

SLINK3

*Protocol Implementation
for FlexMax drives*

Serial communication
manual

POWERTEC
INDUSTRIAL MOTORS

The information held in this manual can be modified without notice and Powertec cannot be held responsible for errors or omissions.

For no reason no part of this manual can be reproduced in any form or by any means (including recording and photocopying) without a written consent of Powertec Industrial Motors.

Before the drive installation, wiring, commissioning and inspection, read carefully this instruction manual.

Keep the manual in a safe place and available to operation and maintenance personnel.

Powertec is not responsible for those mistakes that may be found in this manual and for the damages that they may arouse.

All rights reserved.

Index

1. INTRODUCTION	5
2. FRAME GENERAL DESCRIPTION	5
3. COMMANDS DESCRIPTION	7
4. RESPONSE DESCRIPTION	8
5. PARAMETERS TRANSFER BETWEEN MASTER AND DRIVE	9
5.1. WRITE TYPED PARAMETER: MASTER->SLAVE	11
5.2. WRITE TYPED PARAMETER: SLAVE->MASTER	11
5.2.1. <i>OK response case</i>	11
5.2.2. <i>Error response case</i>	12
5.3. READ TYPED PARAMETER: MASTER->SLAVE	12
5.4. READ TYPED PARAMETER:SLAVE->MASTER	13
5.4.1. <i>OK response case</i>	13
5.4.2. <i>Error response case</i>	13
5.5. WRITE STANDARD PARAMETER: MASTER->SLAVE	14
5.6. WRITE STANDARD PARAMETER: SLAVE->MASTER	14
5.6.1. <i>OK response case</i>	14
5.6.2. <i>Error response case</i>	14
5.7. READ STANDARD PARAMETER: MASTER->SLAVE	15
5.8. READ STANDARD PARAMETER: SLAVE->MASTER	15
5.8.1. <i>OK response case</i>	15
5.8.2. <i>Error response case</i>	15
6. RESET SLAVE	16
6.1. RESET SLAVE COMMAND: MASTER->SLAVE	16
6.2. RESET SLAVE COMMAND: SLAVE->MASTER	16
6.2.1. <i>OK response case</i>	16
6.2.2. <i>Error response case</i>	16
7. STATUS WORD	17
7.1. STATUS WORD COMMAND: MASTER->SLAVE	17
7.2. STATUS WORD COMMAND: SLAVE->MASTER	17
7.2.1. <i>OK response case</i>	17
7.2.2. <i>Error response case</i>	17

8. MALFUNCTION CODE	18
8.1. MALFUNCTION CODE COMMAND: MASTER->SLAVE	18
8.2. MALFUNCTION CODE COMMAND: SLAVE->MASTER	18
8.2.1. <i>OK response case</i>	18
8.2.2. <i>Error response case</i>	18
9. BROADCAST COMMAND VERIFY	19
9.1. BROADCAST COMMAND VERIFY: MASTER->SLAVE	19
9.2. BROADCAST COMMAND VERIFY: SLAVE->MASTER	19
9.2.1. <i>OK response case</i>	19
9.2.2. <i>Error response case</i>	19
10. PARAMETERS TRANSFER BETWEEN MASTER AND DGFC386	20
10.1. WRITE TYPED PARAMETER: MASTER->DGFC386	20
10.2. WRITE TYPED PARAMETER: DGFC386->MASTER	20
10.2.1. <i>OK response case</i>	20
10.2.2. <i>Error response case</i>	21
10.3. READ TYPED PARAMETER: MASTER->DGFC386	21
10.4. READ TYPED PARAMETER: DGFC386->MASTER	22
10.4.1. <i>OK response case</i>	22
10.4.2. <i>Error response case</i>	22
11. SPECIAL ACCESS TO DGFC386.....	23
11.1. SPECIAL DGFC386 ACCESS COMMAND: MASTER->DGFC386	23
11.2. SPECIAL DGFC386 ACCESS COMMAND: DGFC386->MASTER	23
11.2.1. <i>OK response case</i>	23
11.2.2. <i>Error response case</i>	24
12. SERIAL COMMUNICATION REQUIREMENTS	24
12.1. RS485 CONNECTOR SETUP	24
12.2. PORT SETTING	24
12.3. TIME RESTRICTIONS AND CONSIDERATIONS	24
12.3.1. <i>Byte to Byte delay specification</i>	24
12.3.2. <i>Slave answer delay specification</i>	25
12.3.3. <i>Replay by the Slave to Next Master order delay specification</i>	25
12.3.4. <i>Frame transfer duration time calculation</i>	25
13. COMMUNICATION EXAMPLE	25

1. INTRODUCTION

SLINK3 protocol defines the implementation of a data exchange on serial communication link (RS485) between a **Master** (i.e. PC or PLC) and one or more **Slaves** (SIEI Drives or DGFC386).

If the serial line is connected with the Drive it is possible to access the optional DGFC386 card as Destination ID 2.

If the serial line is connected with the DGFC386 it is possible to access the Drive as Destination ID 2.

The purpose of this document is to describe the SLINK3 protocol implementation when interfacing to a SIEI Drive.

Here will be described as the frame is composed, the frame formats for data transfer.

2. FRAME GENERAL DESCRIPTION

SLINK3 specifies a data transfer based on binary coded bytes. Each byte has a special meaning depending on the informations that have to be exchanged.

Here is described how a frame is composed:

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
		255	broadcast
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGHT	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	BIT0: 0 = MASTER frame 1 = SLAVE frame	frame informations
		BIT1...3: destination ID	
		BIT4: 1 = when slave is in fault condition	
		BIT5...7: reserved	
6	COMMAND/RESPONSE (LOW)	0 ... 255	byte low of command/response
7	COMMAND/RESPONSE (HIGH)	0 ... 255	byte high of command/response
6 .. LENGHT -1	DATA	dipending on information type	it is the content of the frame
LENGHT	CHECKSUM	BYTE 2...(LENGHT-1) XOR	vertical parity

slg0010

Columns description:

BYTE:	It represents the order in which bytes are transmitted/received.
NAME :	It is the name of the byte, specifying the information contents.
VALUE:	It is the byte value expressed in hex where specified , otherwise it gives a single bits description..
COMMENTS:	Little description.

Fields description:

START CODE:	It is the frame starting byte, always the same for all frames types.						
ADDRESS:	It identifies the slave addressed from the master; the slave address must be configured by the user on the Drive (Parameter 8511 “Address”) before serial communication connection; there can be 128 slaves connected to a master at once (0..127). Address value 255 specify broadcast communication (transfer to all connected Slaves at a time); in this case the Slaves don’t reply to the Master message; for more informations on broadcast communication refer to chapter 9.						
CONTROL:	It identifies the frame type; there can be two frame types: numbered (information frames) and unnumbered (synchronization frames); unnumbered frames refer to synchronization between Master and Slaves purposes (not described here) . xxxx bits are the frame sequence identifier: each frame sent from the master has a specific sequence number identifier ; the addressed slave always replies to the frame sent by the Master with the same frame sequence identifier; this can help the Master to recognize lost frames, slave answers timeouts or frames overlap.						
LENGTH:	It is the entire frame length expressed in bytes.						
ORDER/INFOS:	Informations of the frame: <table> <tr> <td>BIT 0 :</td> <td>It is set to “0” by the Master and to “1” by the Slave;</td> </tr> <tr> <td>BIT 1..3 :</td> <td>It is the Destination ID (i.e. when the frame is sent to a Slave option): 0 means that the frame is sent to the Slave itself, 1..7 means the frame is sent to a different destination connected to the Slave.</td> </tr> <tr> <td>BIT 4 :</td> <td>It is set by the Slave to allow the Master to monitor the Slave status: it is “0” when the Slave is working correctly, “1” when the Slave is in a fault condition; refer to Slave manual for failures recognition and management.</td> </tr> </table>	BIT 0 :	It is set to “0” by the Master and to “1” by the Slave;	BIT 1..3 :	It is the Destination ID (i.e. when the frame is sent to a Slave option): 0 means that the frame is sent to the Slave itself, 1..7 means the frame is sent to a different destination connected to the Slave.	BIT 4 :	It is set by the Slave to allow the Master to monitor the Slave status: it is “0” when the Slave is working correctly, “1” when the Slave is in a fault condition; refer to Slave manual for failures recognition and management.
BIT 0 :	It is set to “0” by the Master and to “1” by the Slave;						
BIT 1..3 :	It is the Destination ID (i.e. when the frame is sent to a Slave option): 0 means that the frame is sent to the Slave itself, 1..7 means the frame is sent to a different destination connected to the Slave.						
BIT 4 :	It is set by the Slave to allow the Master to monitor the Slave status: it is “0” when the Slave is working correctly, “1” when the Slave is in a fault condition; refer to Slave manual for failures recognition and management.						
COMMAND/RESPONSE:	When the Master is transmitting, it represents the command code (i.e. parameter writing/reading) When the Slave is transmitting, it is the response code. For the complete COMMAND/RESULT codings see chapters 3. and 4.						
DATA:	It is the information transferred with the frame; it will be described deeply later.						
CHECKSUM:	It is the XOR of bytes 2 ... (LENGTH-1).						

3. COMMANDS DESCRIPTION

Here are the complete COMMAND field codings:

COMMAND	VALUE	USED in SIEI drives	USED in DGFC-386
Reset Slave	0020H	YES	NO
State Read	0021H	NO	NO
State Change	0022H	NO	NO
Status Word	0023H	YES	NO
Read standard Parameter Drivecom	0024H	YES	NO
Write standard Parameter Drivecom	0025H	YES	NO
Read Typed Parameter	0026H	YES	YES
Write Typed Parameter	0027H	YES	YES
Broadcast command verify	0028H	YES	NO
Parameter upload	0029H	NO	NO
Parameter dowload	002AH	NO	NO
Continue transfer	002BH	NO	NO
Multiple Parameters Read	002CH	NO	NO
Remote Keypad Command	002DH	NO	NO
Slave Description Command	002EH	NO	NO
Malfunction Code	002FH	YES	NO
Read object dictionary (database format)	0030H	NO	NO
Write object dictionary (database format)	0031H	NO	NO
Sequence Command	0032H	NO	NO
End Sequence	0033H	NO	NO
Reset Sequence	0034H	NO	NO
Run Command sequence	0035H	NO	NO
End transfer	0036H	NO	NO
Not used	0050H-9FH	NO	NO
DGFC Special access	0ABCDH	NO	YES
User defined	0100H-FF00H	NO	NO

slg0020

SIEI and DGFC386 implemented commands will be described later.

4. RESPONSE DESCRIPTION

Here are the complete RESPONSE codings:

RESPONSE	VALUE
OK no error	0000H
Parameter not exist	0001H
Reserved	0002H
Control Access denied	0003H
Reserved	0004H
Attribute Access denied	0005H
Type value error	0006H
Reserved (SIEI)	0007H-000FH
Destination option not exist	0010H
Parameter Access Conflict	0011H
Value out of the maximum range	0012H
Value out of the minimum range	0013H
Value not supported	0014H
Parameter Configuration Conflict	0015H
Command Submitted	0016H
Reserved	0017H
Unknown Command	0018H
Read only Parameter	0019H
Write not allowed	001AH
Value out of constant limits	001BH
State not correct	001CH
Password	001DH
Type Unknown	001EH
Reserved SIEI	001FH-007CH
Transfer ended	0080H
Ok broadcast	0081H
Reserved SIEI	0082H-0096H-00FCH
Frame error	0097H
Reserved SIEI	0098H-00FCH
Protocol errors	00FDH-00FEH
NOK generic	00FFH
User defined	0100H-FFFFH

slg0030

Parameter not exist	The specified parameter does not exist.
Control Access denied	The access is denied because of the control status.
Attribute Access denied	The parameter attributes do not allow the access.
Type value error	The specified value type is incorrect.
Destination option not exist	The destination option does not exist at node.
Parameter Access Conflict	The addressed parameter can not be accessed (for example if the command is write and parameter is connected to an external input).
Value out of the max range	Value is out of the maximum range.
Value out of the min range	Value is out of the minimum range.
Value not supported	Value is in range but not allowed.

Parameter Configuration Conflict	The addressed parameter can not be accessed for sistem configuration conflict (for example the order try to connect an input source to a parameter that is already connected to an input source).
Command Submitted	Command has been submitted but is not possible to know if it has been exucuted .
Unknown Command	The command in the order message is not known.
Read only Parameter	The parameter has read only attribute.
Write not allowed	Write operation is not allowed for the slave conditions.
Value out of constant limits	Value is out of constant fixed limits.
State not correct	The control state doesn't allow the command execution.
Password	The command is not executed because the password is not active.
Type Unknown	The parameter type is not known.
NOK generic	The access is aborted because of an indeterminated error.
Ok broadcast	Response to a Broadcast Command Verify.
Protocol errors	Code FDH: the Slave detected an error in the incoming frame: LENGTH field was wrong.
Transfer ended	Response code when a Parameters upload COMMAND has been completed.
Frame error	The Slave detected an error in the incoming frame (PArity, Overrun, etc.).

5. PARAMETERS TRANSFER BETWEEN MASTER AND DRIVE

Here the parameters transfer between Master and a SIEI drive will be described.

Parameters transfer is divided in two types: Standard (Drivecom) and Typed;

Standard transfer is based on Drivecom* specification that identifies each parameter with an INDEX and a SUBINDEX; the format of the parameter (i.e. integer, long integer etc.) must be known by the master and it is not specified in the frame.

Typed parameter transfer allow parameters transfer with type specification; the parameters are identified by INDEX, SUBINDEX(optional) and format (integer , long integer, floating point et.); in this way the slave can check if the specified parameter format is the same contained in its database.

Here is a list of the supported parameters value types: under DATA TYPE is specified the value type („C“ language notation) and the value length (in bytes); under CODE the FORMAT code for typed parameter transfer is specified (see chapters 5.1 and 5.3.).

***DRIVECOM** is a standard defined by “DRIVECOM Nutzergruppe e.V.”, an association of Drives manufacturers acted to standarize the interfacing between different Drives. The standard defines the most important Drive hardware functions and summarize them in a profile for variable-speed Drives.

The SIEI supported profile is according to:

Power Trasmission, profile n° : 21, DRIVECOM Nutzergruppe e.V.

DATA TYPE	CODE	USED in SIEI drives	USED in DGFC-386
Void	00H	NO	YES
Boolean (1 byte)	01H	NO	YES
Char (byte)	02H	NO	YES
Short Int (2 bytes)	03H	YES	YES
Long Int (4 bytes)	04H	YES	YES
Unsigned Char (byte)	05H	NO	YES
Unsigned Short Int(2 bytes)	06H	YES	YES
Unsigned Long Int (4 bytes)	07H	YES	YES
Float single precision (4 bytes)	08H	YES	YES
String	09H	YES	NO
Octet String	0AH	NO	NO
Reserved	0BH-0EH	NO	NO
Float double precision (8 bytes)	0FH	NO	NO
Long double (10 bytes)	10H	NO	NO
Reserved	11H-1FH	NO	NO

slg0040

During the transfer, values are sent starting from the lower byte of the value; here are some examples:

- Char transfer (1 byte): value 128
The only byte composing VALUE field is 128
- Short int (2 bytes): value 1234H
The order of sent bytes will be
VALUE byte 0: 34H
VALUE byte 1: 12H
- Long int (4 bytes): value 12345678H
The order of sent bytes will be
VALUE byte 0: 78H
VALUE byte 1: 56H
VALUE byte 2: 34H
VALUE byte 3: 12H
- Floating point: the floating point numbers are expressed as specified in IEEE-754 format:

Floating point IEEE-754 Single precision			
Byte 0	Byte 1	Byte 2	Byte 3
mmmmmmmm	mmmmmmmm	emmmmmmm	seeeeeee

slg0050

m = mantissa, e =exponent s = segno

Float (4 bytes) : value 3.141592654

Value byte 0: DBH
Value byte 1: 0FH
Value byte 2: 49H
Value byte 3: 40H

5.1. Write typed parameter: MASTER->SLAVE

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	00H	frame informations
6	COMMAND (LOW)	27H	byte low of command
7	COMMAND (HIGH)	00H	byte high of command
8	FORMAT	BIT 0 ... 4 : format BIT 5, 6 : reserved BIT 7 : 0 ... 1	BIT 0...4: parameter value format BIT 7 "0" subindex not included in the frame; BIT 7 "1" subindex included in the frame
9	INDEX (LOW)	0 ... 255	byte low of parameter INDEX
10	INDEX (HIGH)	0 ... 255	byte high of parameter INDEX

Subindex NOT specified (see FORMAT field):

11	VALUE Byte 0	0 ... 255	first byte of value
12	VALUE Byte 1	0 ... 255	second byte of value
13 ... LENGTH-1	VALUE (continued)	0 ... 255	remaining bytes of value, see FORMAT specification

Subindex specified (see FORMAT field):

11	SUBINDEX	0 ... 255	parameter subindex
12	VALUE Byte 0	0 ... 255	first byte of value
13	VALUE Byte 1	0 ... 255	second byte of value
14 ... LENGTH-1	VALUE (continued)	0 ... 255	remaining bytes of value, see FORMAT specification
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0060

5.2. Write typed parameter: SLAVE->MASTER

5.2.1. OK response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	00H	byte low of response
7	RESPONSE (HIGH)	00H	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0070

5.2.2. Error response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	ERROR CODE (LOW)	byte low of response
7	RESPONSE (HIGH)	ERROR CODE (HIGH)	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0080

5.3. Read typed parameter: MASTER->SLAVE

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	00H	frame informations
6	COMMAND (LOW)	26H	byte low of command
7	COMMAND (HIGH)	00H	byte high of command
8	FORMAT	BIT 0 ... 4 : format BIT 5, 6 : reserved BIT 7 : 0 ... 1	BIT 0...4: parameter value format BIT 7 "0" subindex not included in the frame; BIT 7 "1" subindex included in the frame
9	INDEX (LOW)	0 ... 255	byte low of parameter INDEX
10	INDEX (HIGH)	0 ... 255	byte high of parameter INDEX

Subindex NOT specified (see FORMAT field):

LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity
--------	----------	----------------------------	-----------------

Subindex specified (see FORMAT field):

11	SUBINDEX	0 ... 255	parameter subindex, only if subindex specified in FORMAT field
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0090

5.4. Read typed parameter:SLAVE->MASTER

5.4.1. OK response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	00H	byte low of response
7	RESPONSE (HIGH)	00H	byte high of response
8	FORMAT	BIT 0 ... 4 : format BIT 5, 6 : reserved BIT 7 : 0 ... 1	BIT 0...4: parameter value format BIT 7 "0" subindex not included in the frame BIT 7 "1" subindex included in the frame
9	INDEX (LOW)	0 ... 255	byte low of parameter INDEX
10	INDEX (HIGH)	0 ... 255	byte high of parameter INDEX

Subindex NOT specified (see FORMAT field):

11	VALUE Byte 0	0 ... 255	first byte of value
12	VALUE Byte 1	0 ... 255	second byte of value
13 ... LENGTH-1	VALUE (continued)	0 ... 255	remaining bytes of value, see FORMAT specification

Subindex specified (see FORMAT field):

11	SUBINDEX	0 ... 255	parameter subindex
12	VALUE Byte 0	0 ... 255	first byte of value
13	VALUE Byte 1	0 ... 255	second byte of value
14 ... LENGTH-1	VALUE (continued)	0 ... 255	remaining bytes of value, see FORMAT specification
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0100

5.4.2. Error response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	ERROR CODE (LOW)	byte low of response
7	RESPONSE (HIGH)	ERROR CODE (HIGH)	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0080

5.5. Write standard parameter: MASTER->SLAVE

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	00H	frame informations
6	COMMAND (LOW)	25H	byte low of command
7	COMMAND (HIGH)	00H	byte high of command
8	INDEX (LOW)	0 ... 255	byte low of parameter INDEX
9	INDEX (HIGH)	0 ... 255	byte high of parameter INDEX
10	SUBINDEX	0 ... 255	parameter subindex
11	VALUE Byte 0	0 ... 255	first byte of value
12	VALUE Byte 1	0 ... 255	second byte of value
13 ... LENGTH-1	VALUE (continued)	0 ... 255	remaining bytes of value, see FORMAT specification
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0110

5.6. Write standard parameter: SLAVE->MASTER

5.6.1. OK response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	00H	byte low of response
7	RESPONSE (HIGH)	00H	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0070

5.6.2. Error response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	ERROR CODE (LOW)	byte low of response
7	RESPONSE (HIGH)	ERROR CODE (HIGH)	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0080

5.7. Read standard parameter: MASTER->SLAVE

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	00H	frame informations
6	COMMAND (LOW)	24H	byte low of command
7	COMMAND (HIGH)	00H	byte high of command
8	INDEX (LOW)	0 ... 255	byte low of parameter INDEX
9	INDEX (HIGH)	0 ... 255	byte high of parameter INDEX
10	SUBINDEX	0 ... 255	parameter subindex
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0120

5.8. Read standard parameter: SLAVE->MASTER

5.8.1. OK response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	00H	byte low of response
7	RESPONSE (HIGH)	00H	byte high of response
8	INDEX (LOW)	0 ... 255	byte low of parameter INDEX
9	INDEX (HIGH)	0 ... 255	byte high of parameter INDEX
10	SUBINDEX	0 ... 255	parameter subindex
11	VALUE Byte 0	0 ... 255	first byte of value
12	VALUE Byte 1	0 ... 255	second byte of value
13 ... LENGTH-1	VALUE (continued)	0 ... 255	remaining bytes of value, see FORMAT specification
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0130

5.8.2. Error response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	ERROR CODE (LOW)	byte low of response
7	RESPONSE (HIGH)	ERROR CODE (HIGH)	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0080

6. RESET SLAVE

The Reset Slave command initiates a reset of the serial communication interface of the addressed Slave. This can helpfull after an undetermined error on serial line has occurred . Data field is not used. A reply message is returned to the master with OK response code if COMMAND completed successfully.

6.1. Reset slave command: MASTER->SLAVE

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	00H	frame informations
6	COMMAND (LOW)	20H	byte low of command
7	COMMAND (HIGH)	00H	byte high of command
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0220

6.2. Reset slave command: SLAVE->MASTER

6.2.1. OK response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	00H	byte low of response
7	RESPONSE (HIGH)	00H	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0070

6.2.2. Error response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	ERROR CODE (LOW)	byte low of response
7	RESPONSE (HIGH)	ERROR CODE (HIGH)	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0080

7. STATUS WORD

The Status Word command causes the slave to read the status flags. For Status word specification refer to Drivecom specifications.

7.1. Status word command: MASTER->SLAVE

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	00H	frame informations
6	COMMAND (LOW)	23H	byte low of command
7	COMMAND (HIGH)	00H	byte high of command
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0230

7.2. Status word command: SLAVE->MASTER

7.2.1. OK response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	00H	byte low of response
7	RESPONSE (HIGH)	00H	byte high of response
8	STATUS WORD (LOW)	0 ... 255	first byte of Status word
9	STATUS WORD (HIGH)	0 ... 255	second byte of Status word
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0240

7.2.2. Error response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	ERROR CODE (LOW)	byte low of response
7	RESPONSE (HIGH)	ERROR CODE (HIGH)	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0080

8. MALFUNCTION CODE

The malfunction code command allows the master to read the malfunction present on the slave device. For malfunction code specification refer to the Slave manual.

8.1. Malfunction code command: MASTER->SLAVE

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	00H	frame informations
6	COMMAND (LOW)	2FH	byte low of command
7	COMMAND (HIGH)	00H	byte high of command
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0250

8.2. Malfunction code command: SLAVE->MASTER

8.2.1. OK response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	00H	byte low of response
7	RESPONSE (HIGH)	00H	byte high of response
8	MALFUNCTION CODE (LOW)	0 ... 255	first byte of malfunction code
9	MALFUNCTION CODE (HIGH)	0 ... 255	second byte of malfunction code
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0260

8.2.2. Error response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	ERROR CODE (LOW)	byte low of response
7	RESPONSE (HIGH)	ERROR CODE (HIGH)	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0080

9. BROADCAST COMMAND VERIFY

After a broadcast communication (ADDRESS field = 255), the slaves cannot reply to the message because there would be conflict on the serial bus. Therefore, the Master has to inspect all single slaves afterward to ensure they have received the broadcast command. The broadcast command verify allows the master to accomplish this action. A reply message is returned with the response, OK BROADCAST means broadcast message received.

9.1. Broadcast command verify: MASTER->SLAVE

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	00H	frame informations
6	COMMAND (LOW)	28H	byte low of command
7	COMMAND (HIGH)	00H	byte high of command
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0270

9.2. Broadcast command verify: SLAVE->MASTER

9.2.1. OK response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	81H	byte low of response
7	RESPONSE (HIGH)	00H	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0280

9.2.2. Error response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	01H- slave OK 11H- failure present on the slave	frame informations
6	RESPONSE (LOW)	ERROR CODE (LOW)	byte low of response
7	RESPONSE (HIGH)	ERROR CODE (HIGH)	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0080

10. PARAMETERS TRANSFER BETWEEN MASTER AND DGFC386

The parameters transfer between Master and DGFC386 is accomplished in the same way of Master-Drive one; in this case The Drive operates like a bridge between Master and DGFC386; the supported COMMANDs are Write typed parameter and Read typed parameter. The DGFC386 is addressed specifying the Destination ID value of ORDER/INFOS field: it must be set to 2. Refer to chapter V for DATA field description.

Note: Communication between Master and DGFC386 can also be accomplished by connecting directly the Master to DGFC386 X3 connector: DGFC386 supports SLINK3 Write typed parameter (027H), read typed parameter (26H) and Special access (0ABCDH) COMMANDs; when accessing directly to DGFC386, the Destination ID bits of ORDER/INFOS Field must be set to 0; also for response message the field ORDER/INFOS will be 01 or 11H. In this case it is possible to send the message to the Drive through the DGFC386 specifying the Destination ID in the field ORDER/INFOS as 2. The DGFC386 ADDRESS can be set by means of DGFC386 parameters 511.

10.1. Write typed parameter: MASTER->DGFC386

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	04H	frame informations, DGFC386 destination
6	COMMAND (LOW)	27H	byte low of command
7	COMMAND (HIGH)	00H	byte high of command
8	FORMAT	BIT 0 ... 4 : format BIT 5, 6 : reserved BIT 7 : 0 ... 1	BIT 0...4: parameter value format BIT 7 "0" subindex not included in the frame; BIT 7 "1" subindex included in the frame
9	INDEX (LOW)	0 ... 255	byte low of parameter INDEX
10	INDEX (HIGH)	0 ... 255	byte high of parameter INDEX
11	VALUE Byte 0	0 ... 255	first byte of value
12	VALUE Byte 1	0 ... 255	second byte of value
13 ... LENGTH-1	VALUE (continued)	0 ... 255	remaining bytes of value, see FORMAT specification
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0140

10.2. Write typed parameter: DGFC386->MASTER

10.2.1. OK response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	05H- DGFC386 OK 15H- failure present on the DGFC386	frame informations, DGFC386 destination
6	RESPONSE (LOW)	00H	byte low of response
7	RESPONSE (HIGH)	00H	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0150

10.2.2. Error response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	05H- DGFC386 OK 15H- failure present on the DGFC386	frame informations, DGFC386 destination
6	RESPONSE (LOW)	ERROR CODE (LOW)	byte low of response
7	RESPONSE (HIGH)	ERROR CODE (HIGH)	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0160

10.3. Read typed parameter: MASTER->DGFC386

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	04H	frame informations, DGFC386 addressing
6	COMMAND (LOW)	26H	byte low of command
7	COMMAND (HIGH)	00H	byte high of command
8	FORMAT	BIT 0 ... 4 : format BIT 5, 6 : reserved BIT 7 : 0 ... 1	BIT 0...4: parameter value format BIT 7 "0" subindex not included in the frame; BIT 7 "1" subindex included in the frame
9	INDEX (LOW)	0 ... 255	byte low of parameter INDEX
10	INDEX (HIGH)	0 ... 255	byte high of parameter INDEX
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0170

10.4. Read typed parameter: DGFC386->MASTER

10.4.1. OK response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	05H- DGFC386 OK 15H- failure on the DGFC386	frame informations, DGFC386 destination
6	RESPONSE (LOW)	00H	byte low of response
7	RESPONSE (HIGH)	00H	byte high of response
8	FORMAT	BIT 0 ... 4 : format BIT 5, 6 : reserved BIT 7 : 0 ... 1	BIT 0...4: parameter value format BIT 7 "0" subindex not included in the frame BIT 7 "1" subindex included in the frame
9	INDEX (LOW)	0 ... 255	byte low of parameter INDEX
10	INDEX (HIGH)	0 ... 255	byte high of parameter INDEX
11	VALUE Byte 0	0 ... 255	first byte of value
12	VALUE Byte 1	0 ... 255	second byte of value
13 ... LENGTH-1	VALUE (continued)	0 ... 255	remaining bytes of value, see FORMAT specification
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0180

10.4.2. Error response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	05H- DGFC386 OK 15H- failure present on the DGFC386	frame informations, DGFC386 destination
6	RESPONSE (LOW)	ERROR CODE (LOW)	byte low of response
7	RESPONSE (HIGH)	ERROR CODE (HIGH)	byte high of response
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0160

11. SPECIAL ACCESS TO DGFC386

SLINK3 Special DGFC386 access COMMAND allow data not specified in SLINK3 protocol to be transferred between Master and DGFC386; refer to DGFC386 user specifications for more informations; following, it is described the general frame format for this COMMAND type.

11.1. Special DGFC386 access command: MASTER->DGFC386

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	04H	frame informations, DGFC386 destination
6	COMMAND (LOW)	CDH	byte low of command
7	COMMAND (HIGH)	ABH	byte high of command
8 ... LENGTH -1	DATA	special DGFC386 access data	refer to DGFC386 user manual
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0190

11.2. Special DGFC386 access command: DGFC386->MASTER

11.2.1. OK response case

BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	05H- DGFC386 OK 15H- failure present on the DGFC386	frame informations, DGFC386 specification
6	RESPONSE (LOW)	00H	byte low of response
7	RESPONSE (HIGH)	00H	byte high of response
8 ... LENGTH -1	DATA	special DGFC386 access data	refer to DGFC386 user manual
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0200

11.2.2. Error response case

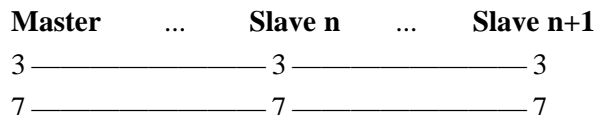
BYTE	NAME	VALUE	COMMENTS
1	START CODE	01H	always the same
2	ADDRESS	0 ... 127	slave address
3	CONTROL	00xxxx00	xxxx is the frame identifier (syncro. purpose)
4	LENGTH	8 ... 255	entire frame length in bytes
5	ORDER/INFOS	05H- DGFC386 OK 15H- failure present on the DGFC386	frame informations, DGFC386 specification
6	RESPONSE (LOW)	ERROR CODE (LOW)	byte low of response
7	RESPONSE (HIGH)	ERROR CODE (HIGH)	byte high of response
8 ... LENGTH -1	DATA	special DGFC386 access data	refer to DGFC386 user manual
LENGTH	CHECKSUM	BYTE 2...(LENGTH-1) XOR	vertical parity

slg0210

12. SERIAL COMMUNICATION REQUIREMENTS

12.1. RS485 connector setup

Here is reported the configuration for RS485 onnector for Master and Slave



12.2. Port setting

The serial communication (RS485) port setting is as follow:

8 data bits, 1 stop bit, even parity, 9600 baud.

12.3. Time restrictions and considerations

12.3.1. Byte to Byte delay specification

Here are reported the delay min/max specification between two subsequent bytes:

Delay between two bytes	Minimum	Maximum
Master -> Slave	0	6 mS
Slave -> Master	0	6 mS

slg0290

12.3.2. Slave answer delay specification

The maximum delay specification between the last byte of the frame sent by the master and the corresponding first byte of the answer by the slave is **1.5 Seconds**.

12.3.3. Replay by the Slave to Next Master order delay specification

The minimum delay between the last byte of the Slave answer and the first byte of the next Master message is **2mS**

12.3.4. Frame transfer duration time calculation

A single frame transfer duration time can be calculated as follow:

$$\text{FRAME_DURATION} = 1.145\text{mS} * n$$

Where n is the number of bytes composing the frame, this considering the delay between two subsequent bytes equal to 0.

The total data transfer between Master and Slave will be:

$$\text{FRAME_DURATION}(\text{MS} \rightarrow \text{SL}) + (\text{Slave answer delay}) + \text{FRAME_DURATION}(\text{SL} \rightarrow \text{MS})$$

13. COMMUNICATION EXAMPLE

Here is reported an example of how the serial communication interface for SLINK3 could be implemented on a Master device ("C" language).

```

/*
*****
Project: SLINK3

File: slink3ex.c

Version: 1.000

Created: 30/06/1996 - 11:55

Author: FER

Description: SLINK3 communication example - MASTER side

*****
*/

```

```

/*
*****
INCLUDE FILES
*/

#include <stdio.h>

/*
*****
LOCAL DEFINITIONS AND MACROS
*/

/* tx/rx frame size */

#define FRAME_SIZE 50 /* frame size for this example */

/* START code definition */
#define SOH 0x01

/*
*****
LOCAL VARIABLES AND CONSTANTS
*/

/* failure signal from the slave */
unsigned char slave_failure;

/*
*****
LOCAL PROTOTYPES
*/

/* send the frame on the serial port routine: its implementation depends
on development environment (i.e. PC , PLC, etc.); here only the interface
definition is given */

unsigned int Send_data_to_serial_port(
    char buffer[], /* address of the buffer to be sent */
    unsigned int length /* number of bytes to be sent */
);

/* receive the frame from the serial port routine: its implementation depends
on development environment (i.e. PC , PLC, etc.); here only the interface
definition is given; see RECEPTION_OK and FAIL for return codes definition
NOTE: the implementation have to consider that the reply length is different
between ok RESPONSE and error RESPONSeE cases*/

```

```

/* serial port reception result codes */
#define RECEPTION_OK 0
#define RECEPTION_FAIL 1 /* parity error, timeouts etc. */

unsigned int Receive_data_from_serial_port(
    char buffer[], /* address of the buffer to be received */
    unsigned int ok_response_length, /* ok RESPONSE replay:

number of bytes to be received */
    unsigned int error_response_length, /* error RESPONSE replay:

number of bytes to be received */
    unsigned int slave_replay_del, /* maximum delay on slave replay start */
    unsigned int bytes_delay, /* maximum delay between two subsequent
                                bytes */
    );

unsigned int Write_typed_int_parameter(
    unsigned int index,
    unsigned char value[]
    );

unsigned int Read_typed_int_parameter(
    unsigned int index,
    unsigned char value[]
    );

unsigned char Calculate_xor( unsigned char buffer[],
    unsigned int start,
    unsigned int end
    );

/*
*****
MODULE BODY
*/

/*
*****

Routine main

Inputs :

Outputs:

```

Called by:

It uses:

Description: main routine example :

- writes a parameter to the slave, displaying the result
- reads a parameter from the slave , displaying the result and the parameter value
- shows the slave state

```
*****
```

```
*/
```

```
void main( void )
{
unsigned int result;
int value;

printf("\nWriting of parameter 8236 with value 1234..");

value = 1234;

result = Write_typed_int_parameter( 8236, (unsigned char *)&value);

printf("\nOperation result : %s", result ? "OK":"ERROR");

if (result ==1)
{
/* displaying of slave state only if result is OK */

printf("\nSlave state: %s",slave_failure ? "OK":"FAIL");
}

printf("\nReading of parameter 8236 ..");

result = Read_typed_int_parameter( 8236, (unsigned char *)&value);

printf("\nOperation result : %s", result ? "OK":"ERROR");

if (result ==1)
{
/* displaying of slave state only if result is OK */
```

```

    printf("\nParameter 8236 value = %d", value);
    printf("\nSlave state: %s",slave_failure ? "OK":"FAIL");
    }

}/* end of main */

/*
*****

Routine Write_typed_int_parameter

Inputs : parameter index, parameter value address

Outputs: returns 1 if transfer completed successfully, otherwise 0
        sets the slave_failure flag according to slave replay

Called by: main

It uses: Send_data_to_serial_port , Receive_data_from_serial_port
        Calculate_xor

Description:
    Writes a typed integer parameter to the slave at ADDRESS 0
    Tests the replay from the slave

*****
*/

unsigned int Write_typed_int_parameter( unsigned int index, unsigned char value[] )
{

/* tx frame buffer */
unsigned char tx_frame[FRAME_SIZE];

/* rx frame buffer */
unsigned char rx_frame[FRAME_SIZE];

unsigned int i;
/* serial port reception result */
unsigned int reception_result;

/* slave response on parameter access */
unsigned int slave_response;

    tx_frame[0] = SOH; /* START code */
    tx_frame[1] = 0x00; /* fixed target ADDRESS : 0 */

```

```

tx_frame[2] = 0x00; /* CONTROL , frame counter fixed at 0 */
tx_frame[3] = 13; /* LENGTH , fixed for Write typed par., int FORMAT */
tx_frame[4] = 0x00; /* ORDER/INFOS */
tx_frame[5] = 0x27; /* COMMAND (LOW), Write typed parameter */
tx_frame[6] = 0x00; /* COMMAND (HIGH), Write typed parameter */
tx_frame[7] = 0x03; /* FORMAT , fixed at int type, no SUBINDEX */
tx_frame[8] = (unsigned char) index; /* INDEX (LOW)*/
tx_frame[9] = (unsigned char) (index << 8); /* INDEX (HIGH)*/
tx_frame[10] = value[0]; /* VALUE byte 0 */
tx_frame[11] = value[1]; /* VALUE byte 1 ; int FORMAT, 2 bytes*/

/* calculates CHECKSUM field */

tx_frame[12] = Calculate_xor(
    &tx_buffer,
    1, /* start from ADDRESS field */
    tx_frame[3]-1 /* stop at LENGTH-1*/
);

/* sends the frame on the serial port */

Send_data_to_serial_port(
    &tx_frame, /* address of the buffer to be sent */
    tx_frame[3] /* number of bytes to be sent */
);

/* receives the replay from the slave */

reception_result = Receive_data_from_serial_port(
    &rx_frame, /* address of the buffer to be received */
    8, /* Write typed parameter replay ok response:
        number of bytes to be received */
    8, /* Write typed parameter replay error response:
        number of bytes to be received */
    SLAVE_ANSWER_TIMEOUT, /* maximum delay on slave replay start */
    MAX_BYTES_DELAY /* maximum delay between two subsequent bytes */
);

/* tests if the frame has been received correctly
(no timeouts, parity errors etc.)*/

if (reception_result == RECEPTION_OK)

    /* reception completed successfully */

    {

```

```

/* calculates the received frame vertical parity and compares it to
the received one*/
if(
  Calculate_checksum(
    &rx_buffer,
    1,          /* starts from ADDRESS field */
    6          /* to LENGTH-1 field*/
  ) != rx_buffer[7]

)

/* CHECKSUM fail */
return (0)

/* reads RESPONSE fields to detect a mistake in parameter access */

slave_response =
(unsigned int) rx_frame[5] +
( (unsigned int) rx_frame[6] << 8);

/* reads the ORDER/INFOS field to detect a slave failure */
slave_failure = rx_frame[4] & 0x10;

/* if parameter access result is not OK returns error code */
if (slave_response != 0x00)
  return (0);

return (1);

}

else

/* reception fail */

return (0);

}/* end of Write_typed_int_parameter */

/*
*****
*****

Routine Read_typed_int_parameter

```

Inputs : parameter index, parameter value address

Outputs: returns 1 if transfer completed successfully, otherwise 0
sets the slave_failure flag according to slave replay

Called by: main

It uses: Send_data_to_serial_port , Receive_data_from_serial_port
Calculate_xor

Description:

Reads a typed integer parameter from the slave at ADDRESS 0
Tests the replay from the slave

*/

```
unsigned int Read_typed_int_parameter( unsigned int index, unsigned char value[] )
```

```
{
```

```
/* tx frame buffer */
```

```
unsigned char tx_frame[FRAME_SIZE];
```

```
/* rx frame buffer */
```

```
unsigned char rx_frame[FRAME_SIZE];
```

```
unsigned int i;
```

```
/* checksum end offset location (depending on RESPONSE field) */
```

```
unsigned int checksum_endoffset;
```

```
/* serial port reception result */
```

```
unsigned int reception_result;
```

```
/* slave response on parameter access */
```

```
unsigned int slave_response;
```

```
tx_frame[0] = SOH; /* START code */
```

```
tx_frame[1] = 0x00; /* fixed target ADDRESS : 0 */
```

```
tx_frame[2] = 0x00; /* CONTROL , frame counter fixed at 0 */
```

```
tx_frame[3] = 11; /* LENGTH , fixed for Read typed parameter */
```

```
tx_frame[4] = 0x00; /* ORDER/INFOS */
```

```
tx_frame[5] = 0x26; /* COMMAND (LOW), Read typed parameter */
```

```
tx_frame[6] = 0x00; /* COMMAND (HIGH), Read typed parameter */
```

```
tx_frame[7] = 0x03; /* FORMAT , fixed at int type, no SUBINDEX */
```

```
tx_frame[8] = (unsigned char) index; /* INDEX (LOW)*/
```

```
tx_frame[9] = (unsigned char) (index << 8); /* INDEX (HIGH)*/
```

```
/* calculates CHECKSUM field */
```

```
tx_frame[10] = Calculate_xor(
    &tx_buffer,
    1,          /* start from ADDRESS field */
    tx_frame[3]-1 /* stop at LENGTH-1 */
);
```

```
/* receives the replay from the slave
```

```
    Rreplay LENGTH dependes on RESPONSE field*/
```

```
reception_result = Receive_data_from_serial_port(
    &rx_frame,          /* address of the buffer to be received */
    13,                /* Read typed parameter replay ok response:
                       number of bytes to be received */
    8,                 /* Read typed parameter replay error response:
                       number of bytes to be received */
    SLAVE_ANSWER_TIMEOUT, /* maximum delay on slave replay start */
    MAX_BYTES_DELAY      /* maximum delay between two subsequent bytes */
);
```

```
/* tests if the frame has been received correctly
   (no timeouts, parity errors etc.)*/
```

```
if (reception_result == RECEPTION_OK)
```

```
    /* reception completed successfully */
```

```
    {
```

```
        /* reads RESPONSE fields to detect a mistake in parameter access */
```

```
        slave_response =
            (unsigned int) rx_frame[5] +
            ( (unsigned int) rx_frame[6] << 8);
```

```
        /* checksum end offset calculation
           (received frame LENGTH depends on RESPONSE field)*/
```

```
        checksum_end_offset = (slave_response)? 6 : 11
```

```
        /* calculates the received frame verical parity and compares it to
           the received one*/
```

```

if(
  Calculate_checksum(
    &rx_buffer,
    1,          /* starts from ADDRESS field */
    checksum_end_offset
  ) != rx_buffer[ checksum_end_offset+1 ]

)
/* CHECKSUM fail */
return (0)

/* reads the ORDER/INFOS field to detect a slave failure */
slave_failure = rx_frame[4] & 0x10;

/* if parameter access result is not OK returns error code */
if (slave_response != 0x00)
  return (0);

/* reads the parameter value from VALUE fields */

value[0] = rx_frame[10]; /* VALUE byte 0 */
value[1] = rx_frame[11]; /* VALUE byte 1 */

return (1);

}
else

/* reception fail */

return (0);

}/* end of Read_typed_int_parameter */

/*
*****
Routine Calculate_xor

Inputs : buffer to calculate xor on, start buffer offset for xor calculation,
        end buffer offset for xor calculation

Outputs: xor

Called by:

```

It uses:

Description: calculates the xor (bitwise) of the given buffer starting from start offset to end offset

```
*****
*/
```

```
unsigned char Calculate_xor( unsigned char buffer[], unsigned int start, unsigned int end)
```

```
{
```

```
unsigned int i;
```

```
unsigned char xor
```

```
for ( i=start; i <= end; i++ )
```

```
{
```

```
  xor ^= buffer[i];
```

```
}
```

```
return xor;
```

```
}/* end of Calculate_xor */
```

```
/*
```

```
*****
```

```
END OF FILE
```

```
*/
```

```
ENDRECORD
```

SLINK PT Manual 02/99
Rev. 0.0 - 22.03.99