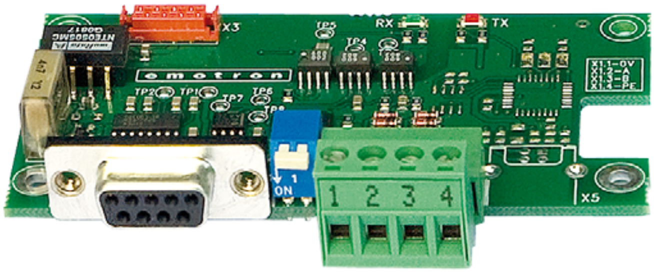
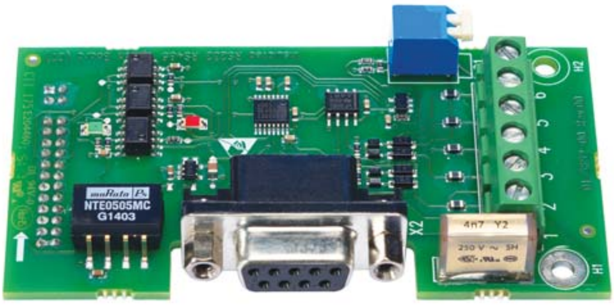
Emotron Isolated RS232/485 2.0 Option

For Emotron VFX/FDU 2.0 AC drive
Emotron VFXR/FDUL
Emotron FlowDrive





Instruction manual
English

emotron
DEDICATED DRIVE
A CG Product

Emotron Isolated RS232/485 2.0 Option

For Emotron VFX/FDU 2.0 AC drive
Emotron VFXR/FDUL
Emotron FlowDrive

Instruction manual – English

Document number: 01-5919-01
Edition: r2
Date of release:15-02-2016

Safety

Instruction manual

Read this instruction manual first!

Since this option is a supplementary part of the AC drive, the user must be acquainted with the original instruction manual of the AC drive. All safety instructions, warnings, etc. as mentioned in this instruction manual are to be known to the user.

Safety instructions

Read the safety instructions in the instruction manual for the AC drive.

Installation

Installation, commissioning, dismounting, making measurements, etc. on the AC drive may only be carried out by personnel who are technically qualified for the task. Installation must also be carried out in accordance with the local standards. Ensure that all necessary safety measures are taken.

WARNING!
Take all necessary safety precautions during installation and commissioning to prevent personal injuries, e.g. by an uncontrolled load.

Opening the AC drive

WARNING!
Always switch off the mains supply before opening the AC drive and wait at least 7 minutes to allow the buffer capacitors to discharge.

Always take adequate precautions before opening the AC drive, even though the connections for the control signals and jumpers are isolated from the mains voltage.

Contents

1. General Information

1.1 Introduction

The isolated RS232/485 option board is an asynchronous serial communication interface for the variable speed drives of the Emotron VFX 2.0 and FDU 2.0 series.

The protocol used for data exchange is based on the Modbus RTU protocol, originally developed by Modicon.

Physical connection can be either RS232 or RS485.

The product acts as a slave with address 1 to 247 in a master-slave configuration. The communication is half-duplex. It has a standard non return to zero (NRZ) format.

Please refer to the instruction manual for the main product for the maximum selectable baud rate.

The character frame format (always 11 bits) has:

- one start bit
- eight data bits
- one or two stop bits (Emotron products use two stop bits)
- even or no parity bit (Emotron products use no parity)

1.2 Users

This instruction manual is intended for:

- installation engineers
- designers
- maintenance engineers
- service engineers

1.3 Safety

Because this option is a supplementary part of the main product, the user must be acquainted with the original instruction manual of the main product. All safety instructions, warnings, etc. as mentioned in these instruction manuals are to be known to the user.

The following indications can appear in this manual. Always read these first and be aware of their content before continuing:

NOTE: Additional information to avoid problems.

CAUTION!
Failure to follow these instructions can result in malfunction or damage to the variable speed drive.

WARNING!
Failure to follow these instructions can result in serious injury to the user in addition to serious damage to the variable speed drive.

1.4 Delivery and unpacking

Check for any visible signs of damage. Inform your supplier immediately of any damage found. Do not install the option board if damage is found.

If the option board is moved from a cold storage room to the room where it is to be installed, condensation can form on it. Allow the option board to become fully acclimatized and wait until any visible condensation has evaporated before installing it in the main product.

2. Connections

2.1 Board layout and connections for Emotron FDU/VFX/VFXR/FDUL/ FlowDrive type IP54, IP20/21 and IP23

This chapter describes the board layout and connections for type IP54, IP20/21 and IP23

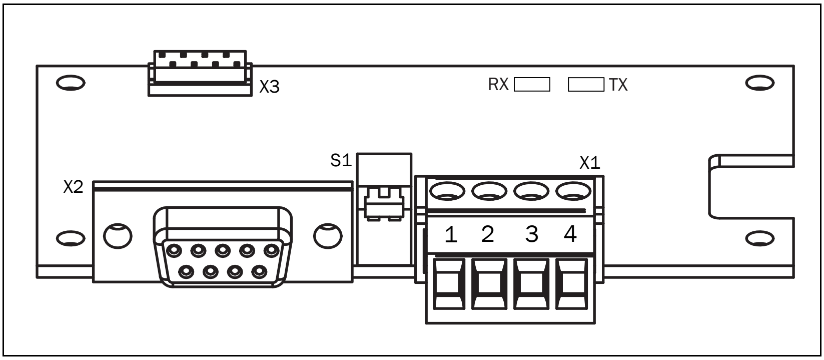


Fig. 1 RS232/485 option board layout for type IP54, IP20/21 and IP23

2.1.1 User connections

Terminal X1

Terminal X1 is used for RS485 communication.

X1	Name	Function
1	Ground	0 V reference
2	A-line	Differential transmit and receive pin.
3	B-line	Differential transmit and receive pin.
4	PE	Protective earth

D-Sub contact, X2

The D-Sub contact, X2, is used for RS232 communication.

X2	Name	Function
2	TX	transmit pin
3	RX	receive pin
5	Ground	0 V reference

Switch S1

Switch	Description
S1	Used to terminate an RS485-network. See section 4.1.2 on page 33 for further information.

LED

These LEDs can be used as simple status indications for the bus system.

LED	Description
RX	Flashes when the node is receiving a message transmitted on the bus.
TX	Flashes when the node is transmitting a response message to the master.

2.2 Board layout and connections for Emotron FDU/VFX/FlowDrive -IP2Y frame sizes A3, B3 and C3

This chapter describes the board layout and connections for type IP2Y frame sizes A3, B3 and C3.

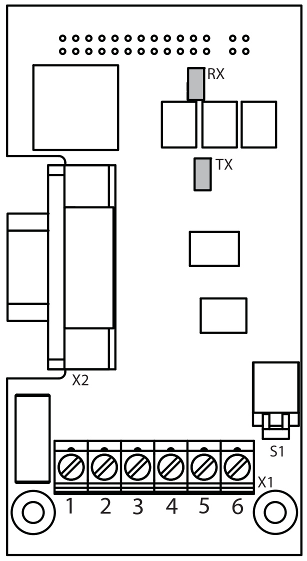


Fig. 2 RS232/485 option board layout for type IP2Y

2.2.1 User connections

Terminal X1

Terminal X1 is used for RS485 communication.

X1	Name	Function
1	Ground	0 V reference
2	A-line	Differential transmit and receive pin.
3	B-line	Differential transmit and receive pin.
4	PE	Protective earth
5	TX	transmit pin
6	RX	receive pin

D-Sub contact, X2

The D-Sub contact, X2, is used for RS232 communication.

X2	Name	Function
2	TX	transmit pin
3	RX	receive pin
5	Ground	0 V reference

Switch S1

Switch	Description
S1	Used to terminate an RS485-network. See section 4.1.2 on page 33 for further information.

LED

These LEDs can be used as simple status indications for the bus system.

LED	Description
RX	Flashes when the node is receiving a message transmitted on the bus.
TX	Flashes when the node is transmitting a response message to the master.

3. Modbus RTU

3.1 General

Devices communicate using a master-slave technique, in which only one device (the master) can initiate transactions (called queries). The other devices (the slaves) respond by supplying the requested data to the master, and by taking the action requested in the query. Typical master devices include host processors and programming panels. Typical slaves include programmable controllers, motor controllers, load monitors, etc., see Fig. 3.

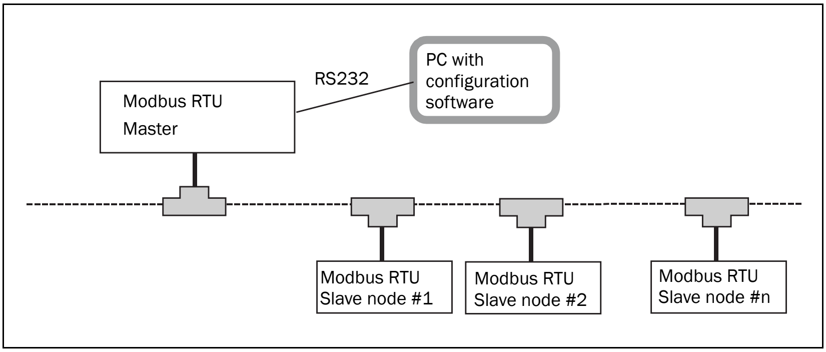


Fig. 3 Network configuration

The master can address individual slaves. Slaves return a message (called a response) to queries that are addressed to them individually, see Fig. 4.

A Modbus RTU telegram consists of a device address, a function code defining the requested action, any data to be sent, and an error-checking field. The response from the slave is constructed in a similar way. It contains fields confirming the action taken, any data to be returned and an error-checking field. If an error occurred in receiving the message, or if the slave is unable to perform the requested action, the slave will construct an error message and send this as its response.

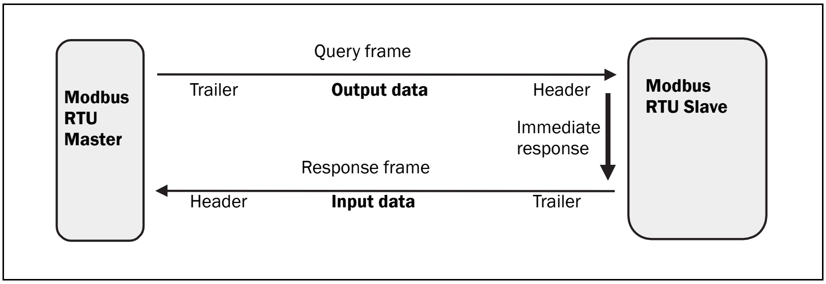


Fig. 4 The Modbus RTU data exchange

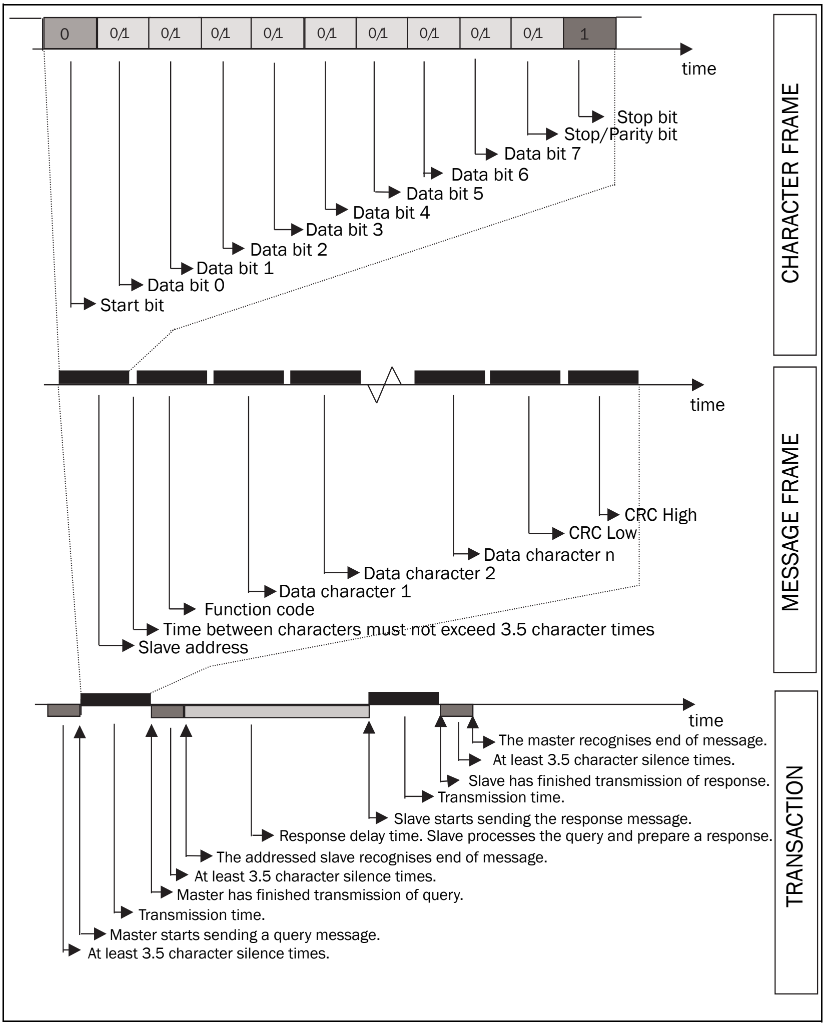
NOTE: All Emotron products use the "no parity" frame with two stop bits, see information below.

If even parity is used, each character (8 bit data) is sent as:

1	Start bit.
8	Data bits, hexadecimal 0-9, A-F, least significant bit sent first.
1	Even parity bit.
1	Stop bit.

If no parity (default for Emotron products) is used, each character (8 bit data) is sent as:

1	Start bit.
8	Data bits, hexadecimal 0-9, A-F, least significant bit sent first.
2	Stop bit.



0
0/1
0/1
0/1
0/1
0/1
0/1
0/1
0/1
0/1
1
time
Stop bit
Stop/Parity bit
Data bit 7
Data bit 6
Data bit 5
Data bit 4
Data bit 3
Data bit 2
Data bit 1
Data bit 0
Start bit
CHARACTER FRAME
time
CRC High
CRC Low
Data character n
Data character 2
Data character 1
Function code
Time between characters must not exceed 3.5 character times
Slave address
MESSAGE FRAME
time
The master recognises end of message.
At least 3.5 character silence times.
Slave has finished transmission of response.
Transmission time.
Slave starts sending the response message.
Response delay time. Slave processes the query and prepare a response.
The addressed slave recognises end of message.
At least 3.5 character silence times.
Master has finished transmission of query.
Transmission time.
Master starts sending a query message.
At least 3.5 character silence times.
TRANSACTION

Fig. 5 Timing diagram for a transaction (query and response messages) (bottom of figure), a message frame (middle of figure) and a character frame (top of figure)

3.2 Framing

Messages start with a silent interval of at least 3.5 character times. This is easily implemented as a multiple of character times at the baud rate used on the network (shown as T1-T2-T3-T4 in the table below). The first field then transmitted is the device address.

The characters permitted to be transmitted for all fields are hexadecimal 0-9, A-F. Network devices monitor the network bus continuously, including during the "silent" intervals. When the first field (the address field) is received, each device decodes it to find out if it is the addressed device.

Following the last transmitted character, a similar interval of at least 3.5 character times marks the end of the message. A new message can begin after this interval.

The entire message frame must be transmitted as a continuous stream. If a silent interval of more than 3.5 character times occurs before completion of the frame, the receiving device flushes the incomplete message and assumes that the next byte will be the address field of a new message.

Similarly, if a new message begins earlier than 3.5 character times following a previous message, the receiving device will consider it a continuation of the previous message. This will set an error, as the value in the final CRC field will not be valid for the combined messages. A typical message frame is shown below:

Header	START	T1-T2-T3-T4
	ADDRESS	8 bits
	FUNCTION	8 bits
Data	DATA	n x 8 bits
Trailer	CRC CHECK	16 bits
	END	T1-T2-T3-T4

3.2.1 Address field

The address field of a message frame contains eight bits. The individual slave devices are assigned addresses in the range of 1 to 247. A master addresses a slave by placing the slave address in the address field of the message. The master itself does not have an address.

When the slave sends its response, it places its own address in the address field of the response to let the master know which slave is responding.

3.2.2 Function field

The function code field of a message frame contains eight bits. Valid codes are in the range of 1 to 6, 15, 16 and 23. See section 3.3, page 17.

When a message is sent from a master to a slave device, the function code field tells the slave what kind of action to perform.

Examples are:

- reading the ON/OFF status of a group of inputs;
- reading the data contents of a group of parameters;
- reading the diagnostic status of the slave;
- writing to designated coils or registers within the slave.

When the slave responds to the master, it uses the function code field to indicate either a normal (error-free) response or that some kind of error occurred (called an exception response). For a normal response, the slave simply echoes the original function code. For an exception response, the slave returns a code that is equivalent to the original function code with its most significant bit set to a logic 1.

In addition to its modification of the function code for an exception response, the slave places an unique code into the data field of the response message. This tells the master what kind of error occurred, or the reason for the exception, see Table 1, page 17.

The master device's application program has the responsibility of handling exception responses. Typical processes are posting subsequent retries of the message, trying to send diagnostic messages to the slave and notifying operators.

Additional information about function codes and exceptions is provided later on in this chapter.

3.2.3 Data field

The data field is constructed using an 8 bit hexadecimal number, in the hexadecimal range of 00 to FF.

The data field of messages sent from a master to slave devices contains additional information which the slave must use to take the action defined by the function code. This can include items such as register addresses, the quantity of items to be handled and the count of actual data bytes in the field.

For example, if the master requests that a slave reads a group of holding registers (function code 03), the data field specifies the starting register and how many registers are to be read. If the master writes to a group of registers in the slave (function code 10 hexadecimal), the data field specifies the starting register, how many registers to write, the count of data bytes to follow in the data field, and the data to be written into the registers.

If no error occurs, the data field of a response from a slave to a master contains the data requested. If an error occurs, the field contains an exception code that the master application can use to determine the next action to be taken.

3.2.4 CRC Error-checking field

The error-checking field contains a 16 bit value implemented as 2 bytes. The error check value is the result of a Cyclical Redundancy Check (CRC) calculation performed on the message contents.

The CRC field is appended to the message as the last field in the message. The low-order byte of the field is appended first, followed by the high-order byte. The CRC high-order byte is the last byte to be sent in the message.

For additional information on CRC calculation, see chapter 5. on page 37.

3.3 Functions

Emotron supports the following Modbus function codes.:

NOTE: The modbus parameter numbers may be different for the true Emotron product compared to what is used in the examples below. Please compare with the latest revision of the parameters in the manual for the main product.

Table 1 Function codes

Function name	Function code
Read Coil Status	1 (01h)
Read Input Status	2 (02h)
Read Holding Registers	3 (03h)
Read Input Registers	4 (04h)
Force Single Coil	5 (05h)
Force Single Register	6 (06h)
Force Multiple Coils	15 (0Fh)
Force Multiple Registers	16 (10h)
Force/Read Multiple Holding Registers	23 (17h)

3.3.1 How to convert modbus numbers into starting addresses

For coils (modbus numbers 1-99) the starting address is calculated by subtracting 1 from the modbus number in the table, e.g. the Run status coil has modbus number 2 but its starting address is defined as 2-1=1.

Input status has modbus numbers 10001-19999 and corresponding starting addresses are calculated by subtracting the 10001 from the modbus number.

For input registers (modbus numbers 30001-39999) the starting addresses are calculated by subtracting 30001 from the modbus number in the table, e.g. modbus number 30011 has a starting address of 30011-30001=10.

For holding registers (modbus numbers 40001 and up) the starting addresses are calculated by subtracting 40001from the Modbus number in the table, e.g. modbus number 41137 has the starting address 1136.

3.3.2 Read Coil Status

Reads the status of changeable digital parameters.

Example

Requesting the RUN status. Result is that the AC drive is stopped.

RUN status:	Modbus no = 2 (02h), start address 1 (01h)
Data:	Stopped = 0
1 byte of data:	Byte count = 01

Request message

Field name	Hex value
Slave address	01
Function	01
Start address HI	00
Start address LO	01
Number of Coils HI	00
Number of Coils LO	01
CRC LO	AC
CRC HI	0A

Response message

Field name	Hex value
Slave address	01
Function	01
Byte count	01
Data	00
CRC LO	51
CRC HI	88

3.3.3 Read Input Status

Modbus numbers 10001-19999.

Example

In the example below a random modbus number has been used (may not exist in the main product).

Request message

Field name	Hex value
Slave address	01
Function	02
Start address HI	00
Start address LO	02
Number of Inputs HI	00
Number of Inputs LO	01
CRC LO	18
CRC HI	0A

Response message

Field name	Hex value
Slave address	01
Function	02
Byte count	01
Data	00
CRC LO	A1
CRC HI	88

3.3.4 Read Holding Registers

Example

Reading the currently selected language, modbus number 43011 with starting address 0BC2h. The result is that the language is set to 1 (Swedish).

Request message

Field name	Hex value
Slave address	01
Function	03
Start address HI	0B
Start address LO	C2
Number of Registers HI	00
Number of Registers LO	01
CRC LO	27
CRC HI	D2

Response message

Field name	Hex value
Slave address	01
Function	03
Byte count	02
Reg no. 0, (0h) data HI	00
Reg no. 0, (0h) data LO	01
CRC LO	79
CRC HI	84

3.3.5 Read Input Registers

Example

Reading modbus register 31002, the output speed, with corresponding starting address 03E9h. The result is that the motor is stopped (zero speed).

If you wish to read register 31003, output torque, at the same time, just increase the number of registers from 01 to 02 in the request message. The response message will then have a byte count of 04 and also contain the data information (2 bytes) of register 31003. However, a new CRC needs to be calculated in this case.

Request message

Field name	Hex value
Slave address	01
Function	04
Start address HI	03
Start address LO	E9
Number of Registers HI	00
Number of Registers LO	01
CRC LO	E0
CRC HI	7A

Response message

Field name	Hex value
Slave address	01
Function	04
Byte count	02
Reg no. 10 (0Ah) data HI	00
Reg no. 10 (0Ah) data LO	00
CRC LO	B9
CRC HI	30

3.3.6 Force Single Coil

Sets the status of one changeable digital parameter.

Example

In the example below a random modbus number has been used (may not exist in the main product).

Request message

Field name	Hex value
Slave address	01
Function	05
Start address HI	00
Start address LO	01
Data HI	FF
Data LO	00
CRC LO	DD
CRC HI	FA

Response message

Field name	Hex value
Slave address	01
Function	05
Start address HI	00
Start address LO	01
Data HI	FF
Data LO	00
CRC LO	DD
CRC HI	FA

3.3.7 Force Single Register

Example

Set parameter with modbus number 43020, Level/Edge, to Edge = 1. The corresponding starting address is 0BCBh.

Request message

Field name	Hex value
Slave address	01
Function	06
Start address HI	0B
Start address LO	CB
Data HI	00
Data LO	01
CRC LO	3B
CRC HI	D0

Response message

Field name	Hex value
Slave address	01
Function	06
Start address HI	0B
Start address LO	CB
Data HI	00
Data LO	01
CRC LO	3B
CRC HI	D0

3.3.8 Force Multiple Coil

Sets the status of multiple changeable digital parameters.

Example

In the example below a random modbus number has been used (may not exist in the main product).

Request message

Field name	Hex value
Slave address	01
Function	0F
Start address HI	00
Start address LO	00
Number of Coils HI	00
Number of Coils LO	02
Byte count	01
Coil no. 0-1 status (0000 0011B)	03
CRC LO	9E
CRC HI	96

Response message

Field name	Hex value
Slave address	01
Function	0F
Start address HI	00
Start address LO	00
Number of Coils HI	00
Number of Coils LO	02
CRC LO	D4
CRC HI	0A

3.3.9 Force Multiple Register

Example

In the example below a random modbus number has been used (may not exist in the main product).

Request message

Field name	Hex value
Slave address	01
Function	10
Start address HI	00
Start address LO	11
Number of Registers HI	00
Number of Registers LO	02
Byte count	04
Data HI reg 17 (11h)	00
Data LO reg 17 (11h)	FA
Data HI reg 18 (12h)	00
Data LO reg 18 (12h)	37
CRC LO	52
CRC HI	88

Response message

Field name	Hex value
Slave address	01
Function	10
Start address HI	00
Start address LO	11
Number of Registers HI	00
Number of Registers LO	02
CRC LO	11
CRC HI	CD

3.3.10 Force/Read Multiple Register

Sets and reads the contents of multiple changeable parameters in the same message.

Example

Set modbus parameter 43064, thermal protection, to PTC=1 and also set next parameter 43065, Motor class, to Class F=5. The corresponding starting address for modbus parameter 43064 will be 0BF7h.

At the same time we will read the contents of modbus numbers 430035 and 430036 which are fieldbus settings for process data size and R/W settings. The result will be 4 = 4 bytes process data and 0 = R/W allowed. The corresponding starting address for modbus number 43035 will be 0BDAh.

Request message

Field name	Hex value
Slave address	01
Function	17
Start read address HI	0B
Start read address LO	DA
Number of read registers HI	00
Number of read registers LO	02
Start write address HI	0B
Start write address LO	F7
Number of write registers HI	00
Number of write registers LO	02
Byte count	04
Data HI register 21 (15h)	00
Data LO register 21 (15h)	01
Data HI register 22 (16h)	00
Data LO register 22 (16h)	05
CRC LO	AB
CRC HI	3C

Response message

Field name	Hex value
Slave address	01
Function	17
Byte count	04
Reg no. 3, (3h) data HI	00
Reg no. 3, (3h) data LO	04
Reg no. 4, (4h) data HI	00
Reg no. 4, (4h) data LO	00
CRC LO	B8
CRC HI	E6

3.4 Errors, exception codes

Two kinds of errors are possible:

- Transmission errors.
- Operation errors.

3.4.1 Transmission errors

Transmission errors are:

- Frame error (stop bit error).
- Parity error (if parity is used).
- CRC error.
- No message at all (time-out).

These errors may be caused by electrical interference from machinery or damage to the communication channel (cables, contact, I/O ports, etc.). This unit will not act on or answer the master when a transmission error occurs. (The result will be the same if a non-existing slave is addressed). The master will eventually cause a time-out condition.

3.4.2 Operation errors

If no transmission error is detected in the master query, the message is examined. If an illegal function code, data address or data value is detected, the message is not acted upon but an answer with an exception code is sent back to the master. This unit can also send back an exception code when a set (force) function message is received during busy periods of operation.

Bit 8 (most significant bit) in the function code byte is set to a “1” in the exception response message. An example is an illegal data address when reading an input register.

Exception response message

Field name	Hex value
Slave address	01
Function	84
Exception code	02
CRC LO	C2
CRC HI	C1

Table 2 Exception codes

Exc. code	Name	Description
01	Illegal function	This unit doesn’t support the function code.
02	Illegal data address	The data address is not within its boundaries.
03	Illegal data value	The data value is not within its boundaries.
06	Busy	The unit is unable to perform the request at this time. Try again later.

4. Interface and Menu System

4.1 RS485 Multipoint network

The RS485 port (see Fig. 6) is used for multi point communication. A host computer (master) can address a maximum of 247 slave stations (nodes). See Fig. 6.

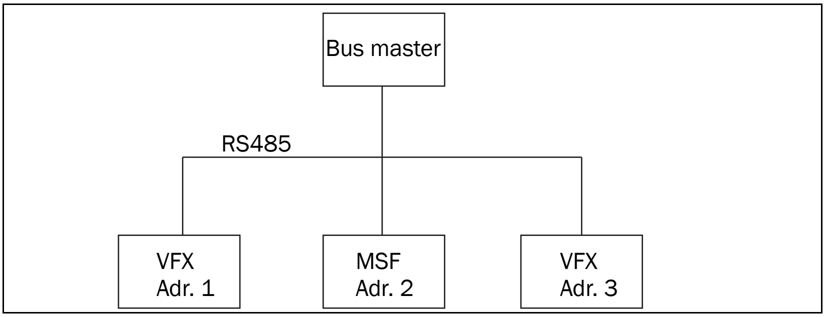


Fig. 6 RS 485 multipoint network

4.1.1 RS485 wiring

For type IP54, IP20/21, IP23 AC drives the connector is a 4-pole male connector. The wiring should be as in Fig. 7.
For IP2Y AC drives the connector is a wire terminal. The wiring should be as in as in Fig. 8.
A common problem when installing new networks is that the A and B wires have been crossed or incorrectly terminated. The A and B wires must never cross and only the end nodes on the network should be terminated.

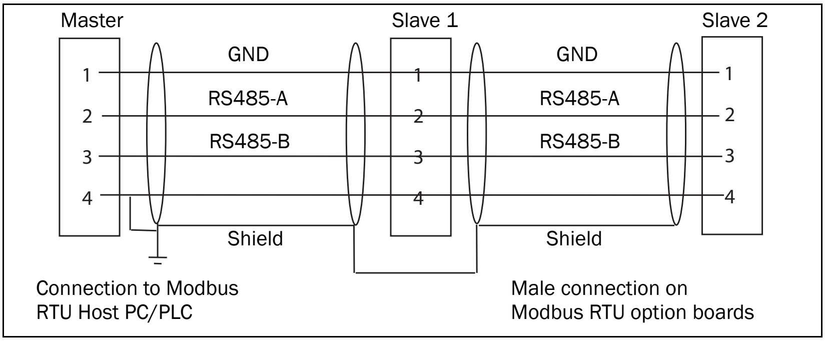

Master
Slave 1
Slave 2
GND
RS485-A
RS485-B
GND
RS485-A
RS485-B
1
2
3
4
1
2
3
4
1
2
3
4
Shield
Shield
Connection to Modbus RTU Host PC/PLC
Male connection on Modbus RTU option boards

Fig. 7 RS485 wiring for type IP54, IP20/21, IP23 AC drives.

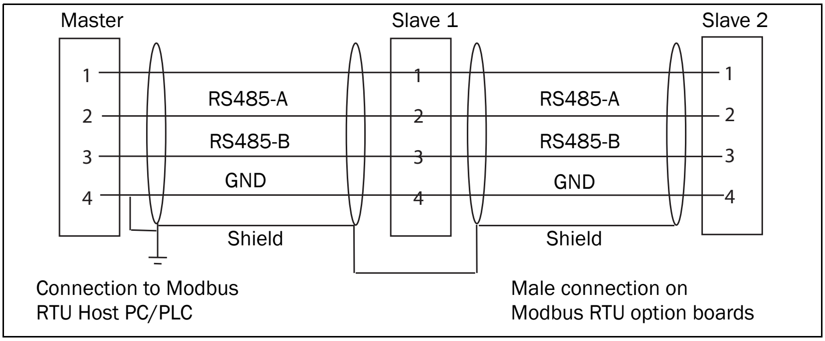

Master
Slave 1
Slave 2
RS485-A
RS485-B
GND
RS485-A
RS485-B
GND
1
2
3
4
1
2
3
4
1
2
3
4
Shield
Shield
Connection to Modbus RTU Host PC/PLC
Male connection on Modbus RTU option boards

Fig. 8 RS485 wiring for IP2Y AC drives.

4.1.2 RS485 termination

The RS485 network must always be terminated to avoid transmission problems. The termination must take place at both ends of the network. In Fig. 7 this means that the termination must take place both at the Master and at Slave 2.

Switch S1 (see Fig. 1) sets the termination ON or OFF as indicated in the Fig. 9 and Fig. 10.

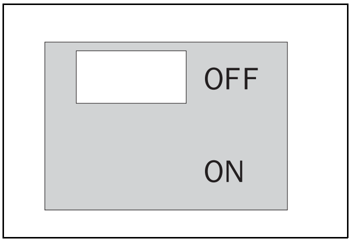


Fig. 9 Termination is OFF

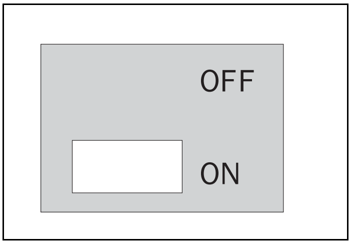


Fig. 10 Termination is ON

NOTE: Physical connection can be either RS232 or RS485, not both at the same time.

4.2 RS232 point to point communication

The RS232 port is used for point to point communication. See Fig. 11. The Emotron product works as a slave node.

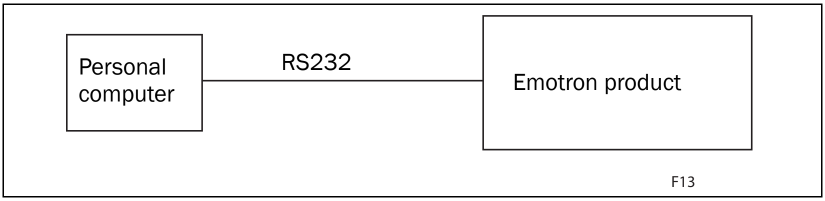


Fig. 11 RS232 point to point network

4.2.1 RS232 wiring

The RS232 port consists of a D-Sub 9-pole female connector. The wiring should be as in Fig. 12.

NOTE: Use an 1:1 cable WITHOUT a pin 2-3 crossing.

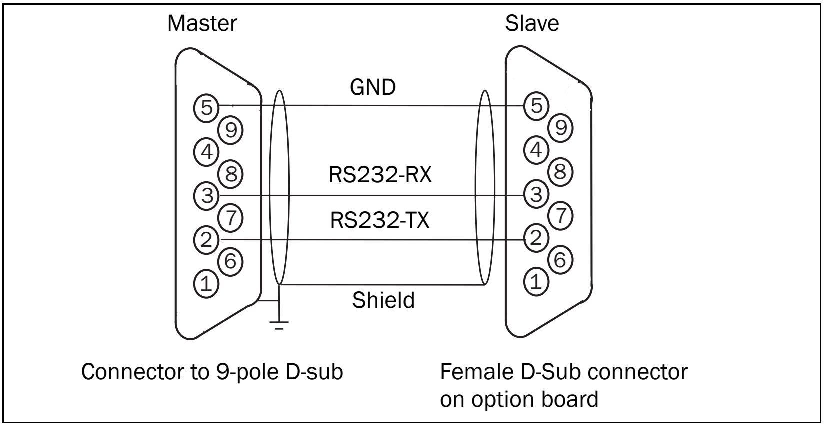

Master
Slave
GND
RS232-RX
RS232-TX
Shield
5
9
4
8
3
7
2
6
1
Connector to 9-pole D-sub
Female D-Sub connector on option board

Fig. 12 RS232 wiring

4.3 Menu description

All menus are described in the manual for the main product.

Table 3 Menus for setting the serial communication

Menu	Function	Default	Range/Selection
261	Communication type	RS232/485	RS232/485, Fieldbus
262	RS232/485		
2621	Baud rate	9600	2400, 4800, 9600, 19200, 38400
2622	Address	1	1-247
264	Interrupt	Warning	Trip, Warning, Off

5. CRC Generation

The CRC is started by first pre-loading a 16-bit register to all 1's. Then a process of applying successive eight-bit bytes of the message to the current contents of the register begins. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits, and the parity bit, do not apply to the CRC.

During generation of the CRC, each eight-bit character is exclusive OR-ed with the register contents. The result is shifted in the direction of the least significant bit (lsb), with a zero filled into the most significant bit (msb) position. The lsb is extracted and examined. If the lsb was a 1, the register is then exclusive OR-ed with a preset, fixed value. If the lsb was a 0, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next eight-bit character is exclusive OR-ed with the register's current value, and the process repeats for eight more shifts as described above. The final contents of the register, after all the characters of the message have been applied, are the CRC value.

Generation in steps:

- **Step 1** Load a 16-bit register with 0xFFFF (all 1's). Call this the CRC register.
- **Step 2** Exclusive OR the first eight-bit byte of the message with the low order byte of the 16-bit CRC register, putting the result in the CRC register.
- **Step 3** Shift the CRC register one bit to the right (toward the lsb), zero-filling the msb. Extract and examine the lsb.
- **Step 4** If the lsb is 0, repeat Step 3 (another shift). If the lsb is 1, Exclusive OR the CRC register with the polynomial value 0xA001 (1010 0000 0000 0001).
- **Step 5** Repeat Steps 3 and 4 until eight shifts have been performed. When this is done, a complete eight-bit byte will have been processed.
- **Step 6** Repeat Steps 2 to 5 for the next eight-bit byte of the message. Continue doing this until all bytes have been processed.
- Result: The final contents of the CRC register are the CRC value.

- **Step** 7 When the CRC is placed into the message, its upper and lower bytes must be swapped as described below.

Placing the CRC into the Message

- When the 16-bit CRC (two eight-bit bytes) is transmitted in the message, the low order byte will be transmitted first, followed by the high order byte e.g. if the CRC value is 0x1241:

Message	
CRC LO	41
CRC HI	12

Example of CRC Generation Function

An example of a C language function performing CRC generation is shown on this page.

The function takes two arguments:

- Unsigned char *puchMsg; — a pointer to the message buffer containing binary data to be used for generating the CRC.
- Unsigned integer usDataLen; — the number of bytes in the message buffer.

The function returns the CRC as an unsigned integer type.

- Unsigned integer CRC16 (unsigned integer usDataLen, unsigned char *puchMsg)

```
#define CRC_POLYNOMIAL    0xA001
   unsigned int crc_reg;
   unsigned char i,k;
   crc_reg = 0xFFFF;
   for (i=0 ; i<usDataLen ; i++)
   {
      crc_reg ^= *puchMsg++;
      for (k=0 ; k<8 ; k++)
      {
         if (crc_reg & 0x0001)
         {
            crc_reg >>= 1;
            crc_reg ^= CRC_POLYNOMIAL;
         }
         else
            crc_reg >>= 1;
      }
   }
   return crc_reg;
```

Fig. 13 CRC example

6. Parameter sets and Trip log lists

Table 4 Trip log list

Trip log list	Modbus number
1	31101 to 31150
2	31151 to 31200
3	31201 to 31250
4	31251 to 31300
5	31301 to 31350
6	31351 to 31400
7	31401 to 31450
8	31451 to 31500
8	31501 to 31550

Table 5 Parameter set list

Parameter sets	Modbus number
A	43001 to 43529
B	44001 to 44529
C	45001 to 45529
D	46001 to 46529

7. Control information

Basic control of the main product over serial communication can easily be performed by using a few modbus parameters.

Table 6

Modbus number	Function	Description
42901	reset	Reset 0->1 (flank trigged)
42902	run/stop	1=run, 0=stop
42903	run right	1=run to the right
42904	run left	1=run to the left
42905	comm.ref	0-4000h <=> 0-100%
42907	comm. set	0=A,1=B,2=C,3=D

Start and stop the AC drive over the serial bus

To be able to start/stop the AC drive menu 215 Run/Stp Ctrl must be set to "Com".

Then decide the rotation direction by writing a 1 to either register 42903 or 42904.

NOTE: Do not set both registers high at the same time since this will cause the AC drive to stop.

Use register 42902 to start or stop the inverter.

Reset alarm over the serial bus

To be able to reset alarms over serial communication menu 216 Reset Ctrl must be set to "Com" or "Com+Keyb". This allows you to reset alarms by changing register 42901 from 0 to 1. Note that this register is flank triggered which means that a change from 0 to 1 has to take place to perform a reset.

Set the reference value over the serial bus

To set the reference value over the serial bus, set menu 214 Ref Control to "Com".

It is then possible to write to register 42905 to set the reference value. The hexadecimal value 0h is equal to min ref and a value of 4000h to max ref. The maximum reference value is depending on selected drive mode (see the manual for the main product).

Change parameter set over serial communication.

Set menu 241 Select set to "Com" and select the parameter set by writing 0-3 (as described in table above) to register 42907.

8. Installation

8.1 Installation in type IP54, IP20/21 and IP23

This chapter describes how to mount option boards in the AC drive.

On these AC drives up to three different option boards and one communication board can be mounted.

8.1.1 Polarisation of flat cables

The flat cable is marked with a colour on one side and has a pin on the micromatch male contact. This side must be matched to the female micromatch contact on the control board and option board respectively, where a small hole in the board is located.

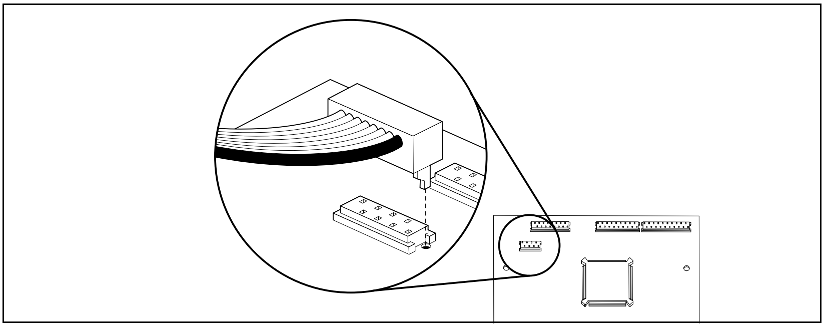


Fig. 14 Polarisation of flat cables.

CAUTION!
Incorrect connection might cause damage to both the option and to the control board/external equipment.

8.1.2 Mechanical mounting

Make sure that the AC drive has been switched off for at least seven minutes to ensure that the capacitor bank is discharged before continuing with installation! Also make sure that no external equipment connected to the drive's interface is powered on.

NOTE: Correct installation is essential for fulfilling the EMC requirements and for proper operation of the module.

8.1.2.1 Mounting the first option board

The communication option board is always mounted on the slot marked C on the mounting plate. In this example we assume that no other option board is installed.

Delivered with the option board kit

- Option board and four screws, M3 x 6.
- 8-pole flat cable for connection between the communication board and the control board.

Mounting

1. Connect the 8-pole flat cable to the X4 connector on the control board with the cable downwards as in Fig. 15.

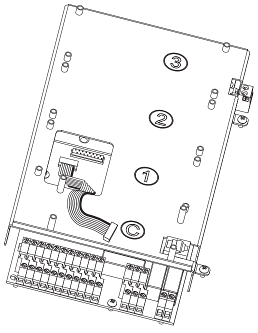


Fig. 15 Flat cable connected to the control board.

NOTE: For polarisation of the flat cable, see section 8.1.1 on page 46.

2. Put the option board on the spacers on the slot marked C. Fasten the board using the four screws.

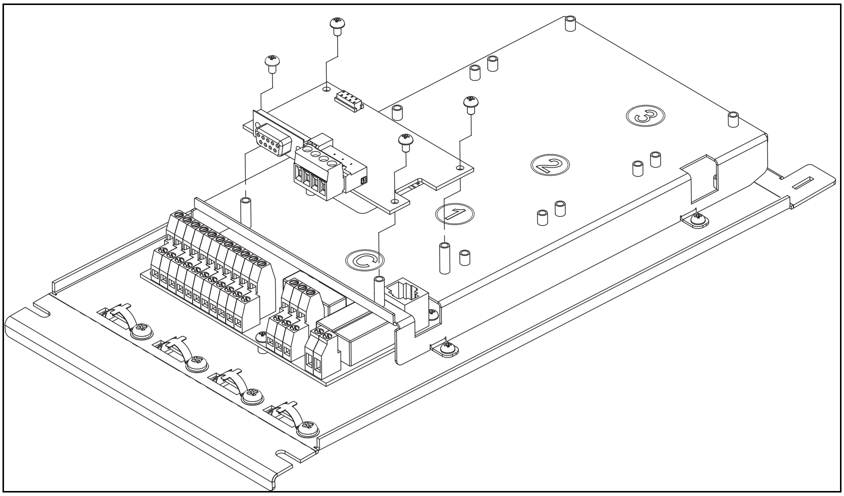


Fig. 16 Mounting the FieldbusRS232/485 option board

3. Connect the other end of the 8-pole flat cable to the X3 connector on the option board. Make sure that the polarisation is correct as in section 8.1.1 on page 46.

NOTE: Connect the micro match male contact to the option in the same manner as on the control board, i.e. the pin on the micro match contact must be fitted into the hole in the PCB.

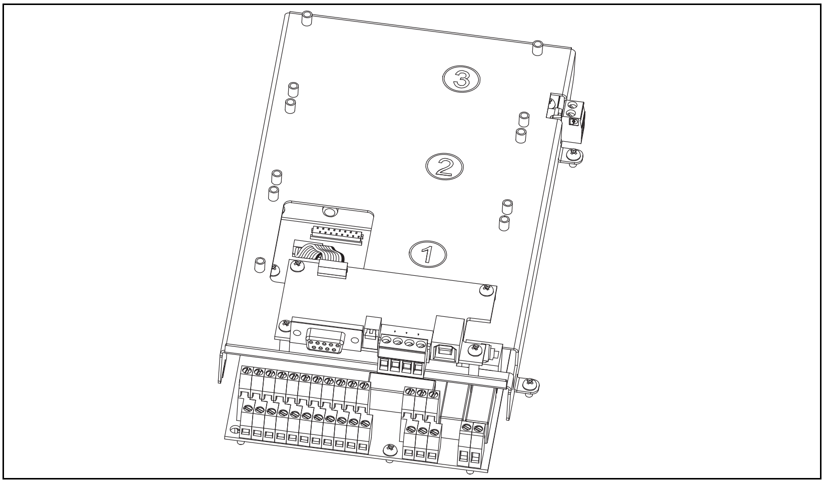


Fig. 17   Flat cable connected to the option board

8.2 Installation in type IP2Y frame sizes A3, B3 and C3

This chapter describes how to mount the option board in the AC drive.

Table 7 Emotron FDU/VFX-IP2Y frame size explanation

Model	Frame size
VFX/FDU48-2P5-2Y	A3
VFX/FDU48-3P4-2Y	
VFX/FDU48-4P1-2Y	
VFX/FDU48-5P6-2Y	
VFX/FDU48-7P2-2Y	
VFX/FDU48-9P5-2Y	
VFX/FDU48-012-2Y	
VFX/FDU48-016-2Y	B3
VFX/FDU48-023-2Y	
VFX/FDU48-032-2Y	C3
VFX/FDU48-038-2Y	

8.2.1 The option kit includes

- option board.
- Two screws, M3 x 6.
- Insulating sheet.

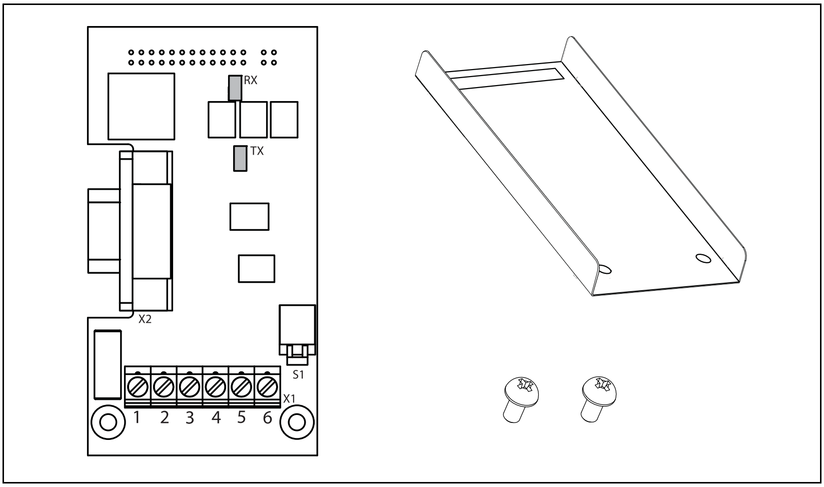


Fig. 18 The IP2Y option kit includes.

CAUTION!
Incorrect connection might cause damage to both the option and to the control board/external equipment.

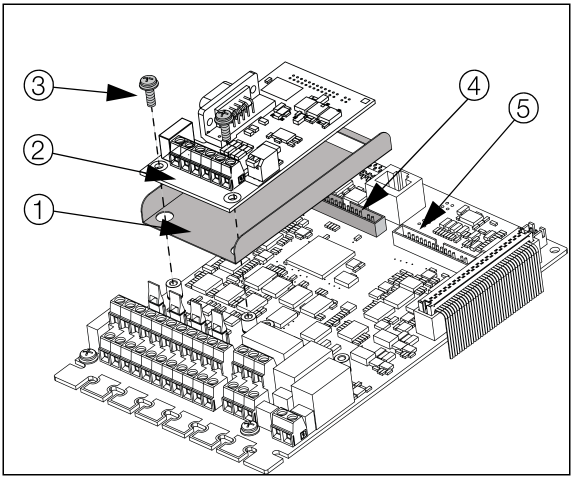
8.2.2 Mounting the option board

Make sure that the AC drive has been switched off for at least ten minutes to ensure that the capacitor bank is discharged before continuing with installation! Also make sure that no external equipment connected to the drive's interface is powered on.

NOTE: Correct installation is essential for fulfilling the EMC requirements and for proper operation of the module.

It is possible to mount two option boards on the control board connectors X7A and X7B.
On frame size A3, if D-sub connector is used, the option board RS232/485-2Y always needs to be mounted on connector X7B. If using RS232,communication wiring must be done via wire terminal X3 Fig. 2, page 8 (D-sub not accessible)
On frame sizes B3 and C3 it does not matter if you mount the option board on place X7A or X7B you are free to choose.
If you intend to mount a second option board, please use connector X7A for the RS232/485 option board.

NOTE: On Frame size A3, the option board RS232/485-2Y always needs to be mounted on connector X7B. Otherwise there is not enough room for the D-Sub connector.



1. Insulation sheet
2. Option board
3. Screws
4. Connector X7A
5. Connector X7B

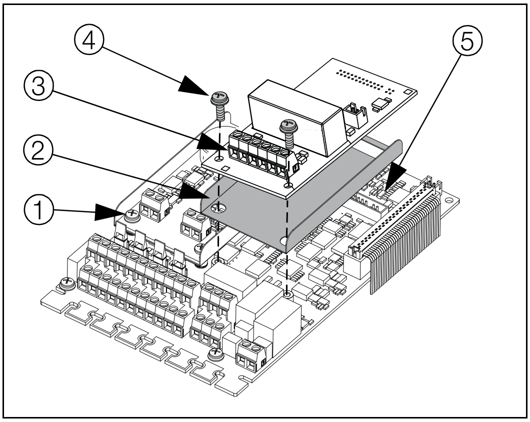
Fig. 19 How to mount the option board on connector X7A.

1. Place the insulating sheet over the short spacers and make sure the slot fits around the X7A/B connector on the control board. Make sure the flaps are bent upwards.
2. Put the option board into position by pressing the connector on the option board into connector X7A/B on the control board. Make sure it rests on the spacers.
3. Secure the option card with the two screws.

8.2.2.1 Mount another option board

A second option board is mounted in the same way as the first, see Fig. 20 where the second board in this case is mounted to connector X7B

.



1. Option board 1
2. Insulation sheet
3. Option board 2
4. Screws
5. Connector X7B

Fig. 20 The IP2Y option kit includes 4 screws, Option board and three different 8-pole flat cables.

CG Drives & Automation Sweden AB
Mörsaregatan 12
Box 222 25
SE-250 24 Helsingborg
Sweden
T +46 42 16 99 00
F +46 42 16 99 49
www.cgglobal.com / www.emotron.com

CG Drives & Automation, 01-5919-01r2, 15-02-2016