OPERATING INSTRUCTIONS

SPECIFICATION HIPERFACE®

Motor feedback protocol

SICK
Sensor Intelligence.

en

HIPERFACE®
by SICK

Manufacturer

SICK STEGMANN GmbH
Dürrheimer Strasse 36
78166 Donaueschingen, Germany
Tel.: +49 (0) 771 / 807 – 0
Fax: +49 (0) 771 / 807 – 100
Internet: http://www.sick.com/
E-mail: info@sick.com

Legal notice

Source document

This document is an original document of the SICK STEGMANN GmbH.

Table of contents

Table of figures

1 Background

The HIPERFACE® interface has been established as a motor feedback protocol by SICK in 1996. It features significant savings and simultaneously high performance for servo motor controllers.

HIPERFACE® is short for HIgh PERformance interFACE. It was specially developed for requirements of digital drive controllers and offers the user standardized and simplified mechanical and electrical interfaces.

1.1 Features

- One electrical interface for the motor controller for all applications and only one type of cable between controller and motor feedback system.
- Implementation of both low-end and high-end applications with only one electrical interface.
- Hybrid interface from
  - Analog process data channel on which sine and cosine signals are transmitted differentially with almost no delay
  - Bidirectional parameter channel according to the EIA-485 specification and UART standard for transmitting absolute position information and various other parameters
- Only 8 wires in encoder cable.
- Electronic type label for identification of the motor feedback system and storage of drive-related information in the motor feedback system.
- Wide temperature range, high shock- and vibration resistance, immunity to electromagnetic interference, and compact dimensions. Devices typically can be fitted inside the servo motor.
- Analog sine/cosine signals are available for speed control. These enable both high resolution for use at low speeds and sufficiently low signal bandwidth for control at high speeds.
- Cable lengths up to 100 m.
- The relationship of the absolute value to the mechanical motor shaft position can be electronically configured to adjust motor commutation.
- For position control of mechanically geared applications, motor feedback systems with the same physical dimensions are also available as absolute multiturn variants for absolute positioning over typically 4096 revolutions.

1.2 Typical Architecture

HIPERFACE® motor feedback systems are a mix of incremental encoders and absolute encoders and combine the advantages of both encoder types.

Initially the absolute value is acquired when the device is powered up and communicated via the bus-enabled parameter interface according to the EIA-485 specification. The drive controller uses this absolute value to initialize its position counter which is afterwards incrementally changed based on the analog sine/cosine signals.

The use of highly linear sine and cosine signals allows very high resolutions needed for speed control through angle interpolation. Simultaneously however the signal frequencies to be transmitted remain relatively low. An encoder with 1024 periods per revolution only produces a frequency of 204.8 kHz at typical maximum speed of 12,000 rpm which can be easily transmitted even over very long distances.

2 Protocol Specification

2.1 Physical Layer

2.1.1 Supply

HIPERFACE® specifies a standard voltage range for Slave supply of 7 ... 12 VDC (including tolerances). The supply voltage is measured on the Slave; accordingly maximum voltage drop over cable length must be considered by Master supply design.

HIPERFACE® specifies a maximum current consumption of 250 mA by the Slave at 7 VDC.

2.1.2 Interfaces

Sine/Cosine Process Channel

The HIPERFACE® process channel uses two sets of pseudo-differential analog voltage lines for transmission of sine/cosine signals. The signal connections are called “SIN” and “COS” while the voltage references are called “REFSIN” and “REFCOS”.

Sine/cosine voltages are transmitted according to the following parameters:

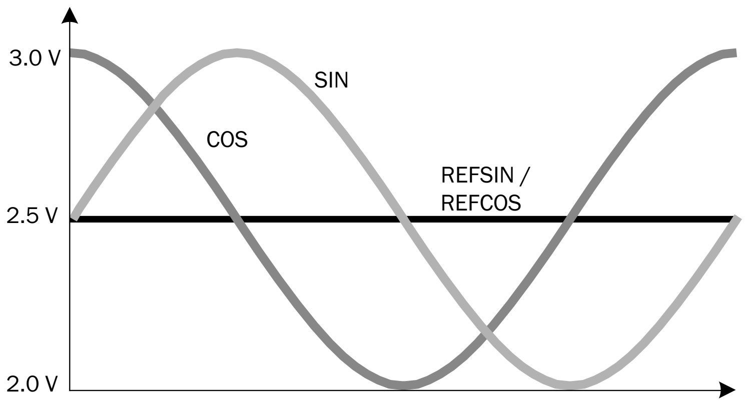


Figure 1: Process channel signal levels

Parameter	Value	Note
Single-ended SIN/COS peak-peak voltage	1.5 5 ... 3.45 V	to ground, over all environmental conditions
Differential SIN/COS peak-peak voltage	0.8 ... 1.2 V	differential to REFSIN/REFCOS
REFSIN/REFCOS level	2.2 ... 2.8 V	to ground, over all environmental conditions
SIN/COS bandwidth	0 ... 204.8 kHz	3 dB signal amplitude
Output load capacity	≥ ± 7 mA	

Table 1: Process channel signal parameters

Process channel schematics are specified in the Master chapter (see Analog Signal Input, p. 62).

EIA-485 Parameter Channel

The HIPERFACE® parameter channel uses signal levels according to the EIA-485 specification. HIPERFACE® uses a half-duplex two line connection (signals “A” resp. “-“ resp. “TxD-/RxD-“ and “B” resp. “+” resp. “TxD+/RxD+”). The optional SC line (common ground) is not used.

EIA-485 transceiver components have to be chosen according to the required data rates (see Data Format, p. 9).

The HIPERFACE® parameter channel requires termination resistors at both ends of the connection lines. Additionally, pull-up/pull-down resistors are also required to reduce interference.

If HIPERFACE® is used in a bus system only the last Slave in the connection line must have a line termination.

Note:

HIPERFACE® bus system is obsolete and not recommended for future use.

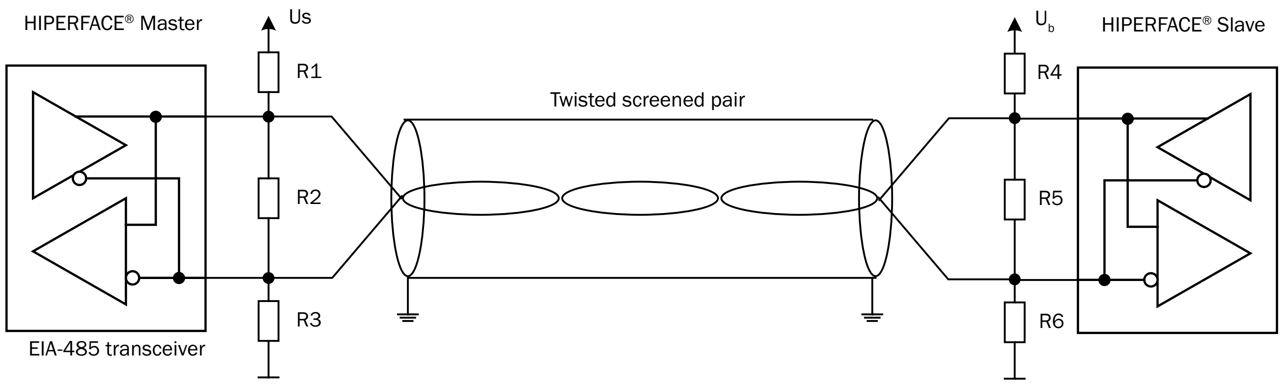


Figure 2: Parameter channel schematics

HIPERFACE®	Part	Value	Note
Master	R1	1 kΩ	In Bus system dimensioning by user
	R2	130 Ω	
	R3	1 kΩ	In Bus system dimensioning by user
Slave	R4	1 kΩ	In Bus system 10 kΩ
	R5	130 Ω	No termination in Bus system
	R6	1 kΩ	In Bus system 10 kΩ

Table 2: Parameter channel components

Note:

Recommended tolerance for all resistors is 1%.

2.1.3 Connectors and Cabling

HIPERFACE® does not specify or recommend specific connectors.

Cabling between Master and Slave foresees a dedicated, separate cable with eight wires and screening. HIPERFACE® specifies a maximum cable length of 100 m.

The system screen concept is very important and influences the achievable signal performance of the whole system. A large area screen applied to both sides – motor and controller – is recommended for best results. The screen should be connected to protective earth. A separate equipotential bonding conductor must be used if large equalizing currents are expected in a system.

Cables for connecting HIPERFACE® Master and Slave are recommended to have the following physical parameters:

Parameter	Typical	Unit	Note
Wire gauge	4 x 2 x 0.34	mm²	Twisted pairs. Diameter corresponds to AWG22.
Capacity	60	pF/m	Wire-to-wire, at 800 Hz
Impedance	100…120	Ω	At 1 MHz
Shielding coverage	>85	%	Braided shield
Rated voltage	36	V	AC/DC
Current load	2	A	
Cable color	Green		Acc. Desina, similar to RAL6018

Table 3: Recommended HIPERFACE® cable properties

Note:

Various motor and drive manufacturers have proprietary standards for encoder cables that can be considered for cable choice.

In addition to the above mentioned characteristics application-specific cable properties (e.g. temperature rating, chemical resistances, UV resistance, flexibility etc.) or compliance ratings (e.g. UL listing) should be considered

ATTENTION:

Running HIPERFACE® signals inside a motor cable is not foreseen by SICK.

2.2 Data Link Layer

2.2.1 Data Format

HIPERFACE® uses an asynchronous half-duplex UART protocol with binary data.

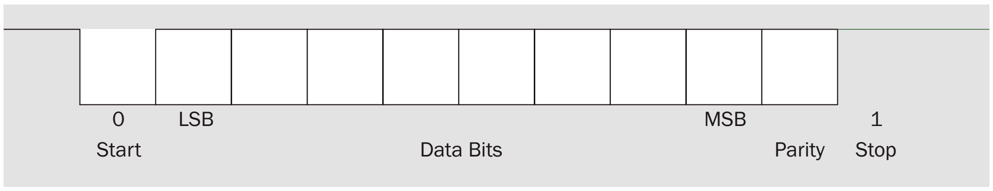


Figure 3: General UART character format

The following UART communication parameters are used:

Parameter	Default setting	Note
Data rate	9600 Baud	Can be programmed to 600, 1200, 2400, 4800, 9600, 19200, 38400 Baud by user
Data bits	8, LSB first	
Start bits	1	
Stop bits	1	
Parity	Even	Can be programmed to Odd, Even, None

Table 4: UART settings

Each data transmission consists of a Master request frame followed by a Slave response frame. The Master transmission must be terminated through a timeout condition after the last byte is transmitted. The Slave transmission will initiate within a specified response time.

The Slave will never initiate communication on its own.

Note:

Data rates of less than 9600 Baud are obsolete and not recommended for future products. Use of data rates less than 9600 Baud in existing HIPERFACE® Master implementations is not known to SICK.

Note:

Previously the default parity for HIPERFACE® Slaves was specified as “Odd”. The actually implemented default parity check always conformed to an “Even” parity definition. Accordingly this specification corrects this definition.

2.2.2 Frame Timing

Each character of the Master transmission must be followed by a subsequent character within the timeout condition. If the Slave receives no further character within the timeout condition the message will be considered terminated and processed.

Accordingly the end of a Master transmission must be followed by a pause of at least the length of the timeout condition.

The end of the Slave transmission is also indicated by a pause of at least the length of the timeout condition.

The timeout condition can be programmed to be either 22 s/Data rate or 55 s/Data rate (default, i.e. 5.73 ms for 9600 Baud).

Note:

The original HIPERFACE® Manual specifies data rates of 11 s/Data rate or 44 s/Data rate. This older figure did not consider the fact that typical UART interrupts of microcontrollers are generated at the end of the first received character.

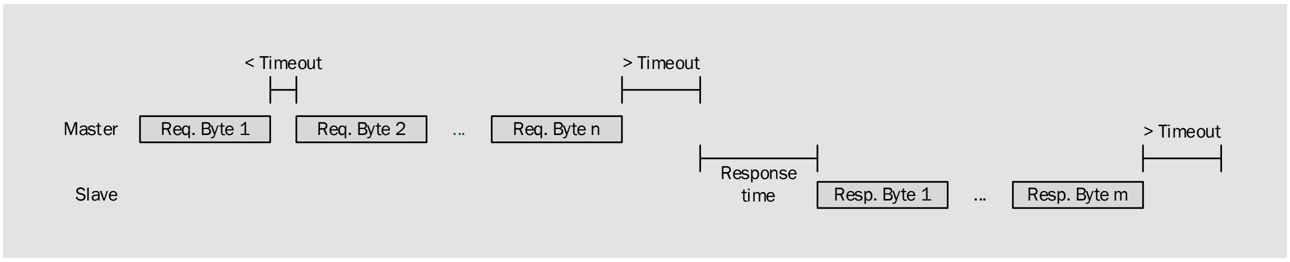


Figure 4: Transmission cycle

2.3 Network Layer

HIPERFACE® allows the use of a bus topology. Accordingly an addressing scheme is implemented.

Each frame contains an address byte, command, and optionally data bytes of varying length and a trailing checksum to detect transmission errors. This specification applies both for Master and Slave transmissions.

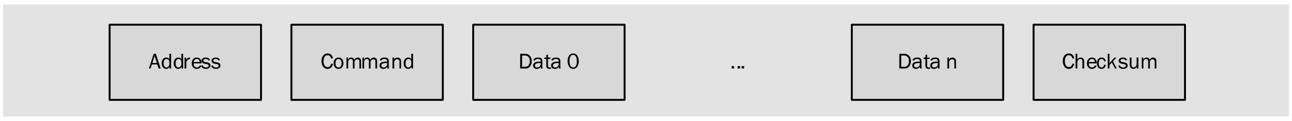


Figure 5: Frame composition

2.3.1 Addressing

A HIPERFACE® address consists of one byte. The available address space allows for 32 different address values.

Bit	Definition	Note
0 ... 4	Adress value	Default: 00h
5	Fixed ‘0’	
6	Fixed ‘1’	
7	Fixed ‘0’	

Table 5: Address byte specification

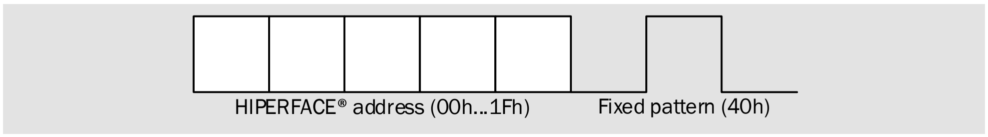


Figure 6: Address format

The default address for a HIPERFACE® device is 0 (40h).

HIPERFACE® also implements a broadcast address (FFh) with which a Master can address all bus-linked HIPERFACE® Slaves to perform a selected command. Each Slave will acknowledge with their specific address.

2.3.2 Command Format

A HIPERFACE® command consists of at least one byte. The available address space allows for 64 different command values. Depending on the specific command additional bytes are required (see Application Layer, p. 14). Additional bytes can use all 256 byte values.

Bit	Definition	Note
0 ... 5	Command value	
6	Fixed '1'	
7	Error bit	

Table 6: Command byte specification

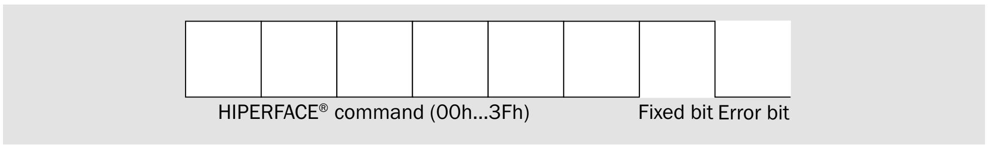


Figure 7: Command format

2.3.3 Checksum

In the HIPERFACE® protocol each frame is terminated by a checksum, both by Master and Slave. The receiver of a frame performs a transmission failure check with the help of this checksum and can react accordingly.

The HIPERFACE® checksum is defined through an XOR operation on all frame bytes including address.

Frame	Checksum
40h 42h	02h
40h 4Eh 70h 00h	7Eh

Table 7: Checksum examples

2.3.4 Fault Indications

HIPERFACE® uses two different ways to indicate detected faults. Faults are always only communicated from Slave to Master.

Warning Bit

If a HIPERFACE® Slave detects an operational fault unrelated to a current command request, a warning bit will be issued and a corresponding error code is stored for query by the "Read Encoder Status" command.

Operational faults are identified in Fault Messages, p. 45.

A HIPERFACE® Slave can store error codes of up to four operational faults in parallel.

The warning bit is active if the Error bit (bit 7) of the HIPERFACE® command is set by the Slave.

In this condition the original command request by the Master will be correctly responded to by the Slave.

The Slave will issue the warning bit with each following command response until the Master has read out the error code (with command 50h) or performed a Slave reset (command 53h). If multiple operational faults are stored, the Master must read out each associated error code separately to reset the Error bit.

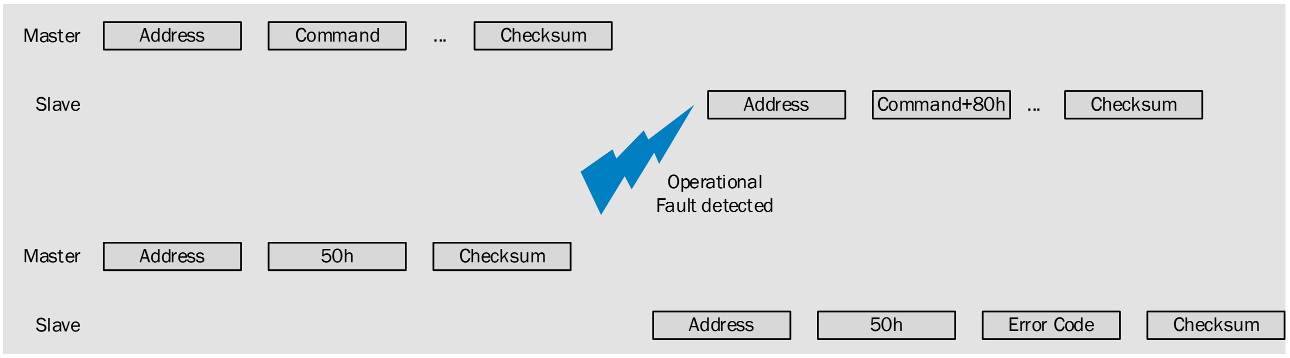


Figure 8: Warning Indication (one Operational Fault)

Error Response

If a HIPERFACE® Slave detects a condition that does not allow processing a Master command request an error response will be issued.

Typical conditions for this behavior are protocol errors, bad command arguments, or internal encoder faults. Error conditions are identified in Fault Messages, p. 45.

In case of an error condition the Slave will cancel processing of the command request and respond with an error response. The error response contains the status command value 50h followed by an appropriate error code.

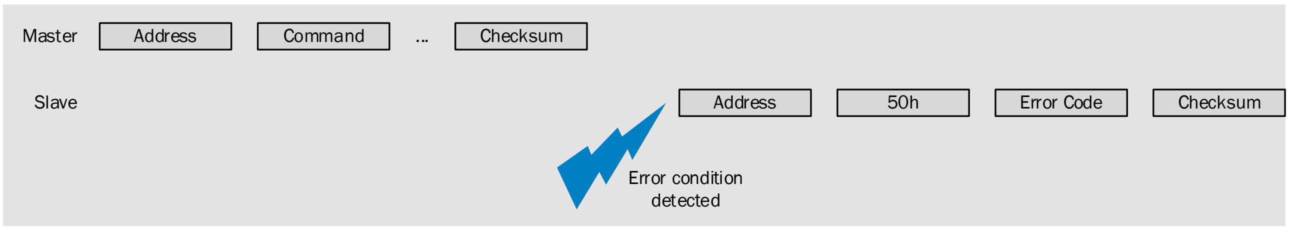


Figure 9: Error Response

If both a warning and an error condition are detected for the same command response the Slave will issue an error response with a set warning bit (command value D0h). The error code corresponding to the warning has to be read out as lined out in Warning Bit, p. 12.

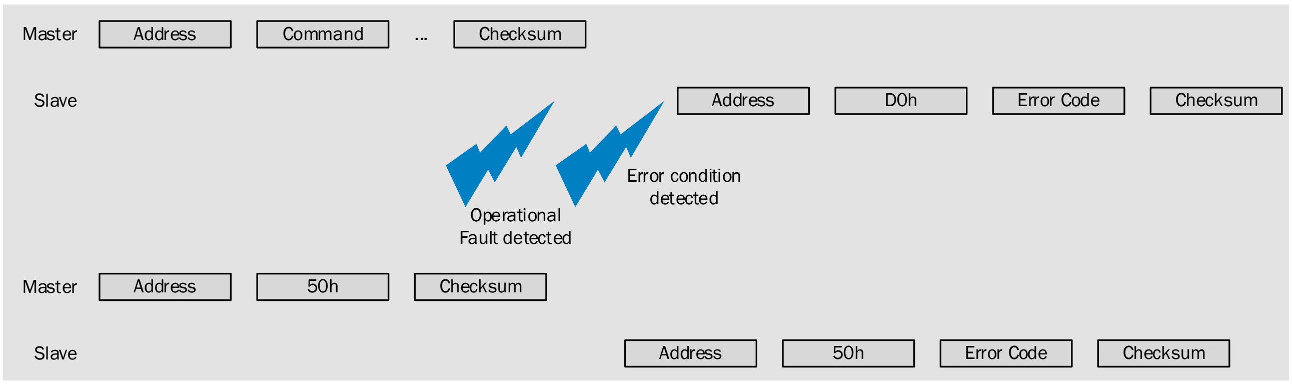


Figure 10: Error and warning indication in parallel

Note:
If the Slave detects error conditions critical to position measurement further Master position requests will be answered with error messages until the Slave is reset (command 53h). Critical error conditions are indicated in Fault Messages, p. 45.

2.4 Application Layer

The application layer of the HIPERFACE® protocol is defined through the state machine, the available commands, and warning/error codes.

Commands are defined by the following values:

Parameter	Description	Note
Command identifier	Command address according to Command Format, p. 12	40h … 7Fh
Access code	Specification if command can only be accessed with transmitted code byte ("code 0"). Default value for code 0 is 55h.	Other codes only used with data fields
Response time	Specification of maximum response time of Slave after Master request timeout (in ms).	Response time does not include transmission times and protocol timeout
Master data length	Byte count of necessary data bytes for request	Does not include address, command identifier, or checksum
Slave data length	Byte count of data bytes for response	Does not include address, command identifier, or checksum
Master data values	Definition of data bytes for request	
Slave data values	Definition of data bytes for response	
Error conditions	Associated error conditions	Does not include global warning indications

Table 8: Command definitions

2.4.1 HIPERFACE® State Machine

The following figure shows the states of the HIPERFACE® communication and the transition conditions.

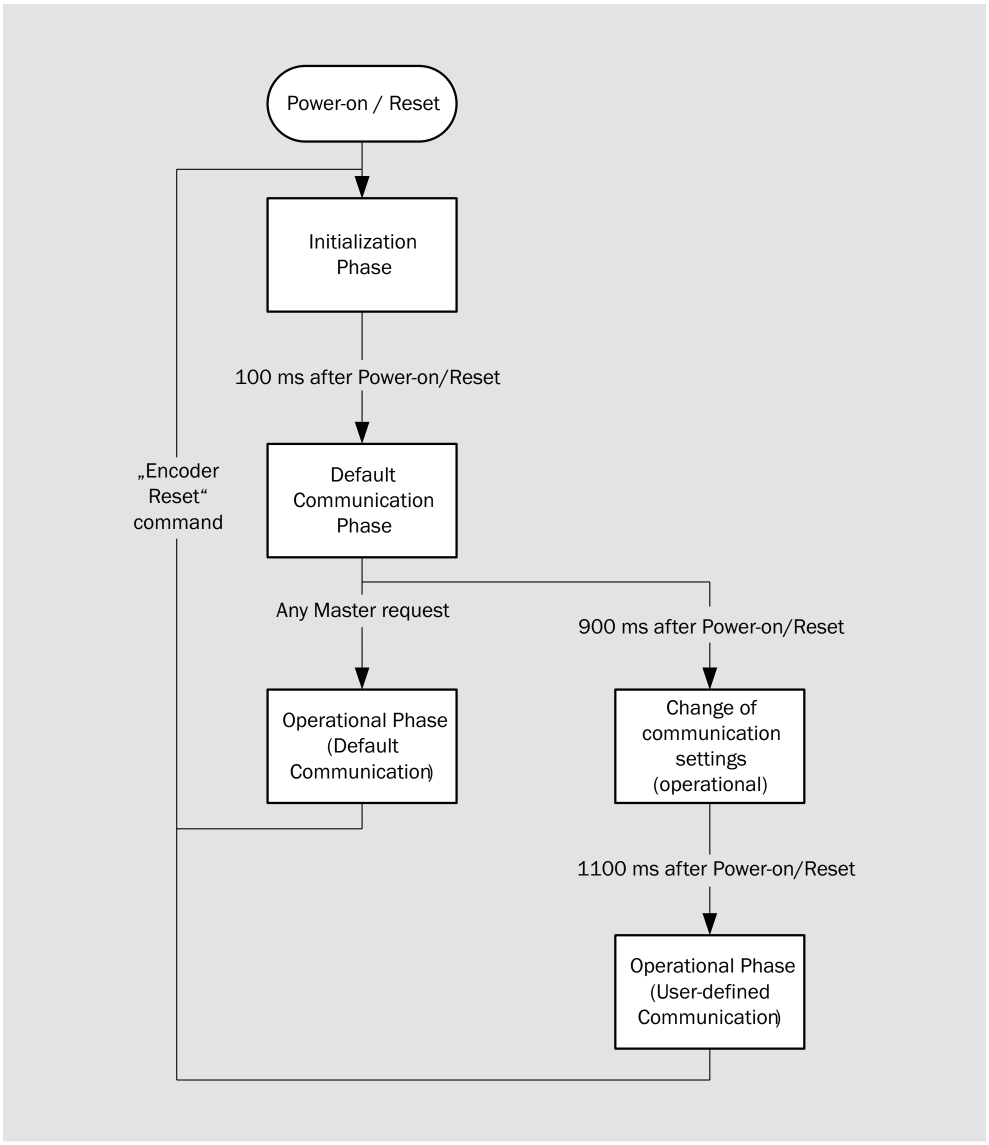

Power-on / Reset
Initialization Phase
100 ms after Power-on/Reset
Default Communication Phase
„Encoder Reset“ command
Any Master request
900 ms after Power-on/Reset
Operational Phase (Default Communication)
Change of communication settings (operational)
1100 ms after Power-on/Reset
Operational Phase (User-defined Communication)

Figure 11: HIPERFACE® State Machine

The states have the following meaning:

State	Available functions	Exit condition
Initialization	No UART communication **NOTE:** In legacy encoders Sin/Cos signals are available during initialization but usually not calibrated. HIPERFACE® Master must not use Sin/Cos signals during initialization.	To “Default Communication”: Automatically 100 ms after power-on/reset **NOTE:** Newer HIPERFACE® Slaves might deviate from this figure, especially if they support safety functions. Product datasheets will indicate the actual value. Even in these cases the duration shall not exceed 200 ms after power-on/reset.
Default Communication	UART communication with default settings (9600 Baud, even parity, 5x timeout) User commands	To “Operational Phase (default communication)”: Any single Master request
		To “Change of communication settings (operational)”: No message until 900 ms after power-on/reset **NOTE:** Some legacy encoders exceed this value. Product datasheets will indicate the actual value.
Operational (default communication)	UART communication with default settings (9600 Baud, even parity, 5x timeout) User commands	To “Initialization”: “Encoder Reset” command
Change of communication settings (operational)	No UART communication Sin/Cos signals available	To “Operational (user-defined communication)”: Automatically 1100 ms after power-on/reset
Operational (user-defined communication)	UART communication with user-defined settings User commands	To “Initialization”: “Encoder Reset” command

Table 9: HIPERFACE® States and Transitions

2.4.2 User Commands

Note:

For all commands the following error conditions can be detected and transmitted:

- “Parity error” (09h)
- “Checksum error” (see Checksum error (0Ah), p. 51)
- “Unknown command” (see Unknown command (0Bh), p. 51)
- “Wrong command length” (see Wrong command length (0Ch), p. 52)

The following table gives a summary of all existing user commands.

Command identifier	Function	Max. respon-se time [ms]	Reference
42h	Read Position	10	Read Position, p. 18
43h	Set Position	40	Set Position, p. 19
44h	Read Analog Value	5	Read Analog Value, p. 21
46h	Read Counter	5	Read Counter, p. 22
47h	Increment Counter	30	Increment Counter, p. 22
49h	Reset Counter	30	Reset Counter, p. 23
4Ah	Read Data	30	Read Data, p. 26
4Bh	Store Data	250	Store Data, p. 27
4Ch	Data Field Status	5	Data Field Status, p. 29
4Dh	Create Data Field	70	Create Data Field, p. 30
4Eh	Memory Status	5	Memory Status, p. 32
4Fh	Set Access Code	40	Set Access Code, p. 32
50h	Read Encoder Status	5	Read Encoder Status, p. 33
52h	Read Type Label	5	Read Type Label, p. 33
53h	Encoder Reset	-	Encoder Reset, p. 35
55h	Set Encoder Address	40	Set Encoder Address, p. 36
56h	Read Version	5	Read Version, p. 36
57h	Set Serial Interface	40	Set Serial Interface, p. 37
67h	Temporarily Set Serial Interface	5	Temporarily Set Serial Interface, p. 39
6Ah	Set Position with Synchro-nization	40	Set Position with Synchronization, p. 41
6Bh	Sensor Adjustment	40	Sensor Adjustment, p. 43
6Ch	Read Synchronization Offset	30	Read Synchronization Offset, p. 44

Table 10: HIPERFACE® user command summary

Read Position

The “Read Position” command allows the motor controller to read out the absolute position value of the encoder.

The datasheet of the HIPERFACE® Slave must specify the maximum speed during which the absolute position value can be acquired.

Note:

This value is typically 6000 min^{-1} for rotary encoders.

The position acquisition is latched to the falling edge of the start bit of the first byte of the Slave response (address byte).

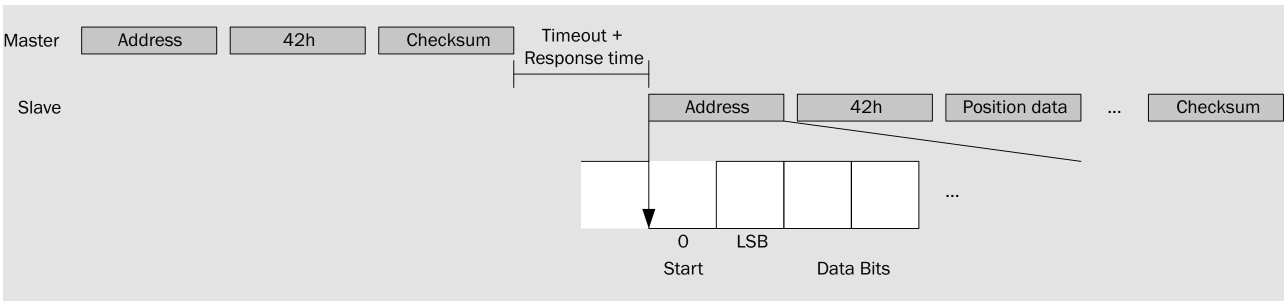


Figure 12: “Read Position” latching point

The absolute position value serves to uniquely identify one of the analog sine/cosine periods. It has a resolution of 5 bits per period in all HIPERFACE® Slaves.

Encoder family	Sine/Cosine periods	Absolute position resolution
SEy34/37/52	16	512 steps/turn (9 bits)
SKx36	128	4096 steps/turn (12 bits)
SRx50	1024	32768 steps/turn (15 bits)
TTK50	1/mm	32 steps/mm (max. 940 mm)
TTK70	1/mm	32 steps/mm (max. 4000 mm)

Table 11: HIPERFACE® Slave position resolution examples

Parameter	Value	Note
Command identifier	42h	
Access code	-	
Response time	10 ms	
Master data length	0 bytes	
Slave data length	4 bytes	
Master data values	-	
Slave data values	Absolute position as unsigned 32 bit value, MSB first	
Error conditions	02h (Angle offset fault) 0Dh (Wrong argument) 1Dh (LED current high) 1Fh (Speed high) 20h (Singleturn fault) 21h (Multiturn amplitude fault) 22h (Multiturn sync fault) 23h (Multiturn vectorlength fault)	

Table 12: Read Position

Set Position

The “Set Position” command allows the motor controller to store a position offset in the HIPERFACE® Slave. This command is used to simplify usage of the absolute position for the commutation of the motor. Typically the HIPERFACE® Master will want to set the Slave absolute position to 0 at a locked motor shaft position.

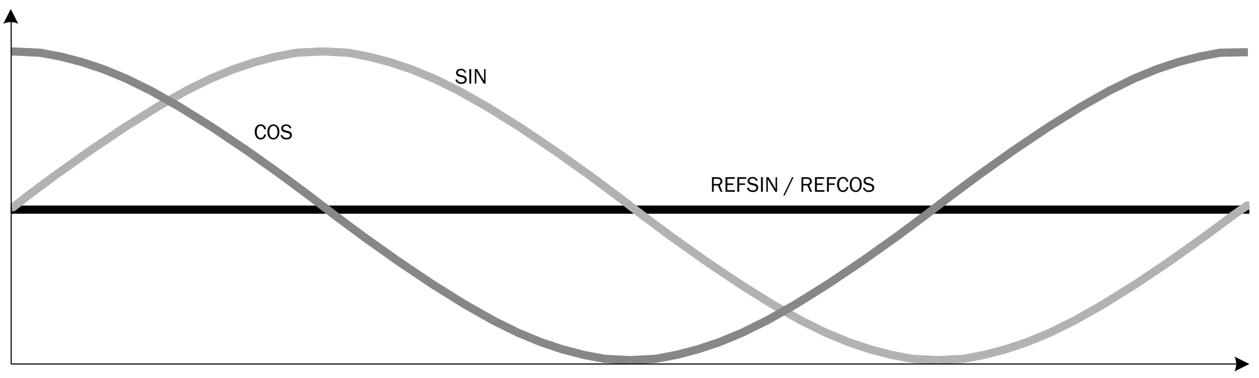
The HIPERFACE® Master must transmit the desired position value (preset) with the command “Set Position”. The Slave will calculate the corresponding position offset and store it internally.

ATTENTION:

As no synchronization is provided for this function the command must be called at standstill of the motor.

The command “Set Position” can only be activated if “Code0” is transmitted to guard against improper use and impairment of motor function.

The command “Set Position” allows setting of any position preset within the measuring range. This allows changing the phase shift between absolute position and analog signals of the HIPERFACE® Slave. When shipped a Slave always has a phase shift of 0°.



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Default alignment of absolute position (5 LSBs) to one Sin/Cos period

27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Example for shifted alignment of absolute position to Sin/Cos period

Figure 13: “Set position” phase shift

Note:

The phase shift is determined by the 5 LSBs of the absolute position.

Note:

A phase shift other than 0° limits the ability of the Master to synchronize absolute position to analog signals. Therefore it is recommended to only select position presets where the phase shift remains 0°. On the other hand, for encoders with low sine/cosine period count this can lead to inacceptable commutation offset errors. In this case it is recommended to store the commutation offset in a Slave data field and perform the offset calculation Master-side.

Parameter	Value	Note
Command identifier	43h	
Access code	Code0	Default: 55h
Response time	40 ms	
Master data length	5 bytes	
Slave data length	0 bytes	
Master data values	Byte 0...3: Absolute position preset as unsigned 32 bit value, MSB first Byte 4: Code0	
Slave data values	-	
Error conditions	02h (Angle offset fault) 05h (I2C communication fault) 06h (EE checksum fault) 0Dh (Wrong argument) 0Fh (Wrong access code) 1Dh (LED current high) 20h (Singleturn fault) 21h (Multiturn amplitude fault) 22h (Multi sync fault) 23h (Multiturn vectorlength fault)	

Table 13: Set Position

Read Analog Value

The “Read Analog Value” command allows accessing sensor sources in the HIPERFACE® Slave other than position. A typical sensor source might be encoder temperature.

Individual sensor sources are addressed by the Master with a “CHANNEL” selector.

The available sensor sources and their encoding and resolution are always specified in the product data sheet of a HIPERFACE® Slave.

Sensor Source	CHANNEL selector	Encoding, Resolution	Encoder families
Encoder temperature	48h	Value measured in 1 °C steps, returned as signed 16 bit value (2’s complement)	SEy34/37/52, SEK90/160/260, SKx36, SRx50-G2, TTK50/70
Legacy encoder temperature	F0h	Value measured in 2.048 °C steps, returned as signed 16 bit value (2’s complement) Temperature = ([F0h]+40)/2.048	SCx, SEy37, SKx36, SRx50-G2

Table 14: HIPERFACE® Slave sensor source examples

Note:

HIPERFACE® Slaves typically measure temperature on their main circuit board. Accordingly the returned temperature value will include power dissipation of electronic components. A specified ambient temperature refers to a temperature measurement point that can be accessed external to the HIPERFACE® slave.

Parameter	Value	Note
Command identifier	44h	
Access code	-	
Response time	5 ms	
Master data length	1 byte	
Slave data length	3 bytes	
Master data values	CHANNEL selector	Choices according to product data sheet
Slave data values	Byte 0: CHANNEL selector Byte 1...2: Sensor value	Encoding, resolution as specified in data sheet
Error conditions	0Dh (Wrong argument)	

Table 15: Read Analog Value

Read Counter

The “Read Counter” command returns the value of an implemented non-volatile counter of the HIPERFACE® Slave.

The counter has a width of 24 bits and allows 1 million write cycles.

Parameter	Value	Note
Command identifier	46h	
Access code	-	
Response time	5 ms	
Master data length	0 bytes	
Slave data length	3 bytes	
Master data values	-	
Slave data values	Byte 0…2: Counter as unsigned 24-bit value	
Error conditions	06h (EE checksum fault)	

Table 16: Read Counter

Increment Counter

The “Increment Counter” command increments the implemented non-volatile counter value of the HIPERFACE® Slave by one.

The counter has a width of 24 bits and allows 1 million write cycles.

Parameter	Value	Note
Command identifier	47h	
Access code	-	
Response time	30 ms	
Master data length	0 bytes	
Slave data length	0 bytes	
Master data values	-	
Slave data values	-	
Error conditions	05h (I2C communication fault) 06h (EE checksum fault) 08h (Counter fault)	

Table 17: Increment Counter

Reset Counter

The “Reset Counter” command writes 0 to the implemented non-volatile counter value of the HIPERFACE® Slave.

The counter has a width of 24 bits and allows 1 million write cycles.

Parameter	Value	Note
Command identifier	49h	
Access code	Code0	Default: 55h
Response time	30 ms	
Master data length	1 byte	
Slave data length	0 bytes	
Master data values	Byte 0: Code0	
Slave data values	-	
Error conditions	05h (I2C communication fault) 06h (EE checksum fault) 0Fh (Wrong access code)	

Table 18: Reset Counter

Data Field Specification

HIPERFACE® Slave devices implement a function that allows users to store arbitrary data in non-volatile memory of the Slave. Typical use cases for this function include storage of motor data (type label, serial no) or diagnostic data (motor and/or encoder faults).

HIPERFACE® allows storage in the EEPROM of the Slave with the following specification:

– Data is stored in “data field” units that have a size of 16 to 128 bytes (in multiples of 16 bytes). Prior to data storage a data field must be created with a predetermined size.

– Data fields are identified by a consecutive “data field number” starting with 0.

– Data fields can be linked to one of four access code values (Code0, Code1, Code2, or Code3). The access code must be transmitted by the user to read or write a data field.

– Data fields can be set as “read-only” or “write-enabled”. Only with “write-enabled” can a data field be written to.

– The status of a data field can be changed (linked code, read/write mode). For the last data field (highest data field number) size of the data field can also be increased.

– Data fields can be deleted.

The datasheet of a HIPERFACE® Slave must indicate the amount of bytes that a user can store in data fields. The amount of bytes specified in this way is 100% user content. All values needed for data field management are stored separately.

Note:

Typical user memory size in HIPERFACE® Slaves is 1792 bytes.

Extended Type Label Specification

Data fields are also used for the “Extended Type Label” function.

Newer HIPERFACE® Slaves allow the Master to read out encoder parameters explicitly. The access to encoder parameters of the Extended Type Label is identical to reading out user data fields, using the virtual data field number FFh.

Note:

Extended Type Label data is not stored in nonvolatile memory for users but in a separate memory.

Note:

The Extended Type Label is not counted as an active data field in the “data field count” of the “Memory Status” command.

Note:

Legacy HIPERFACE® Slaves identify with an implicit “type code” (see Read Encoder Status, p. 33). Masters have to implement the corresponding encoder parameters for a given code during design. Some HIPERFACE® Slaves implement the Extended Type Label while also indicating the legacy “type code”. This has to be indicated in the product data sheet. Slaves that only implement the Extended Type Label have to use the “type code” FFh.

Encoder type	Type Code	Extended Type Label
Legacy	Not FFh	No
Recent	Not FFh	Yes
New and future	FFh	Yes

Table 19: Extended Type Label scenarios

The Extended Type Label is accessible without access code and is “read-only”. It cannot be deleted.

The Extended Type Label has a minimum length of 64 bytes. The actual size can be higher if bit 15 of the parameter selector is enabled (see following table, offset 1Ch).

The HIPERFACE® Master can read out any or all bytes within the Extended Type Label as specified in a “Read Data” command (see Read Data, p. 26).

Data in the Extended Type Label is specified according to the following table.

Offset	Length (bytes)	Description	Note
00h	1	XOR checksum complete Extended Type Label	
01h	1	Encoder type Bit 0: 0 = rotary, 1 = linear Bit 1: 0 = unipolar counting, 1 = bipolar counting Other bits: not implemented, read “0”	
02h	4	Encoder resolution (unsigned 32 bit value) No. of periods/turn for rotary encoder Period length in 1 nm steps for linear encoder	
06h	4	Encoder range (unsigned 32 bit value) No. of coded revolutions for rotary encoder No. of periods for linear encoder	
0Ah	18	Encoder designation (ASCII string, left-justified, with trailing 00h characters)	
1Ch	2	Parameter selector (specification of available values in Extended Type Label) Bit 0: Temperature CHANNEL selector disabled/enabled Bit 1: Temperature min value disabled/enabled Bit 2: Temperature max value disabled/enabled Bit 3: LED current CHANNEL selector disabled/enabled Bit 4: LED current min value disabled/enabled Bit 5: LED current max value disabled/enabled Bit 6: Vector length CHANNEL selector disabled/enabled Bit 7: Vector length min value disabled/enabled Bit 8: Vector length max value disabled/enabled Bit 9: Speed max value disabled/enabled Bit 10: Acceleration max value disabled/enabled Bit 11...14: not implemented, read “0” Bit 15: Next selector disabled/enabled	If next selector enabled (bit 15 = 1), 2nd parameter selector follows at offset 40h, and 2nd parameter entries at offset 42h etc.
1Eh	30	Parameter values as defined in Parameter selector. Each parameter is a 16 bit value Byte 0...1: Temperature CHANNEL value (for command 44h) Byte 2...3: Temperature min value (signed, in °C) Byte 4...5: Temperature max value (signed, in °C) Byte 6...7: LED current CHANNEL value (for command 44h) Byte 8...9: LED current min value (unsigned, in mA) Byte 10...11: LED current max value (unsigned, in mA) Byte 12...13: Vector length CHANNEL value (for command 44h) Byte 14...15: Vector length min value (unsigned, arbitrary units) Byte 16...17: Vector length max value (unsigned, arbitrary units) Byte 18...19: Speed max value (unsigned, in rpm or m/min) Byte 20...21: Acceleration max value (unsigned, in krad/s^2 or m/s^2) Byte 22...29: not implemented, read “0”	
3Ch	4	Not implemented, read “0”	

Table 20: Extended Type Label data

Read Data

The “Read Data” command allows reading out data field values out of a HIPERFACE® Slave.

The user specifies the following information:

- Data field number (as created with command 4Dh, see Create Data Field, p. 30) or Extended Type Label code FFh (see Extended Type Label Specification, p. 24).
- Offset within the data field; must be within data field size (0 ... size-1).
- Number of bytes to read out (min. 1 byte, max. 128 bytes). Offset plus byte count must not be larger than data field size.
- Access code. If the data field is created without requirement for access code this byte can contain any value but must be included in the Master request.

Parameter	Value	Note
Command identifier	4Ah	
Access code	Code0 ... Code3	Depending on user definition of data field
Response time	30 ms	
Master data length	4 bytes	
Slave data length	4...131 bytes	Depending on requested number of bytes (1...128)
Master data values	Byte 0: Data field number Byte 1: Offset in data field Byte 2: Number of bytes Byte 3: Access code	
Slave data values	Byte 0: Data field number Byte 1: Offset in data field Byte 2: Number of bytes Byte 3...130: Data	
Error conditions	03h (Data field partition table fault) 05h (I2C communication fault) 0Fh (Wrong access code) 11h (Wrong data field offset) 12h (Wrong data field number)	

Table 21: Read Data

Store Data

The “Store Data” command allows storing user data to data fields of a HIPERFACE® Slave.

The user cannot store data to a data field that was created with or changed to “read-only” status.

Note:

Data fields will not be automatically increased in size if the maximum size is reached. A data field must be explicitly enlarged with the Create Data Field command (see Create Data Field, p. 30). This is only possible for the data field created last (highest data field number).

The user specifies the following information:

– Data field number (as created with command 4Dh, see Create Data Field, p. 30).
– Offset within the data field; must be within data field size (0 ... size-1).
– Number of bytes to write (min. 1 byte, max. 128 bytes). Offset plus byte count must not be larger than data field size.
– Access code. If the data field is created without requirement for access code this byte can contain any value but must be included in the Master request.
– Data bytes.

Parameter	Value	Note
Command identifier	4Bh	
Access code	Code0 … Code3	Default: 55h
Response time	250 ms	
Master data length	5…132 bytes	
Slave data length	3 bytes	
Master data values	Byte 0: Data field number Byte 1: Offset in data field Byte 2: Number of bytes Byte 3: Access code Byte 4…131: Data	
Slave data values	Byte 0: Data field number Byte 1: Offset in data field Byte 2: Number of bytes	
Error conditions	03h (Data field partition table fault) 05h (I2C communication fault) 06h (EE checksum fault) 0Eh (Data field is read-only) 0Fh (Wrong access code) 11h (Wrong data field offset) 12h (Wrong data field number)	

Table 22: Store Data

Data Field Status

The “Data Field Status” command allows reading out data field status of a HIPERFACE® Slave.

The data field number can be of an existing data field (as created with command 4Dh, see Create Data Field, p. 30) or Extended Type Label code FFh (see Extended Type Label Specification, p. 24).

Data field status is returned as one byte with the same encoding as for the “Create Data Field” command (see Create Data Field, p. 30).

Parameter	Value	Note
Command identifier	4Ch	
Access code	-	
Response time	5 ms	
Master data length	1 byte	
Slave data length	2 bytes	
Master data values	Byte 0: Data field number	
Slave data values	Byte 0: Data field number Byte 1: Status Bit 0...2: Data field size in 16 byte blocks (size/16 – 1), i.e. 0 = 16 bytes, 1 = 32 bytes, ..., 7 = 128 bytes Bit 3: Code disabled/enabled Bit 4..5: Access code, 0 = Code0, 1 = Code1, 2 = Code2, 3 = Code3 Bit 6: 0 = Read-only, 1 = Write-enabled Bit 7: 1 = Data field exists	Note: Code0 for data fields is identical to Code0 for access to some commands
Error conditions	03h (Data field partition table fault) 05h (I2C communication fault) 12h (Wrong data field number)	

Table 23: Data Field Status

Create Data Field

The “Create Data Field” command allows creating new or modifying existing data fields of a HIPERFACE® Slave.

For creating new data fields the user specifies the following information:

– Data field number. Must be identical to the next available number (i.e.: No existing data fields – 0, three existing data fields – 3 etc.)

– Data field status as one byte with the same encoding as for the “Data Field Status” command (see Data Field Status, p. 29). Bit definitions see below in table.

Note:

For creating a data field bit 7 must be set to 1.

– Access code. The transmitted access code byte must correspond to the selected access code for the data field. This is even necessary during data field creation if the access code is disabled for the data field.

For modifying data fields the user specifies the following information:

– Data field number (as created previously).

– Data field status as one byte with the same encoding as for the “Data Field Status” command (see Data Field Status, p. 29). Bit definitions see below in table.

Note:

For modifying a data field bit 7 must be set to 1, for deleting a data field the bit must be 0.

Note:

Deletion or data field size change is only possible for the data field created last (highest data field number).

Note:

For deletion all status bits (except bit 7) must be identical to the stored values of the data field.

– Access code. The transmitted access code byte must correspond to the selected access code for the data field. This is even necessary during data field modification if the access code is disabled for the data field.

Parameter	Value	Note
Command identifier	4Dh	
Access code	Code0 ... Code3	Depending on user definition of data field
Response time	70 ms	
Master data length	3 bytes	
Slave data length	2 bytes	
Master data values	Byte 0: Data field number Byte 1: Status Bit 0...2: Data field size in 16 byte blocks (size/16 – 1), i.e. 0 = 16 bytes, 1 = 32 bytes, ..., 7 = 128 bytes Bit 3: Code disabled/enabled Bit 4..5: Access code, 0 = Code0, 1 = Code1, 2 = Code2, 3 = Code3 Bit 6: 0 = Read-only, 1 = Write-enabled Bit 7: 0 = Delete data field, 1 = Create/Modify data field Byte 2: Access code	
Slave data values	Byte 0: Data field number Byte 1: Status Bit 0...2: Data field size in 16 byte blocks (size/16 – 1), i.e. 0 = 16 bytes, 1 = 32 bytes, ..., 7 = 128 bytes Bit 3: Code disabled/enabled Bit 4..5: Access code, 0 = Code0, 1 = Code1, 2 = Code2, 3 = Code3 Bit 6: 0 = Read-only, 1 = Write-enabled Bit 7: 0 = Data field deleted, 1 = Data field crea-ted/modified	Note: Code0 for data fields is identical to Code0 for access to some commands
Error conditions	03h (Data field partition table fault) 05h (I2C communication fault) 06h (EE checksum fault) 0Fh (Wrong access code) 10h (Fixed data field size) 12h (Wrong data field number)	

Table 24: Create Data Field

Memory Status

The “Memory Status” command returns information on free user memory and existing data fields of the HIPERFACE® Slave.

Free user memory is returned as number of 16 byte blocks available (e.g. for 1792 free bytes a value of 112 / 70h is returned).

Parameter	Value	Note
Command identifier	4Eh	
Access code	-	
Response time	5 ms	
Master data length	0 bytes	
Slave data length	2 bytes	
Master data values	-	
Slave data values	Byte 0: Free user memory (in 16 byte blocks) Byte 1: Number of existing data fields	
Error conditions	06h (EE checksum fault)	

Table 25: Memory Status

Set Access Code

The “Set Access Code” command allows changing access code bytes of the HIPERFACE® Slave.

Each of the four different code bytes (Code0, Code1, Code2 and Code3) can be changed.

Note:

The default value of each code byte is 55h.

Note:

HIPERFACE® offers no function for users to reset access code bytes to default values.

Parameter	Value	Note
Command identifier	4Fh	
Access code	-	
Response time	40 ms	
Master data length	3 bytes	
Slave data length	1 byte	
Master data values	Byte 0: Access code number (00h...03h) Byte 1: Old access code Byte 2: New access code	Default codes: 55h
Slave data values	Byte 0: Access code number (00h...03h)	

Parameter	Value	Note
Error conditions	05h (I2C communication fault) 06h (EE checksum fault) 0Dh (Wrong argument) 0Fh (Wrong access code)	

Table 26: Set Access Code

Read Encoder Status

The “Read Encoder Status” command returns active warning and error codes of the HIPERFACE® Slave.

For each “Read Encoder Status” command one active warning or error is returned from the Slave error stack. HIPERFACE® Slaves can store up to four active warnings or errors in the stack.

If a warning or error occurs multiple times the corresponding code will only be stored once in the stack until it has been read out.

If more than four warning or errors occur before the stack is read out older entries will be deleted from the stack.

If no warning or error is active “Read Encoder Status” will return status code 00h.

An active warning indication (see Warning Bit, p. 12) can only be reset by the user if he reads out all active warnings and errors in the stack.

The meaning of a warning or error codes is defined in the product data sheet. Typical indications and their codes are listed in Fault Messages, p. 45.

Parameter	Value	Note
Command identifier	50h	
Access code	-	
Response time	5 ms	
Master data length	0 bytes	
Slave data length	1 byte	
Master data values	-	
Slave data values	Byte 0: Encoder status byte	Status 00h, warning, or error code
Error conditions		

Table 27: Read Encoder Status

Read Type Label

The “Read Type Label” command returns device and interface information of the HIPERFACE® Slave.

"Read Type Label" returns the following data:

– Parameter interface (UART) settings with the same encoding as for command "Set Serial Interface" (see Set Serial Interface, p. 37).
– Encoder type code, see Extended Type Label Specification, p. 24 for more details.
– Total memory size in 16 byte blocks (e.g. for 2048 implemented memory bytes a value of 128 / 80h is returned)
– Option code for specific available hardware/software options, HIPERFACE® version. This is a legacy value; for new HIPERFACE® Slaves this value is always 00h.

Parameter	Value	Note
Command identifier	52h	
Access code	-	
Response time	5 ms	
Master data length	0 bytes	
Slave data length	4 bytes	
Master data values	-	
Slave data values	Byte 0: UART settings Bit 0...2: Data rate 000 = 600 Baud 001 = 1200 Baud 010 = 2400 Baud 011 = 4800 Baud 100 = 9600 Baud (default) 101 = 19200 Baud 110 = 38400 Baud Bit 3: not implemented Bit 4...5: Data bits, parity 00 = 8 data bits, no parity 10 = 8 data bits, even (default) 11 = 8 data bits, odd Bit 6: Timeout 0 = 2*11/data rate 1 = 5*11/data rate (default) Bit 7: HIPERFACE® Bus 0 = Bus Slave 1 = Standard Slave (default) Byte 1: Encoder type 00...FEh = specific encoder type FFh = extended type label available Byte 2: Memory size in 16 byte blocks Byte 3: Option code	Byte 0 default value = E4h
Error conditions	05h (I2C communication fault)	

Table 28: Read Type Label

Encoder Reset

The “Encoder Reset” command allows a software reset of the HIPERFACE® Slave.

Software reset leads to the same initialization process as after power-on.

Note:

Some legacy products have slight differences between power-on and software reset. New and future products must observe identical behavior.

Configuration data like Slave address and interface settings are not changed by an “Encoder Reset”.

Note:

A correct “Encoder Reset” request will not be followed by a Slave response message.

Note:

“Encoder Reset” is executed by the HIPERFACE® Slave after the timeout of the request has elapsed. A HIPERFACE® Master must wait for this time plus the initialization time of 100 ms before communication can resume.

Parameter	Value	Note
Command identifier	53h	
Access code	-	
Response time	-	No response The Slave shall respond to new requests after initialization time (100 ms) plus timeout for this request
Master data length	0 bytes	
Slave data length	-	No response
Master data values	-	
Slave data values	-	
Error conditions		

Table 29: Encoder Reset

Set Encoder Address

The “Set Encoder Address” command allows setting the Slave address of the HIPERFACE® Slave.

Slave addresses are valid from 40h (default) to 5Fh (see Addressing, p. 11).

Note:

After successful execution of this command the Slave response message will already use the new Slave address.

Parameter	Value	Note
Command identifier	55h	
Access code	Code0	Default: 55h
Response time	40 ms	
Master data length	2 bytes	
Slave data length	0 bytes	
Master data values	Byte 0: New Slave address (40...5Fh) Byte 1: Code0	Default address: 40h
Slave data values	-	
Error conditions	05h (I2C communication fault) 06h (EE checksum fault) 0Dh (Wrong argument) 0Fh (Wrong access code)	

Table 30: Set Encoder Address

Read Version

The “Read Version” command returns serial number and firmware information of the HIPERFACE® Slave.

“Read Version” returns the following data:

- Serial number in ASCII encoding.
- Firmware version in ASCII encoding with trailing 00h characters.
- Firmware date in ASCII encoding (“DD.MM.YY” format).

Parameter	Value	Note
Command identifier	56h	
Access code	-	
Response time	5 ms	
Master data length	0 bytes	
Slave data length	37 bytes	
Master data values	-	
Slave data values	Byte 0…8: Serial number in ASCII Byte 9…28: Firmware version in ASCII Byte 29…36: Firmware date in ASCII	
Error conditions	05h (I2C communication fault)	

Table 31: Read Version

Set Serial Interface

The “Set Serial Interface” command allows setting the UART parameters of the HIPERFACE® Slave.

“Set Serial Interface” uses parameter interface (UART) settings with the same encoding as for command “Read Type Label” (see Read Type Label, p. 33).

Note:

After successful execution of this command the Slave will only use the new UART settings after an encoder reset (see Encoder Reset, p. 35) or a power cycle.

Note:

After power-on each HIPERFACE® slave will only respond to default UART settings for a certain time. Different settings will only be used if this initialization time has elapsed without communication (see HIPERFACE® State Machine, p. 15).

Note:

Previously the default parity for HIPERFACE® Slaves was specified as “Odd”. The actually implemented default parity check always conformed to an “Even” parity definition. Accordingly this specification corrects this definition.

Parameter	Value	Note
Command identifier	57h	
Access code	Code0	Default: 55h
Response time	40 ms	
Master data length	2 bytes	
Slave data length	1 byte	
Master data values	Byte 0: UART settings Bit 0…2: Data rate 000 = 600 Baud 001 = 1200 Baud 010 = 2400 Baud 011 = 4800 Baud 100 = 9600 Baud (default) 101 = 19200 Baud 110 = 38400 Baud Bit 3: not implemented Bit 4…5: Data bits, parity 00 = 8 data bits, no parity 10 = 8 data bits, even (default) 11 = 8 data bits, odd Bit 6: Timeout 0 = 2*11/data rate 1 = 5*11/data rate (default) Bit 7: HIPERFACE® Bus 0 = Bus Slave 1 = Standard Slave (default) Byte 1: Code0	Byte 0 default value = E4h The actually available data rates are specified in the product data sheet of a HIPERFACE® Slave
Slave data values	Byte 0: UART settings (as above)	
Error conditions	05h (I2C communication fault) 0Dh (Wrong argument) 0Fh (Wrong access code)	

Table 32: Set Serial Interface

Temporarily Set Serial Interface

Note:

The command “Temporarily Set Serial Interface” is an optional HIPERFACE® command.

The “Temporarily Set Serial Interface” command allows setting the UART parameters of the HIPERFACE® Slave immediately without nonvolatile memory storage.

“Temporarily Set Serial Interface” uses parameter interface (UART) settings with the same encoding as for command “Set Serial Interface”
(see Set Serial Interface, p. 37).

Note:

After successful execution of this command the Slave will directly use the new UART settings. After an encoder reset (see Encoder Reset, p. 35) or a power cycle the settings are discarded.

Note:

The Slave response message will still use the previous UART settings; the new settings are applied afterwards.

Note:

Previously the default parity for HIPERFACE® Slaves was specified as “Odd”. The actually implemented default parity check always conformed to an “Even” parity definition. Accordingly this specification corrects this definition.

Parameter	Value	Note
Command identifier	67h	
Access code	-	
Response time	5 ms	
Master data length	1 byte	
Slave data length	1 byte	
Master data values	Byte 0: UART settings Bit 0…2: Data rate 000 = 600 Baud 001 = 1200 Baud 010 = 2400 Baud 011 = 4800 Baud 100 = 9600 Baud (default) 101 = 19200 Baud 110 = 38400 Baud Bit 3: not implemented Bit 4…5: Data bits, parity 00 = 8 data bits, no parity 10 = 8 data bits, even (default) 11 = 8 data bits, odd Bit 6: Timeout 0 = 2*11/data rate 1 = 5*11/data rate (default) Bit 7: HIPERFACE® Bus 0 = Bus Slave 1 = Standard Slave (default)	Byte 0 default value = E4h The actually available data rates are specified in the pro-duct data sheet of a HIPER-FACE® Slave
Slave data values	Byte 0: UART settings (as above)	
Error conditions	0Dh (Wrong argument) 0Fh (Wrong access code)	

Table 33: Temporarily Set Serial Interface

Set Position with Synchronization

Note:

The command “Set Position with Synchronization” is an optional HIPERFACE® command.

The “Set Position with Synchronization” command allows the motor controller to store a position offset in the HIPERFACE® Slave like the “Set Position” command (see Set Position, p. 19).

The difference to this command is that “Set Position with Synchronization” will only allow setting absolute position offsets that are in phase to the Sin/Cos signals. The commanded offset will be rounded towards the next appropriate value when using this command.

All other aspects of “Set Position” also pertain to this command.

The HIPERFACE® Master must transmit the desired position value (preset). The Slave will calculate the corresponding position offset, round it to the next appropriate value for zero phase shift and store it internally.

ATTENTION:

As no synchronization is provided for this function the command must be called at standstill of the motor.

The command “Set Position with Synchronization” can only be activated if “Code0” is transmitted to guard against improper use and impairment of motor function.

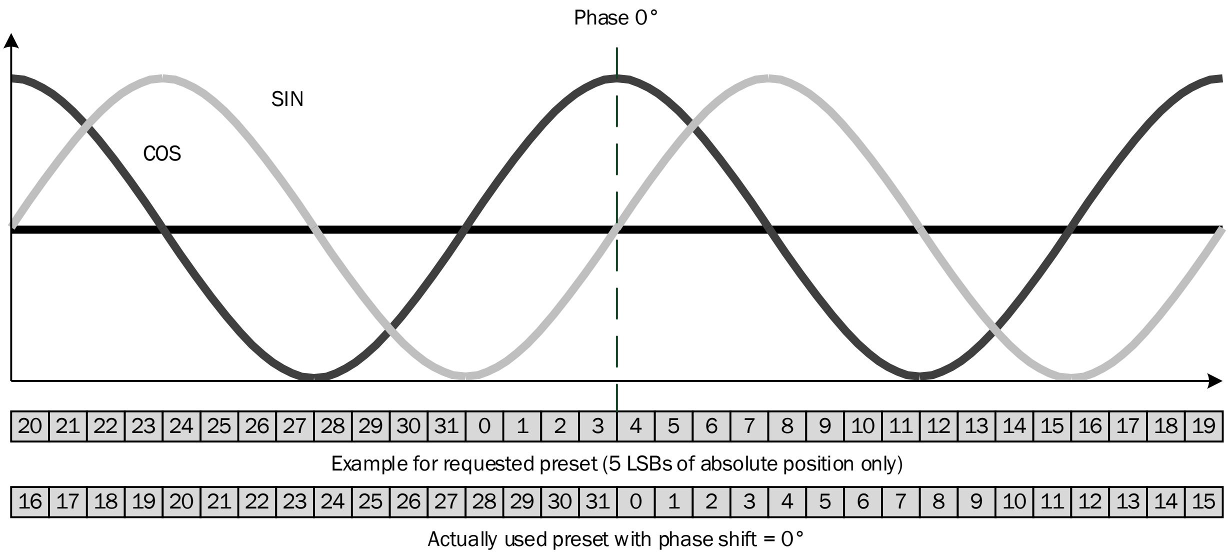


Figure 14: “Set position with Synchronization” operation

Note:

A phase shift other than 0° limits the ability of the Master to synchronize absolute position to analog signals. Therefore it can be preferable to use this command to force a phase shift of 0°. On the other hand, for encoders with low sine/cosine period count this can lead to inacceptable commutation offset errors. In this case it is recommended to store the commutation offset in a Slave data field and perform the offset calculation Master-side.

Parameter	Value	Note
Command identifier	6Ah	
Access code	Code0	Default: 55h
Response time	40 ms	
Master data length	5 bytes	
Slave data length	4 bytes	
Master data values	Byte 0…3: Absolute position preset as unsigned 32 bit value, MSB first Byte 4: Code0	
Slave data values	Byte 0…3: Actually used absolute position preset as unsigned 32 bit value, MSB first	Can be shifted if requested preset would have led to a phase shift ≠ 0°
Error conditions	02h (Angle offset fault) 05h (I2C communication fault) 06h (EE checksum fault) 0Dh (Wrong argument) 0Fh (Wrong access code) 1Dh (LED current high) 20h (Singleturn fault) 21h (Multiturn amplitude fault) 22h (Multi sync fault) 23h (Multiturn vectorlength fault)	

Table 34: Set Position with Synchronization

Sensor Adjustment

Note:

The command “Sensor Adjustment” is an optional HIPERFACE® command. It is useful for HIPERFACE® Slaves that have to be partially assembled by the user (e.g. linear read-head + measurement tape, rotary kit).

This command is defined in regard to command identifier, command length, and access code. Implementation-specific behavior can be addressed through one byte in the Master request message. The response of the Slave is also implementation-specific and can be indicated through one byte in the Slave response message.

Possible faults that arise from usage of this command are also implementation-specific (except for default faults for wrong message formats).

Parameter	Value	Note
Command identifier	6Bh	
Access code	Code0	Default: 55h
Response time	Implementation-specific	To be specified in data sheet of Slave
Master data length	2 bytes	
Slave data length	2 bytes	
Master data values	Byte 0: Implementation-specific option code Byte 1: Code0	Byte 0 to be specified in data sheet of Slave
Slave data values	Byte 0: Master option code Byte 1: Implementation-specific response status	Byte 1 to be specified in the data sheet of Slave
Error conditions	0Dh (Wrong argument) 0Fh (Wrong access code)	Other possible faults to be specified in the data sheet of Slave

Table 35: Sensor Adjustment

Read Synchronization Offset

Note:

The command “Read Synchronization Offset” is an optional HIPERFACE® command. It is useful for HIPERFACE® Slaves that have low period count where both using command “Set Position” (see Set Position, p. 19) is not optimal (due to potential loss of synchronization between Sin/Cos and absolute position) and “Set Position with Synchronization” (see Set Position with Synchronization, p. 41) is also not good (due to potential large mismatch between 0 position and motor commutation angle).

“Read Synchronization Offset” returns the phase shift between the absolute position (as set with “Set Position”) and the Sin/Cos position. This value allows the HIPERFACE® Master to have a well-calibrated commutation angle (due to using “Set Position”) and to achieve synchronization by using the phase shift value from this command.

The returned value has to be added to the current position value for performing synchronization to Sin/Cos during Master start-up.

Parameter	Value	Note
Command identifier	6Ch	
Access code	-	
Response time	10 ms	
Master data length	0 bytes	
Slave data length	1 byte	
Master data values	-	
Slave data values	Byte 0: Phase shift (0...31)	
Error conditions	0Dh (Wrong argument)	Other possible faults to be specified in the data sheet of Slave

Table 36: Read Synchronization Offset

2.4.3 Fault Messages

HIPERFACE® defines fault messages as defined in the subsequent sections.

Each HIPERFACE® Slave uses a subset of these messages as specified in the product data sheet.

Faults are either transmitted with the warning bit as indicated in Warning Bit, p. 12 or as error response (see Error Response, p. 13). This will be specified in the following sections under "fault indication".

Fault messages can be understood as belonging to different fault groups. Fault groups cluster faults according to functionality of the HIPERFACE® Slave.

Fault group	Behavior
Initialization	During start-up of the HIPERFACE® Slave the different electronic components are configured with data from an EEPROM. This data is protected against unintentional changes by checksums. If the check fails for position-relevant data, subsequent position commands (read position or set position) will yield an error code "02h = incorrect internal angular offset". If the check fails for other data the error code "06h = internal CRC error" is issued. To identify what specific checksum failed a further error code is issued. Errors during initialization will impact the correct operation of the HIPERFACE® Slave and will cause messages from other error groups later. If error messages occur during start-up it is recommended to carry out a "software reset" first. If this didn't succeed a "power cycle" should be done.
Protocol	This group contains errors in reference to the command transmission and analysis. These messages are sent as response to the command request and will not be stored in the error stack. They won't cause follow-up errors.
Data	This group contains all errors in reference to the user data storage, which can occur during carry out of a command or during initialization. They are not stored in the error stack. They won't cause follow-up errors within other areas.
Position	This group contains all errors that can occur carrying out the command read position (42h) or write position (43h). These messages are sent as response to the command request and will not be stored in the error stack. In this case the position reading is stopped. During more complex error situations when more than one error occurs the first one is sent and the other error messages are stored. This is indicated by the error bit at the status answer.
Others	This group contains warnings or errors caused by the monitoring of values for temperature, the LED-current or the internal user counter. The error is generated when the value exceeds the limit. The errors are stored in the error stack and will be deleted during read out.

Table 37: HIPERFACE® fault groups

Incorrect alignment data (01h)

Parameter	Value
Fault indication	Error response
Fault group	Initialization
Code	01h

Table 38: Incorrect alignment data

Description:

“Incorrect alignment data” indicates that one of the following values stored in nonvolatile memory was corrupted:

- Singleturn resolution and Multiturn range settings
- Sensor calibration

Slave reaction:

The indication will be stored in the error stack along with the “Internal checksum error”.

Position output is locked if this fault was detected. A “Read position” or “Set position” command will always receive a Slave error response with the Incorrect internal angular offset (02h), p. 47.

Possible causes:

The fault could result from electrical disturbances during start-up or from hardware faults of the memory component.

Analog signals outside specification (01h)

Parameter	Value
Fault indication	Error response
Fault group	Position
Code	01h

Table 39: Analog signals outside specification

Description:

During operation “Analog signals outside specification” indicates that one of the following signal faults was detected:

- Supply voltage outside valid range
- Internal voltages outside valid range

Slave reaction:

If this condition is detected the position output will not be locked.

Possible causes:

The fault could result from a bad supply voltage of the Master or from hardware faults of the Slave.

Incorrect internal angular offset (02h)

Parameter	Value
Fault indication	Error response
Fault group	Initialization
Code	02h

Table 40: Incorrect internal angular offset

Description:

At start-up “Incorrect internal angular offset” indicates that one of the following values stored in nonvolatile memory was corrupted:

- Position offset
- Sensor synchronization

Slave reaction:

The indication will be stored in the error stack along with the Internal checksum error (06h), p. 49.

Position output is locked if this fault was detected. A “Read position” or “Set position” command will always receive a Slave error response with this fault code.

Possible causes:

The fault could result from electrical disturbances during start-up or from hardware faults of the memory component.

Data field partitioning table destroyed (03h)

Parameter	Value
Fault indication	Error response
Fault group	Initialization/Data
Code	03h

Table 41: Data field partitioning table destroyed

Description:

At start-up “Data field partitioning table destroyed” indicates that one of the following values stored in nonvolatile memory was corrupted:

- UART settings
- User memory size
- Access codes

During operation “Data field partitioning table destroyed” indicates a data corruption in user data fields during data field operations (“Read Data”, “Store Data”, “Data Field Status”, and “Create Data Field”).

Slave reaction:

The indication will be stored in the error stack along with the “Internal checksum error”.

Writing to nonvolatile memory data is locked if this fault was detected. Related command requests will always receive a Slave error response with this fault code.

Possible causes:

The fault could result from electrical disturbances during start-up or from hardware faults of the memory component.

Analog limit values not available (04h)

Parameter	Value
Fault indication	Error response
Fault group	Initialization
Code	04h

Table 42: Analog limit values not available

Description:

At start-up “Analog limit values not available” indicates that one of the following values stored in nonvolatile memory was corrupted:

- Temperature warning range
- LED current warning range
- Synchronization check warning range
- Other diagnostic ranges

Slave reaction:

The indication will be stored in the error stack along with the Internal checksum error (06h), p. 49.

Position output is locked if this fault was detected. A “Read position” or “Set position” command will always receive a Slave error response with fault Incorrect internal angular offset (02h), p. 47.

Possible causes:

The fault could result from electrical disturbances during start-up or from hardware faults of the memory component.

Internal I2C bus not operational (05h)

Parameter	Value
Fault indication	Error response
Fault group	Initialization/Position
Code	05h

Table 43: Internal I2C bus not operational

Description:

“Internal I2C bus not operational” indicates that the HIPERFACE® Slave could not establish communication to its nonvolatile memory or detected a fault in the communication protocol to its nonvolatile memory.

Additionally “Internal I2C bus not operational” is used at start-up if one of the following values stored in nonvolatile memory was corrupted:

- Encoder type code

Slave reaction:

“Internal I2C bus not operational” is used during operation if an Internal checksum error (06h), p. 49 was detected at start-up and the user tries to use commands that use nonvolatile memory of the HIPERFACE® Slave.

Possible causes:

The fault could result from electrical disturbances during start-up or operation or from hardware faults of the memory component.

Internal checksum error (06h)

Parameter	Value
Fault indication	Error response
Fault group	Initialization
Code	06h

Table 44: Internal checksum error

Description:

“Internal checksum error” indicates that the HIPERFACE® Slave has detected data corruption in its internal configuration.

Note:

Other error codes are typically also set along with “Internal checksum error” to narrow down the affected data. This is documented in Incorrect internal angular offset (02h), p. 47, Data field partitioning table destroyed (03h), p. 47, Counter overflow (08h), p. 50, Wrong data field number (12h), p. 55 and Multiturn amplitude fault (21h), p. 59.

Slave reaction:

If an “Internal checksum error” is detected at start-up no command will be executed that reads critical or writes any data to nonvolatile memory (“Read Counter”, “Increment Counter”, “Reset Counter”, “Store Data”, “Create Data Field”, “Set Access Code”, “Set Encoder Address”, “Set Serial Interface”). If a user accesses these functions this error will be indicated.

Possible causes:

The fault could result from electrical disturbances during start-up or operation or from hardware faults of the memory component.

Program watchdog fault (07h)

Parameter	Value
Fault indication	Error response
Fault group	Initialization
Code	07h

Table 45: Program watchdog fault

Description:

The “Program watchdog fault” is indicated if the HIPERFACE® Slave was reset due to a watchdog violation.

Slave reaction:

The fault indication will have no further consequences during operation.

Possible causes:

The fault could result from electrical disturbances during start-up or operation or from hardware faults of the processor component.

Counter overflow (08h)

Parameter	Value
Fault indication	Error response
Fault group	Others
Code	08h

Table 46: Counter overflow

Description:

At start-up “Counter overflow” indicates that the counter value of the HIPERFACE® Slave in nonvolatile memory is corrupted.

During operation “Counter overflow” indicates that the counter of the HIPERFACE® Slave has the maximum value (FF FF FFh) and the user tried to increment (see Increment Counter, p. 22). The counter will stay at maximum value until it is reset (see Reset Counter, p. 23).

Slave reaction:

The indication will be stored in the error stack along with the Internal checksum error (06h), p. 49.

Counter operations are locked if this fault was detected. A “Read counter”, “Increment counter”, or “Reset counter” command will always receive a Slave error response with fault Internal checksum error (06h), p. 49.

Possible causes:

The fault could result from electrical disturbances during start-up or from hardware faults of the memory component.

Parity error (09h)

Parameter	Value
Fault indication	Error response
Fault group	Protocol
Code	09h

Table 47: Parity error

Description:

During operation “Parity error” indicates that the HIPERFACE® Slave has detected a UART fault in a Master request message.

These faults typically include parity faults, frame errors, noise errors, and/or overflow errors of the UART peripheral.

Slave reaction:

The message will not be parsed further in this case and no action will be performed by the HIPERFACE® Slave.

Possible causes:

The fault could result from electrical disturbances on the physical interface (connectors, cable).

Checksum error (0Ah)

Parameter	Value
Fault indication	Error response
Fault group	Protocol
Code	0Ah

Table 48: Checksum error

Description:

During operation “Checksum error” indicates that the HIPERFACE® Slave has detected a checksum fault in a Master request message.

Slave reaction:

The message will not be parsed further in this case.

Possible causes:

This could be a result from electrical disturbance on the line, wire break, or a bad request sent from the Master.

Unknown command (0Bh)

Parameter	Value
Fault indication	Error response
Fault group	Protocol
Code	0Bh

Table 49: Unknown command

Description:

During operation “Unknown command” indicates that the HIPERFACE® Slave has received a Master request message with an unknown command value.

Slave reaction:

The message will not be parsed further in this case.

Possible causes:

This could be a result from electrical disturbance on the line, wire break, or a bad request sent from the Master.

Wrong command length (0Ch)

Parameter	Value
Fault indication	Error response
Fault group	Protocol
Code	0Ch

Table 50: Wrong command length

Description:

During operation “Wrong command length” indicates that the HIPERFACE® Slave has received a Master request message with an invalid number of message bytes.

Slave reaction:

The message will not be parsed further in this case.

Possible causes:

This could be a result from electrical disturbance on the line, wire break, or a bad request sent from the Master.

Wrong command argument (0Dh)

Parameter	Value
Fault indication	Error response
Fault group	Protocol
Code	0Dh

Table 51: Wrong command argument

Description:

During operation “Wrong command argument” indicates that the HIPERFACE® Slave has received a Master request message with an invalid value in one of its command argument bytes.

Slave reaction:

The message will not be parsed further in this case.

Possible causes:

This could be a result from electrical disturbance on the line, wire break, or a bad request sent from the Master.

Read-only data field (0Eh)

Parameter	Value
Fault indication	Error response
Fault group	Data
Code	0Eh

Table 52: Readonly data field

Description:

During operation “Read-only data field” indicates that the HIPERFACE® Slave has received a “Store data” request that either targets a data field with “read-only” setting or the “Extended Type Label”.

Slave reaction:

The message will not be parsed further in this case.

Possible causes:

This could be a result from electrical disturbance on the line, wire break, or a bad request sent from the Master.

Incorrect access code (0Fh)

Parameter	Value
Fault indication	Error response
Fault group	Data
Code	0Fh

Table 53: Incorrect access code

Description:

During operation “Incorrect access code” indicates that the HIPERFACE® Slave has received a Master message request that requires one of the access codes (Code0, Code1, Code2 or Code3) and the transmitted access code byte was wrong.

Slave reaction:

The message will not be parsed further in this case.

Possible causes:

This could be a result from electrical disturbance on the line, wire break, or a bad request sent from the Master.

Out of memory fault (10h)

Parameter	Value
Fault indication	Error response
Fault group	Data
Code	10h

Table 54: Out of memory fault

Description:

During operation “Out of memory fault” indicates one of the following fault conditions:

– The HIPERFACE® Slave has received a “Create data field” request to create a new data field with a data field size that does not fit in the remaining user memory.

– The HIPERFACE® Slave has received a “Create data field” request to change the size of an existing data field with a data field size that does not fit in the remaining user memory.

– The HIPERFACE® Slave has received a “Create data field” request that targets the Extended Type Label.

Note:

If the user tries to modify the size of a data field other than the last one created a “Wrong data field number” indication is used.

Slave reaction:

The message will not be parsed further in this case.

Possible causes:

This could be a result from a bad request sent from the Master.

Wrong data field offset (11h)

Parameter	Value
Fault indication	Error response
Fault group	Data
Code	11h

Table 55: Wrong data field offset

Description:

During operation “Wrong data field offset” indicates that the HIPERFACE® Slave has received a Master request message to read or write in a data field with an offset outside the data field size. The indication is also used if a read access to the Extended Type Label uses an offset outside the type label size (64 bytes).

Slave reaction:

The message will not be parsed further in this case.

Possible causes:

This could be a result from a bad request sent from the Master.

Wrong data field number (12h)

Parameter	Value
Fault indication	Error response
Fault group	Data
Code	12h

Table 56: Wrong data field offset

Description:

At start-up “Wrong data field number” indicates that one of the following values stored in nonvolatile memory was corrupted:

– Extended Type Label

Note:

This check is only performed if the HIPERFACE® Slave has a “true” Extended Type Label with type code FFh (see Extended Type Label Specification, p. 24).

During operation “Wrong data field number” indicates one of the following fault conditions:

– The HIPERFACE® Slave has received a data field operation request with an invalid data field number.

– The HIPERFACE® Slave has received a “Create data field” request to create a new data field with a “delete” bit set.

– The HIPERFACE® Slave has received a “Create data field” request to modify a data field size for a data field that is not the last one.

Slave reaction:

The indication at start-up will be stored in the error stack along with the “Internal checksum error”.

Position output is locked if this fault was detected at start-up. A “Read position” or “Set position” command will always receive a Slave error response with the Incorrect internal angular offset (02h), p. 47.

During operation the message will not be parsed further if this fault was detected.

Possible causes:

The fault at start-up could result from electrical disturbances during start-up or from hardware faults of the memory component.

The fault during operation could be a result from a bad request sent from the Master.

Value monitoring analog signals (1Ch)

Parameter	Value
Fault indication	Error response
Fault group	Position/Others
Code	1Ch

Table 57: Value monitoring analog signals

Description:

The “Value monitoring analog signals” indicates that the HIPERFACE® Slave has detected significant signal faults in the analog sine/cosine signals.

Slave reaction:

If this condition is detected the position output will not be locked.

Note:

This diagnostic typically is also done inside the motor controller.

Possible causes:

A typical transient reason for this fault is excessive mechanical stress on the Slave (e.g. shocks).

Transmitter current critical (1Dh)

Parameter	Value
Fault indication	Error response
Fault group	Initialization/Position/ Others
Code	1Dh

Table 58: Transmitter current critical

Description:

The “Transmitter current critical” indicates that the HIPERFACE® Slave has detected an error condition with the sensor LED. For optical encoders this condition will be checked prior to every Slave response message.

Note:

Some legacy HIPERFACE® Slaves transmit this condition with the Warning Bit.

Slave reaction:

If this condition is detected the position output will not be locked.

Possible causes:

Potential reasons for this fault include end of lifetime of the LED, various hardware faults in the optical pick-up system, or defects of the code disc.

Encoder temperature critical (1Eh)

Parameter	Value
Fault indication	Warning bit
Fault group	Others
Code	1Eh

Table 59: Encoder temperature critical

Description:

The “Encoder temperature critical” indicates that the HIPERFACE® Slave has detected an invalid ambient temperature.

This condition will be checked prior to every Slave response message.

Slave reaction:

If this condition is detected the position output will not be locked.

Possible causes:

Typically this fault indicates bad application conditions.

Speed too high (1Fh)

Parameter	Value
Fault indication	Error response
Fault group	Others
Code	1Fh

Table 60: Speed too high

Description:

The “Speed too high” indicates one of the following conditions:

– The HIPERFACE® Slave has detected an invalid shaft speed during a “Read position” request. An error indication will be output.

– The HIPERFACE® Slave has detected shaft motion during a “Set position” request. An error indication will be output and the command will not be processed.

Note:

Some legacy HIPERFACE® Slaves transmit this condition with the Warning Bit.

Slave reaction:

If this condition is detected the position output will not be locked.

Possible causes:

Typically this fault indicates bad application conditions.

Singleturn position unreliable (20h)

Parameter	Value
Fault indication	Error response
Fault group	Position
Code	20h

Table 61: Singleturn position unreliable

Description:

The “Singleturn position unreliable” is used for rotary encoders and indicates that the HIPERFACE® Slave detected an invalid singleturn sensor signal. An error indication will be output.

Note:

Some legacy HIPERFACE® Slaves transmit this condition with the Warning Bit.

Slave reaction:

If this condition is detected the position output will not be locked.

Possible causes:

Potential reasons for this fault include end of lifetime of the LED, various hardware faults in the optical pick-up system, or defects of the code disc.

Sensor not adjusted or in adjustment mode (20h)

Parameter	Value
Fault indication	Error response
Fault group	Position
Code	20h

Table 62: Sensor not adjusted or in adjustment mode

Description:

The “Sensor not adjusted or in adjustment mode” is used for linear encoders and indicates that the HIPERFACE® Slave detected an invalid adjustment of the measure embodiment to the encoder read-head.

Slave reaction:

Position output is locked if this fault was detected. A “Read position” or “Set position” command will always receive a Slave error response with the Incorrect internal angular offset (02h), p. 47.

Possible causes:

Potential reasons for this fault include bad Master requests (Slave is not commanded to leave adjustment mode) or bad alignment between measure embodiment and encoder.

Multiturn amplitude fault (21h)

Parameter	Value
Fault indication	Error response
Fault group	Position
Code	21h

Table 63: Multiturn amplitude fault

Description:

The “Multiturn amplitude fault” is used for rotary encoders and indicates that the HIPERFACE® Slave detected invalid multiturn sensor signal amplitude. An error indication will be output.

In new and future HIPERFACE® Slaves this fault is also used for indication of faults in multiturn parameters stored in nonvolatile memory. In this case the fault will be stored in the error stack along with the Internal checksum error (06h), p. 49.

Note:

Some legacy HIPERFACE® Slaves transmit this condition with the Warning Bit.

Slave reaction:

If this condition is detected the position output will not be locked.

Possible causes:

Potential reasons for this fault include magnetic disturbances, defects of multiturn sensors, or loss of multiturn magnets.

Distance measure/sensor too high (21h)

Parameter	Value
Fault indication	Error response
Fault group	Position
Code	21h

Table 64: Distance measuresensor too high

Description:

The “Distance measure/sensor too high” is used for linear encoders and indicates that the HIPERFACE® Slave detected an invalid distance of the measure embodiment to the encoder read-head.

Slave reaction:

If this condition is detected the position output will not be locked.

Possible causes:

Potential reasons for this fault include wrong mounting of linear encoder or measure embodiment.

Multiturn sync fault (22h)

Parameter	Value
Fault indication	Error response
Fault group	Position
Code	22h

Table 65: Multiturn sync fault

Description:

The “Multiturn sync fault” is used for rotary encoders and indicates that the HIPERFACE® Slave detected invalid multiturn sensor synchronization. An error indication will be output.

ATTENTION:

Some legacy HIPERFACE® Slaves transmit this condition with the Warning Bit.

Note:

A multiturn sync diagnostic cannot reliably detect gear synchronization faults over extended time. After first indication of this fault gear synchronization can wear out even more and result in undetected false position output.

Slave reaction:

Position output is locked if this fault was detected. A “Read position” or “Set position” command will always receive a Slave error response with the Incorrect internal angular offset (02h), p. 47.

Possible causes:

The typical reason for this fault is end of lifetime of the multiturn gear.

Multiturn vectorlength fault (23h)

Parameter	Value
Fault indication	Error response
Fault group	Position
Code	23h

Table 66: Multiturn vectorlength fault

Description:

The “Multiturn vectorlength fault” is used for rotary encoders and indicates that the HIPERFACE® Slave detected invalid multiturn sensor signal vectorlength. An error indication will be output.

Note:

Some legacy HIPERFACE® Slaves transmit this condition with the Warning Bit.

Slave reaction:

If this condition is detected the position output will not be locked.

Possible causes:

A typical transient reason for this fault is excessive mechanical stress on the Slave (e.g. shocks).

Linear position fault (23h)

Parameter	Value
Fault indication	Error response
Fault group	Position
Code	23h

Table 67: Linear position fault

Description:

The “Linear position fault” is used for linear encoders and indicates that the HIPERFACE® Slave detected a fault in its position measurement.

Slave reaction:

If this condition is detected the position output will not be locked.

Possible causes:

Potential reasons for this fault include various hardware faults in the measurement system or defects of the measure embodiment.

Multiturn counter fault (24h)

Parameter	Value
Fault indication	Error response
Fault group	Position
Code	24h

Table 68: Multiturn counter fault

Description:

The “Multiturn counter fault” is used for rotary encoders and indicates that the HIPERFACE® Slave detected invalid multiturn signals in a counting multiturn module. An error indication will be output.

Slave reaction:

Position output is locked if this fault was detected. A “Read position” or “Set position” command will always receive a Slave error response with the “Incorrect internal angular offset” 02h (see Incorrect internal angular offset (02h), p. 47).

Possible causes:

Potential reasons for this fault include various hardware faults in the measurement or counter system.

3 HIPERFACE® Master

The overall interface requirements expected of a HIPERFACE® Master are lined out in chapter Protocol Specification, p. 6. This chapter details recommendations for additional implementation aspects of drive controllers.

3.1 Hardware Implementation

3.1.1 Encoder Supply

The supply details (voltage range, current) for HIPERFACE® Slaves is specified in Supply, p. 6.

The recommended supply voltage for powering HIPERFACE® Slaves is 8 V as measured on the drive controller output. Typically HIPERFACE® Slaves are equipped with linear voltage regulators to generate internal voltage rails. By using a low supply voltage thermal dissipation within the encoder is reduced so that lifetime at maximum specified temperatures is improved.

The 8 V figure considers typical voltage drops over the full cable length so that 7 V are guaranteed on the Slave-side termination of the cable.

3.1.2 Analog Signal Input

The sine/cosine signals are transmitted pseudo-differentially as detailed in Sine/Cosine Process Channel, p. 6.

This interface determines the actual performance of the speed control-loop of the drive controller. Accordingly the interface circuit must be carefully designed.

The sine/cosine signal input of the HIPERFACE® Master should consider the following items:

– Differential input amplifiers with low offset voltage, low noise, high common-mode rejection

– Passive components with small tolerances (≤ 1%)

It is strongly recommended to implement an analog input filter circuit with low-pass characteristic to filter noise. Since signal bandwidth of different HIPERFACE® Slaves varies greatly the Master implementation should consider switchable filter characteristics depending on connected Slave. The following table gives an overview over signal bandwidth for existing HIPERFACE® Slaves.

Slave type	Maximum output frequency [kHz]
SEK/L34, SEK/L37, SEK/L52	3.2
SEK90, SEK160, SEK260	3.2
SFS/M60	153.6
SKS36, SKS36S	25.6
SKM36, SKM36S	19.2
SKS/M36(S) Stand-alone	12.8
SRS/M50, SRS/M50S	204.8
SRS/M50 Stand-alone	102.4
TTK50/70	10.0

Table 69: Slave signal bandwidth

The filtered sine/cosine signals are fed to one comparator each for generating incremental counting signals (“A signal” resp. “B signal).

The combination of A and B signals in a quadrature counter logic allows determination of the incremental position with 4 steps per sine/cosine signal period. It is possible to perform the counter logic in either dedicated digital hardware or through software. A hardware counting is recommended to reduce jitter and processor load.

Typically the signals are separately also fed to a 2-channel analog-digital converter (ADC) to capture voltage levels of sine and cosine signals and perform a fine angle interpolation. The ADC should have a resolution of at least 10 bits.

Comparator, quadrature counter, and ADC functions are usually available as peripherals in microcontrollers or DSPs.

For good common-mode rejection the PCB layout should foresee a symmetric arrangement for the differential input signal tracks (Sin/RefSin resp. Cos/RefCos).

The following figure shows a suggested input circuit for the analog signals.

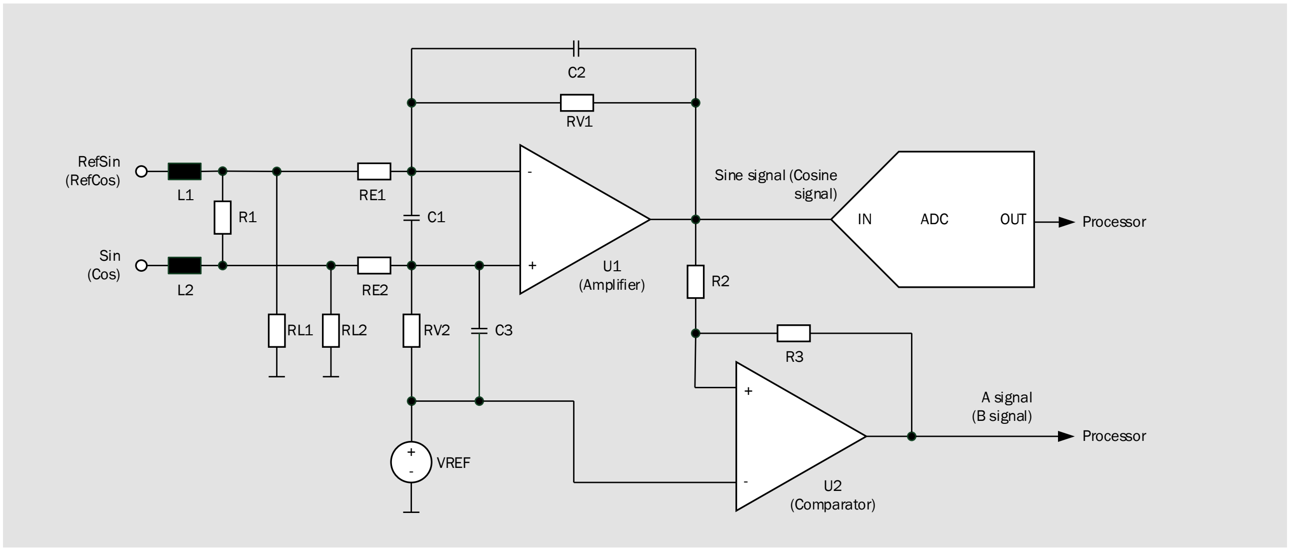


Figure 15: Typical analog signal input schematics (one of two channels)

Part	Value	Note
L1, L2	50 µH	Typical value
R1	120 Ω	
RL1, RL2	1 kΩ	Typical value
RE1, RE2	10 kΩ	Typical value
C1		Optional, depending on filter requirements
RV1, RV2		Identical value, depending on amplifier requirements
C2, C3		Identical value, depending on amplifier requirements
U1		Depending on amplifier requirements
R2	33 kΩ	Typical value
R3	1 MΩ	Typical value
U2		Depending on comparator requirements

Table 70: Analog signal input components

Note:

For resistors and capacitors tolerances of 1% or better are recommended. Capacitors should be of C0G type. The target is to have symmetrical input circuits in order to achieve both a high common mode rejection at the differential amplifier and low phase shifts between sine and cosine signal.

A suitable simple signal filter to improve signal-noise ratio is a first order low-pass formed with RE1, RE2, and C1.

The corner frequency f_c of the low-pass filter should be chosen as low as possible, taking the maximum analog signal bandwidth into account (as demanded by application or as provided by encoder).

The corner frequency impacts attenuation and phase shift (resp. latency/group delay) of the analog encoder signals.

The attenuation impacts the range of an implemented sine/cosine signal vector length check. The signal latency (resp. phase shift) will impact the feedback loop performance; adding to the inherent latency of the HIPERFACE® Slave.

The following table lists examples based on the maximum signal bandwidths of selected encoder families.

Slave type	Maximum output frequency [kHz]	3 dB corner frequency fc [kHz]	Capacity C1 [nF]	Phase shift [°]	Latency [µs]
SEy34/37/52	3.2	6.4	2.5	26.6	23
SKx36	25.6	51.2	0.3	26.6	2.9
SRx50	204.8	409.6	0.04	26.6	0.36

Table 71: Analog lowpass filter settings

Note:

The capacity values for C1 are chosen based on RE1 = RE2 = 10 kΩ. Phase shift and latency are listed for maximum output frequency.

3.1.3 EIA-485 Data Interface

The EIA-485 driver output signals A, /B are realized like detailed in the protocol specification (see chapter EIA-485 Parameter Channel, p. 7).

3.1.4 Position Signal Synchronization

It is recommended to provide a signal connection between the internal RxD signal line and the capture input of the incremental quadrature counter.

This allows highly synchronous comparison between a received absolute position on the EIA-485 interface and the counted position from the analog signal interface even at high shaft speeds.

3.2 Software Implementation

3.2.1 Serial Protocol

The HIPERFACE® serial protocol is handled with a standard UART, see protocol specification Data Link Layer, p. 9.

Note:

Data rates of less than 9600 Baud are deprecated and are no longer supported by future generations of HIPERFACE® Slaves.

3.2.2 Command Handler

HIPERFACE® Slaves implement individual subsets of those commands listed in User Commands, p. 17. A list of available commands is specified in the product specification and data sheet.

It is strongly recommended to implement fault handling for detailed fault reaction management.

As the HIPERFACE® Slave only issues fault indications upon command requests a sufficient cyclic request should be implemented in the drive software. If no other information is required the command "Read Position" (see Read Position, p. 18) should be used to poll fault indications.

If the command handler receives a warning bit indication (see Warning Bit, p. 12) the underlying error code(s) should be read out.

With error codes available an appropriate fault reaction should be implemented, see Safe Position Functions, p. 72.

3.2.3 Quadrature Counting

A and B signals from the analog signal input (see Analog Signal Input, p. 62) are used for quadrature counting. If this is not implemented in hardware, a software task must perform this operation.

The software task must have a higher processing rate than the bandwidth of the sine/ cosine signal (see Table Slave signal bandwidth, p. 62) to avoid undersampling and missing counting a signal period.

The software task must perform the following counter operations based on the A/B signals as sampled currently (A_n, B_n) and the previous signals (A_{n-1}, B_{n-1}):

A_{n-1}	B_{n-1}	A_n	B_n	Counter
0	0	0	0	No action
0	0	0	1	+1
0	0	1	0	-1
0	0	1	1	Invalid
0	1	0	0	-1
0	1	0	1	No action
0	1	1	0	Invalid
0	1	1	1	+1
1	0	0	0	+1
1	0	0	1	Invalid
1	0	1	0	No action
1	0	1	1	-1
1	1	0	0	Invalid
1	1	0	1	-1
1	1	1	0	+1
1	1	1	1	No action

Table 72: Quadrature counting operations

Note:

"Invalid" refers to an invalid transition of comparator signals. In this case a fault in the HIPERFACE® Slave or the Master input stage or Master sample time for this task must be assumed.

3.2.4 Fine Angle Interpolation

Typically the position resolution from quadrature counting is not sufficient for speed control loops. Interpolation of the angle that is indicated through the sine/cosine signals allows for a much higher resolution.

In principle resolution is only limited by the choice of analog-digital converter components and signal noise.

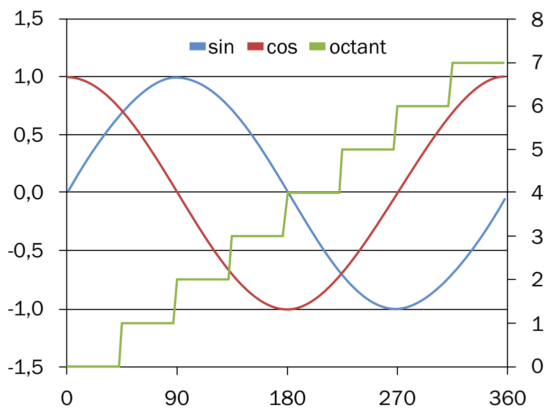
The angle interpolation (φ) is performed through an arctan calculation on the sine/cosine voltages:

$$\varphi = \arctan \frac{V_{\sin}}{V_{\cos}}$$

The following example algorithm is a good compromise between code size and execution speed.

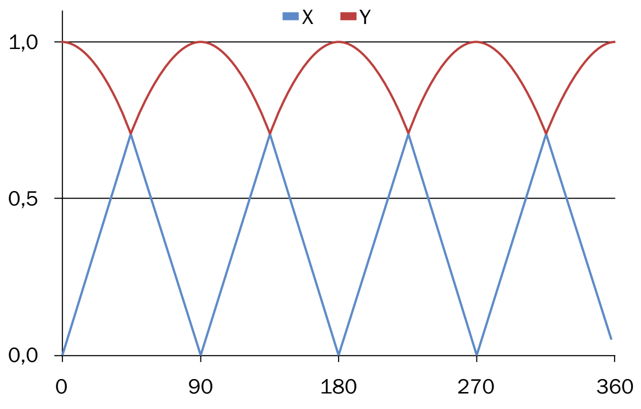
1. Determination of octant

$V_{sin} \geq 0$	$V_{cos} \geq 0$	$\lvert V_{sin} \rvert < \lvert V_{cos} \rvert$	Octant
True	True	True	0
True	True	False	1
True	False	False	2
True	False	True	3
False	False	True	4
False	False	False	5
False	True	False	6
False	True	True	7

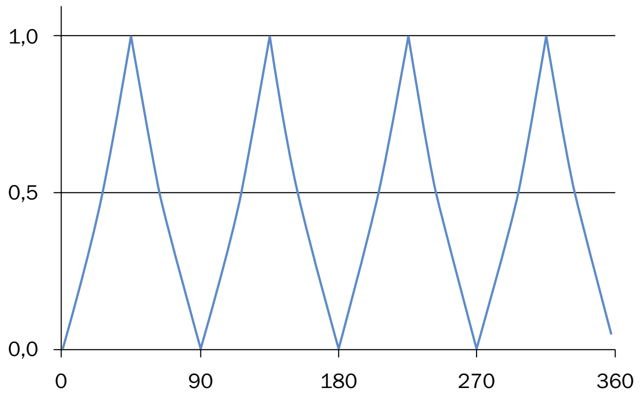


2. Sine/cosine transformation

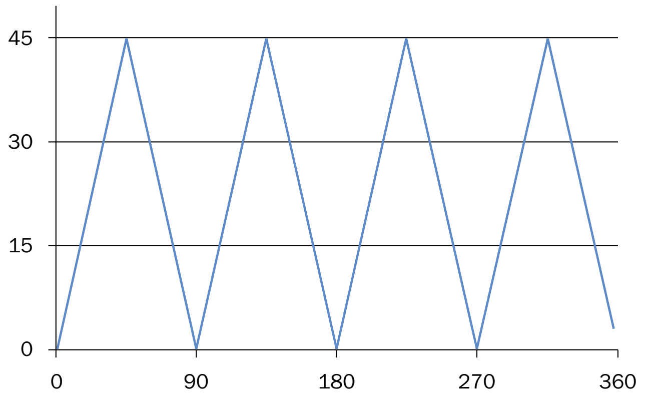
Octant	X	Y
0	V_{sin}	V_{cos}
1	V_{cos}	V_{sin}
2	$- V_{cos}$	V_{sin}
3	V_{sin}	$- V_{cos}$
4	$- V_{sin}$	$- V_{cos}$
5	$- V_{cos}$	$- V_{sin}$
6	V_{cos}	$- V_{sin}$
7	$- V_{sin}$	V_{cos}



3. Calculation of T = X/Y (has always result ≤ 1)



4. Table-based linearization to approximate A = arctan(T) function. Typically 32 nodes are sufficient.



5. Retransformation of A to angle φ

Octant	φ
0	A
1	90°-A
2	90°+A
3	180°-A
4	180°+A
5	270°-A
6	270°+A
7	360°-A

3.2.5 Position Synchronization

The exact position and/or time of a position step will invariably differ between the different position sources (absolute position over EIA-485 vs. incremental position over analog signal). Typical sources for this difference are mechanical tolerances of the separate code tracks, signal propagation delays, input filters, and comparator hysteresis.

Accordingly, a synchronization algorithm must be implemented for correct alignment of the different position components to the quadrature counter start value and subsequent full position values.

The following recommended synchronization algorithm allows for compensation of tolerances within 50% of a sine/cosine period. This makes it possible to assign the analog signals to a specific absolute position value even at high speeds.

1. Synchronous measurement of absolute position and fine angle

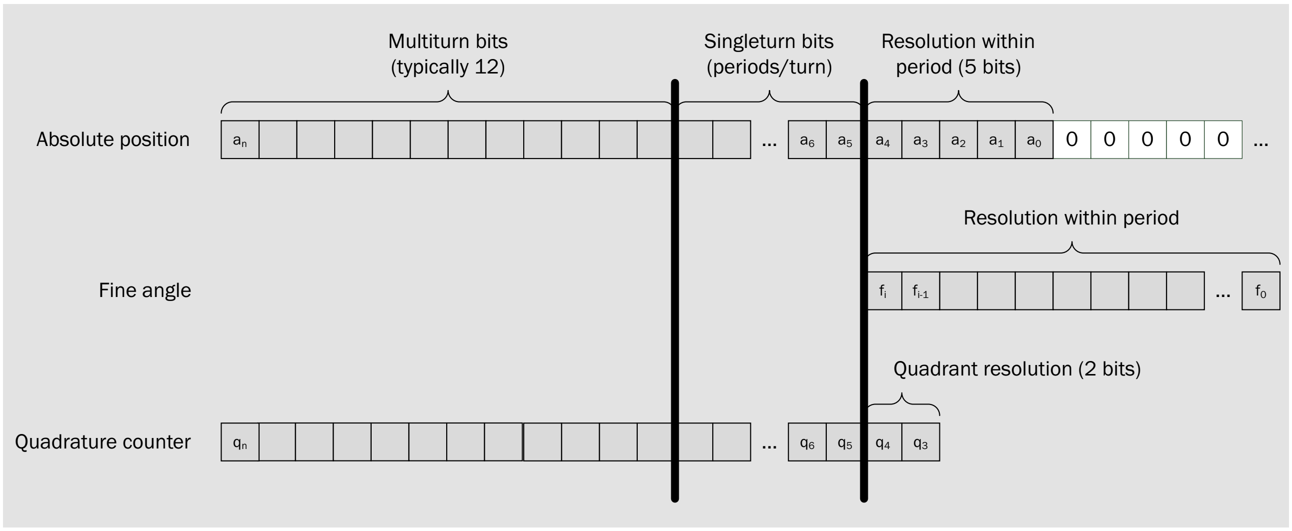
The different measurements (pos_{abs} and φ_{fine}) should be performed with a minimum of time separation to reduce the synchronization difference.

2. Valence alignment

The position values have to be aligned to their appropriate resolution valence.

- The absolute position pos_{abs} $[a_n a_{n-1} ... a_0]_2$ of a HIPERFACE® Slave is always supplied with a resolution of 32 steps (5 bits) per sine/cosine signal period.
- The quadrature counter $q = [q_n q_{n-1} ... q_3]_2$ requires a resolution of quadrants (2 bits) per sine/cosine signal period (please note that $q_2 ... q_0$ don't exist as a quadrature counter does not have this high resolution).

- The fine angle $\varphi_{fine} = [f_i f_{i-1} \dots f_0]_2$ is resolved according to the chosen ADC resolution and fine angle interpolation algorithm. The fine angle has a range of one sine/cosine period.



Multiturn bits (typically 12)
Singleturn bits (periods/turn)
Resolution within period (5 bits)
Absolute position
a_n … a_6 a_5 a_4 a_3 a_2 a_1 a_0 0 0 0 0 0 …
Resolution within period
Fine angle
f_i f_{i-1} … f_0
Quadrant resolution (2 bits)
Quadrature counter
q_n … q_6 q_5 q_4 q_3

Figure 16: Position value valence

3. Quadrature counter initialization

For the quadrant value of the counter, the fine angle value is used (two most significant bits).

$q_3 = f_{i-1}$; $q_4 = f_i$

For all other (higher) bits of the counter, the absolute position bits are used.

$q_k = a_k$ ($k = 5 \dots n$)

If the quadrant value of the fine angle differs from the quadrant value of the absolute position, the period count of the quadrature counter must be selected according to the following table:

$f_{i:i-1}$	$a_{4:3}$	$q_{n:5}$
00	00	$a_{n:5}$
00	01	$a_{n:5}$
00	10	Invalid
00	11	$a_{n:5+1}$
01	00	$a_{n:5}$
01	01	$a_{n:5}$
01	10	$a_{n:5}$
01	11	Invalid
10	00	Invalid
10	01	$a_{n:5}$
10	10	$a_{n:5}$
10	11	$a_{n:5}$
11	00	$a_{n:5-1}$
11	01	Invalid
11	10	$a_{n:5}$
11	11	$a_{n:5}$

Note:

“Invalid” refers to an invalid combination of position signals. In this case a fault in the HIPERFACE® Slave or the Master signal synchronization must be assumed.

Note:

$q_{n:5} = a_{n:5} \pm 1$ refers to an arithmetic addition/subtraction and must be propagated to all higher bits.

4. Runtime position synchronization

During runtime typically the full resolution position must be calculated by the HIPERFACE® Master from the quadrature counter value and the fine angle.

The synchronization is performed similar to the quadrature counter initialization except that the counter is used and not the absolute position value.

Please note that a possible period count change must be stored in the full resolution position value and not in the counter itself.

3.3 Typical Use Cases

3.3.1 Encoder Start-up

At power-up of the encoder the drive controller has to follow the state machine of the HIPERFACE® Slave according to chapter HIPERFACE® State Machine, p. 15.

A typical flow chart for the drive controller is shown below:

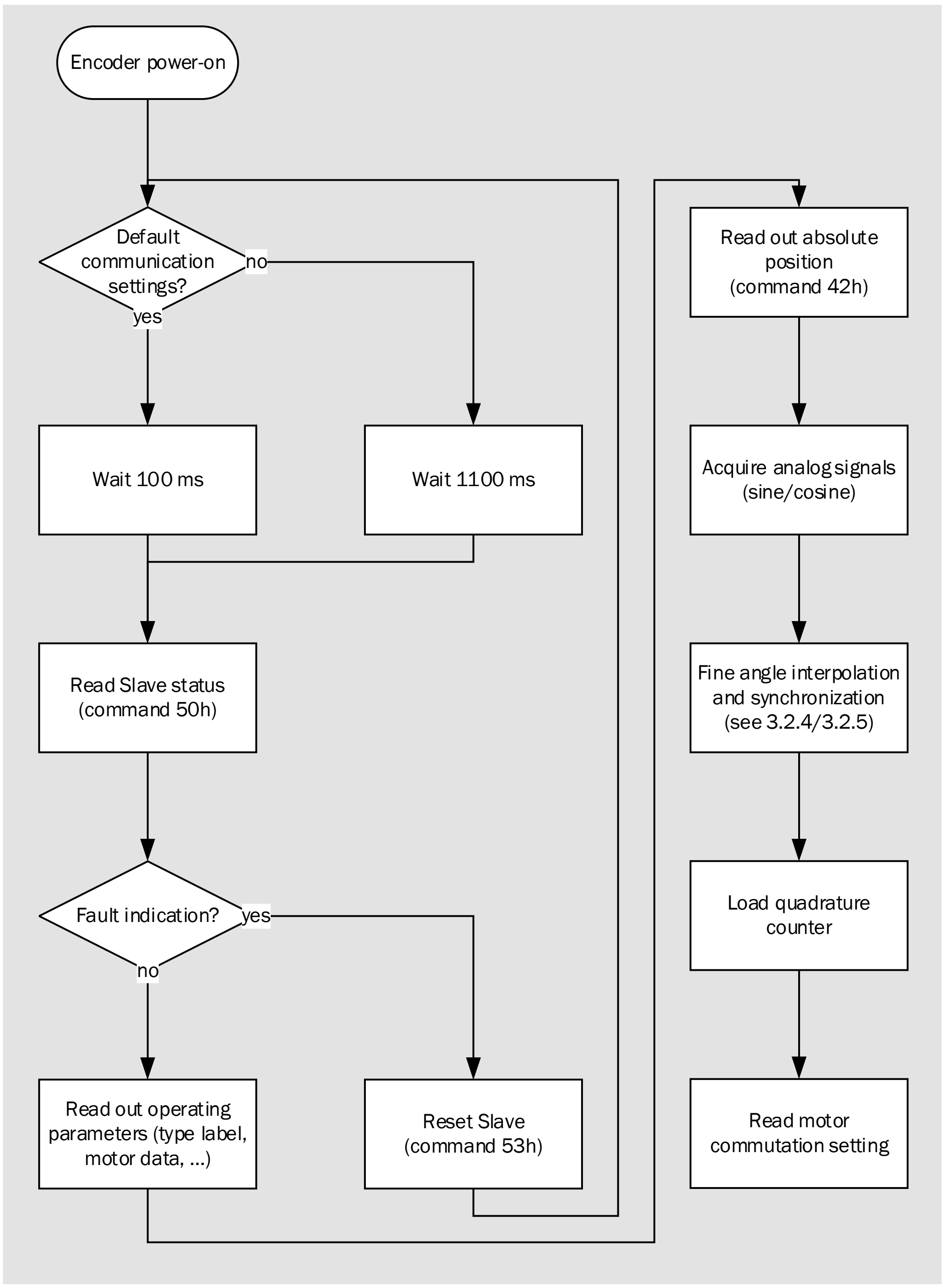

Encoder power-on
Default communication settings?
no
yes
Wait 100 ms
Wait 1100 ms
Read Slave status (command 50h)
Fault indication?
yes
no
Read out operating parameters (type label, motor data, ...)
Reset Slave (command 53h)
Read out absolute position (command 42h)
Acquire analog signals (sine/cosine)
Fine angle interpolation and synchronization (see 3.2.4/3.2.5)
Load quadrature counter
Read motor commutation setting

Figure 17: Start-up flow chart

Please note the following details:

- Some legacy HIPERFACE® Slave devices output Sin/Cos signal voltages directly after power-on. These signal voltages are typically not calibrated (offset/amplitude adjusted) until the initialization phase is over. Use of these signals by the HIPERFACE® Master during Initialization phase is not recommended.
- Depending on required communication settings the drive controller must wait for the “Default Communication Phase” to pass. This is reflected in the flowchart above with the differentiation between waiting 100 ms or 1100 ms.
- It is strongly recommended to read the Slave status after a power-on and react according to potential fault indications (see Fault Reaction, p. 72). Depending on electrical environment (voltage ramps, noise, EMI) it is even advisable to include a Slave reset as a default after power-on. This allows the encoder to stabilize and restart from a defined supply situation.

3.3.2 Position Acquisition

For the speed control loop of the drive controller the highly linear sine/cosine analog signals are used. The signals permit a high interpolation factor and thus a high resolution of the speed value.

For the position control loop different concepts are feasible.

- Typically, the incremental counting signals are used as base for the positioning control loop.
- If the cycle time of the position control loop is slow enough, the digital absolute position value through the RS485 interface can be directly accessed.

3.3.3 Safe Position Functions

Some HIPERFACE® Slaves offer safety functions for drive controllers as lined out in their datasheets resp. operating manuals. These safety functions are designed according to functional safety standards for machinery (IEC 61508 etc.).

Available safety functions are limited to position values that can be derived from the analog sine/cosine signals. Accordingly only safety functions based on position increment can be realized with one HIPERFACE® Slave. Other safety functions are possible when including additional position sources in a safety system.

Details on available safety functions, limits, and requirements for drive controller and/or motor manufacturer are specified in the respective operating manual or the Implementation Manual HIPERFACE® Safety (Order No 8014120).

3.3.4 Fault Reaction

HIPERFACE® fault messages should be treated individually to allow for maximum availability of the drive controller.

In a first step it is recommended to define basic drive controller actions based on the severity of a fault message. The following table lists different severity levels.

Severity	Basic drive controller action.
Minor	No action required. Can be used for long-term diagnostics.
Major	Typically stop of motor not required. Drive controller must perform specified action.
Critical	Fault handling must be performed during stop of motor (including a possible stand-still during start-up). If a critical fault occurs during motion, drive controller must initiate motor stop.

Table 73: HIPERFACE® fault severity

In a second step it is recommended to refine drive controller actions based on individual fault codes as lined out in the following table.

Fault code	Reference	Fault Group	Seve-rity	Controller response
01h	Incorrect alignment data (Incorrect alignment data (01h), p. 46)	Initialization	Critical	SW reset of Slave.
01h	Analog signals outside specification (Analog signals outside specification (01h), p. 46)	Others	Major	Retry command. If error persists, SW reset of Slave.
02h	Incorrect internal angular offset (Incorrect internal angular offset (02h), p. 47)	Initialization	Critical	SW reset of Slave.
02h	Position error (Incorrect internal angular offset (02h), p. 47) (follow-up to previous error when reading position)	Position	Critical	Retry command. If error persists, SW reset of Slave.
03h	Data field partitioning table destroyed (Data field partitioning table destroyed (03h), p. 47)	Initialization	Critical	SW reset of Slave.
03h	Data field partitioning table destroyed (Data field partitioning table destroyed (03h), p. 47)	Data	Critical	Verify/restore contents of data fields.
04h	Analog limit values not available (Analog limit values not available (04h), p. 48)	Initialization	Critical	SW reset of Slave.
05h	Internal I2C bus not operational (Internal I2C bus not operational (05h), p. 48)	Initialization	Critical	SW reset of Slave.
05h	Internal I2C bus not operational (Internal I2C bus not operational (05h), p. 48)	Position	Critical	Retry command. If error persists, SW reset of Slave.
06h	Internal checksum error (Internal checksum error (06h), p. 49)	Initialization	Critical	SW reset of Slave.
06h + 01h	CRC error; position invalid	Initialization	Critical	SW reset of Slave.
06h + 02h	CRC error; position/multi-turn invalid	Initialization	Critical	SW reset of Slave.
06h + 03h	CRC error; wrong HIPERFACE settings	Initialization	Critical	SW reset of Slave.
06h + 04h	CRC error; invalid limit values	Initialization	Critical	SW reset of Slave.
06h + 05h	CRC error; invalid Slave type label	Initialization	Critical	SW reset of Slave.
06h + 08h	CRC error; invalid user counter value	Initialization	Minor	-
06h + 12h	CRC error; extended type label invalid	Initialization	Critical	SW reset of Slave.
06h + 21h	CRC error; multiturn parameters invalid	Initialization	Critical	SW reset of Slave.

Fault code	Reference	Fault Group	Seve-rity	Controller response
07h	Program watchdog fault	Initialization	Minor	Read and delete message.
08h	Counter overflow	Others	Major	Reset user counter.
09h	Parity error	Protocol	Major	Retry command.
0Ah	Checksum error	Protocol	Major	Retry command.
0Bh	Unknown command	Protocol	Major	Retry command.
0Ch	Wrong command length	Protocol	Major	Retry command.
0Dh	Wrong command argument	Protocol	Major	Retry command.
0Eh	Read-only data field	Data	Major	Check/change sta-tus of data field.
0Fh	Incorrect access code	Data	Major	Check/change access code.
10h	Out of memory fault	Data	Major	Reduce data field size.
11h	Wrong data field offset	Data	Major	Check/change offset.
12h	Wrong data field number	Data	Major	Check/change number.
1Ch	Value monitoring analog signals	Position	Major	Retry command. Alternative: Start position estimation/ ride-through. If error persists, SW reset of Slave.
1Ch	Value monitoring analog signals	Others	Critical	Retry command. If error persists, SW reset of Slave.
1Dh	Transmitter current critical	Initialization	Critical	SW reset of Slave.
1Dh	Transmitter current critical	Position	Critical	Retry command. If error persists, SW reset of Slave.
1Dh	Transmitter current critical	Others	Critical	Retry command. If error persists, SW reset of Slave.
1Eh	Encoder temperature critical	Others	Major	Change motion profile to fit tempe-rature range.
1Fh	Speed too high	Position	Major	Check that maxi-mum allowed speed for HIPERFACE position read-out is considered.
20h	Singleturn position unreli-able	Initialization	Critical	SW reset of Slave.
20h	Singleturn position unreli-able	Position	Critical	Retry command. If error persists, SW reset of Slave.
20h	Sensor not adjusted or in adjustment mode	Position	Critical	Check sensor mode. Retry command. If error persists, SW reset of Slave.
21h	Multiturn amplitude fault	Initialization	Critical	SW reset of Slave.

Fault code	Reference	Fault Group	Seve-rity	Controller response
21h	Multiturn amplitude fault	Position	Critical	Retry command. If error persists, SW reset of Slave.
21h	Distance measure/sensor too high	Position	Critical	Check assembly of sensor.
22h	Multiturn sync fault	Position	Critical	Only use sine/cosi-ne based position until shut-down. Replace Slave.
23h	Multiturn vectorlength fault	Position	Critical	Retry command. If error persists, SW reset of Slave.
23h	Linear position fault	Position	Critical	Retry command. If error persists, SW reset of Slave.
24h	Multiturn counter fault	Position	Critical	Retry command. If error persists, SW reset of Slave.

Table 74: HIPERFACE® fault controller actions

Additional fault messages can be defined for new and future HIPERFACE® Slaves or are legacy codes typically no longer in use. It is expected that a HIPERFACE® Master can only react with the worst-case fault reaction to unknown fault codes (i.e. motor stop).

3.3.5 Motor Commutation Setting

For motor commutation the offset between the feedback (HIPERFACE® Slave) output and the motor shaft must be known during assembly of the motor.

The following methods are available:

Data field-based

With a blocked motor shaft the high-resolution position is measured and calculated from the digital absolute position and the interpolated angle value (see Position Acquisition, p. 72). This position value is then stored in a data field of the Slave (see Store Data, p. 27).

This method requires that the drive controller knows where this information is stored in a given motor type. It permits the accurate setting of the commutation, even for high pole motors.

Using “Set position”

With a blocked motor shaft the drive controller acquires the interpolated angle value (see Fine Angle Interpolation, p. 66). Then the function “Set position” (see Set Position, p. 19) is called where the 5 LSB of the preset value in this function call match the 5 MSB of the interpolated angle value in order to keep the phase shift between digital absolute position and sine/cosine signals at 0°.

This commutation setting method can only be set with the resolution of whole periods of the Slave. For Slaves with small period counts the setting of the commutation may be too inaccurate and limit the achievable performance of the drive system.

This method is universally applicable even if the drive controller has no detailed information on the motor type.

Note:

A motor manufacturer planning to use motors with third-party motor controllers should note that some controllers will inhibit motion control if the absolute position and sine/cosine signals are not synchronized.

Using “Set position with synchronization”

This method is similar to Using “Set position”, p. 75 except that the drive controller does not need to acquire an interpolated angle value beforehand.

The advantages and disadvantages are identical to Using “Set position”, p. 75.

Note:

This command is optional for HIPERFACE® Slaves.

Using “Read synchronization offset”

With a blocked motor shaft the drive controller performs a commutation offset setting with the method as lined out in Using “Set position”, p. 75.

The “missing” LSBs of the interpolated angle value can be returned by the Slave through the command “Read synchronization offset” (see Read Synchronization Offset, p. 44). With this information the drive controller can perform commutation to the highest available resolution.

This method is universally applicable even if the drive controller has no detailed information on the motor type.

Note:

This command is optional for HIPERFACE® Slaves.

3.3.6 Absolute Position Cross-check

If a motor controller implements position acquisition by reading the absolute position only once at start-up (see Position Acquisition, p. 72) there remains a risk of counter-based faults. Such faults could lead to slow build-up of a wrongly incremented position value. If the incremental fault gets high enough commutation faults of the motion control loop or even a critical self-excitation could happen leading to uncontrolled movement.

In order to control this behavior a cross-check between counted and absolute position can be implemented in the motor controller.

ATTENTION:

This position cross-check is not foreseen for use in safety functions. HIPERFACE® Slaves only support safety functions based on their sine/cosine signals.

A good implementation of the position cross-check should take note of the following items:

- The leading information in case of discrepancies is the absolute position, not the counted position.
- Use of a hardware-based signal synchronization as lined out in Position Signal Synchronization, p. 64 is strongly recommended to allow for position cross-check with small tolerance values even at high shaft speeds.
- For comparison of absolute and counted positions please note that HIPERFACE® Slaves resolve absolute position values to 5 bits within one sine/cosine period. A position cross-check with higher resolution is not possible.
- When choosing the cycle time of the position cross-check the probability (and effect) of a counting fault must be weighed with the probability of signal transmission faults on the EIA-485 interface during operation. It is strongly recommended to include fault tolerance mechanisms to counter sporadic transmission faults. Typical cycle times fall in the range of 50 ms to 1 s.

8010701/YUV9/2016-02-12 · COMAT (2016-02) · A4 4c int44

Australia
Phone +61 3 9457 0600
1800 334 802 – tollfree
E-Mail sales@sick.com.au

Austria
Phone +43 (0)22 36 62 28 8-0
E-Mail office@sick.at

Belgium/Luxembourg
Phone +32 (0)2 466 55 66
E-Mail info@sick.be

Brazil
Phone +55 11 3215-4900
E-Mail marketing@sick.com.br

Canada
Phone +1 905 771 14 44
E-Mail information@sick.com

Czech Republic
Phone +420 2 57 91 18 50
E-Mail sick@sick.cz

Chile
Phone +56 2 2274 7430
E-Mail info@schadler.com

China
Phone +86 4000 121 000
E-Mail info.china@sick.net.cn

Denmark
Phone +45 45 82 64 00
E-Mail sick@sick.dk

Finland
Phone +358-9-2515 800
E-Mail sick@sick.fi

France
Phone +33 1 64 62 35 00
E-Mail info@sick.fr

Gemany
Phone +49 211 5301-301
E-Mail info@sick.de

Great Britain
Phone +44 (0)1727 831121
E-Mail info@sick.co.uk

Hong Kong
Phone +852 2153 6300
E-Mail ghk@sick.com.hk

Hungary
Phone +36 1 371 2680
E-Mail office@sick.hu

India
Phone +91–22–4033 8333
E-Mail info@sick-india.com

Israel
Phone +972-4-6881000
E-Mail info@sick-sensors.com

Italy
Phone +39 02 27 43 41
E-Mail info@sick.it

Japan
Phone +81 (0)3 5309 2112
E-Mail support@sick.jp

Malaysia
Phone +603 808070425
E-Mail enquiry.my@sick.com

Netherlands
Phone +31 (0)30 229 25 44
E-Mail info@sick.nl

New Zealand
Phone +64 9 415 0459
0800 222 278 – tollfree
E-Mail sales@sick.co.nz

Norway
Phone +47 67 81 50 00
E-Mail sick@sick.no

Poland
Phone +48 22 837 40 50
E-Mail info@sick.pl

Romania
Phone +40 356 171 120
E-Mail office@sick.ro

Russia
Phone +7-495-775-05-30
E-Mail info@sick.ru

Singapore
Phone +65 6744 3732
E-Mail sales.gsg@sick.com

Slovakia
Phone +421 482 901201
E-Mail mail@sick-sk.sk

Slovenia
Phone +386 (0)1-47 69 990
E-Mail office@sick.si

South Africa
Phone +27 11 472 3733
E-Mail info@sickautomation.co.za

South Korea
Phone +82 2 786 6321
E-Mail info@sickkorea.net

Spain
Phone +34 93 480 31 00
E-Mail info@sick.es

Sweden
Phone +46 10 110 10 00
E-Mail info@sick.se

Switzerland
Phone +41 41 619 29 39
E-Mail contact@sick.ch

Taiwan
Phone +886 2 2375-6288
E-Mail sales@sick.com.tw

Thailand
Phone +66 2645 0009
E-Mail tawiwat@sicksgp.com.sg

Turkey
Phone +90 (216) 528 50 00
E-Mail info@sick.com.tr

United Arab Emirates
Phone +971 (0) 4 88 65 878
E-Mail info@sick.ae

USA/Mexico
Phone +1(952) 941-6780
1 (800) 325-7425 – tollfree
E-Mail info@sick.com

Vietnam
Phone +84 8 62920204
E-Mail Ngo.Duy.Linh@sicksgp.com.sg

More representatives and agencies
at **www.sick.com**

SICK AG | Waldkirch | Germany | www.sick.com

SICK
Sensor Intelligence.