

4.5 Expansion Module MAC00-FC2/FC4



4.5.1 Introduction to this section

Section 4.3 in the MAC050-141 & MAC400-800 user manual deals with JVLs expansion modules MAC00-FC2/FC4, which are used too build in a MAC motor on a CANopen® network.

This sections covers:

- General introduction, a section with general information about CANopen®, from section 4.3.1 to section 4.3.6
- Setting up the Baud-rate, node-id and termination of the CAN bus. Covers also the wiring of the CAN bus. From section 4.3.7 to section 4.3.15
- How to use CanOpenExplorer and Mac-Talk debug window. From section 4.3.16 to section 4.3.20.
- Survey over Communication specific objects and manufacturer specific objects in the DS301 standard. Communication objects are the general information about the settings in the module, where the Manufacturer specific object are the settings of input/output and the motor parameters. This section also covers the settings of the transmit and receive PDOs in the module. From section 4.3.21 to section 4.3.37.
- Survey over objects which are used in connection with the DSP-402 standard. From section 4.3.38 to section 4.3.45.
- Cables for the MAC00-FC4 section 4.3.46
- Section with more detail explanations to the CANopen® theory, particularly DS-301. From section 4.3.47 to section 4.3.53.

4.5 Expansion Module MAC00-FC2/FC4

4.5.2 CANopen® Introduction

The MAC00-FC2 and FC4 expansion modules are CANopen® slaves. With these modules, all of the registers in the MAC motor can be accessed over a CANopen® network. The modules implement an object dictionary that follows the CiA® DS-301 standard. The modules contain a number of statically mapped PDOs that can be used to access the most common registers. The MAC00-FC2 and FC4 also support the DSP-402 standard from CiA®.

Expansion modules MAC00-FC2 and FC4 can be mounted on the standard MAC motors MAC50, MAC95, MAC140, MAC141, MAC400 and MAC800.

Both modules offer the same functionality but with the following hardware differences:

Type	Protection class	Connectors		
		I/O and interface	Power supply	Bus interface
MAC00-FC2	IP67/IP65*	Cable glands (Mini crimp connectors internally)	Cable glands (Screw terminals internally)	Cable glands x 2 (Screw terminals internally)
MAC00-FC4	IP67/IP65*	M12	M12	M12 (x2)

Note*: IP65 on MAC400-800

The MAC00-FC2 module can be delivered with cable in selected lengths. Cables with M12 connectors can also be supplied for the MAC00-FC4 module.

The MAC00-FC2/FC4 expansion modules is designed to be used on a CANbus, CANopen® DS-301 and CANopen® DSP-402, do not use the module together with CAN-Kingdom or DeviceNet.

4.5.3 CiA® membership

CiA® (CAN in Automation) is a non-profit society, the object of the society is to promote CAN (Controller-Area-Network) image and to provide a path for future developments of the CAN protocol. CiA® specifications cover physical layer definitions as well as application layer and device profile descriptions.

In order to receive the CANopen® standard, is it necessary to obtain a membership of the society. The fee for the membership is depending on how many employees you company has. A membership runs from January 1st until December 31st every year. Your membership is renewed automatically unless you cancel it in writing by the end of a calendar year. Companies applying for membership after July 1st pay just 50 % of the membership for that year.

On www.can-cia.org/cia/application.html can you download a application file in PDF format and fill it in.

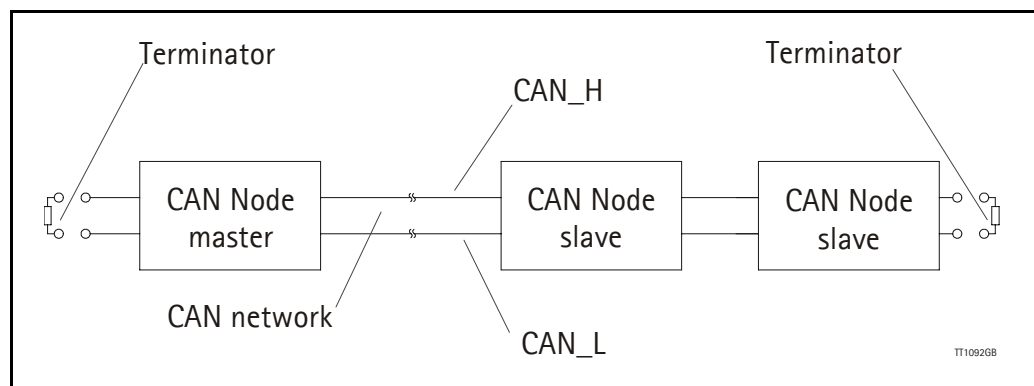
When you have received you license from CiA®, you have to be aware that the standards will be sent on a CD. All of the CiA specifications can be ordered on this web-page:

<http://www.can-cia.org/index.php?id=6>

4.5 Expansion Module MAC00-FC2/FC4

4.5.4 CANopen® network

The CAN bus, is a serial bus with multi-master capabilities where different products from different manufacturers can communicate with each other. This could be devices as PLCs, motors, sensors and actuators. Message types have higher priority and are sent first, for time critical applications. New devices can easily be integrated on a existing bus, without the need to reconfigure the entire network. The devices are connected through a 2 wire bus cable, and data is transmitted serially.



4.5.5 CANopen®, general information

CANopen® is a CAN-based higher level protocol. The purpose of CANopen® is to give an under stable and unique behaviour on the CAN network. The CAN network is the hardware level of the system, and CANopen® is the software level. CANopen® is based on the communication profile described in CiA® DS-301, and it specifies all of the basic communication mechanisms.

CiA® DS-301 contains message types on the lowest software level. The DSP-402 CANopen® standard defines the device profile and the functional behaviour for servo drive controllers, frequency inverters and stepper motor. The DSP-402 is a higher software level, and it use the DS-301 communication, but is making the device independent of the manufacturer. If the devices using only the DSP- 402 it is possible that some general data can be lost.

The CAN bus with real-time capabilities work in accordance with the ISO 11898 standard. The major performance features and characteristic of the CAN protocol are described below:

Message-oriented protocol:

The CAN protocol does not exchange data by addressing the recipient of the message, but rather mark each transmitted message with a message identifier. All nodes in the network check the identifier when they receive a message to see whether it is relevant for them, messages can there for, be accepted by none, one, several or all participants.

Prioritisation of messages:

As the identifier in a message also determines its priority for accessing the bus, it is possible to specify a correspondingly rapid bus access for messages according to their importance. Especially important messages can thus gain access to the bus without a prolonged wait-time, regardless of the loading on the bus at that moment.

This characteristic mean that especially important messages are transmitted with priority even in exceptional situations, thereby ensuring proper functioning of a system even during phases of restricted transmission capacity.

4.5 Expansion Module MAC00-FC2/FC4

Multi-Master capability:

Bus access rights are not issued by a mean-level control unit (bus master) per network. Each participant can rather start to send a message with equal rights as soon as the bus has become free. If several participants access the bus at the same time, an arbitration process allocates each participant the bus access right in line with the priority of the message they want to send at that particular moment. Each participant can therefore communicate directly with every other participant. As the transmission of a message can be initiated by the message source itself, then in the case of event-controlled transmission of messages, the bus is only occupied when a new message is on-hand.

No-loss bus arbitration:

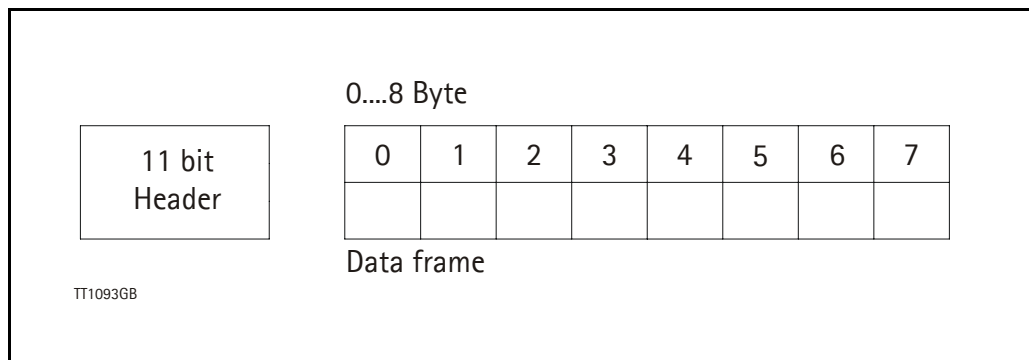
As the bus is accessed at random under the CAN protocol, it is possible that several participants want to occupy the bus at the same time. In other random bus access routines, this causes the destruction of the suppressed messages. In order to solve such a bus access conflict, a repeated occupation of the bus is required using an appropriate triggering strategy. The CAN protocol therefore deploys a routine to ensure that the message with the highest priority at any given time is sent without any destruction of message contents.

Short block length:

The maximum data length of a CAN message is limited to 8 bytes. This data length is usually sufficient to transmit the information occurring in the lowest field area in a CAN message.

4.5.6 Header

A CAN message transmits the communications object and a variety of management and control information. The management and control information bits are used to ensure error free data transmission, and are automatically removed from the received message and inserted before a message is sent. A simplified CANopen® message could be as the figure below:



The two bit fields “Header” and “Data” form the simplified CANopen® message. The 11-bit Header are also designated as the identifier or as the COB-ID (Communication Object identifier).

JVL uses 11-bit format type CAN A, but not 29-bit format type CAN B.

The COB-ID carries out two task for the controller communications object.

- Bus arbitration: Specification of transmission priorities.
- Identification of communications objects.

The COB-ID comprising two sections:

- Function code, 4 bit in size (0...15)
- Node address (Node ID), 7 bit in size (0...127). See section 4.3.12.

4.5 Expansion Module MAC00-FC2/FC4

The function code classifies the communications objects, and controls the transmission priorities. Objects with a small function code are transmitted with high priority. For example, in the case of a simultaneous bus access an object with the function code “1” is sent before an object with the function code “3”.

Node address:

Every device is configured before network operation with a unique 7-bit long node address between 1 and 127. The device address “0” is reserved for broadcast transmissions, in which messages are sent simultaneously to all devices.

PDO, SDO, EMCY, NMT and heartbeat are using the header frame for communication on the CANopen® bus.

4.5.7 Connecting MAC00-FC2/FC4 to the CAN bus

Before you connect the MAC00-FC2/FC4 to the CAN bus the Baud-rate, the Node-ID and the termination must be selected.

On the serial bus it is possible to have a transmission speed (Baud-rate) of max. 1000 Kbit/s and a min. of 10 Kbit/s. The Baud-rate depends on the cable length, and the wires cross-section, the table below has some recommendations for networks with less than 64 nodes. Recommended bus cable cross-sections are according to CiA®:

Bus Distance (m)	Cross-section (mm ²)	Terminator (ohm)	Baud-rate (Kbit/s)
25	0.25-0.34	120	1000
100	0.34-0.6	150-300	500
250	0.34-0.6	150-300	250
500	0.5-0.6	150-300	125
500	0.5-0.6	150-300	100
1000	0.75-0.8	150-300	50

The bus wires may be routed in parallel, twisted and/or shielded, depending on EMC requirements. The layout of the wiring should be as close as possible to a single line structure, in order to minimize reflections. The cable stubs for connection of the bus node shall be as short as possible, especially at high bit rates. The cable shielding in the house shall have a large contact area. For a drop cable a wire cross-section of 0.25 to 0.34 mm² would be an appropriate choice in many cases. In section 4.3.46 of this chapter there is an overview showing various JVL standard cables. All the JVL cables are twisted and shielded.

For bus lengths greater than 1 km, a bridge or repeater device is recommended. Galvanic isolation between the bus nodes is optional. The MAC00-FC2 and FC4 modules have the galvanic isolation integrated to obtain the best possible immunity against noise and differences in the voltage potential between the nodes.

4.5 Expansion Module MAC00-FC2/FC4

4.5.8 Necessary accessories to MAC-FC2/FC4:

On our web page www.jvl.dk you can, under the downloads menu find the EDS file for the MAC00-FC2/FC4 module, in the menu Field bus Interface Specifications Files. EDS means Electronic Data Sheet. This file contains the information about the MAC00-FC2/FC4 settings, that is required to configure the setup and program in the master.

The MAC00-FC2/FC4 is a slave module on the CAN-bus, the master can be for example a PLC or a PC. If you are using a PLC as master, then make sure that it is provided with a CANopen® communications module, and that the correct programming tools are available. For getting support to the PLC master, it is more rewarding to use the PLC vendor.

If you are using a PC as master JVL have some tools that can help you when you are installing and using the MAC00-FC2/FC4.

The latest firmware for the MAC00-FC2/FC4 module is in the menu downloads/firmware. In the menu for programs you can find the program CanOpen Explorer, this is a free-ware program.

The CanOpen Explorer program can be used to load the EDS file, and operate with the motor. The CanOpenExplorer program shall use a special dongle for communication with the PC, see section 4.3.17 for further information about the dongle. The PC has to be provided with a CANopen® communications module. In section 4.3.46 there is a survey of cables JVL can supply, for the CAN-bus.

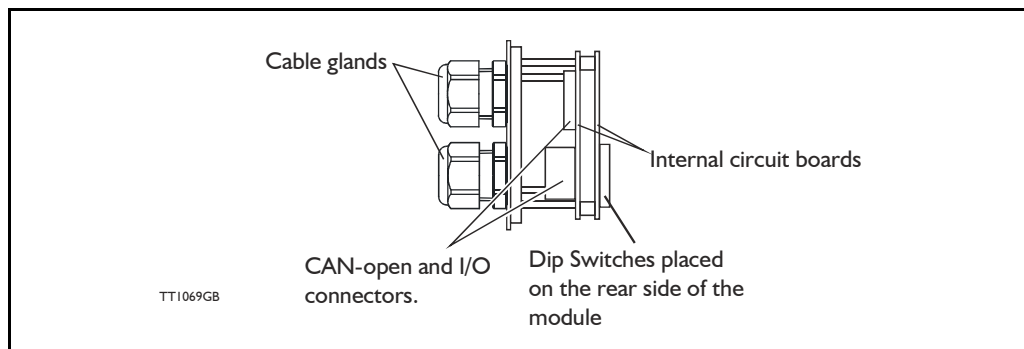
The MacTalk program can be used to monitor various operations and make the initial set up on the motor see section 1.1 for setting up the MAC motor. In the menu for programs you can find the program MacTalk, but be aware that this is not a free-ware program. Please contact your JVL representative for further information.

4.5.9 EDS (Electronic data Sheet)

In order to give the user of CANopen® more support, are the device description available in a standardised way, and it gives the opportunity to create standardised tools for configuration of CANopen® devices, designing networks with CANopen® devices and managing project information on different platforms. The EDS file are ASCII-coded.

4.5.10 Preparing the hardware

To make the selection of the Baud-rate, Node-ID and Line termination on the MAC00-FC2/FC4 module is it necessary to dismantle the module from the motor, and select it via the two Dip switches on the rear side of the module, notice that MAC00-FC4 include one more Dip switch, see section 4.3.15:



4.5 Expansion Module MAC00-FC2/FC4

4.5.11 Baud-rate:

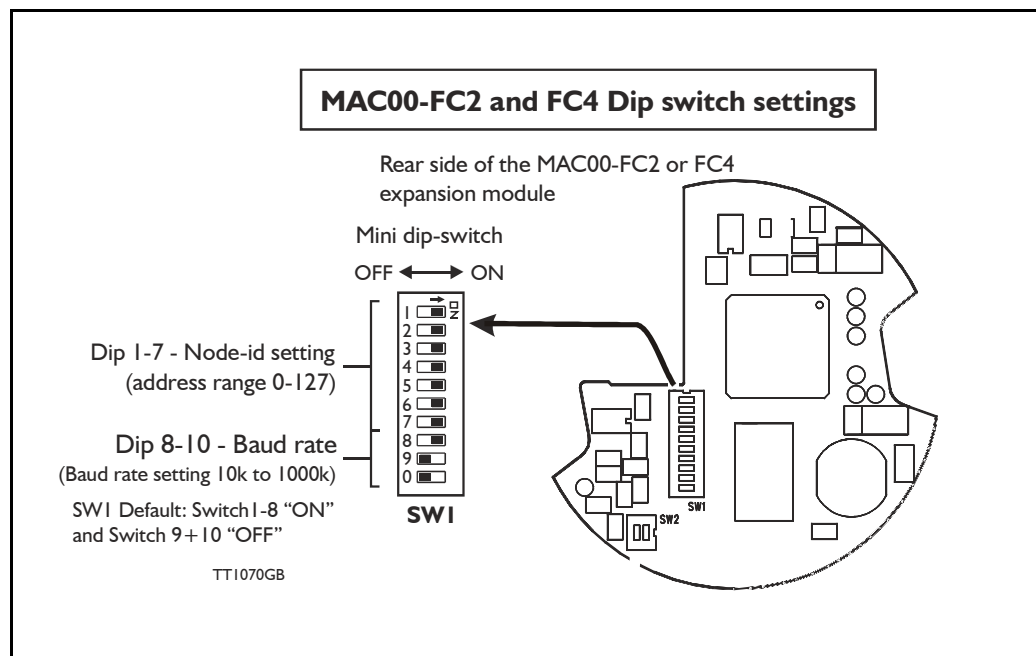
The Baud-rate can be set according to the following table, and is selected on the Dip switch SW1 dip 8-10, as shown on the figure below:

Baud-rate	Dip Switch no. (SW1)		
	10	9	8
1000 kbit	OFF	OFF	OFF
500 kbit (factory default)	OFF	OFF	ON
250 kbit	OFF	ON	OFF
125 kbit	OFF	ON	ON
100 kbit	ON	OFF	OFF
50 kbit	ON	OFF	ON
20 kbit	ON	ON	OFF
10 kbit	ON	ON	ON

The factory default settings sets the module to have a Baud-rate of 500 kbit.

The Baud-rate is the external communication speed. Please notice that internal execution time can be the main limitation meaning that data will be received at the selected Baud-rate but not necessarily executed at the same time.

The Baud-rate setting can only be done in the hardware, it is not possible to set this by using the MacTalk software.



4.5 Expansion Module MAC00-FC2/FC4

4.5.12 Node-ID:

The node-ID can be selected on the Dip switch SW1 Dip 1-7. The address can be set according to the following table:

If the node-id is set to 127, the node address will be set to the same as the motor address (can be defined in MacTalk), which is the factory setting for the modules.

Node-id	Dip Switch no. (SW1)							Node-id	Dip Switch no. (SW1)						
	7	6	5	4	3	2	1		7	6	5	4	3	2	1
0	Reserved (illegal setting)							31	OFF	OFF	ON	ON	ON	ON	ON
1	OFF	OFF	OFF	OFF	OFF	OFF	ON	32	OFF	ON	OFF	OFF	OFF	OFF	OFF
2	OFF	OFF	OFF	OFF	OFF	ON	OFF	33	OFF	ON	OFF	OFF	OFF	OFF	ON
3	OFF	OFF	OFF	OFF	OFF	ON	ON	34	OFF	ON	OFF	OFF	OFF	ON	OFF
4	OFF	OFF	OFF	OFF	ON	OFF	OFF	35	OFF	ON	OFF	OFF	OFF	ON	ON
5	OFF	OFF	OFF	OFF	ON	OFF	ON	36	OFF	ON	OFF	OFF	ON	OFF	OFF
6	OFF	OFF	OFF	OFF	ON	ON	OFF	37	OFF	ON	OFF	OFF	ON	OFF	ON
7	OFF	OFF	OFF	OFF	ON	ON	ON	38	OFF	ON	OFF	OFF	ON	ON	OFF
8	OFF	OFF	OFF	ON	OFF	OFF	OFF	39	OFF	ON	OFF	OFF	ON	ON	ON
9	OFF	OFF	OFF	ON	OFF	OFF	ON	40	OFF	ON	OFF	ON	OFF	OFF	OFF
10	OFF	OFF	OFF	ON	OFF	ON	OFF	41	OFF	ON	OFF	ON	OFF	OFF	ON
11	OFF	OFF	OFF	ON	OFF	ON	ON	42	OFF	ON	OFF	ON	OFF	ON	OFF
12	OFF	OFF	OFF	ON	ON	OFF	OFF	43	OFF	ON	OFF	ON	OFF	ON	ON
13	OFF	OFF	OFF	ON	ON	OFF	ON	44	OFF	ON	OFF	ON	ON	OFF	OFF
14	OFF	OFF	OFF	ON	ON	ON	OFF	45	OFF	ON	OFF	ON	ON	OFF	ON
15	OFF	OFF	OFF	ON	ON	ON	ON	46	OFF	ON	OFF	ON	ON	ON	OFF
16	OFF	OFF	ON	OFF	OFF	OFF	OFF	47	OFF	ON	OFF	ON	ON	ON	ON
17	OFF	OFF	ON	OFF	OFF	OFF	ON	48	OFF	ON	ON	OFF	OFF	OFF	OFF
18	OFF	OFF	ON	OFF	OFF	ON	OFF	49	OFF	ON	ON	OFF	OFF	OFF	ON
19	OFF	OFF	ON	OFF	OFF	ON	ON	50	OFF	ON	ON	OFF	OFF	ON	OFF
20	OFF	OFF	ON	OFF	ON	OFF	OFF	51	OFF	ON	ON	OFF	OFF	ON	ON
21	OFF	OFF	ON	OFF	ON	OFF	ON	52	OFF	ON	ON	OFF	ON	OFF	OFF
22	OFF	OFF	ON	OFF	ON	ON	OFF	53	OFF	ON	ON	OFF	ON	OFF	ON
23	OFF	OFF	ON	OFF	ON	ON	ON	54	OFF	ON	ON	OFF	ON	ON	OFF
24	OFF	OFF	ON	ON	OFF	OFF	OFF	55	OFF	ON	ON	OFF	ON	ON	ON
25	OFF	OFF	ON	ON	OFF	OFF	ON	56	OFF	ON	ON	ON	OFF	OFF	OFF
26	OFF	OFF	ON	ON	OFF	ON	OFF	57	OFF	ON	ON	ON	OFF	OFF	ON
27	OFF	OFF	ON	ON	OFF	ON	ON	58	OFF	ON	ON	ON	OFF	ON	OFF
28	OFF	OFF	ON	ON	ON	OFF	OFF	59	OFF	ON	ON	ON	OFF	ON	ON
29	OFF	OFF	ON	ON	ON	OFF	ON	60	OFF	ON	ON	ON	ON	OFF	OFF
30	OFF	OFF	ON	ON	ON	ON	OFF	61	OFF	ON	ON	ON	ON	OFF	ON

Table continued on next page

4.5 Expansion Module MAC00-FC2/FC4

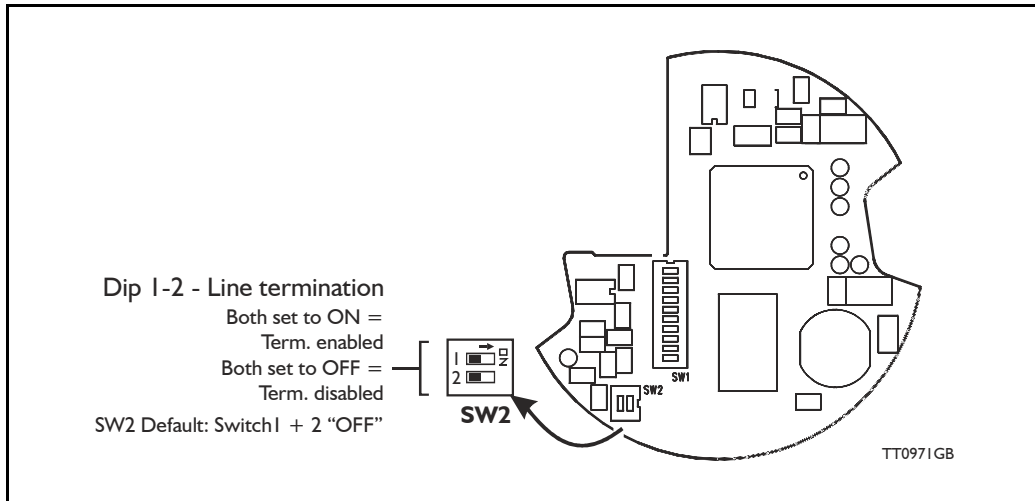
Address table continued from previous page

Node-id	Dip Switch no. (SW1)							Node-id	Dip Switch no. (SW1)						
	7	6	5	4	3	2	1		7	6	5	4	3	2	1
62	OFF	ON	ON	ON	ON	ON	OFF	95	ON	OFF	ON	ON	ON	ON	ON
63	OFF	ON	ON	ON	ON	ON	ON	96	ON	ON	OFF	OFF	OFF	OFF	OFF
64	ON	OFF	OFF	OFF	OFF	OFF	OFF	97	ON	ON	OFF	OFF	OFF	OFF	ON
65	ON	OFF	OFF	OFF	OFF	OFF	ON	98	ON	ON	OFF	OFF	OFF	ON	OFF
66	ON	OFF	OFF	OFF	OFF	ON	OFF	99	ON	ON	OFF	OFF	OFF	ON	ON
67	ON	OFF	OFF	OFF	OFF	ON	ON	100	ON	ON	OFF	OFF	ON	OFF	OFF
68	ON	OFF	OFF	OFF	ON	OFF	OFF	101	ON	ON	OFF	OFF	ON	OFF	ON
69	ON	OFF	OFF	OFF	ON	OFF	ON	102	ON	ON	OFF	OFF	ON	ON	OFF
70	ON	OFF	OFF	OFF	ON	ON	OFF	103	ON	ON	OFF	OFF	ON	ON	ON
71	ON	OFF	OFF	OFF	ON	ON	ON	104	ON	ON	OFF	ON	OFF	OFF	OFF
72	ON	OFF	OFF	ON	OFF	OFF	OFF	105	ON	ON	OFF	ON	OFF	OFF	ON
73	ON	OFF	OFF	ON	OFF	OFF	ON	106	ON	ON	OFF	ON	OFF	ON	OFF
74	ON	OFF	OFF	ON	OFF	ON	OFF	107	ON	ON	OFF	ON	OFF	ON	ON
75	ON	OFF	OFF	ON	OFF	ON	ON	108	ON	ON	OFF	ON	ON	OFF	OFF
76	ON	OFF	OFF	ON	ON	OFF	OFF	109	ON	ON	OFF	ON	ON	OFF	ON
77	ON	OFF	OFF	ON	ON	OFF	ON	110	ON	ON	OFF	ON	ON	ON	OFF
78	ON	OFF	OFF	ON	ON	ON	OFF	111	ON	ON	OFF	ON	ON	ON	ON
79	ON	OFF	OFF	ON	ON	ON	ON	112	ON	ON	ON	OFF	OFF	OFF	OFF
80	ON	OFF	ON	OFF	OFF	OFF	OFF	113	ON	ON	ON	OFF	OFF	OFF	ON
81	ON	OFF	ON	OFF	OFF	OFF	ON	114	ON	ON	ON	OFF	OFF	ON	OFF
82	ON	OFF	ON	OFF	OFF	ON	OFF	115	ON	ON	ON	OFF	OFF	ON	ON
83	ON	OFF	ON	OFF	OFF	ON	ON	116	ON	ON	ON	OFF	ON	OFF	OFF
84	ON	OFF	ON	OFF	ON	OFF	OFF	117	ON	ON	ON	OFF	ON	OFF	ON
85	ON	OFF	ON	OFF	ON	OFF	ON	118	ON	ON	ON	OFF	ON	ON	OFF
86	ON	OFF	ON	OFF	ON	ON	OFF	119	ON	ON	ON	OFF	ON	ON	ON
87	ON	OFF	ON	OFF	ON	ON	ON	120	ON	ON	ON	ON	OFF	OFF	OFF
88	ON	OFF	ON	ON	OFF	OFF	OFF	121	ON	ON	ON	ON	OFF	OFF	ON
89	ON	OFF	ON	ON	OFF	OFF	ON	122	ON	ON	ON	ON	OFF	ON	OFF
90	ON	OFF	ON	ON	OFF	ON	OFF	123	ON	ON	ON	ON	OFF	ON	ON
91	ON	OFF	ON	ON	OFF	ON	ON	124	ON	ON	ON	ON	ON	OFF	OFF
92	ON	OFF	ON	ON	ON	OFF	OFF	125	ON	ON	ON	ON	ON	OFF	ON
93	ON	OFF	ON	ON	ON	OFF	ON	126	ON	ON	ON	ON	ON	ON	OFF
94	ON	OFF	ON	ON	ON	ON	OFF	127	Node id will be the same as for the motor						

4.5 Expansion Module MAC00-FC2/FC4

4.5.13 Bus termination.

If the MAC00-FC2/FC4 is the last device on the CAN bus the module have a build-in terminator of 120 ohm. The dip switch (SW2) is used to enable termination. When both switches are on, the termination is enabled. In order to guarantee perfect operation of the CAN bus, bus terminating resistors must be provide at both ends of the bus cable. The figure below shows the termination selection:



The factory default settings are OFF on both dip switches, an the line termination are off.

CAN bus connectors:

The MAC00-FC2/FC4 are not using 9-pin D-sub connectors and none of the cables JVL supplies are provided with 9-pin D-sub, but the PIN configuration is also shown in table below.

Signal	Description	MAC00-FC2	MAC00-FC4	D-sub
-	Reserved			Pin 1
CAN_L	CAN_L bus line (Low)	B+	Pin 5	Pin 2
CAN_GND	CAN Ground	GND	Pin 3	Pin 3
-	Reserved			Pin 4
(CAN_SHLD)	Optional CAN Shield	GND	Pin 1	Pin 5
(GND)	Optional CAN Ground			Pin 6
CAN_H	CAN_H bus line (High)	A-	Pin 4	Pin 7
-	Reserved (error line)			Pin 8
CAN_V+	Optional CAN ext. + supply			Pin 9

On the next page are there drawings off the 9-pin D-sub and the 5-pin style connector.

4.5 Expansion Module MAC00-FC2/FC4

4.5.14 CanOpenExplorer program

The CanOpenExplorer is a program that was developed for internal use only, especially in the production, but the program have features that is nice to have, and make it more easily to start up the MAC motor when this is supply with a MAC00-FC2/FC4 modul. The program can write and send SDOs, PDOs, SYNC and heartbeat message, and finally it can reads EDS files.

4.5.15 An overall method for communication test

Depending on which type of master and software solution there are available, shall these parts be available:

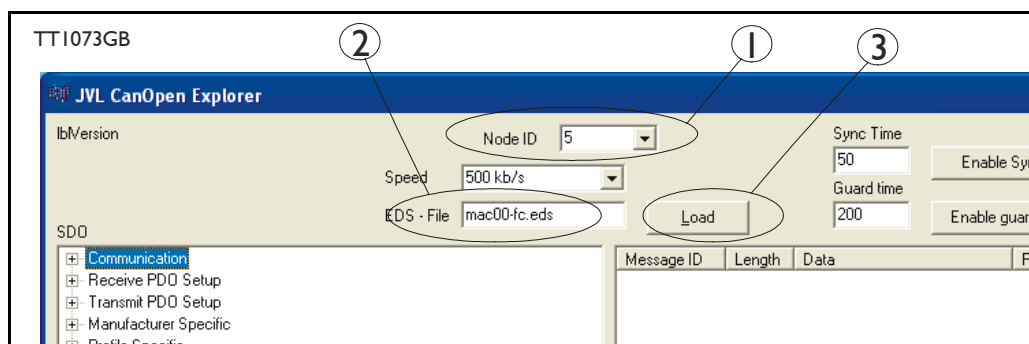
PLC: PLC with a CANopen® module and software that can communicate with this module. The CANopen® module shall be connected to a CAN bus, as shown in section 4.3.14 and section 4.3.15. To set up the master, download the EDS file from the JVL web page, see section 4.3.8. This file contains all register set-up data for in the MAC00-FC2/FC4. The node-ID, the Baud-rate, and the termination resistor, has to be selected on the module, see from section 4.3.11 to section 4.3.13. And the power supply has to be connected to the motor as shown in section 3.2.5.

PC: PC with a CAN adaptor and software that can communicate with this module, or if the CanOpen Explorer software is used the PCAN-USB Dongle from Peak-system that is connected to a USB port on the PC. Peak systems web page are www.peak-system.com here are a list of distributors. If MacTalk is used, the PC and the MAC00-FC2/FC4 are connected via the RS232 interface on the MAC00-FC2/FC4module. To set up the master, download the EDS file from the JVL web-page, see section 4.3.8. This file contains all register set-up in the MAC00-FC2/FC4. The node-ID, the Baud-rate, and the termination resistor, has to be selected on the module, see from section 4.3.11 to section 4.3.13, and the power supply has to be connected to the motor as shown in section 3.2.5.

If CanOpenExplorer is used, see the following method to test the motor communication:

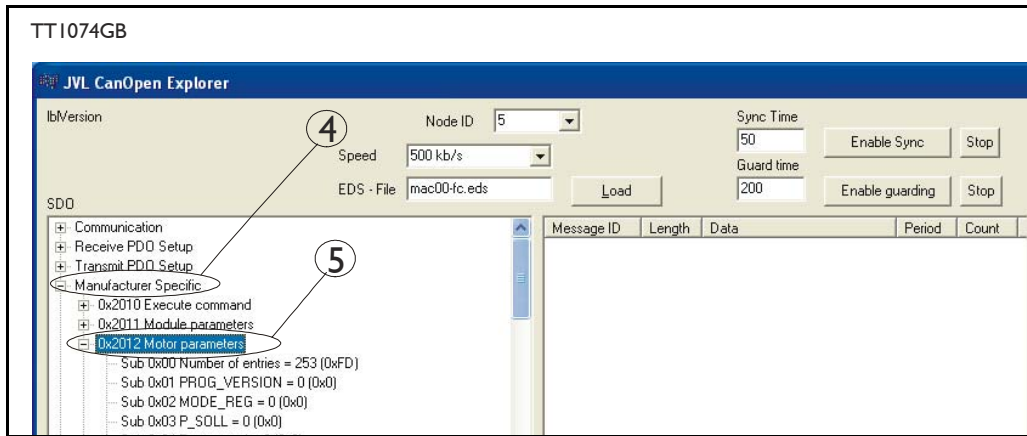
- Load CanOpenExplorer.
- Connect the motor to the USB port via the Dongle.
- Connect power supply, see section 3.2.5.
- Run the CanOpenExplorer program on the PC.

- 1: Select the correct node ID, in the slave.
- 2: Select the EDS file, for all the MAC motors it is MAC00-fc.eds.
- 3: Load the EDS file by pressing load.

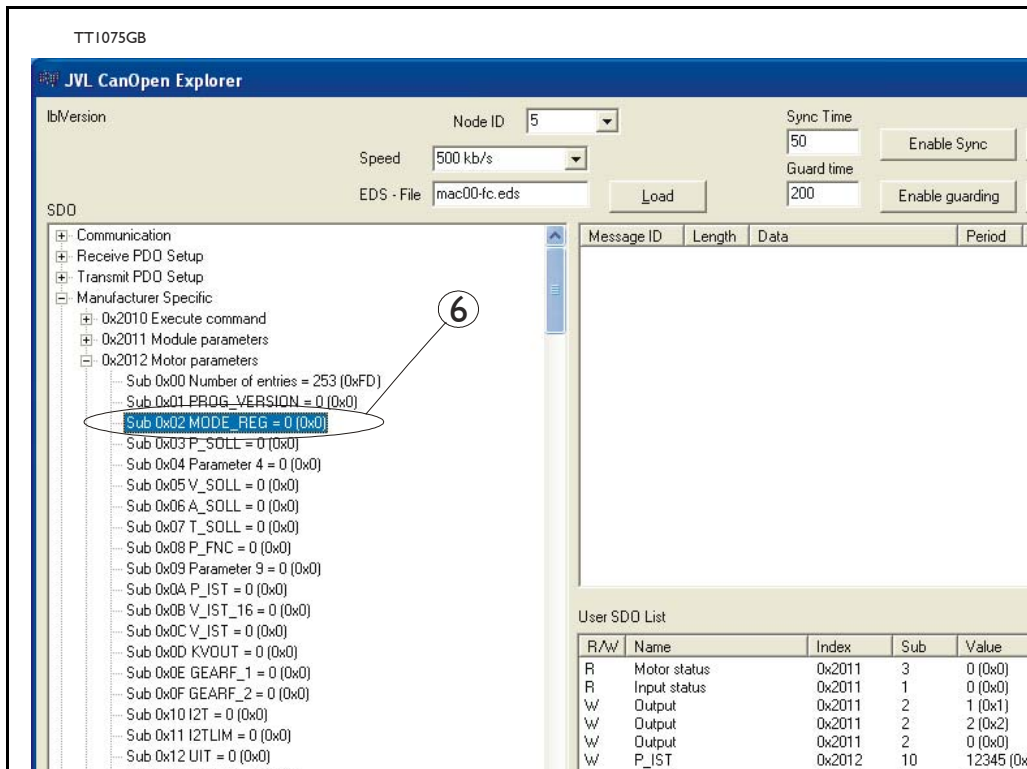


4.5 Expansion Module MAC00-FC2/FC4

- 4: Select here on the + the manufacturer specific register.
- 5: Select thereafter the object 0x2012. Object 0x2012 contains the motor parameters.

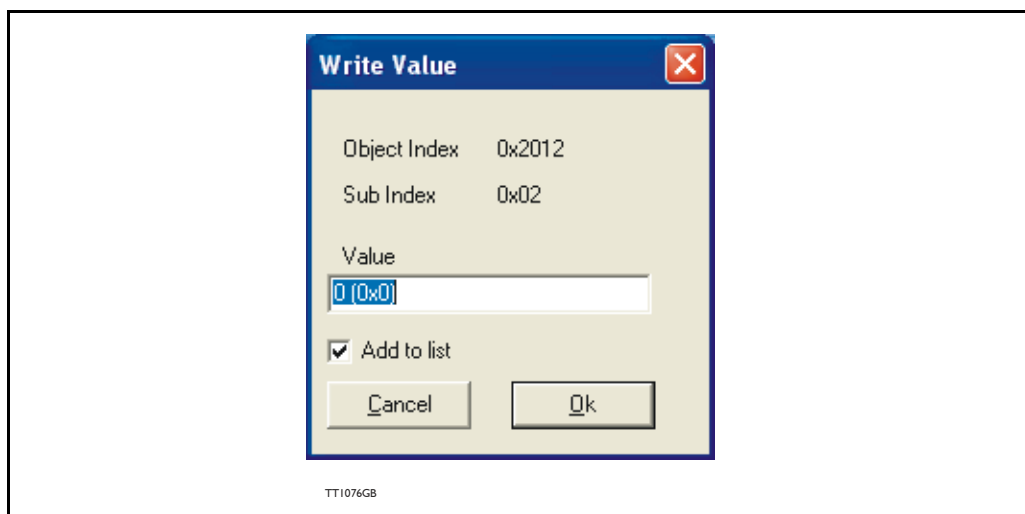


- 6: Point to the sub register 0x02, which is the register which determines in which mode the motor will operate.



4.5 Expansion Module MAC00-FC2/FC4

Press W on the keyboard, and the following screen appears:



- 7: Type 02 in the window, and press OK.
- 8: Click on the sub register 0x05, which is the register to choose which velocity the motor will run in. Press W on the keyboard, type 100 in the window, and press OK. 100 is in Counts/Sample.
- 9: Click on the sub register 0x03, which is the register to choose which distance the motor shall run. Press on W on the keyboard, type 20000 in the window, and type OK. 20000 is in Encoder Counts

Now shall the motor shaft rotate slowly, until the motor has counted 20000 Encoder pulses. If you want to stop the motor, when click on sub register 0x02 and write 0 in the window, and it will switch to passive mode. Now it is possible to change the value in the register and change the speed and distance for the motor.

If using other software the test could be described as, (using object 2012h):

Sub-register	Name	Width	Unit	Operation	Value
02h	Mode_Reg	16 bit		Set up the motor in position mode	02h
05h	V_SOLL	16 bit	Counts/sample	Sets up the desired velocity	100h
03h	P_SOLL	32 bit	Encoder count	The motor rotates the desired numbers of encoder pulses	20000
02h	Mode_Reg	16 bit		Sets the motor to passive mode	00h
Returning the motor with higher velocity					
02h	Mode_Reg	16 bit		Set up the motor in position mode	02h
05h	V_SOLL	16 bit	Counts/sample	Sets up the desired velocity	200h
03h	P_SOLL	32 bit	Encoder count	The motor rotates the desired numbers of encoder pulses	-20000
02h	Mode_Reg	16 bit		Sets the motor in passive mode	00h

4.5 Expansion Module MAC00-FC2/FC4

4.5.16 How to use CanOpenexplorer

After startup, the name and details of the HW-interface, like PCAN_USB should appear upper left.

When you turn on a motor/CAN node after having started CanOpenexplorer, the Data Window (large center right), there will come a message with the number 0x7xx, where xx is the node ID - for instance, 0x704 will indicate node 4. Set the Node ID field top center to that value (4).

Make sure the right EDS file is loaded. The program loads a hard-coded default file - either smc75.eds or mac00_fc.eds. It is also possible to load another EDS file by writing the file name in the "EDS file" field top center and pressing the load button. Note that the EDS view (large center left panel) will add the new file at the bottom but not clear the existing file(s) loaded.

Normal operation will be to select an object in the EDS view pane, and press either R for read or W for write. Pressing R should read the value, and that is successful if no error pops up. pressing w for write will pop up a small window, where the present value is displayed in both decimal and hex. It is then possible to write a new value either in decimal or in hex using a 0x prefix, like 0x185 to enable the first TPDO on node 5 (by clearing the high bit). If the Add to list checkbox is checked, the object will be added to the user SDO list as a write SDO. Pressing A performs a read and Adds it to the user SDO list pane (lower right) as a read SDO.

The SDOs in the user SDO pane can be rearranged by dragging them with the mouse. Double click on a user SDO list, will execute the operation, either reading or writing. The bus state can be changed using the NMT buttons lower left, like Operational to enable PDOs.

The button read user SDOs will read all of the "R" type object in the user SDO list. This is useful for updating a larger number of values in the EDS view.

The button read user SDOs will write all of the "W" type object in the user SDO list. This is useful for automated testing.

Entries can be deleted from the user SDO list by selecting them with the mouse and pressing the delete key.

The sync Time field top right sets the time in milli-seconds for the SYNC messages to be sent out. SYNCs can be started and stopped using the buttons Enable Sync and the Stop button to the right of it.

The Guard Time field below the Sync Time field works like SYNC-just for the Guarding message.

The close button exits the program after saving the list of user SDOs, which will be automatically reloaded on the next program start.

4.5 Expansion Module MAC00-FC2/FC4

4.5.17 MacTalk CAN debug window

The hardware CAN Node chip have a Node Control Register, this controls the initialization, defines the node specific interrupt handling and selects an operation mode. The Node Control Register have a field call LEC (Last Error Code), this bit field indicates if the latest CAN message has been correct (No error) or it indicates the type of error, which has been detected.

The register contains the following fields:

Bit	15-8	7	6	5	4	3	2	1	0
Field	0	BOFF	EWRN	0	RXOK	TXOK	LEC		
Read	X	X		X	X	X		X	
Write					X	X		X	

Where:

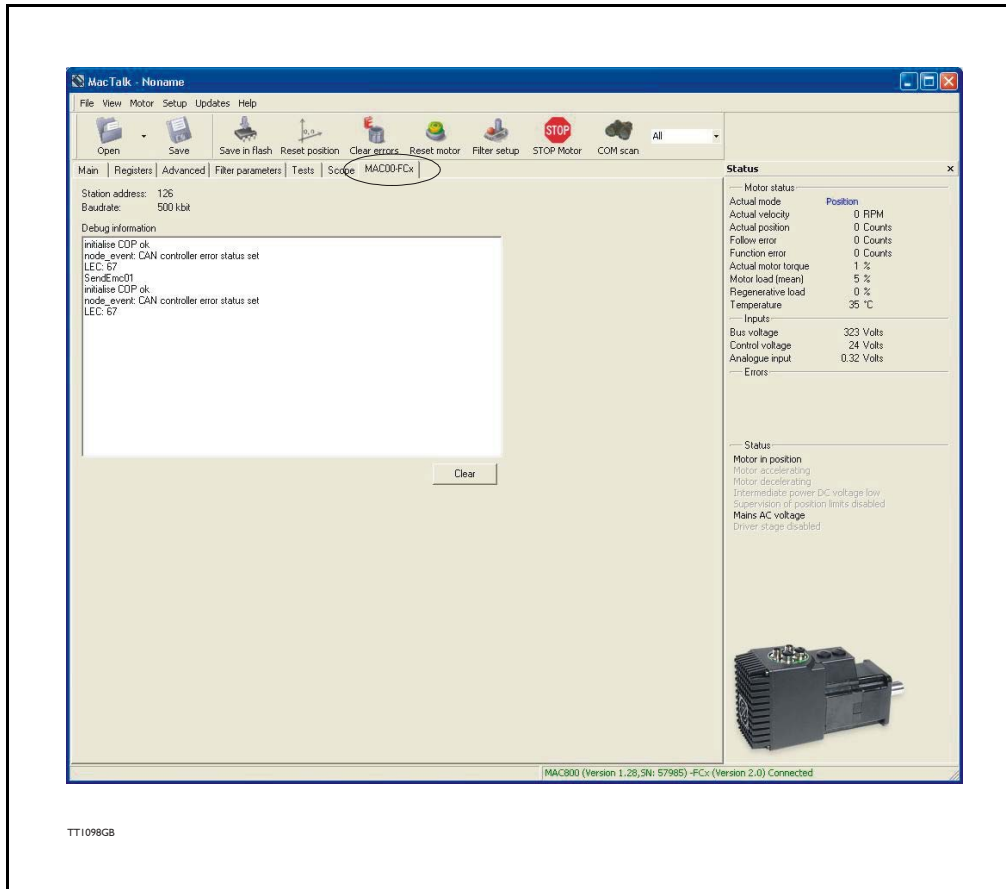
Field	Meaning	Value	Description
LEC	Last Error Code		See table below
TXOK	Message Transmitted Successfully	0	No successful transmission since last flag reset
		1	A message has been transmitted successfully (error free and acknowledged by at least one other node)
RXOK	Message Received Successfully	0	No successful reception since last flag reset.
		1	A message has been received successfully.
EWRN	Error Warning Status	0	No warning limit exceeded.
		1	One of the error counters in the Error Management Logic reached the error warning limit of 96
BOFF	Bus-Off Status	0	CAN controller is not in the bus-off state.
		1	CAN controller is in the bus-off state

Table for last Error code

LEC	Meaning	Description
000	No error	The latest transfer on the CAN bus has been completed successfully
001	Stuff error	More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed
010	Form error	A fixed format part of a received frame has the wrong format
011	Ack error	The transmitted message was not acknowledged by another node
100	Bit1 error	During a message transmission the CAN node tried to send a recessive level (1), but the monitored bus value was dominant (outside the arbitration field and the acknowledge slot)
101	Bit0 error	Two different conditions are signaled by this code: 1. During transmission of a message (or acknowledge bit, active error flag, overload flag), the CAN node tried to send a dominant level (0), but the monitored bus value has been recessive. 2. During bus-off recovery, this code is set each time a sequence of 11 recessive bits has been monitored. The CPU may use this code as an indication, that the bus is not continuously disturbed
110	CRC error	The CRC check sum of the received message was incorrect
111	Reserved	

4.5 Expansion Module MAC00-FC2/FC4

Select the **MAC00-FCx** tab. See the figure below:



And example of an error message are shown in the figure above

On this error message is there a “Bit0 error” condition on the CAN bus, and the CAN bus is in the bus-off-state, and a error counter in EWRN has reached the error limits. To get this information convert 67h to binary 1100111.

4.5 Expansion Module MAC00-FC2/FC4

4.5.18 The DS301 specified Communications objects are:

The different communications objects are shown in table below, to get the default value in CanOpenExplorer, press on R on the keyboard, and the actual value will be shown.

Name	Index (hex)	Sub Index	Data Type	Read only	Default	Description			
Device type	1000		UNSIGNED32	X	0x20192	Contains information about the device type. See note at top of next page. Mandatory.			
Error Register	1001		UNSIGNED8	X		This is the mapping error register, and it is part of the emergency object. If some of the sub index are high, an error has occurred. See also section 4.3.21. Mandatory			
		0				Generic error. Mandatory			
		1				Current			
		2				Voltage			
		3				Temperature			
		4				Communication (Overrun)			
		5				Device profile specific			
		6				Reserved			
Reservation register	1004					Reservation of PDOs			
		0				Reserved numbers of PDOs			
		1				Reserved numbers of syncPDOs			
		2				Reserved numbers of asyncPDOs			
Manufacturer device name	1008		VISIBLE STRING	X	JVL A/S				
Manufacturer hardware version	1009		VISIBLE STRING	X	1.0				
Manufacturer software version	100A		VISIBLE STRING	X		Example: Version x.x			
Guard time	100C		UNSIGNED16			Inform about the Guard time in milliseconds. Is only mandatory if the module does not support heartbeat			
Life time factor	100D		UNSIGNED8			Is the factor, that guard time is multiplied with, to give the life time for the node guarding protocol			
Heartbeat time	1017		UNSIGNED8			If the Heartbeat timer is not 0, Heartbeat is used.			
Identity object	1018		IDENTITY	X		Contain general information about the module			
		0				1..4	X	4h	Number of entries. Mandatory
		1				UNSIGNED32	X	0x0117	Vendor ID, contains a unique value allocated to each manufacturer. 117h is JVLs vendor ID. Mandatory.
		2				UNSIGNED32	X	0x0100	Product Code, identifies a specific device version. The MAC00-FC2 /FC4 has the product code 100h
		3				UNSIGNED32	X	0x20020	Revision number.
		4				UNSIGNED32	X		Serial number

4.5 Expansion Module MAC00-FC2/FC4

Note to “device type” (index 1000).

The device type register is composed of 2 16-bit registers, one register describes which device profile the module is supports, and the other state which type of motors the module is supports, and possible I/O module. The default value 0192h inform that the DSP402 Device profile are supported, and the value 0002h announce that the MAC00-FC2/FC4 module supports servo drives.

4.5.19 Emergency object

The EMCY (emergency) object is used to transfer an error message to the CANopen® master, or also to another node which can process the error message. The reaction on the emergency object is not specified. An emergency object is transmitted only once per “error event”

The MAC00-FC supports the EMC object (Emergency).
The following error codes can be generated:

Errorcode 1001h: Generic error - Motor error

Errorcode 1002h: Generic error - Limit switch error

Errorcode 1003h: Generic error - Internal communication error

Errorcode 1004h: Generic error - Queue overflow in communication queue.

The EMCY object 1001h are sent as an 8 bit header, an have the following structure:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANopen® error code: MSB 10	CANopen® error code: LSB 01	CANopen® 8-bit error register	MAC motors ERR_STAT MSB 16-bit error register	MAC motors ERR_STAT LSB 16-bit error register	Reserved		

Byte 0-1:

Shows which Generic error the module is sending.

Byte 2:

Error register. In the error register it is indicated to which error class the error belongs.
Motor status table:

Bit 0	Overload
Bit 1	Follow error
Bit 2	Function error
Bit 3	Regenerative error
Bit 4	In position
Bit 5	Accelerating
Bit 6	Decelerating
Bit 7	Position

Byte 3-4:

The ERR_STAT register is located in the motor, not in the MAC00-FC2/4 module, but the SendEmc01 message is sent from the module firmware whenever it receives a status byte from the motor where the error-bit is set, it then reads register 35 from the motor. When the error is no longer present, the module will send a NoError EMCY object once.

4.5 Expansion Module MAC00-FC2/FC4

The EMCY object 1002h is sent as an 8 byte message, and has the following structure:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANopen® error code: MSB 10	CANopen® error code: LSB 02	CANopen® 8-bit error register	0	0	0	0	0

EMCY/object 1002h is sent when any of the HW end limits are active. No additional information in bytes 3-7.

The EMCY object 1003h is sent as an 8 byte message, and has the following structure:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANopen® error code: MSB 10	CANopen® error code: LSB 03	CANopen® 8-bit error register	0	0	0	0	0

EMCY/object 1003h is sent when internal communication between the module and the motor has been disconnected.

The EMCY object 1004h is sent as an 8 byte message, and has the following structure:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANopen® error code: MSB 10	CANopen® error code: LSB 04	CANopen® 8-bit error register	0	0	0	0	0

EMCY/object 1004h is sent in case of overflow in the communications queue between the module and the motor.

In MAC00-FC2/FC4 none of the error control is enabled then the modules are started up, because if there is any fault in the system it is impossible to get in contact with the module. After the module has started up and there is communication between the master and the slave, then turn on the wanted error control mechanism in the object Dictionary, see section 4.3.20.

4.5 Expansion Module MAC00-FC2/FC4

4.5.20 Manufacturer specific object dictionary

	Index (hex)	Sub Index	Type	Read only	Default	Description
Command	2010	0	UNSIGNED 8			Execute a MAC00-FCx command
Module parameters	2011	0	UNSIGNED 8	X	8	Subindex count
		1	UNSIGNED 8	X		Input status IN1 - IN4, NL, PL
		2	UNSIGNED 8			Output
		3	UNSIGNED 8	X		Motor Status
		4	UNSIGNED 16	X		Last Motor Error
		5	UNSIGNED 8			Output setup
		6	UNSIGNED 8		0x3F	Input active level
		7	UNSIGNED 8			Input setup
		8	UNSIGNED 8			Setup bits
Motor parameters	2012	0	UNSIGNED 8	X	254	Subindex count
		n	UNSIGNED 32			Access to the motor parameter n
FastMac Command	2013	0	UNSIGNED 8			Executes a FastMac command
Homing Torque	2100	0	UNSIGNED 16		1000	Defines the torque limit used during homing with DSP-402

Writing to these object in CanOpenExplorer is done by pressing W, on the keyboard when the register in the folder Manufacturer specific is selected. Reading is done by pressing R.

4.5.21 Object 2010h-Subindex 0 Reset

When writing to this object (sub index 0), it is possible to execute some special commands for the MAC00-FCx module. The following commands are available:

Number	Function
0	No operation
1	Reset limit error
2	Reset communication error
3-255	Reserved

4.5 Expansion Module MAC00-FC2/FC4

4.5.22 Object 2011h – Subindex 1 Input status

This object is used to read out the actual value of the inputs.

Bit	7	6	5	4	3	2	1	0
Input	Reserved		PL	NL	IN4	IN3	IN2	IN1

PL is the positive limit switch input, and NL is the negative limit switch input. IN1-IN4 is the digital user inputs. On the MAC00-FC2 module are the input connected via J2, and on the MAC00-FC4 module are the input connected via the M12 connectors marked I/O.

4.5.23 Object 2011h – Subindex 2 Outputs

With this object the outputs can be controlled.

The value written to this object is directly shown on the outputs if the outputs are not set to use the default function (see subindex 5).

Bit	7	6	5	4	3	2	1	0
Output	Reserved						O2	O1

O1 and O2 are the digital user output 1 and 2. On the MAC00-FC2 module the outputs are connected via J4, and on the MAC00-FC4 module the outputs are connected via the M12 connectors marked I/O.

4.5.24 Object 2011h – Subindex 3 Motor status

With this object the status of the motor can be monitored.

Bit	7	6	5	4	3	2	1	0
Data	Reserved	Deceleration	Acceleration	In position	Reserved	Limit switch Error	Disconnected	Motor Error

Bit 6: Equals 1, if the velocity is decreasing.

Bit 5: Equals 1, if the velocity is increasing.

Bit 4: Equals 1, if the motor is at the commanded position.

Bit 2: Equals 1, if a limit switch has been activated.

Bit 1: Equals 1, if there is a communication error between the MAC00-FC and the motor. This could occur if the motor has been reset due to a voltage drop.

Bit 0: Equals 1, if there is a fatal motor error. Read subindex 4 to get extended information.

4.5 Expansion Module MAC00-FC2/FC4

4.5.25 Object 2011h – Subindex 4 Last motor status

When a fatal motor error occurs, the ERR_STAT register from the MAC motor is received and can be read from this object.

Motor status table:

Bit 0	Overload
Bit 1	Follow error
Bit 2	Function error
Bit 3	Regenerative error
Bit 4	In position
Bit 5	Accelerating
Bit 6	Decelerating
Bit 7	Position

4.5.26 Object 2011h – Subindex 5 Output setup

This object is used to control the function of the outputs. When bit $x = 0$, the outputs are controlled by the object 2011h, subindex 2.

When bit $x = 1$, the output is controlled by the default function. The default function for O1 is “In position” and for O2 “Error”.

Bit	7	6	5	4	3	2	1	0
Output	Reserved						O2	O1

4.5.27 Object 2011h – Subindex 6 Input active level

With this object the active level of the inputs can be selected. When bit $x = 0$, the input is active low and when bit $x = 1$, the input is active high.

The default setup for the output is active high.

Bit	7	6	5	4	3	2	1	0
Input	Reserved		PL	NL	IN4	IN3	IN2	IN1

4.5.28 Object 2011h – Subindex 7 Input setup

With this object the dedicated function of the inputs can be enabled. When the corresponding bit is 0, the input functions as a normal input. When the corresponding bit is 1, the dedicated function of the input will be enabled. When the end limit inputs NL or PL are enabled and one of these is activated, the error action will be executed. The error action is defined in object 2011h subindex 8.

Bit 1 - “Input Mirror”. Setting this bit will transfer the state of the inputs NL, PL, IN1-4 to the “Input” register in the basic motor. This is useful if the inputs are used in for example a “eRxP” program (graphic programming).

Notice that only MAC00-FCx Firmware newer than 3.00 supports this feature.

Bit	7	6	5	4	3	2	1	0
Input	Reserved		PL	NL	Reserved		Input Mirror	

4.5 Expansion Module MAC00-FC2/FC4

4.5.29 Object 2011h – Subindex 8 Setup bits

This object is used for auxiliary setup of the module

Bit	7	6	5	4	3	2	1	0
Setup	Endless relative	Error action	Reserved				SCAN_V_IST	SCAN_P_IST

SCAN_P_IST: When this bit is 1, the P_IST is scanned all the time. The transmit PDO21 will then send the last scanned position instead of reading the position.

SCAN_V_IST: When this bit is 1, the V_IST is scanned all the time. The transmit PDO22 will then send the last scanned velocity instead of reading the velocity.

Endless relative: When this bit is 1, the endless relative position mode is used when doing relative positioning in DSP-402. When using this mode, absolute positioning can no longer be used.

Error action: 0 = set motor in passive mode, 1 = stop motor by setting velocity to zero.

4.5 Expansion Module MAC00-FC2/FC4

4.5.30 Object 2012h – Motor parameters

With this object all the registers of the MAC motor can be accessed. All the registers are accessed as 32 bit. When reading and writing to 16 bit registers, the values are automatically converted in the module. In addition to these features listed in the table below, many more are accessible. In total, the MAC motor contains more than 150 internal registers such as nominal velocity, actual position, etc. But please note that several registers are not for the normal user and damage may occur if the contents of these registers is changed. The table shows the most commonly used registers.

Sub-index (Hex)	Name	Data type	Read/Write	Default (HEX)	Unit	Description
00	Number of entries	UNSIGNED8	Read	253		
01	PROG_VERSION	VISIBLE_STRING	Read	120		
02	MODE_REG	UNSIGNED16	Write			0: Passive mode 1: Velocity mode 2: Position mode 3: Gear mode 4: Analog Torque mode 5: Analog Velocity mode 6: Analog Velocity/Gear mode 7-11: Reserved 12: Torque Zero Search 13: Sensor type1 Zero search 14: Sensor type2 Zero search
03	P_SOLL	UNSIGNED32	Write		Encoder counts	The commanded position
05	V_SOLL	UNSIGNED16	Write		Counts/sample	Desired velocity
06	A_SOLL	UNSIGNED16	Write		Counts/sample ²	The maximum allowed acceleration
07	T_SOLL	UNSIGNED16	Write			The maximum allowed torque
0A	P_IST	UNSIGNED32	Read		Encoder counts	The actual position
0C	V_IST	UNSIGNED16	Read		Counts/sample	The actual velocity
0E	GEAR_1=0	Integer	Write			Gear output factor used in gear mode
0F	GEAR_2=0	Word	Write			Gear input factor used in gear mode
10	I2T	Word	Read			Motor temperature calculated
11	I2tLIM	Word	Read			Error trip level used for I2T register
1C	MIN_P_IST	Long int	Read		Encoder counts	Software position limit-positive

Continued next page

4.5 Expansion Module MAC00-FC2/FC4

Sub-index (Hex)	Name	Data type	Read/Write	Default (HEX)	Unit	Description
1E	MAX_P_IST	Long int	Read		Encoder counts	Software position limit negative
20	ACC_EMERG	Word	Write		Counts/sample ²	The maximum allow deceleration when a Unrecoverable error has occurred
21	INPOSWIN	Word	Write		Encoder counts	If actual position is within this window, the motor is in position
22	INPOSCNT	Word			Samples	The number of samples the motor has to be within the pos. interval spec.in INPOSWIN
23	ERR_STAT	Unsigned16	Read			Motor status: Bit 0: Overload Bit 1: Follow error Bit 2: Function error Bit 3: Regenerative error Bit 4: In position Bit 5: Accelerating Bit 6: Decelerating Bit 7: Position limits error

4.5.31 Object 2013h – Subindex 0 FastMac command.

When writing to this object, a FastMac command is executed. Please refer to the MAC00-FPx section for a description of the FastMac commands.

4.5.32 Enable and Disable PDOs

In the CANOpen® profile it is only possible to have four transmit and four receive PDOs enabled at the same time. In the MAC00-FC2/FC4 all PDOs are disabled when the module is booted up, the user has to choose which PDOs the application will use, and enable these.

To enable or disable a PDO it is necessary to write to the MSB (bit 31) in the PDO COB-ID entry in the PDO communication parameter Record. The COB-ID register is sub-index 1h, and the value range of this register is UNSIGNED32.

The PDOs are enabled when bit 31 is 0, and is disabled when bit 31 is 1.

4.5 Expansion Module MAC00-FC2/FC4

The table below shows default value of the COB-ID:

PDO	Sub-index	Type	Description	Default	Access type
21	1	Receive	COB-ID	Nodeid+0x80000200	r/w
	1	Transmit	COB-ID	Nodeid+0x80000180	r/w
22	1	Receive	COB-ID	Nodeid+0x80000300	r/w
	1	Transmit	COB-ID	Nodeid+0x80000280	r/w
23	1	Receive	COB-ID	Nodeid+0x80000400	r/w
	1	Transmit	COB-ID	Nodeid+0x80000380	r/w
24	1	Receive	COB-ID	Nodeid+0x80000500	r/w
	1	Transmit	COB-ID	Nodeid+0x80000480	r/w
25	1	Transmit	COB-ID	Nodeid+0x80000480	r/w

Remark: Some PLC's count PDO's starting from 1. Other PLC's count from 0. If counting from 0 please subtract 1 from the above mentioned PDO numbers.

4.5.33 Receive PDOs

The PDOs 1-20 are reserved for use with the DSP-402 (CANopen® motion control profile). The following receive PDOs are available:

Receive PDO 21:

This PDO can be used to update the position. The data in the PDO is written directly to the position register and if the motor is in position mode, it will start moving to that position.

Byte	0	1	2	3	4	5	6	7
Data	P_SOLL				Reserved	Reserved	Reserved	Reserved
Object	2012h, sub 3							

Receive PDO 22:

With this PDO it is possible to update the velocity, acceleration and torque.

Byte	0	1	2	3	4	5	6	7
Data	V_SOLL		A_SOLL		T_SOLL		MODE_REG	
Object	2012h, sub 5		2012h, sub 6		2012h, sub 7		2012, sub 2	

Receive PDO 23:

This PDO sets a new operating mode for the motor.

Byte	0	1	2	3	4	5	6	7
Data	FastMac Command	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Object	2013h, sub 0							

Add 96 to the FastMac command number. For example command 23 becomes 119 (decimal)

4.5 Expansion Module MAC00-FC2/FC4

Receive PDO 24:

This PDO updates the outputs.

Byte	0	1	2	3	4	5	6	7
Data	Output data	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Object	2011h, sub 2							

4.5.34 Transmit PDOs

The transmit PDOs 1-20 are reserved for use with the DSP-402 (CANopen® motion control profile).

All of the transmit PDOs support synchronous transmission, and PDO 25 also supports asynchronous transmission.

Transmit PDO 21:

With this PDO the actual position can be read.

Byte	0	1	2	3	4	5	6	7
Data	P_IST			Motor Status	Inputs	Reserved	Reserved	
Object	2012h, sub 10			2011h, sub 3	2011h, sub 1			

Transmit PDO 22:

With this PDO the actual velocity can be read.

Byte	0	1	2	3	4	5	6	7
Data	V_IST		Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Object	2012h, sub 12							

Transmit PDO 23:

With this PDO the actual torque can be read.

Byte	0	1	2	3	4	5	6	7
Data	VF_OUT		Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Object	2012h, sub 121							

Transmit PDO 24:

With this PDO the value of the analog input can be read.

Byte	0	1	2	3	4	5	6	7
Data	ANINP		Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Object	2012h, sub 122							

4.5 Expansion Module MAC00-FC2/FC4

Transmit PDO 25:

With this PDO the motor status, inputs and last error can be read.

This PDO also supports asynchronous transmission. If this PDO is in asynchronous mode, it will be transmitted every time the run status or inputs are changed.

Byte	0	1	2	3	4	5	6	7
Data	Motor Status	Inputs	Last motor error		Reserved	Reserved	Reserved	Reserved
Object	2011h, sub 3	2011h, sub 1	2011h, sub 4					

4.5.35 Transmission time

Due to the internal communication between the motor and the MAC00-FCx, the PDOs takes a certain time to process. The following table shows the processing time for the PDOs

PDO number	21	22	23	24	25
Receive PDO	8.5ms	21ms	<1ms	<1ms	-
Transmit PDO	12.5ms (<1ms)*	10.5ms (<1ms)**	10.5ms	10.5ms	<1ms

* : Note that Transmit PDO21 is faster if P_IST scanning is enabled. (See object 2011h subindex 8).

** : Note that Transmit PDO22 is faster if V_IST scanning is enabled. (See object 2011h subindex 8).

If the received PDOs are transmitted faster than the internal processing time, an internal queue overflow occurs (See emergency object). If the SYNC object interval is smaller than the processing time of the active transmit PDOs, an internal queue overflow error occurs.

4.5 Expansion Module MAC00-FC2/FC4

4.5.36 DSP-402 Support

Introduction

The MAC00-FCx supports the DSP-402 standard from CiA® (<http://www.can-cia.com/>).

Please refer to this standard for full details of the functions.

The DSP-402 is only a standard proposal and might be changed in the future. We reserve the right to change future firmware versions to conform to new versions of the standard. Not all of the functionality, described in DSP-402, is supported. But all the mandatory functions are supported.

The following operation modes is supported:

- Profile position mode
- Velocity mode
- Homing mode

Precondition:

Before the DSP-402 mode can be used, the firmware in the FCx module must be updated to at least version 1.3.

The start mode of the motor must be set to passive.

No power up *Zero searches* must be selected.

If absolute movement is used, the 'resynchronize after passive mode' must be set.

When using DSP-402 mode, manipulating parameters with object 2012h or 2013h can corrupt the behaviour of the DSP-402 functions. Also be aware that manipulating parameters in MacTalk should be avoided when using DSP-402.

4.5 Expansion Module MAC00-FC2/FC4

4.5.37 Supported objects

Most of the DSP402 parameters start up in the module with-coded values. A few of them are set depending on the motor type the module is attached to - either MAC50-141 or MAC400-800. None of the parameters can be saved to flash in the module.

The following table shows the additional object dictionary defined for DSP-402 support. The numbers in brackets, in the update/write field, **bold**, refer to the formula for the factors in the section 4.3.40

Name	Desc	COB ID (hex)	Sub-index	Motor register	InitValue	Scalefactor to motor	Update/write
Device data							
Motor_type		6402	0		10		
Motor_catalog_number		6403	0		MACxxx		
Motor_manufacturer		6404	0		JVL A/S		
http_motor_catalog_address		6405	0		www.JVL.dk		
Supported_drive_modes		6502	0		45		
Drive_catalog_number		6503	0		MACxxx		
Drive_manufacturer		6504	0		JVL A/S		
http_drive_catalog_address		6505	0		www.JVL.dk		
Digital I/O							
Digital_inputs		60FD	0	Motor status			When HW inputs or motor status (change) See formula (6) in section 4.3.40
Digital_outputs_numbers_of_entries		60FE	0		2		
Digital_outputs_Physical_outputs		60FE	1	HW output	0		Imm. See formula (7) in section 4.3.40
Digital_outputs_Bit_mask		60FE	2	HW output	0		Imm. See formula (7) in section 4.3.40
Device control							
Abort_connection_option_code	N/U	6007	0				
Error_code	N/U	603F	0				
Controlword		6040	0				
Statusword		6041	0				
Quick_stop_ortion_code		605A	2				Used in state machine
Modes_of_operation		6060	0				
Mode_of_operation_display		6061	0				
Profile Position parameters							
Position_actual_value		6064	0			1/Position_factor	BusyRead
Target_position		607A	0	P1		Position_factor	Positionmode when bit in Controlword is set
Software_position_limit_number_of_entries		607D	0		0		
Software_position_limit_Min_position_limit		607D	1		0		0
Software_position_limit_Max_position_limit		607D	2		0		0
Position_window		6067	0	Z1	100	Position_factor	Imm.
Position_window_time		6068	0	INPOSCNT	6	SamleFreq/1000	Imm
Max_motor_speed	N/U	6080	0		5000 or 4000		
Profile_velocity	N/U	6081	0	V1	100	Velocity_factor	Imm
Profile_acceleration		6083	0	A1	15000	Acceleration_factor	Imm
Quick_stop_deceleration		6085	0	A2	50000	Acceleration_factor	Imm
Motion_profile_type	N/U	6086	0		0		

Continued on next page

4.5 Expansion Module MAC00-FC2/FC4

Name	Desc	COB ID (hex)	Sub-index	Motor register	Initial Value	Scalefactor to motor	Update/write
Profile velocity mode							
Velocity_sensor_actual_value		6069	0	V_IST			BusyRead
Velocity_demand_value	N/U	606B	0				(Copied from target velocity on updated)
Velocity_actual_value		606C	0	V_IST		1/(Velocity_factor*16)	BusyRead
Velocity_window		606D	0	Z1	100	Velocity_factor/16	Imm
Velocity_window_time		606E	0	INPOSCNT	6	SampleFreq/1000	Imm
Target_velocity		60FF	0	V1	50	Velocity_factor or Velocity_factor depending on polarity	Imm + Start-Velocity mode
Max_torque		6072	0	T1 and TSOLL	1000	1.023	Imm
Homing mode							
Home_offset		607C	0	P_HOME			During homing See (8) in section 4.3.40
Homing_method		6098	0				See homing description.
Homing_speeds_number_of_entries	RO	6099	0		2		
Homing_speeds_Speeds_during_search_for_switch		6099	1	V1	50	+/- Velocity_factor	At homing
Homing_speeds_Speeds_during_search_for_zero		6099	2	V2	50	Velocity_factor	At homing
Homing_acceleration		609A	0	ASOLL	5000	Acceleration_factor	At homing
Factors							
Position_notation_index	N/U	6089	0		0		
Position_dimension_index	N/U	608A	0		0xAC		
Velocity_notation_index	N/U	608B	0		0		
Velocity_dimension_index	N/U	608C	0		0xA4		
Acceleration_Notation_index	N/U	608D	0		0		
Acceleration_dimension_index	N/U	608E	0		0		
Position_encoder_resolution_number_of_entries	RO	608F	0		2		
Position_encoder_resolution_Encoder_increment		608F	1		4096 or 8000		Not CF_Upd, >Position_factor
Position_encoder_resolution_Motor_revolution		608F	2		1		CF_Upd, >Position_factor
Velocity_encoder_resolution_number_of_entries	RO	6090	0		2		
Velocity_encoder_resolution_encoder_increments_per_second	N/U	6090	1		4096 or 8000		
Velocity_encoder_resolution_motor_resolution_s_per_second	N/U	6090	2		1		
Gear_ratio_number_of_entries	RO	6091	0		2		
Gear_ratio_Motor_revolutions		6091	1				CF Upd, >Position_factor
Gear_ratio_Shaft_revolutions		6091	2				CF Upd, >Position_factor
Feed_constant_number_of_entries	RO	6092	0		2		See formula (4) in section 4.3.40
Feed_constant_Feed	N/U	6092	1		4096 or 8000		In CF_Upd
Feed_constant_Shaft_revolutions		6092	2		1		CF Upd, >Position_factor >Feed_constant >PFactorNumerator
Position_factor_number_of_entries	RO	6093	0		2		See formula (1) in section 4.3.40
Position_factor_Numerator	N/U	6093	1		1		In CF_Upd See formula (5) in section 4.3.40
Position_factor_Feed_constant		6093	2				

Continued on next page

4.5 Expansion Module MAC00-FC2/FC4

Name	Desc.	COB ID (hex)	Sub-index	Motor register	Initial Value	Scalefactor to motor	Update/write
Velocity_encoder_factor_number_of_entries	RO	6094	0		2		See formula (2) in section 4.3.40
Velocity_encoder_factor_Numerator		6094	1		4096 or 8000		CF_Upd, >Velocity_factor
Velocity_encoder_factor_Divisor	RPM	6094	2		60		CF_Upd, >Velocity_factor
Acceleration_factor_number_of_entries	RO	6097	0		2		See formula (3) in section 4.3.40
Acceleration_factor_Numerator		6097	1		4096 or 8000		CF_Upd.>Acceleration_factor
Acceleration_factor_Divisor	RPM	6097	2		60		CF_Upd Acceleration_factor
Polarity	Bit7: InvPos. Bit6: InvVel.	607E	0				CF_Upd, >Position_factor >Velocity_factor
SampleFreq					520.833 or 770		Not CF_Upd, >Velocity_factor >Acceleration_factor
Homing_Torque		2100	0	T_HOME	500 in V2.0, 30 in V2.1	1.023	At start of homing
Module Parameters:							
Inputs status		2011	1				See section 4.3.24
Outputs		2011	2		0		See section 4.3.25
Motor status		2011	3				See section 4.3.26
Last motor status		2011	4		0		See section 4.3.27
Output setup		2011	5		0		See section 4.3.28
Input active level		2011	6		0x3F		See section 4.3.29
Input setup		2011	7		0		See section 4.3.30
Setup bits		2011	8				See section 4.3.31
Data (256 motor registers)		2012	x				See section 4.3.32
Fastcommand (Send FastMac command)		2013	0				See section 4.3.33

4.5 Expansion Module MAC00-FC2/FC4

4.5.38 Factors

Position_factor. (1) in section 4.3.39

The position factor is the relation between the user unit and the internal position unit (counts). The position factor is automatically calculated when the feed constant (Object 6092h) and gear ratio (Object 6091h) are set.

Example:

We have a MAC motor with a 3.5:1 gear box connected to a belt drive. The diameter of the drive wheel is 12.4 cm. We want the unit of position to be in millimetres.

The circumference of the drive wheel is 389.56mm (124mm*pi). The parameters should be set as follows:

Object	Name	Value
6091h subindex 1	Gear_ratio_Motor_revolutions	35
6091h subindex 2	Gear_ratio_Shaft_revolutions	10
6092h subindex 1	Feed_constant_Feed	38956
6092h subindex 2	Feed_constant_Shaft_revolutions	100

Please note that it is not necessary to set the encoder resolution. This is automatically set by the module.

Positions_factor formula:

$$\text{Position_factor} = \frac{\text{Gear_ratio_Motor_rev.} * \text{Feed_constant_Shaft_Rev.} * \text{Position_encoder_res.} * \text{Encoder_Increments}}{\text{Feed_constant_Feed} * \text{Feed_constant_Shaft_rev.} * \text{Position_encoder_res.} * \text{Motor_rev.}}$$

or as and object:

$$\text{Position_factor} = \frac{\text{Object 6091sub1} * \text{Object 6092sub2} * \text{Object 608Fsub1}}{\text{Object 6092sub1} * \text{Object 6092sub2} * \text{Object 608Fsub2}}$$

The Position_factor is calculated to in the above example:

$$\text{Position_factor} = \frac{35 * 100 * 4096}{38956 * 10 * 1} = 36,8$$

for a MAC50-141. For and MAC800, shall 4096 be changed to 8000.

Velocity_encoder_factor. (2) in section 4.3.39.

This factor is used to convert the user unit into the internal unit (counts/sec).

The factor is adjusted via object 6094h.

Example 1:

We have a MAC800 motor with 8000 counts/revolution. We want the user unit of the velocity to be in RPM. The parameters should be set as follows:

Object	Name	Value
6094h subindex 1	Velocity_encoder_factor_Numerator	8000
6094h subindex 2	Velocity_encoder_factor_Divisor	60

4.5 Expansion Module MAC00-FC2/FC4

Velocity_encoder_factor formula:

$$\text{Velocity_encoder_factor} = \frac{\text{Velocity_encoder_factor_Numerator}}{\text{Velocity_encoder_factor_Divisor}} * \frac{16}{\text{SampleFreq}}$$

notice that the samplefrequens is differens in MAC050-141.

Or as and object:

$$\text{Velocity_encoder_factor} = \frac{\text{Object 6094sub1}}{\text{Object 6094sub2}} * \frac{16}{\text{Samplefreq}}$$

The Velocity_encoder_factor in example 1 is calculated to:

$$\text{Velocity_encoder_factor} = \frac{8000}{60} * \frac{16}{770} = 2,77 \text{ RPM}$$

Example 2:

We have a MAC800 motor with 8000 counts/revolution and the same belt drive as in the above example under Position_Factor. We want the user unit of the velocity to be in mm/s. The parameters should be set as follows:

Object	Name	Calculated value	Value
6094h subindex 1	Velocity_Encoder_Factor_Numerator	$389.56/(3.5*8000)$ =0.013913	13913
6094h subindex 2	Velocity_Encoder_Factor_Divisor	1	1000000

The Velocity_encoder_factor in example 2 is calculated to:

$$\text{Velocity_encoder_factor} = \frac{13913}{1000000} * \frac{16}{770} = 0,000289 \text{ mm}$$

Acceleration_factor. (3) in section 4.3.39.

This factor is used to convert the user unit into the internal unit (counts/sec²). The factor is adjusted via object 6097h.

Example 1:

We have a MAC800 motor with 8000 counts/revolution. We want the user unit of the acceleration to be in RPM/s. The parameters should be set as follows:

Object	Name	Value
6097h subindex 1	Acceleration_factor_Numerator	8000
6097h subindex 2	Acceleration_factor_Divisor	60

Acceleration_factor formula:

$$\text{Acceleration_factor} = \frac{\text{Acceleration_factor_Numerator}}{\text{Acceleration_factor_Divisor}} * \frac{16}{\text{SampleFreq} * \text{SampleFreq}}$$

4.5 Expansion Module MAC00-FC2/FC4

or as and object:

$$\text{Acceleration_factor} = \frac{\text{Object 6097sub1}}{\text{Object 6097sub2}} * \frac{16}{\text{Samplefreq} * \text{Samplefreq}}$$

notice that the samplefrequens is differens in MAC050-141.

The Acceleration_factor in example 1 is calculated to:

$$\text{Accelerationr_factor} = \frac{8000}{60} * \frac{16}{770 * 770} = 0,003598 \text{ RPM/s}$$

Example 2:

We have a MAC800 with 8000 counts/rev. and the same belt drive as in the above example under Position_Factor. We want the user unit of the acceleration to be in mm/s². The parameters should be set as follows:

Object	Name	Calculated value	Value
6097h subindex 1	Acceleration_Encoder_Factor_Numerator	389.56/(3.5*8000) =0.013913	13913
6097h subindex 2	Acceleration_Encoder_Factor_Divisor	1	1000000

The Acceleration_factor in example 2 is calculated to:

$$\text{Accelerationr_factor} = \frac{13913}{1000000} * \frac{16}{770 * 770} = 3,75 * 10^{-7} \text{ mm/s}^2$$

Feed_constant_factor. (4) in section 4.3.39.

Feed_constant_factor formula:

$$\text{Feed_constant_factor} = \frac{\text{Feed_constat_Feed}}{\text{Feed_constant_Shaft_revolutions}}$$

or as and object:

$$\text{Feed_constant_factor} = \frac{\text{Object 6092sub1}}{\text{Object 6092sub2}}$$

Position_factor_Numerator (5) in section 4.3.39.

Position_factor_Numerator formula

$$\text{Position_factor_Numerator} = \frac{\text{Feed_constant_Feed} * \text{Position_factor}}{\text{Feed_constant_Shaft_revolutions}}$$

or as and object:

$$\text{Position_factor_Numerator} = \frac{\text{Object 6092sub1} * \text{Position_factor}}{\text{Object 6092sub2}}$$

4.5 Expansion Module MAC00-FC2/FC4

Digital Inputs (6) in section 4.3.39.

Bits 31: 24 The Motor Status register (2011-3)
Bits 23: 16 HardwareInputs (2011-1) - the 6 HW-inputs on the module
Bit 2: Home sensor state
Bits 1: 0
Bits 5: 4 from (HardwareInputs ANDed with InputSetup (2011-7)) - (in reality in puts 5 and 6)
DigitalInputs = ((UINT32)Inputs << 16)
((UINT32)MotorStatus << 24
(((Inputs & InputSetup) >> 4) & 0x3)
(unsigned char)P5_P0 << 2

Outputs (7) in section 4.3.39.

Outputs = ((DigitalOutputs & OutputMask) >> 16) & 0x3

Note that the module outputs come from the manufacturer specific bits 17:16 and that bit 0, Set Brake, is not implemented.

Home offset (8) in section 4.3.39.

P_Home = -(PositionFactor * HomeOffset) + HomeTable[Method].HomeOffset * IndexDistance * UseIndex

4.5.39 Changing operation mode

A change of operation mode is only possible when the operation mode is not enabled. There are two exceptions and one is when changing from homing mode to profile position mode. This is possible when the homing sequence is completed and can be done even though the operation mode is enabled. The other exception is when changing from profile position mode into velocity mode.

4.5.40 Profile position mode

This mode can be used for positioning where a movement profile can be set up. The acceleration and maximum velocity can be programmed.

In this mode, both absolute and relative moves are supported. The type of move is selected via bit 6 (abs/rel) in the status word. When a relative move is selected, the type of relative move is dependent on the setup in object 2011h subindex 8.

It is also possible to select different movement modes. This is done using bit 5 (change set immediately) in the status word. When this bit is 0 and a move is in progress, the new set-point is accepted. But the new set-point and profile are not activated before the previous movement is finished. When this bit is 1, the new set-point is activated instantly and the motor will move to the new position with the new profile parameters.

Please note:

- The torque limit that is used during the profile can be set via object 6072h.
- The register L1 (object 2012 subindex 81) is used to select the load factor when the profile is started. If a different load factor is required, this register must be set correctly.

4.5 Expansion Module MAC00-FC2/FC4

4.5.41 Velocity mode

In this mode the motor runs at a selected velocity. A new velocity can be selected and the motor will then accelerate/decelerate to this velocity.

The maximum slippage error is not supported in this mode.

Please note:

- The torque limit can be set via object 6072h.

4.5.42 Homing mode

In this mode different homing sequences can be initiated. The standard homing modes from section 4.3.22 are supported.

The home sensor must be connected to the AIN input on the module.

If the end limit inputs must be active during the homing sequence, they must be enabled via object 2011h subindex 7.

The sensors should be connected to the appropriate inputs NL and PL.

The torque limit used during homing is selected via object 2100h. The unit of this is object is the same as other torque objects, e.g. Object 6072h.

There are also 4 manufacturer specific methods. These are listed in the table below.

Method	Uses index	Description
-1	Yes	Torque homing in negative direction and afterwards homing on the index pulse.
-2	Yes	Torque homing in positive direction and afterwards homing on the index pulse.
-3	No	Torque homing in negative direction.
-4	No	Torque homing in positive direction.

Please note that you should always use a home offset (object 607Ch) when using torque homing. This is to ensure that the motor moves away from the end limit. The sign of the home offset should be the opposite of the homing direction. For example, when using a negative homing direction, the home offset could be 5000.

4.5 Expansion Module MAC00-FC2/FC4

4.5.43 Supported PDOs

Receive PDOs

PDO no.	Mapping object index	Mapping object name	Comment
1	6040h	Controlword	Controls the state machine
2	6040h 6060h	Modes of operation	Controls the state machine and modes of operation
3	6040h 607Ah	Controlword Target position (pp)	Controls the state machine and the target position (pp)
4	6040h 60FFh	Controlword Target velocity (pv)	Controls the state machine and the target velocity (pv)
7	6040h 60FEh	Controlword Digital outputs	Controls the state machine and the digital outputs

Transmit PDOs

PDO no.	Mapping object index	Mapping object name	Event driven	Comment
1	6041h	Statusword	Yes	Shows status
2	6041h 6061h	Modes of operation	Yes	Shows status and the current mode of operation
3	6041h 6064h	Statusword Position actual value	No	Shows status and the current position (pp)
4	6041h 606Ch	Statusword Velocity actual value	No	Shows status and the current velocity (pv)
7	6041h 60FDh	Statusword Digital inputs	Yes	Controls the state machine and the digital inputs

4.5 Expansion Module MAC00-FC2/FC4

4.5.44 CANopen® DS-301 device profiles

Standardized devices in CANopen® have their characteristics described in a device profile. For each device profile, particular data and parameters are strictly defined, data and parameters are known as objects in CANopen. Objects perform all processes in CANopen®, they can perform various tasks, it can be as a communication object or as device specific objects, where they are directly related to the device. A communication object can transport data to the bus control and establish connection, or supervise the network devices.

The application layer makes it possible to exchange meaningful real-time-data across the CAN network, the format of this data and its meaning must be known by the producer and the consumer(s). There are encoding rules that define the representation of values of data types and the CAN network transfer syntax for the representations. Values are represented as bit sequences. Bit sequences are transferred in sequences of octets (byte). For numerical data types the encoding is with the lowest byte first.

Every object is described and classified in the object dictionary (or index) and is accessible through the network. They are addressed using a 16 bit index so that the object dictionary may contain a maximum of 65536 entries.

Index (Hex)	Object	Supported by MAC00-FC2/FC4
0000-	Not used	
0001-001F	Static data types	
0020-003F	Complex data types	
0040-005F	Manufacturer specific Data Types	
0060-0FFF	Reserved for further use	
1000-1FFF	Communication Profile area DS301	Yes
2000-5FFF	Manufacturer specific profile area	Yes
6000-9FFF	Standardised Device Profile area (DSP-402)	Yes
A000-FFFF	Reserved for further use	

Index 0001-001F:

Static data types contain type definitions for standard data types like boolean, integer, floating point etc. These entries are included for reference only, they cannot be read or written.

Index 0020-003F:

Complex data types are pre-defined structures that are composed out of standard data types and are common to all devices.

Index 0040-005F:

Manufacturer specific data types are also structures composed of standard data types but are specific to a particular device.

Index 1000-1FFF:

The communication Profile area contains the parameters for the communication profile on the CAN network. These entries are common to all devices.

Index 2000-5FFF:

The manufacturer specific profile area, for truly manufacturer specific functionality.

4.5 Expansion Module MAC00-FC2/FC4

Index 6000-9FFF:

The standardised device profile area, contains all data objects common to a class of devices that can be read or written via the network. The drives profile uses entries from 6000h to 9FFFh to describe the drive parameter and the drive functionality. Within this range up to 8 devices can be described. In such a case the device are denominated Multi Device Modules. Multi Device Module are composed of up to 8 device profile segments. By this feature it is possible to build devices with multiple functionality. The different device profile entries are shifted with 800h.

A 16-bit index is used to address all entries within the object dictionary. In case of a simple variable this references the value of this variable directly. In case of records and arrays however, the index addresses the whole data structure. To allow individual elements of structures of data to be accessed via the network a sub-index has been defined. For single object dictionary entries such as and Unsigned8, Boolean, Integer32, the value for the sub-index is always zero. For complex object dictionary entries such as arrays or records with multiple data fields the sub-index refers to fields within a data-structure pointed to by the main index. Index counting starts with one.

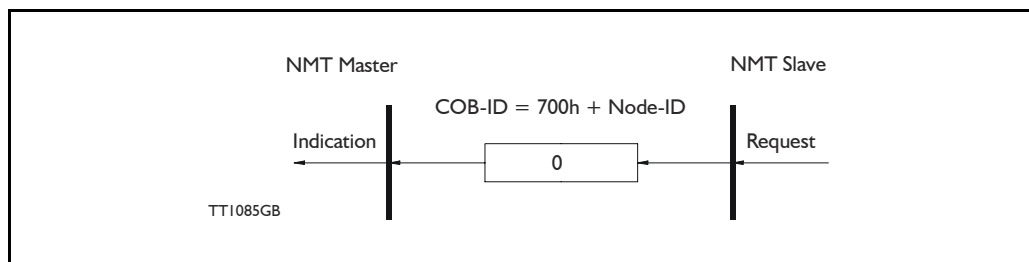
The DS-301 standard is the application and the communications profile for a CANopen® bus, and is the interface between the devices and the CAN bus. It defines the standard for common data and parameter exchange between other bus devices, and it controls and monitors the devices in the network. In the table below are listed some of the communications profile objects:

Data Transfer	Parameter Transfer	Special functions	
PDO			Process Data Objects
	SDO		Service Data Objects
		SYNC	Synchronisation
		EMCY	Emergency

The access from the CAN network is done through data objects PDO (Process Data Object) and SDO (Service Data Object).

4.5.45 Boot up telegram

After the initialization phase, a CANopen® slave log on with a boot up message. The node address of the slave is contained in this. This allows a CANopen® master to know which slaves are connected to the network. The protocol uses the same identifier as the error control protocols, see the figure below:



One data byte is transmitted with value 0.

4.5 Expansion Module MAC00-FC2/FC4

4.5.46 PDO (Process Data Object):

PDO: Performs real time transfers, and the transfer of PDOs are performed without a protocol. PDOs are used in two ways, for data transmission and for data reception. PDOs can bundle all objects from the object data directory, and a PDO can handle max 8 bytes of data in the same PDO. The PDO can consist of multiple objects. Other PDOs characteristic is, that it doesn't reply when it is receiving data, this for making the data transfer fast. It has a high priority identifier.

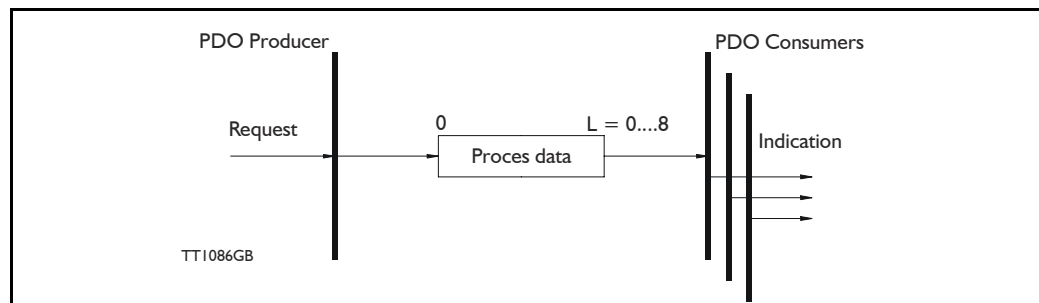
PDO connections follow the Producer/Consumer model. Whereby a normal PDO connection follows the Push model and a RTR connection the Pull model.

Objects are mapped in a PDO. This mapping is an agreement between the sender and receiver as to which object is located at which position in the PDO. This means that the sender knows at which position in the PDO it should write data and the receiver knows to where it should transfer data which it received.

The PDOs correspond to entries in the Device Object Dictionary and provide the interface to the application objects. Data type and mapping of application objects into a PDO is determined by a corresponding PDO mapping structure within the Device object Dictionary. Number and length of PDOs of a device is application specific and have to be specified within the device profile

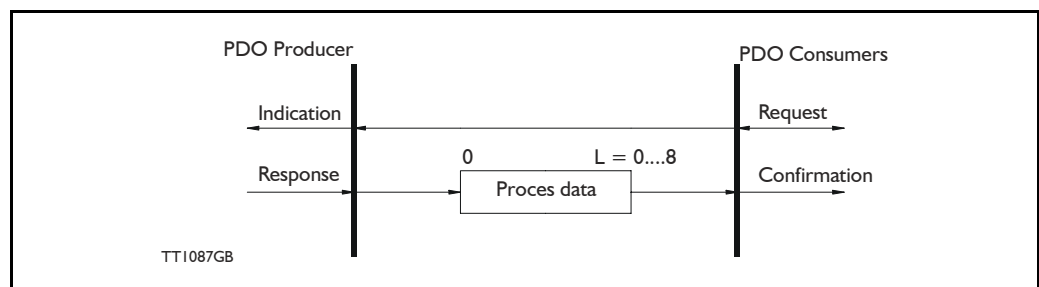
Write PDO service:

The Write PDO service is unacknowledged. There is a PDO producer which sends its PDO to the PDO consumer. There can be 0 or more consumers in the network. For receive PDOs, MAC00-FC2/FC4 is the consumer and for Transmit PDOs, the producer. The following figure shows a Write PDO service:



Read PDO service:

The read PDO service is an acknowledged service. One of the several PDO consumers send a RTR message to the network. After it has received the RTR message, the PDO producer sends the requested PDO. This service is used for RTR queries. Using this service, an actual value can be interrogated independently of the selected cycle time. The following figure show a read PDO service:



4.5 Expansion Module MAC00-FC2/FC4

PDO identifier:

In the CANOpen® profile it is only possible to have four transmit and four receive PDOs enabled at the same time. In the MAC00-FC2/FC4 all PDOs are disabled when the module is booted up, the user has to choose which PDOs the application is to use, and enable these.

The PDO configuration can be seen either in the EDS-file or in the CanOpen Explorer program, here the communication and the mapping parameter are shown.

There are two standard methods to map the PDOs in CAN-Open, there is a static mapping and a dynamic mapping. In the static PDO mapping all PDOs are mapped in accordance with some fixed non-modifiable setting in the relevant PDO. In the dynamic PDO mapping the setting for a PDO can be modified. It is also allowed to have flexible combination of different process data during operation. The MAC00-FC2/FC4 module, use only static mapping.

4.5.47 SDO (Service Data Objects):

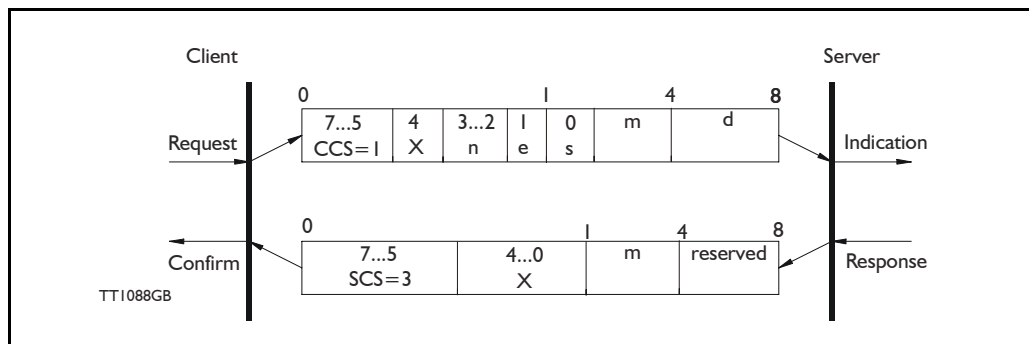
SDO: can access all entries in the object directory, but they are normally used in the initialization, during the boot up procedure. Some SDOs characteristic are:

- Confirmed transfer of objects
- Data transfer/exchange is always non-synchronous
- Values greater than 4 bytes are transferred (Normal transfer)
- Values not more than 4 bytes are transferred (Expedited transfer)

Basically a SDO is transferred as a sequence of segments. Prior to transferring the segment there is an initialization phase where client and server prepare themselves for transferring the segment. For SDOs, it is also possible to transfer a data set of up to four bytes during the initialisation phase. This mechanism is called an expedited transfer.

Down loading SDO protocol

The download SDO protocol is used to write the values of the object directory into the drive



4.5 Expansion Module MAC00-FC2/FC4

Upload SDO protocol

The upload SDO protocol is used to read the values in the object directory of the drive.

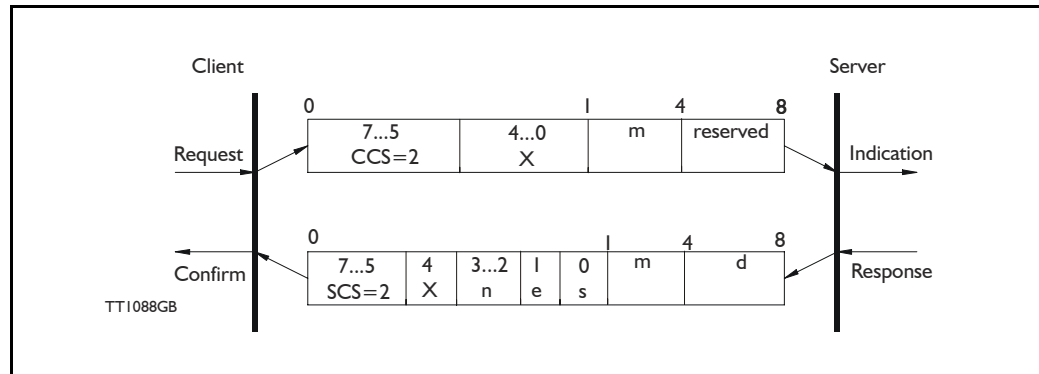


Table for upload and download SDO protocol.

	CCS:	SCS:	n:	e:	s:	m:
Down-load	1: Initiate down-load request	3: Initiate download response	Only valid if e=1 and s=1 otherwise 0. If valid it indicates the number of bytes in d that do not contain data. Bytes [8-n,7] do not contain data	Transfer type: 0= normal transfer 1= expedited transfer	Size indicator: 0=data set size is not indicated 1=data set size is indicated	Multiplexer. It represents the index/sub-index of the data to be transfer by the SDO
Upload	2: Initiate upload request	2: Initiate upload response	Only valid if e=1 and s=1 otherwise 0. If valid it indicates the number of bytes in d that do not contain data. Bytes [8-n,7] do not contain data	Transfer type: 0= normal transfer 1= expedited transfer	Size indicator: 0=data set size is not indicated 1=data set size is indicated	Multiplexer. It represents the index/sub-index of the data to be transfer by the SDO

CCS:Client command specified.

SCS:Server commander specified.

4.5 Expansion Module MAC00-FC2/FC4

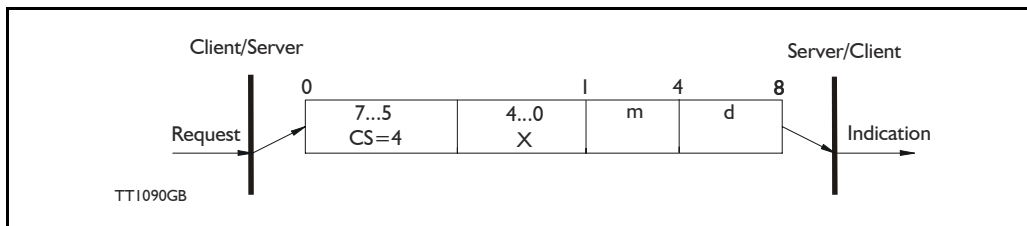
Table for upload and download SDO protocol (continued)

	d:	X:	Reserved:
Download	e=0, s=0: d is reserved for further use e=0, s=1: d contains the number of bytes to be downloaded. Byte 4 contain the lsb and byte 7 contain the msb e=1, s=1: d contain the data of length 4-n to be downloaded, the encoding depends on the type of the data reference by index and sub-index.	not used, always 0	Reserved for further use, always 0
Upload	e=0, s=0: d is reserved for further use e=0, s=1: d contains the number of bytes to be uploaded. Byte 4 contain the lsb and byte 7 contain the msb e=1, s=1: d contain the data of length 4-n to be uploaded, the encoding depends on the type of the data reference by index and sub-index.	not used, always 0	Reserved for further use, always 0

Abort SDO transfer protocol:

SDO tasks, which the MAC00-FC2/FC4 cannot process are responded to using an abort SDO protocol. If the module does not respond in the expected time, the CANOpen® master also sends an abort SDO.

The following figure show a abort SDO transfer protocol:



There are various abort codes in CANopen®, these are listed in the table below:

Abort code	Description
0503 0000h	Toggle bit not alternated
0504 0000h	SDO Protocol timed out
0504 0001h	Client/server command specified not valid or unknown
0504 0002h	Invalid block size (block mode only)
0504 0003h	Invalid sequence number (block mode only)
0504 0004h	CRC error (block mode only)
0504 0005h	Out of memory
0601 0000h	Unsupported access to an object
0601 0001h	Attempt to read a write only object
0601 0002h	attempt to write a read only object
0602 0000h	Object does not exist in the object dictionary
0604 0041h	Object cannot be mapped to the PDO

Table continued on next page.

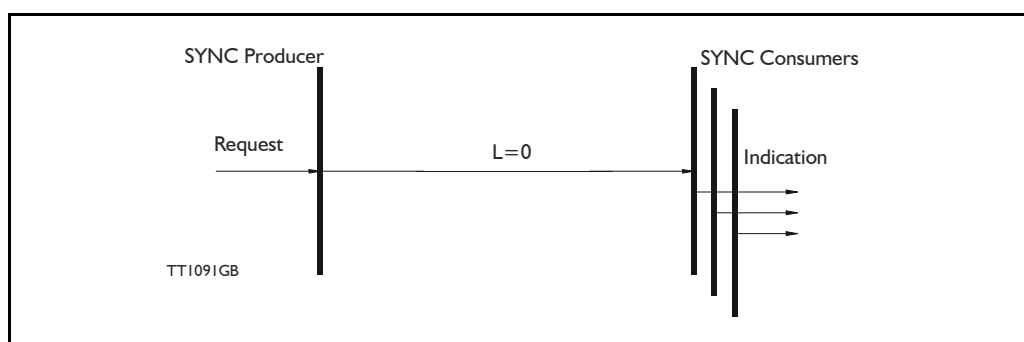
4.5 Expansion Module MAC00-FC2/FC4

Address table continued from previous page:

Abort code	Description
0604 0042h	The number and length of the objects to be mapped would exceed PDO length
0604 0043h	General parameter incompatibility reason
0606 0000h	Access failed due to an hardware error
0607 0010h	Data type does not match, length of service parameter does not match
0607 0012h	Data type does not match, length of service parameter too high
0607 0013h	Data type does not match, length of service parameter too low
0609 0011h	Sub-index does not exist
0609 0030h	Value range of parameter exceeded (only for write access)
0609 0031h	Value of parameter written too high
0609 0032h	Value of parameter written too low
0609 0036h	Maximum value is less than minimum value
0800 0000h	General error
0800 0020h	Data cannot be transferred or stored to the application
0800 0021h	Data cannot be transferred or stored to the application because of local control
0800 0022h	Data cannot be transferred or stored to the application because of the present device state
0800 0023h	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of a file error).

4.5.48 SYNC (Synchronisation Object)

A SYNC producer sends the synchronization object cyclically a broadcast telegram. The SYNC telegram defines the basic clock cycle of the network. The time between the SYNC telegram is set using the object Communication Cycle period (1006h). In order to obtain a precise (accurate) cycle between the SYNC signals, the SYNC telegram is sent with a high-priority identifier. This can be modified using the object (1005h). The SYNC transfer applies the producer/consumer push model and is non-confirmed.



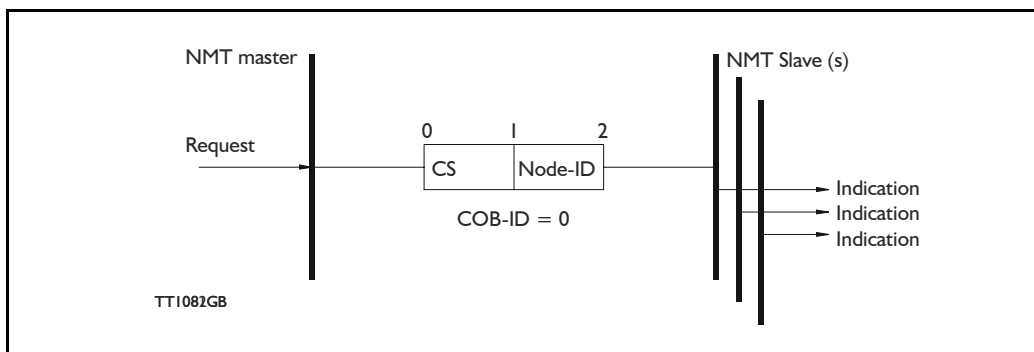
The SYNC does not carry any data (L=0). The identifier of the SYNC object is located at object 1005h.

4.5 Expansion Module MAC00-FC2/FC4

4.5.49 NMT (Network Management services)

The Network Management is structured according to nodes and follows a master-slave structure. NMT objects are used for executing NMT services. Through NMT services, nodes are initialised started, monitored, resetted or stopped. All nodes are regarded as NMT slaves. An NMT slave is uniquely identified in the network by its Node-ID. NMT requires that one device in the network fulfils the function of the NMT master. The NMT master controls the state of the NMT slaves. The state attribute is one of the values (Stopped, Pre-operational, Operational, initialising). The module control services can be performed with a certain node or with all nodes simultaneously. The NMT master controls its own NMT state machine via local services, which are implementation dependent. The Module Control Service except Start Remote Node can be initiated by the local application.

A general NMT protocol:



Where **CS** is the NMT command specified. The Node-ID of the NMT slave as assigned by the NMT master in the Node Connect protocol, or 0. If 0, the protocol addresses all NMT slaves.

CS =	Operation
1	Start Remote Node
2	Stop Remote Node
128	Enter Pre Operational
129	Reset Node
130	Reset Communication

Start Remote Node:

This is an instruction to transition from the Pre-Operational to Operational communications state. The drive can only send and receive process data when it is in the Operational state.

Stop Remote Node:

This is an instruction to transition from Pre-Operational into stopped or from Operational into Stopped. In the stopped state, the nodes can only process NMT instructions.

Enter Pre Operational:

This is an instruction to transition from Operational or Stopped into Pre-Operational. In the Pre-Operational state, the node cannot process any PDOs. However, it can be parameterized or operated via SDO. This means setpoint can also be entered.

4.5 Expansion Module MAC00-FC2/FC4

Reset Node:

This is an instruction to transition from Operational, Pre-Operational or Stopped to initialization. After the Reset Node instruction, all objects (1000h-9FFFh) are reset into the Voltage On stage.

Reset Communication:

This is an instruction to transition from Operational or Stopped to Initialization. After the Reset Communication instruction, all communication objects (1000h-1FFFh) are reset into the initial state.

In the various communication states, nodes can only be accessed via CANOpen® using specific communication services. Further, the nodes in the various states only send specific telegram. This is clearly shown in the following table:

	Initializing	Pre-Operational	Operational	Stopped
PDO			X	
SDO		X	X	
Synchronization Object		X	X	
Emergency Object		X	X	
Boot-Up Object	X			
Network Management object		X	X	X

4.5.50 Error Control Services

There exist two possibilities to perform Error Control:

- Node Guarding/Life Guarding
- Heartbeat

With Node Guarding, the CANopen® master sends, to each slave an RTR telegram (Remote Transmit request) with the COB-ID 1792 (700h) + node-ID.

The slave responds, with the same COB-ID, with its communications state. This means either Pre-Operational, Operational or stopped.

The CANopen® slave also monitors the incoming RTR telegram from the master.

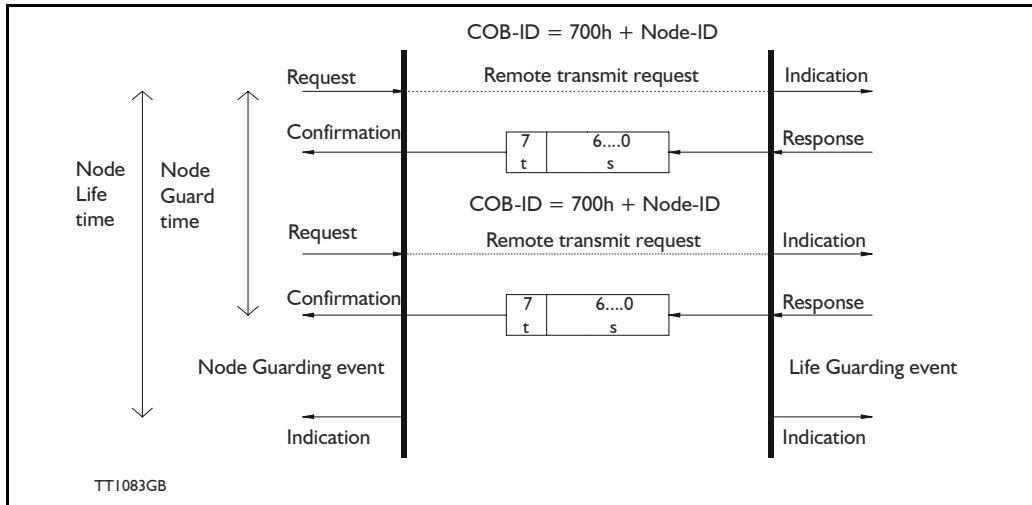
The cycle of the incoming RTR telegrams is set by using the Guard Time Object.

The numbers of RTR telegrams which can fail as a maximum before the slave initiates a Life Guarding event is defined using the Life time factor object.

The Node Life Time is calculated from the product of the Guard Time and Life Time Factor. This is the maximum time which the slave waits for an RTR telegram.

4.5 Expansion Module MAC00-FC2/FC4

The figure below show a Node Guarding/Life Guarding protocols



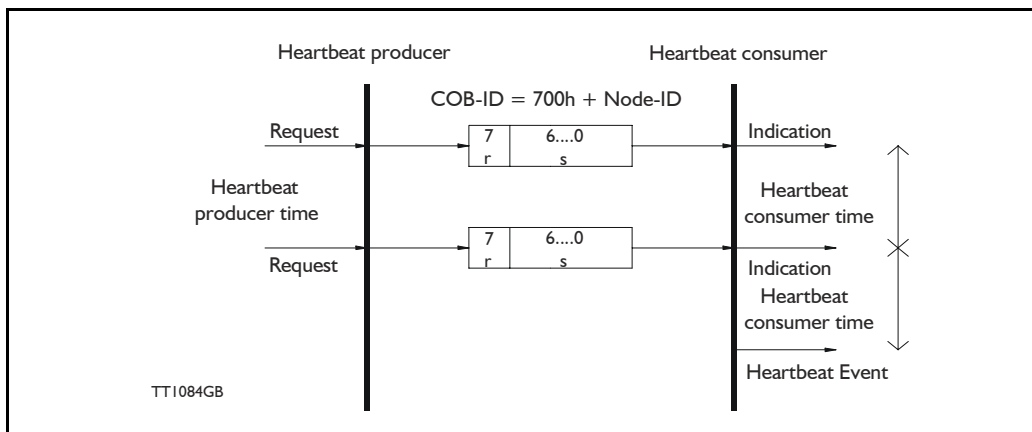
Where s is the state of the NMT slave:

s	NMT state
4	Stopped
5	Operational
7	Pre-operational

t: is the toggle bit, it alternate between 2 consecutive responses from the NMT Slave. The value of the toggle-bit of the first response after the guarding protocol becomes active, is 0. The Toggle Bit in the guarding protocol is only reset to 0 when the NMT message Reset Communication is passed (no other change of state resets the toggle bit). If a response is received with the same value of the toggle-bit as in the preceding response then the new response is handled as if it was not received.

Heartbeat:

With the Heartbeat protocol, a Heartbeat Producer cyclically sends its communications state to the CAN bus. One or more Heartbeat Consumers receive the indication. The relationship between producer and consumer is configured via the object dictionary. The Heartbeat Consumer guards the reception of the Heartbeat within the Heartbeat Consumer time. If the Heartbeat is not received within the Heartbeat Consumer Time a Heartbeat Event will be generated.



4.5 Expansion Module MAC00-FC2/FC4

Where r is reserved (always 0).

s: is the state of the Heartbeat producer:

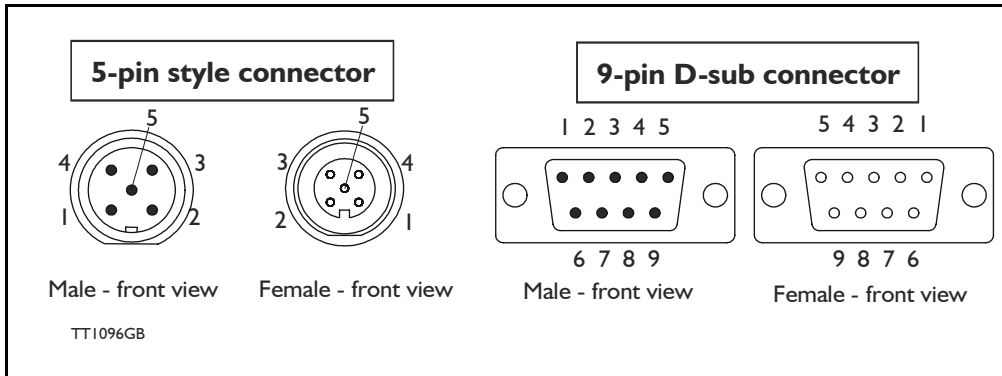
s	NMT state
0	Boot up
4	Stopped
5	Operational
7	Pre-operational

Only one communication monitoring service may be activated. This is either Node Guarding/Life Guarding or Heartbeat. If the Heartbeat Producer Time is configured on a device the Heartbeat Protocol begins immediately. If a device starts with a value for the Heartbeat Producer Time different from 0 the Heartbeat Protocol starts on the state transition from Initialising to Pre-operational. In this case the Boot-up Message is regarded as first heartbeat message. If the Heartbeat producer time is not 0 the heartbeat protocol is used.

In MAC00-FC2/FC4 none of the error control is enabled then the modules are started up, because if there is any fault in the system it is impossible to get in contact with the module. After the module has started up and there is communication between the master and the slave, then turn on the wanted error control mechanism in the object Dictionary, see section 4.4.20.

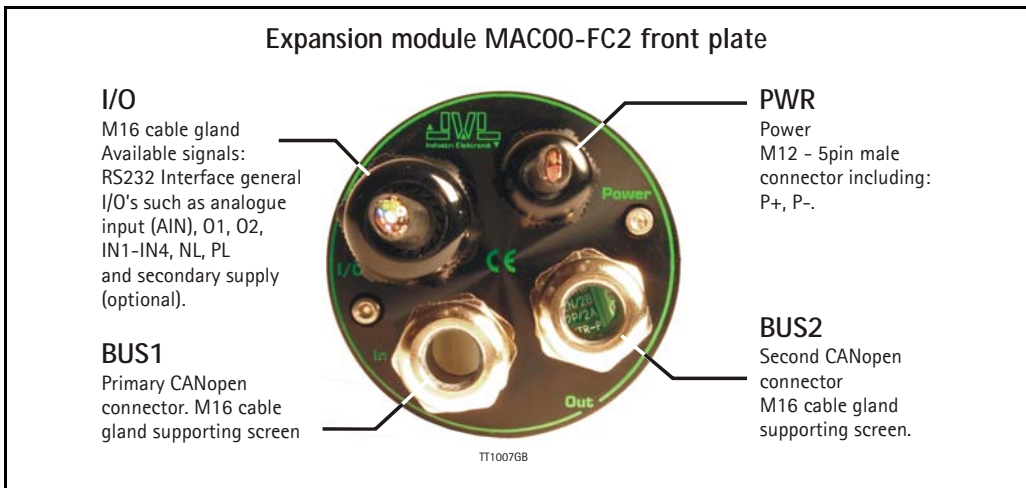
4.5 Expansion Module MAC00-FC2/FC4

CAN bus connectors - continued.

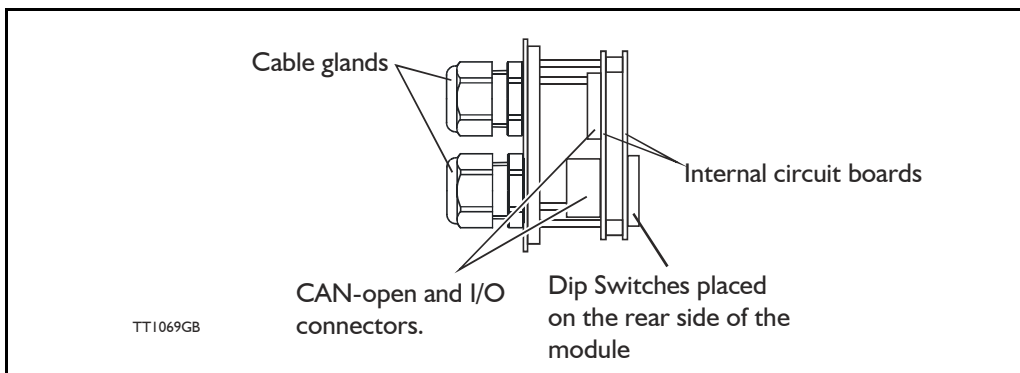


4.5.51 MAC00-FC2 Connectors

Rear plate layout:

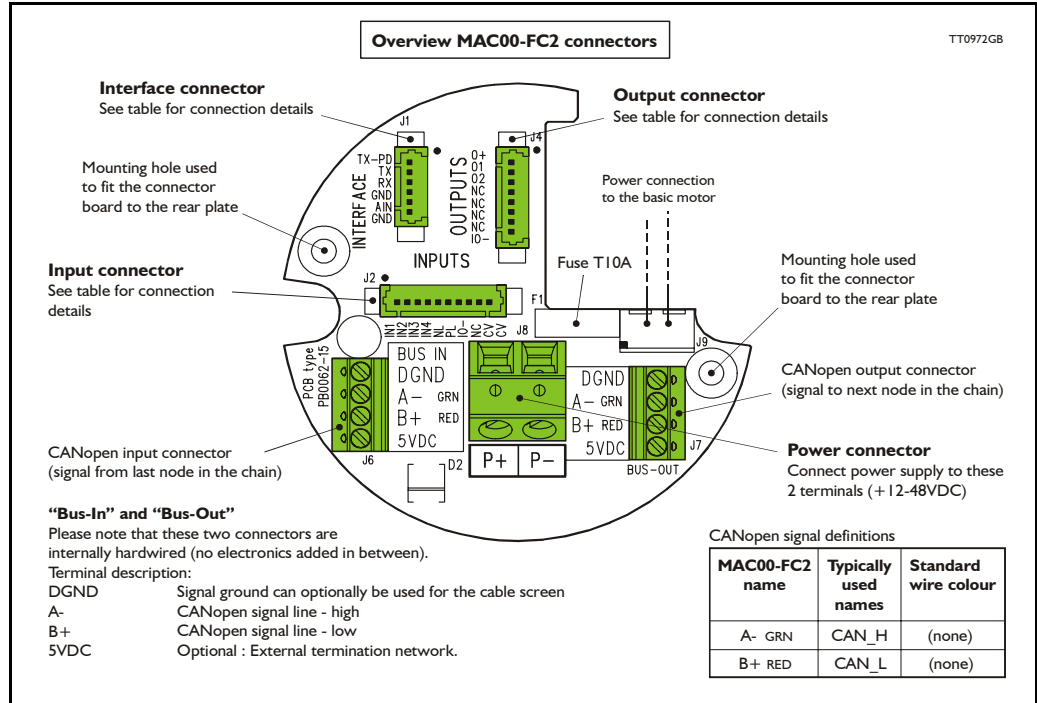


The MAC00-FC2 module is function as node in the CAN bus network, and the following terminals are available : B+ and A- are “Bus in”, and B+ and A- are “Bus out”. The connectors are placed on the dismantled module as the figure show below:



4.5 Expansion Module MAC00-FC2/FC4

The illustration below shows all the internal connectors in the module. The CAN bus and power connectors are easy-to-use screw terminals. If the I/Os are used, they require a JVL cable type WG0402 (2m), WG0410 (10m) or WG0420 (20m). See also the appendix for cable and connector accessories.



The MAC00-FC2 type number only covers the basic module, i.e. without any cables.

4.5 Expansion Module MAC00-FC2/FC4

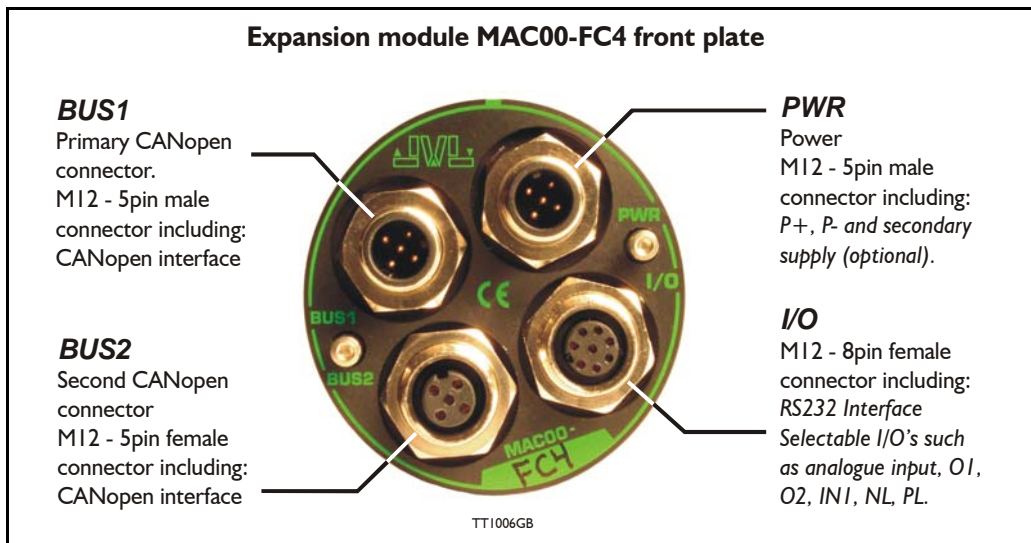
4.5.52 MAC00-FC2 with cables (optional)

If a number is added after the basic type number, for example MAC00-FC2-10, this suffix indicates that the module is fitted with 10 m of cable in the I/O. The I/O cable covers all the signal lines, i.e. RS232, Digital input 1-4, Limit inputs NL and PL and the Digital outputs 1-4

Digital Inputs - Internal connector J2			
Signal name	Pin no.	Description	Wire colour
IN1	1	Digital input 1	Red/black
IN2	2	Digital input 2	Green/black
IN3	3	Digital input 3	Violet
IN4	4	Digital input 4	Violet/white
NL	5	Negative limit input - If not used, do not connect.	Grey
PL	6	Positive limit input - If not used, do not connect.	Grey/black
IO-	7	I/O ground. This ground is shared with the output ground	Pink/black
NC	8	(Reserved)	Black/white
CV	9	Secondary supply. Used during emergency stop	Light green **
CV	10	Secondary supply. Used during emergency stop	White
Digital Outputs - Internal connector J4			
Signal name	Pin no.	Description	Wire colour
O+	1	Supply for outputs - Must be connected to an ext. supply.	Red/white
O1	2	Digital output 1 - PNP output	Green/white
O2	3	Digital output 2 - PNP output	Yellow/black
NC	4	(Reserved)	Blue/white
NC	5	(Reserved)	Orange/white
NC	6	(Reserved)	Brown/white
NC	7	(Reserved)	Pink
IO-	8	I/O ground. This ground is shared with the input ground	Black
Interface - including analogue input - Internal connector J1			
Signal name	Pin no.	Description	Wire colour
TXP	1	Transmit pull-down (Connect to TX if addr. not used).	Red
TX	2	RS232 Transmit (Connect to TXPD if addr. not used).	Green **
RX	3	RS232 Receive	Yellow
GND	4	Ground for RS232	Blue
AIN	5	Analogue input +/-10V or Zero sensor input	Orange
GND	6	Ground for AIN	Brown
Cable Screen			
The cable-screen is internally connected to motor housing. Externally it must be connected to earth.			
Unused wire			
Orange/Black - is not used internally. It must be left unconnected.			

** : The light green wire (CV) can be difficult to distinguish from the green wire (TX) on some cables.

4.5 Expansion Module MAC00-FC2/FC4



4.5.54 MAC00-FC4 connectors, rear plate layout

The set up of Baud-rate, Node-ID and terminator are selected in the same way as in the MAC00-FC2 module.

Expansion MAC00-FC4 Hardware description:

The MAC00-FC4 offers IP67 on MAC050-141 and IP65 on MAC400-800 protection and M12 connectors which makes it ideal for automation applications where no additional protection is desired. The M12 connectors offer solid mechanical protection and are easy to unplug compared to the FC2 module which has cable glands. The signals available are slightly restricted compared to the FC2 module since only 4 I/O terminals are available. The I/Os connected to these 4 terminals must be selected by a small dip-switch, see the drawing below the I/O table on the next page.

The connector layout:

"PWR" - Power input. M12 - 5-pin male connector				
Signal name	Description	Pin no.	JVL Cable WI1000M12 F5T05N	Isolation group
P+	Main supply +12-48VDC. Connect with pin 2 *	1	Brown	1
P+	Main supply +12-48VDC. Connect with pin 1 *	2	White	1
P-	Main supply ground. Connect with pin 5 *	3	Blue	1
CV	Control voltage +12-48VDC.	4	Black	1
P-	Main supply ground. Connect with pin 3 *	5	Grey	1
* Note: P+ and P- is each available at 2 terminals. Make sure that both terminals are connected in order to split the supply current in 2 terminals and thereby avoid an overload of the connector.				
"BUS1" - CAN-open interface. M12 - 5-pin male connector				
Signal name	Description	Pin no.	Cable: WI1006-M12F5SxxR	Isolation group
CAN_SHLD	Shield for the CAN interface - internally connected to the motor housing	1	Bare	2
CAN_V+	Reserved for future purpose - do not connect	2	Red	2
CAN_GND	CAN interface ground	3	Black	2
CAN_H	CAN interface. Positive signal line	4	White	2
CAN_L	CAN interface. Negative signal line	5	Blue	2

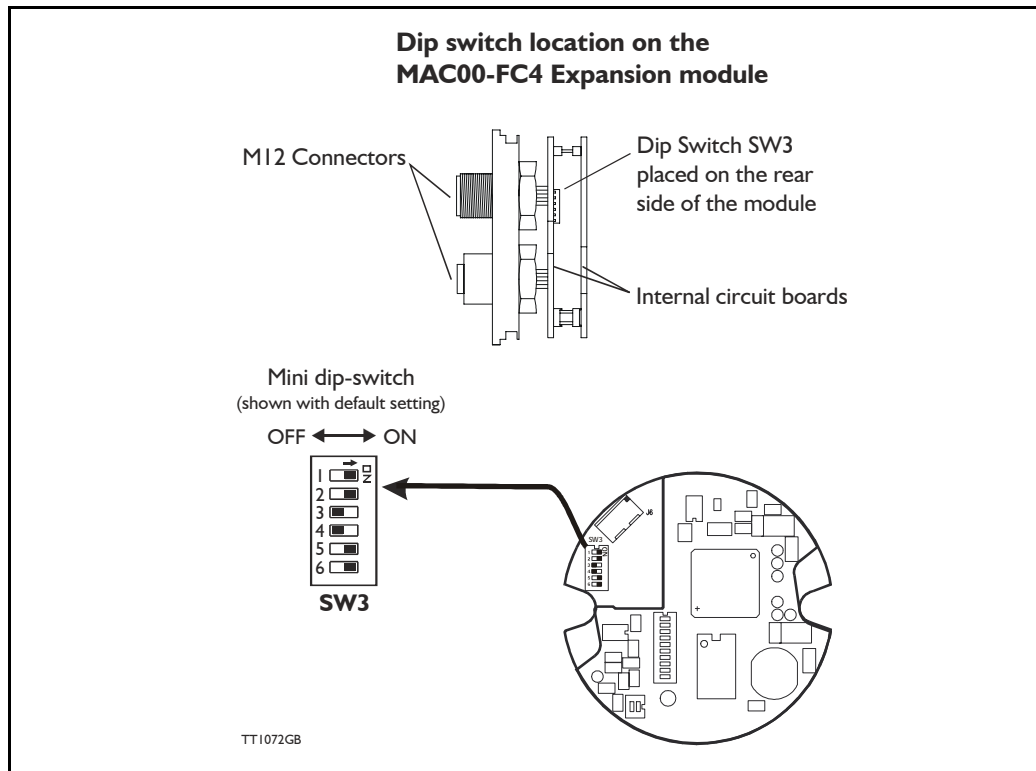
4.5 Expansion Module MAC00-FC2/FC4

"BUS2" - CANopen® interface. M12 - 5-pin female connector					
Signal name	Description	Pin no.	Cable: WI1006- M12M5SxxR	Isolation group	
CAN_SHLD	Shield for the CAN interface - internally connected to the motor housing	1	Bare	2	
CAN_V+	Reserved for future purpose - do not connect	2	Red	2	
CAN_GND	CAN interface ground	3	Black	2	
CAN_H	CAN interface. Positive signal line	4	White	2	
CAN_L	CAN interface. Negative signal line	5	Blue	2	
"IO" - I/Os and RS232 interface. M12 - 8-pin female connector.					
Signal name	Description	Function	Pin no.	JVL Cable WI1000-M12 M8T05N	Isolation group
IOC	I/O terminal C.	SW3 DIP 5 = OFF : <i>PL</i> input SW3 DIP 5 = ON : <i>O1</i> output	1	White	3
Tx	RS232 interface - transmit output Important !: DIP1 must be turned ON. If addressing is used it must be turned ON at minimum one of the connected motors.		2	Brown	1
Rx	RS232 interface - receive input		3	Green	1
GND	RS232 Ground - also used with analogue input		4	Yellow	1
IOA	I/O terminal A.	SW3 DIP 2 = ON and DIP3 = OFF : <i>A/N</i> (Analogue input) SW3 DIP2 = OFF and DIP 3 = ON : <i>O2</i> (output 2) (<i>A/N</i> is the analogue input. Remember to use the GND terminal with <i>A/N</i> !).	5	Grey	3 (1 when used as <i>A/N</i>)
IOB	I/O terminal B.	SW3 DIP 4 = OFF : <i>IN1</i> (input 1) SW3 DIP 4 = ON : <i>O1</i> (output 1)	6	Pink	3
IO-	I/O ground to be used with <i>IN1</i> , <i>NL</i> , <i>PL</i> , <i>O1</i> , <i>O2</i>		7	Blue	3
IOD	I/O terminal D.	SW3 DIP 6 = OFF : <i>NL</i> (negative limit input) SW3 DIP 6 = ON : <i>O+</i> (output supply)	8	Red	3
Cable Screen					
Some standard cables with M12 connector offer a screen around the cable. This screen on some cables is fitted to the outer metal at the M12 connector. When fitted to the MAC00-FC4 module, this means that the screen will have contact with the complete motor housing and thereby also the power ground (main ground).					
Isolation groups					
The MAC00-FC4 offers optical isolation at the digital inputs and outputs (<i>IN1</i> , <i>NL</i> , <i>PL</i> and <i>O1-2</i>). The table shows a number for each pin. This number refers to the isolation group to which each pin is connected. Isolation group 1 means that the terminal refers to the main ground (<i>P-</i> , <i>GND</i> and the motor housing). Isolation group 2 means that the terminal refers to the CAN interface ground (<i>CAN_GND</i>). Isolation group 3 means that the terminal refers to the I/O ground (<i>IO-</i>)					

Regarding the setting of SW3, see on next page.

4.5 Expansion Module MAC00-FC2/FC4

The drawing below shows the SW3 Dip-switch location. The various settings of SW3 is shown on the previous page.



Switch description:

SW3	Description	Function	Signal name
Dip 1	RS232 interface - transmit output	ON = Enable	Tx
Dip 2 Dip 3	I/O terminal A	DIP2=ON and DIP3=OFF : AIN (Analogue input)	IOA
Dip 2 Dip 3	I/O terminal A	DIP2=OFF and DIP3=ON : O2 (output 2)	IOA
Dip 4	I/O terminal B	DIP4=ON : Output 1 DIP4=OFF : Input 1	IOB
Dip 5	I/O terminal C	DIP5=ON : O1 output DIP5=OFF : PL (positive limit input)	IOC
Dip 6	I/O terminal D	DIP6=ON : O+ (Output supply) DIP6=OFF : NL (Negative limit input)	IOD









The factory default setting is:

SW3	ON	OFF	Function
Dip 1	X		RS232 interface Enable
Dip 2 Dip 3	X	X	O2 (output 2)
Dip 4		X	Input 1
Dip 5	X		O1 output
Dip 6	X		O+ (output supply)

4.5 Expansion Module MAC00-FC2/FC4

4.5.55 Cables for the MAC00-FC4

The following cables equipped with M12 connector can be supplied by JVL.

MAC00-FC4 Connectors				Description	JVL Order no.	Photo
"BUS1" 5-pin Male B-cod- ed	"BUS2" 5-pin Female B-cod- ed	"I/O" 8-pin Fe- male	"PWR" 5-pin Male			
		X		RS232 Interface cable. Connects directly from MAC00-FC4 to PC Length: 5m (197 inch)	RS232-M12-1-8	
			X	Cable (Ø5.5mm) with M12 female 5-pin connector loose wire ends 0.35mm² (22AWG) and foil screen. Length: 5m (197 inch)	WI1000-M12F5T05N	
			X	Same as above but 20m (787 inch)	WI1000-M12F5T20N	
		X		Cable with M12 male 8-pin connector loose wire ends 0.22mm² (24AWG) and screen. Length: 5m (197 inch)	WI1000-M12M8T05N	
		X		Same as above but 20m (787 inch)	WI1000-M12M8T20N	
	X			CANopen® cable with M12 male 5-pin connector, loose ends and screen. Length: 5m (197 inch).	WI1006-M12M5S05R	
	X			Same as above but 15m (591 inch)	WI1006-M12M5S15R	
X				CANopen® cable with M12 female 5-pin connector, loose ends and screen. Length: 5m (197 inch)	WI1006-M12F5S05R	
X				Same as above but 15m (591 inch)	WI1006-M12F5S15R	
Termination resistor						
	X			CANopen® male M12 termination resistor.	WI1008-M12M5STR4	
Protection caps. Optional if connector is not used, to protect from dust / liquids.						
	X	X		IP67 protection cap for M12 female connector.	WI1000-M12FCAP1	
X			X	IP67 protection cap for M12 male connector.	WI1000-M12MCAP1	

Important: Please note that the cables are a standard type. They are not recommended for use in cable chains or where the cable is repeatedly bent. If this is required, use a special robot cable (2D or 3D cable). See also *Accessories*, page 394 where additional M12 connectors are shown.