no reference or base value, [ME15] mantissa-exponent notation can be used instead.

$$f[M15] = (fmant;fesp) \Leftrightarrow f = \frac{fmant}{2^{15}} \times 2^{fesp}$$

Note that this notation is not unambiguous. One number can be represented in different ways in [ME15] notation.

For example the number 15.5 can be represented in either of the following 2 ways:

$$f_1[M15] = (31744;4) \Leftrightarrow f_1 = \frac{31744}{2^{15}} \times 2^4 = 15,5$$

$$f_2[M15] = (15872;5) \Leftrightarrow f_1 = \frac{15872}{2^{15}} \times 2^5 = 15,5$$

Proportional-integrative regulator gains for current, speed and position

The gains that can be set on the motor's proportional-integrative regulators can be divided into:

- feedback gains
- feedforward gains

Feedback gains are based on a proportional-integrative structure that limits output and corrects the integral component.

There are 3 such gains:

- proportional-integrative gain for the mantissa (KxxxPIMANT)
- proportional-integrative gain for the exponent (KxxPIEXP)
- correction factor gain

The relationship of these gains to the classic proportional gains (Kp) and integrative gains (Ki) are as follows:

$$\begin{cases} K_{pi} = K_p + K_i \\ K_{cor} = \dfrac{K_i}{K_p + K_i} \end{cases}$$

Also:

$$\begin{Bmatrix} \text{KxxxPIMANT\_M;KxxxPIEXP\_M} = K_{pi}[ME15] \\ \text{KxxxKCOR\_M} = K_{cor}[Q15] \end{Bmatrix}$$

The base value for Kcor is 1.

Feedforward gain is expressed as a percentage in [Q15] notation taking 100 as base value.
Example:

$$\begin{cases} K_p = 0,5 \\ K_i = 0,08 \Leftrightarrow \\ K_{FF} = 15 \end{cases} \begin{cases} K_{pi} = 0,5 + 0,08 = 0,58 \\ K_{cor} = \dfrac{0,08}{0,58} = 0,1379 \\ K_{FF} = 15 \end{cases} \begin{cases} \text{KxxxPIMANT\_M} = 0,58 \times 2^{15} = 19005 \\ \text{KxxxPIEXP\_M} = 0 \\ \text{KxxxKCOR\_M} = 0,1379 \times 2^{15} = 4520 \\ \text{KxxxKFF\_M} = \dfrac{15}{100} \times 2^{15} = 4915 \end{cases}$$

Output from the proportional-integrative regulator is also limited by minimum and maximum values, KxxxLIMMIN and KxxxLIMMAX respectively, both expressed in [Q15] notation.

Base values for [Q15] notation

Unit of measure	Base value for SM137	Base value for SM140
Voltage in Volts [V]	19.4	19.6
Current or Torque in Amps A]	11.3137	45.2548
Speed in revs per minute [rpm]	8000	8000
Acceleration in revs/sec^2	9411	9411

SM137 quantification

The AMAXHOLD parameter is quantified at 4.096 [revs/sec2] and the AMAXTRAJ parameter at:

- 78.125 [giri/sec^2] for revision B
- 19.53125 [giri/sec^2] for revision C

This implies that if, for example, with a step of 0.32 mm/rev, you set an acceleration AMAXTRAJ of 430 [mm/sec2], you obtain an acceleration of 17x78.125x0.32=425 [mm/sec2]

8 Internal states

The motor always powers up in AXALARM state.
The current motor state can be read using the Get motor's internal state command CmdGetSmStat (Section 4.18).
The following table lists all the motor's internal states.

Code	Meaning	Name
0x0000	The motor is out of regulation mode, so the motor applies no resistance if the rotor is moved from its current position	AXNOREG
0x0001	The motor is in alarm state. From the mechanical viewpoint the motor is out of regulation mode. Unlike AXNOREG state, however, the motor cannot return directly to regulation mode	AXALARM
0x0002	The motor is in regulation mode and therefore attempts to maintain the current rotor position	AXSTOP
0x0003	The motor is calibrating the offsets of the current sensors	AXADCOFF
0x0004	The motor is resetting the electrical position of the rotor	AXAZZEL
0x0006	The motor is performing an automatic reset and is currently seeking the encoder's zero notch	AXAZZAUTO
0x0007	The motor is stopping with the deceleration ramp set in the AMAXHOLD parameter	AXHOLD
0x0008	The motor is running at constant speed	AXFREERUN
0x0009	The motor is performing a positioning movement with the acceleration ramp set in the AMAXTRAJ parameter and the maximum speed set in VMAXTRAJ	AXEXEC
0x000A	Not used	AXTORQUE
0x000B	The motor is interpolating (only firmware versions >=110)	AXINTERP
0x000D	The motor is latching its position on to the zero notch (only firmware versions >=110).	AXLATCH
0x000E	The motor is attempting to effect a controlled braking in order to stop with the acceleration described in parameter 0x119. As soon as the motor has stopped, its new status will become AXALARM. This status is available on SM140 from version 119 onwards	PREAXALARM
0x000F	The motor is attempting to effect a controlled braking in order to stop with the acceleration described in parameter 0x119. As soon as the motor has stopped, its new status will become AXNOREG. This status is available on SM140 from version 119 onwards	PREAXNOREG

The following tables show all the possible combinations of motor state, sent command and command effect.

Symbols legend:

- "=": State remains the same
- "error": The command is not accepted and the motor returns a CMDNACK response
- "State 1 ⇒ State 2": aWhen the motor receives the command, it enters State1. On completion of the operations required in State1, state changes to State2
- "∗" : Commands and state implemented only from revision C on the SM137 and on the SM140

N.B.: The motor's initial state is always AXALARM

Command	Status						
	PREAXALARM (6)	AXALARM	AXADCOFF	PREAXNOREG (6)	AXNOREG	AXAZZEL	AXSTOP
CMDEMERG	== error	AXALARM	AXALARM	== error	AXALARM	AXALARM	AXALARM
CMDADCOFF	== error	AXADCOFF ⇒AXALARM	= error	== error	= error	= error	= error
CMDNOREG	== error	AXNOREG	= error	== error	=		AXNOREG
CMDAZZEL	== error	= error	= error	== error	AXAZZEL ⇒ AXNOREG	= error	= error
CMDREG	== error	= error	= error	== error	(1)AXAZZEL⇒ AXSTOP	= error	=
CMDAZZ	== error	= error	= error	== error	= error	= error	= error or AXAZZAUTO⇒AXHOLD⇒ AXSTOP
CMDMAZZ	== error	= Performs a manual reset	= error	== error	= Performs a manual reset	= error	= Performs a manual reset
CMDJOG	== error	= error	= error	== error	= error	= error	AXFREERUN
CMDTRAJ	== error	= error	= error	== error	= error	= error	AXEXEC ⇒ AXSTOP (3)
CMDHOLD	== error	= error	= error	== error	= error	= error	=
CMDCHGPAR	== error	=	= error	== error	=	= error	=
CMDGETPAR	==	=	=	==	=	=	=
CMDGETALARM	==	=	=	==	=	=	=
CMDGETPOS	==	=	=	==	=	=	=
CMDGETPOST	==	=	=	==	=	=	=
CMDGETVEL	==	=	=	==	=	=	=
CMDGETTOR	==	=	=	==	=	=	=
CMDGETVER	==	=	=	==	=	=	=
CMDGETSMSTAT	==	=	=	==	=	=	=
CMDSAMPLE	==	=	=	==	=	=	=
CMDGETSAMP	==	=	=	==	=	=	=
CMDCHGPARN ∗	== error	=	= error	== error	=	= error	=
CMDGETPARN ∗	==	=	=	==	=	=	=
CMDTRAJVEL ∗	== error	= error	= error	== error	= error	= error	AXEXEC⇒ AXSTOP(3)
CMDSETOVERR ∗	==	=	=	==	=	=	=
CMDGETOVERR ∗	==	=	=	==	=	=	=
CMDJOGN ∗	== error	= error	= error	== error	= error	= error	AXFREERUN
CMDSAVEPARFL ∗	== error			== error		= error	= error

Command	Status						
	PREAXALARM (6)	AXALARM	AXADCOF F	PREAXNOREG (6)	AXNOREG	AXAZZEL	AXSTOP
CMDERASEFIR ∗	== error			== error		= error	= error
CMDCHGBOOT ∗	== error			== error		= error	= error
CMDGOTOBOOT ∗	== error			== error		= error	= error
CMDGOINTERP ∗	== error	= error	= error	== error	= error	= error	AXINTERP
CMDSTOPINTERP ∗	== error	=	=	== error	=	=	=
CMDLATCHINTERP∗	== error	= error	= error	== error	= error	= error	= error

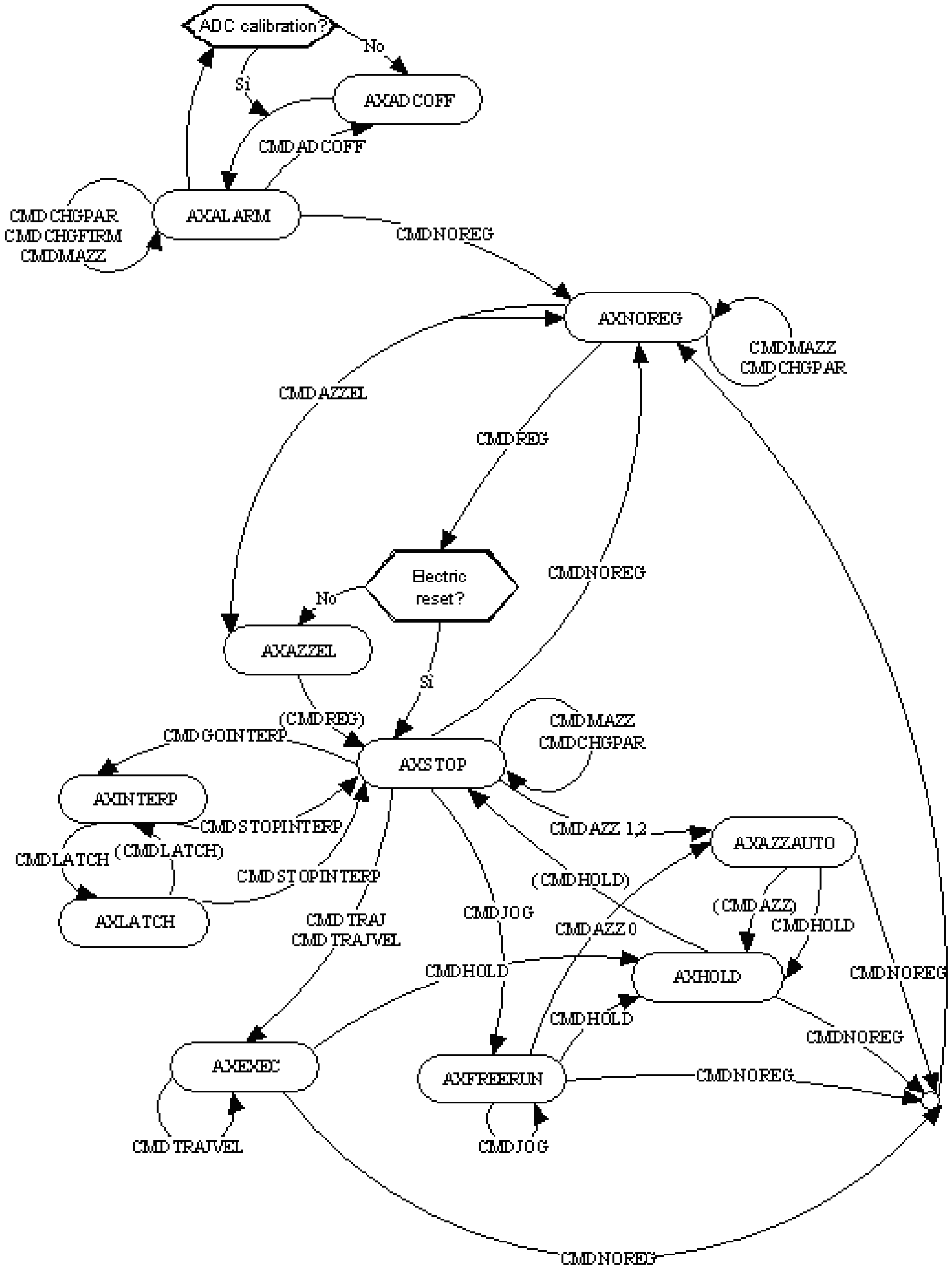
Command	Status					
	AXAZZAUTO	**AXFREERUN**	**AXEXEC**	**AXHOLD**	**AXINTERP(*)**	**AXLATCH (*)**
CMDEMERG	AXALARM	AXALARM	AXALARM	AXALARM	= error	= error
CMDADCOFF	= error	= error	= error	= error	= error	= error
CMDNOREG	AXNOREG	AXNOREG	AXNOREG	AXNOREG	= error	= error
CMDAZZEL	= error	= error	= error	= error	= error	= error
CMDREG	= error	= error	= error	= error	= error	= error
CMDAZZ	=	AXAZZAUTO ⇒ AXHOLD⇒AXSTOP or = error	= error	= error	= error	= error
CMDMAZZ	= error	= error	= error	= error	= error	= error
CMDJOG	= error	=(2)	= error	= error	= error	= error
CMDTRAJ	= error	= error	= error	= error	= error	= error
CMDHOLD	AXHOLD⇒ AXSTOP	AXHOLD⇒ AXSTOP	AXHOLD⇒ AXSTOP	=	= error	= error
CMDCHGPAR	= error	= error	= error	= error	= error	= error
CMDGETPAR	=	=	=	=	=	= error
CMDGETALARM	=	=	=	=	=	= error
CMDGETPOS	=	=	=	=	=	= error
CMDGETPOST	=	=	=	=	=	= error
CMDGETVEL	=	=	=	=	=	= error
CMDGETTOR	=	=	=	=	=	= error
CMDGETVER	=	=	=	=	=	= error
CMDGETSMSTAT	=	=	=	=	=	= error
CMDSAMPLE	=	=	=	=	=	= error
CMDGETSAMP	=	=	=	=	=	= error
CMDCHGPARN ∗	= error	= error	= error	= error	= error	= error
CMDGETPARN ∗	=	=	=	=	= error	= error
CMDTRAJVEL ∗	= error	= error	= (5)	= error	= error	= error
CMDSETOVERR ∗	=	=	=	=	= error	= error
CMDGETOVERR ∗	=	=	=	=	= error	= error
CMDJOGN ∗	= error	= (2)	= error	= error	= error	= error
CMDSAVEPARFL ∗	= error	= error	= error	= error	= error	= error
CMDERASEFIR ∗	= error	= error	= error	= error	= error	= error
CMDCHGBOOT ∗	= error	= error	= error	= error	= error	= error
CMDGOTOBOOT ∗	= error	= error	= error	= error	= error	= error
CMDGOINTERP ∗	= error	= error	= error	= error	= error	= error
CMDSTOPINTERP∗	= error	= error	= error	= error	AXSTOP	AXSTOP
CMDLATCHINTER∗	= error	= error	= error	= error	AXLATCH	= error

Notes:

- (1) If the motor has not yet performed an electrical reset, it passes through AXAZZEL for the time necessary to complete an electrical reset before it can enter AXSTOP
- (2) If necessary, the jog speed is modified with ramps between one speed and the next
- (3) State becomes AXEXEC only if the motor has been manually or automatically reset
- (4) These commands are strictly linked to communication state. Contact the supplier for further details
- (5) If necessary, movement speed is modified with ramps between one speed and the next
- (6) This status is available on SM140 from version 119 onwards

The movement's target position is also modified if necessary. If speed or position cannot be changed, the motor returns the invalid command response CmdNACK (Appendix F).

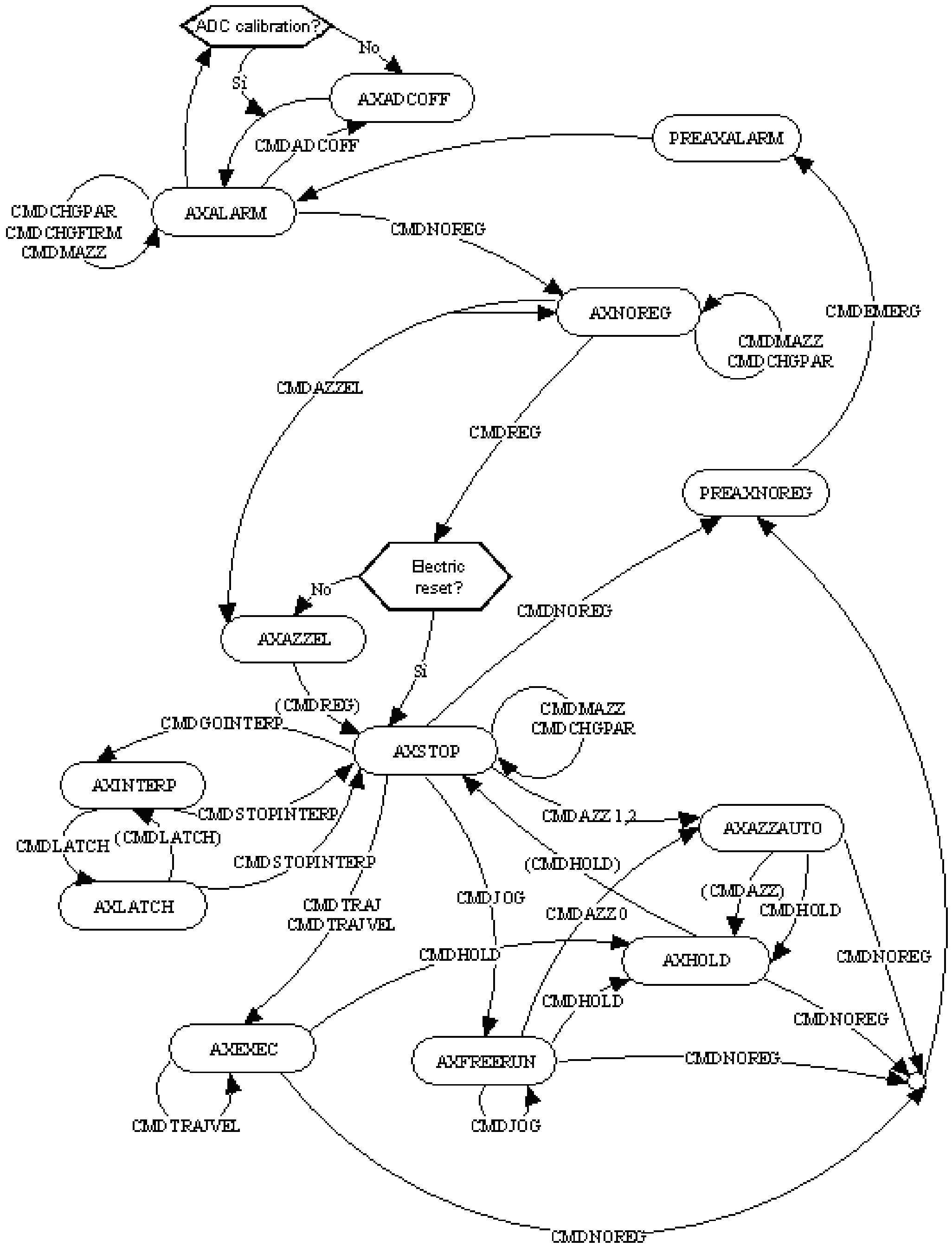
State transitions:



Notes:

If a number appears alongside the CMDAZZ command, it is the value for the TIPOAZZ parameter which permits the specified state transition.
State transitions associated with the overrun cams are not shown.
To exit the alarm status, it is recommended to use the CMDRESET command (Section 3.2).

With SM140 motors, if you have not disabled controlled braking, whenever the motor exits regulation mode, state transition adopts the logic shown below.



Note the presence of two new states. The purpose of these states is to stop the motor with a deceleration ramp identical to the jog ramp.
When the motor stops, it enters either AXNOREG or AXALARM state, depending on the case.
To exit the alarm status, it is recommended to use the CMDRESET command (Section 3.2).

9 Reset state

The CmdGetStatAzz command reads the motor's current reset state.
The following table shows the states that the motor may return, depending on the type of automatic reset being performed, as determined by the TIPOAZZ (0x012F) parameter:

Code	Name	Meaning	Reset Type (TIPOAZZ)
0x0000	NOAZZ	The motor has not been manually or automatically reset, or the last reset has not been completed	0 -1 - 2
0x0001	AZZMAN	The motor has been reset manually	0 -1 - 2
0x0002	SEARCHINGMICRO	The motor is performing an automatic reset and is currently seeking the reset microswitch	1 - 2
0x0003	AZZAUTO	The motor has been reset automatically	0 - 1 - 2
0x0004	LEAVINGMICRO	The motor is performing an automatic reset, has reached the microswitch and is about to leave it	1 - 2
0x0006	SARCHINGTACCA	The motor is waiting to reach the encoder's zero notch to complete an automatic reset	0 - 1

10 Messages

The motor can generate two types of message:

1. ALARMS: these messages inform you of serious error conditions that place the motor in an EMERGENCY state
2. WARNINGS: these messages inform you of non-serious error conditions that do not place the motor in an EMERGENCY state

Messages can be read using the `CmdGetAlarm` command (Section 4.15).

10.1 ALARM messages

The following table lists all possible ALARM messages and any actions required to restore normal motor operating conditions.

Code	Name	Meaning	Action required
0x0000	NOALARM	The motor is functioning normally	None
0x0001	ALOVERHEATED	The motor has exceeded a temperature of 70° C	Check the real mechanical load on the motor and/or the idle time to working time ratio
0x0002	ALOVERCURR	An overcurrent has passed through the motor's power stage	Possible short circuit in the power stage. Contact the supplier
0x0003	ALOVERLOAD	The torque demanded of the motor has exceeded maximum torque for more than the permitted time	Increase the value of the `MAXTORQUE` parameter in conjunction with the `KVELLIMAX` and `KVELLIMIN` parameters.
0x0004	ALGENTRAIETT	Trajectory generator error	Contact the supplier
0x0005	ALOVERLIM	The motor has moved beyond one of the software limits. This can occur if control of the limits is enabled and the motor is reset	Check possible causes: ■ overruns at the end of movements ■ jogging beyond the limits ■ external loads
0x0006	ALMAXERRORP	The motor has generated a position tracking error greater than the maximum permissible set in the parameters	Check motor calibration. Check for any mechanical obstacles. Increase the position tracking error threshold
0x0007	ALMAXERRORV	The motor has generated a speed tracking error greater than the maximum permissible set in the parameters	Check motor calibration. Check for any mechanical obstacles. Increase the speed tracking error threshold

Code	Name	Meaning	Action required
0x0008	ALCOMERROR	Communication error between motor and master (1)	Check the cabling. If the TIMEOUTFB (0x012D) parameter is other than 0, increase it. Check whether the master was switched off while the motor was still switched on
0x0009	ALNOPOWER	The power supply has dropped below 16 V while the motor was in regulation mode	Check whether the machine entered emergency state while the motor was still in regulation mode. Check the power supply
0x000A	ALNOPARAMINFL(*)	Incorrect parameters saved in flash memory	Resend the parameters to the motor. Save them in flash memory to avoid the message repeating
0x000B	ALNOPRGINFLASH (*)	Incorrect parameters saved in flash memory	Download the right application
0x000C	ALERASINGFL(*)	Flash memory erase problems have been encountered during an application download	Download the application again
0x000D	ALPRGMINGFL(*)	Flash memory write problems have been encountered during an application download	Download the application again
0x000E	ALWRONGDATA2 INTERP(*)	The motor received a packet with a coding error during interpolation	Check for communication errors. Check the communication software on the master that drives the motor and/or the cable wiring
0x000F	ALWROGSETP(*)	The motor received a position setpoint demanding overspeed during interpolation	Check the communication software on the master that drives the motor
0x0010	ALNOSETP2 INTERP(*)	The motor received no position setpoint within the allowed time during interpolation	■ Check that the declared bus frequency parameter is correct ■ Check for communication errors ■ Check the communications software on the master that drives the motor and/or the cable wiring
0x0011	ALWRONGFREQ(*)	Bus frequency differs from that declared in the motor parameter	Check that the parameter declaring bus frequency is correct

Code	Name	Meaning	Action required
0x0012	ALNOTIPICPARAM	Certain parameters needed for correct motor functioning were not found on power-up	Contact CNI
0x0013	ALTIMEOUTCOMU NIC	Error in motor communications protocol	Check the wiring and condition of the connection cable. If necessary increase the value of parameter 0x012 (D-TIMEOUTFB)
0x0014	ALOVERPOWER	Power supply voltage has exceeded 38 V	Reduce inertia at the rotor or the maximum acceleration value, or use an electric brake (module P144 from CNI)
0x0020	ALDSPOVERLOAD	Internal software error	Contact the supplier
0x0021	ALWRONGIRQ	Internal software error	Contact the supplier

Legend:

- (1) With 485 serial communications, communication errors may be caused by:
  - Timeouts: the master has not sent any further data
  - Reception of too long a packet of characters (or a missing ETX)
  - Reception of 2 STX characters without an ETX.
  - An incorrect CRC
  - The arrival of an ESC followed by the wrong character (ETX, STX or ESC)
- (*) Messages found on the SM137 from revision C on, and on the SM140

10.2 WARNING messages

The following table lists all possible WARNING messages and any actions required to restore normal motor operating conditions.

Code	Name	Meaning	Action required
0x4000	ALNOAZZ	A positioning movement has been commanded without resetting the motor first	Perform a reset before attempting a positioning movement
0x4001	ALPARNONCORR	A non-existent parameter was sent or received	Check the software on the master
0x4002	ALCMDLOOSED	Internal software error	Contact the supplier
0x4003	ALWRONGCMD	A non-existent command was sent	Contact the supplier
0x4004	ALAXALREADYINPOS	A positioning movement to the motor's current position has been commanded	Check the software on the master
0x4005	ALREQPOSOVERLIM	The positioning movement requested is beyond the software limits	Check the software limits or check the software on the master
0x4006	ALNOTPOT(*)	The motor has reached the upper or lower software limit	Check the software limits set in the parameters or commands sent to the motor
0x4007	ALFLNOTERASED*	A request to write to flash memory refers to an unerased area	Contact the supplier
0x4008	ALFLREADING*	Error writing to flash memory	Contact the supplier
0x4009	ALNOTHW_M(1)	The motor has reached the negative overrun cam	Move the motor in the positive direction or disable control of the negative overrun cam
0x400a	ALPOTHW_M(1)	The motor has reached the positive overrun cam	Move the motor in the negative direction or disable control of the positive overrun cam
0x400b	ALMOVTOOLONG(2)	A positioning movement longer than 67108863 encoder counts has been commanded	Break the movement down into smaller movements, each within the maximum permissible length, without stopping

Legend:

- (*) Messages found on the SM137 from revision C on (firmware versions from 110 on) and on the SM140
- (1) Messages found on the SM140
- (2) From firmware version 119 on

11 Status bits

Packets sent in response to commands always contain 4 status bits. The following table lists their meanings.

Bit	Name	Meaning
3	MASKBITALLARME_SM	1: the motor is in alarm state
2	MASKBITWARNING_SM	1: the motor has a message for the master
1	MASKBITCommand_SM	0: the last command sent to the motor has not yet been completed. This can occur with the automatic reset and line commands
0	MASKBITNOQUOTA_SM	1: the motor is responding to the default command not with its real position but with its theoretical position, speed, or torque

12 Responses to invalid commands

Whenever the motor is unable to perform a command it always responds in the same way, as follows:

Byte	Response	
01	STX (0x02)	
02	Node	
03	0x0	Status Bit
04	CmdNACK (0xB0)	
05	Byte 05 of sent command	
06	Byte 06 of sent command	
07	Byte 07 of sent command	
08	CRC	
09	ETX (0x03)	

13 Digital inputs

13.1 Introduction

The SM140 motor is equipped with 3 digital inputs. The logic level of these inputs can be toggled by the INVDIN (0x012E) parameter as shown in the following tables.

INVDIN	Corresponding digital input	Description
Bit 0	1	Negative overrun cam
Bit 1	2	Positive overrun cam
Bit 2	3	Microswitch for automatic reset

Value of Bit n of INVDIN	Input voltage (V)	Logic level of signal
0	1	Low
0	24	High
1	0	High
1	24	Low

Thus, if bit 2 of INVDIN is 1 and voltage to the third digital input is 0, the logic level of the microswitch will be high.

13.2 Overrun cams

The system handles two overrun cams, one positive and one negative.
The positive overrun cam is the cam that the motor eventually reaches when it increases its real position, while the negative overrun cam is the one that it reaches when it decreases its real position, irrespective of the motor's direction of rotation, which can be set by means of bit 3 of parameter BIT_A (0x011E).

If overrun cam control has been enabled by bits 8 and 9 of parameter BIT_A, and the motor reaches one of the cams during a movement, the motor enters alarm state (AXALARM) or stops and enters regulation mode (AXSTOP) depending on bit 10 of the BIT_A parameter.
The motor also generates a message showing which of the two cams it has reached.

At this stage the motor can only perform movements that take it away from the engaged cam. The default logic of the overrun signals is reversed for obvious safety reasons: the motor detects the overrun cam when its logic signal is low. As already shown, the logic level of this signal can be reversed by the INVDIN (0x012E) parameter but in this event the level of safety provided by the overrun diminishes at the user's own risk.

Reset microswitch

The motor uses the third digital input to perform a fully independent automatic reset (TIPOAZZ=1 or TIPOAZZ=2). This input must therefore be connected to the reset microswitch.
The motor detects the reset microswitch when the relevant logic signal is high.
The logic level of this signal can be reversed as explained above.
For a description of the various types of reset and the way in which this digital input is used in those conditions, see the section dealing with the Automatic position assign (Automatic reset) command CmdAzz (Section 3.7).

14 Interpolation

Interpolation is currently only possible with the Enet-X bus.

The following tables illustrate the byte containing the protocol’s significant bits.

Significant bits for protocol from Master to Slave

Meaning	Bit 7	Bit 6	Bit 5	Bit 4	
Sending default packet	0	0	0	0	M0
Sending single packet	1	1	1	x	M1
Start of multiple packet transmission	0	1	1	x	M2
Continuation of multiple packet transmission	0	0	1	x	M3
End of multiple packet transmission	1	0	1	x	M4
Interpolating: sending setpoint	1	0	0	0	M5
Interpolating: requesting latch	1	1	0	0	M6
Not used	0	1	0	x	

Significant bits for protocol from Slave to Master

Meaning	Bit 7	Bit 6	Bit 5	Bit 4	Bit 2	Bit 1	
Sending default packet	1	1	0	0	x	x	S0
Sending single packet	1	1	1	x	x	x	S1
Start of multiple packet transmission	0	1	1	x	x	x	S2
Continuation of multiple packet transmission	0	0	1	x	x	x	S3
End of multiple packet transmission	1	0	1	x	x	x	S4
Slave interpolating. (SM140: microswitch input high)	1	0	0	0	1	0	S5
Slave interpolating. (SM140: microswitch input low)	1	0	0	0	0	1	S5
Interpolation: position latched on latch	1	0	0	0	x	1	S6
Not used	0	0	0	x	x	x	
Not used	0	1	0	x	x	x	

If the master needs to send an interpolation command it switches the protocol to M5 state, and immediately sends the first setpoint. The slave confirms interpolation by responding with S5 state and sending the default datum requested (normally the real position).

If the master needs to perform a latch, it sends the M6 frame just once. The slave immediately starts to seek the zero notch and returns the real latched position with frame S6.