# CANopen Starter Kit KIT-152

## References for the first Start-up

**Edition May 2011**

Further information:

| | EUROPE | NORTH AMERIKA |
|---|---|---|
| Address: | SYS TEC electronic GmbH<br>August-Bebel-Str. 29<br>D-07973 Greiz<br>GERMANY | PHYTEC America LLC<br>255 Ericksen Avenue NE<br>Bainbridge Island, WA 98110<br>USA |
| Quotation Hotline: | +49 (3661) 6279-0<br>info@systec-electronic.com | +1 (800) 278-9913<br>order@phytec.com |
| Technical Hotline: | +49 (3661) 6279-0<br>support@systec-electronic.com | +1 (800) 278-9913<br>support@phytec.com |
| Fax: | + 49 (3661) 6279-99 | +1 (206) 780-9135 |
| Web Site: | http://www.systec-electronic.com | http://www.phytec.com |

5. Edition May 2011

## List of Figures and Tables

# KIT-152 - CANopen Starter Kit

The CANopen Starter Kit is the cost-efficient access to the development of own CANopen products and demonstrates the capabilities of the CANopen software and CAN- hardware. It contains all components of a CANopen Network, such as microcontroller-boards, an adapter from PC to CAN-Bus, CANopen IO-Box, CANopen slave and master software, CANopen slave and master software for Windows 2000/XP and a CANopen configuration tool for the object directory. The software components are fully executable demos with a limited range of functions. The user interface is identical to the full version. The CANopen Starter Kit has the following system requirements:

- MS-Windows 2000/XP or more recent
- 1 free COM- Interface
- 1 free USB- Interface
- Microsoft Visual C/C++ starting at version 7.1

# 1 Extent of Delivery:

## 1.1 Hardware:

| | |
|---|---|
| 1x MM-217 | DIPmodule F40 with Motherboard |
| 1x 3204000 | USB-CANmodule11x phyPS-451-Y CANopen IO-Box |
| 1x WK054 | CAN Cable Set |
| 1x WK800 | RS 232- Cable1x SV007 Power Supply Unit 9V/500mA |

## 1.2 Documentation:

| | |
|---|---|
| 1x L-1000 | Motherboard Developmentboard DIPmodule F40 |
| 1x L-1062 | CANopen ChipF40 System Manual |
| 1x L-487 | USB-CANmodule Hard- and Software Manual |
| 1x L-1048 | IO-Box System Manual |
| 1x L-1020 | CANopen Software Manual |
| 1x L-1078 | CANopen Starter Kit KIT-152, References for first start-up (this manual) |
| 1x L-1055 | CANopen Configuration Manager User Manual |
| 1x L-1056 | CANopen Device Monitor User Manual |
| 1x L-1022 | ODBuilder Manual |

## 1.3 Software:

| | |
|---|---|
| 1x SO-387 | USB-CANmodule Utility Disk |
| 1x SO-1018 | CANopen Kit-Library forKIT-152 |
| 1x SO-1007 | ODBuilder Demo |
| 1x SO-1008 | CANopen-Demo for Windows |
| 1x SO-1047 | CANopen Configuration Suite |

# 2 Setup and Implementing of the Hardware

## 2.1 Implementing of the DIPmodule F40

Unpack the single components. Make sure to do this in an electrostatic protected area.
The motherboard is already adjusted according to the attached DIPmodule F40. Still, please verify the settings of the jumpers on the motherboard that are described below.
The DIPmodule F40 has to be plugged into the 40-pole slot (*see Figure 1*).

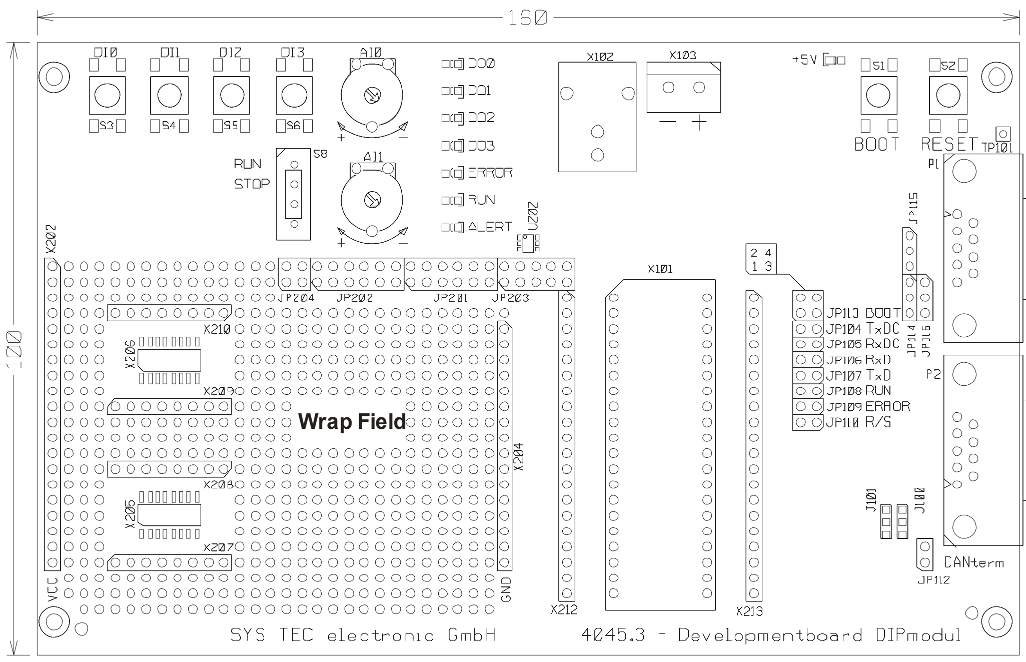


*Figure 1: DIPmodule F40 motherboard*

| Jumper | Function | Condition |
|---|---|---|
| JP104 | TxD CAN | open |
| JP105 | RxD CAN | open |
| JP106 | RxD RS232 | closed |
| JP107 | TxD RS232 | closed |
| JP108 | Run-LED | open |
| JP109 | Error-LED | open |
| JP110 | R/S | open |
| JP112 | CAN Termination | as required |
| JP113 | Boot | (1-3) (2-4) |
| JP114 | onboard RS232 enable | (1-2) |
| JP115 | onboard RS232 enable | (1-2) |
| JP116 | onboard RS232 enable | (1-2) |
| | | |

Please take the exact meaning of the Jumpers on the motherboard from the manual of the motherboard (*L 1000*).

## 2.2 Implementing of the CANopen IO-Box

The CANopen IO Box contains a completely programed DIPmodule F40 with CANopen Software.
With the DIP switch of the CANopen IO Box, the node address and the bitrate can be adjusted. Please make sure that both are adjusted as described in ***Fehler! Verweisquelle konnte nicht gefunden werden.***. If this is not the case, please adjust the setting according to ***Fehler! Verweisquelle konnte nicht gefunden werden.***.

| DIP1 | DIP2 | DIP3 | DIP4 | DIP5 | DIP6 | DIP7 | DIP8 |
|---|---|---|---|---|---|---|---|
| OFF | OFF | OFF | OFF | OFF | OFF | ON | OFF |

*Figure 2: Jumper configuration for DIP switches at CANopen IO Box*

Please take the exact meaning of the jumpers from the manual of CANopen ChipF40 (*L 1062*).

## 2.3 Connecting the Power Supply

The port for the power supply is located on the motherboard for the DIPmodule F40 at the low voltage connection jack X102. Please use the included Power Supply Unit SV007 with 9V/500mA. The LED D102 (green, next to the boost- button, *see Figure 1*) shows that the voltage is applied to the motherboard for the DIPmodule F40.

**Reference:** Please avoid changing the DIPmodule F40 or the jumper configuration when voltage is applied.

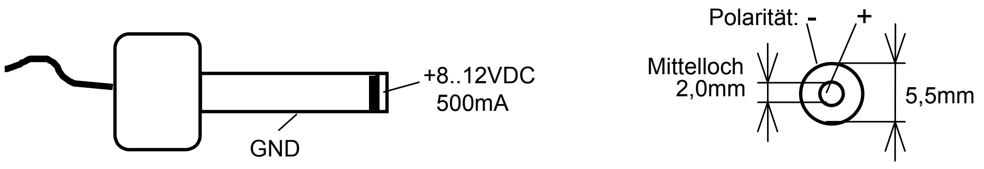


*Figure 3: Connection of power supply to motherboard*

The power supply of the CANopen IO-Box is being connected via the CAN connection jack. For this please use the terminal resistance with 2 cables that was included in the delivery. Please screw the ends of this 2 cables together with the screw terminal X103 on the motherboard of the DIPmodule F40. *Table 1* shows the pin configuration of the power supply. Connect the SUB-D 9 connector to a free connection jack at the end of the CAN-bus cable.

| Pin at SUB-D 9 | Description | Color of the cable at the terminal resistance | Pin at X103 on motherboard |
|---|---|---|---|
| 3 | GND | black | - |
| 9 | VCC (+8..12VDC) | red | + |

*Table 1: Pin configuration of the power supply at the CANopen IO box*

## 2.4 Setup of the RS 232 Interface

The serial RS 232 interface is used for communication with the PC and for downloading software onto the microcontroller board. All printf()-instructions are emitted at this interface within the CANopen kit software. The connection is established with a RS 232 extension cord which is included in the delivery of the CANopen Starter Kit. Please connect the cable to the top D-SUB connection jack (P1) on the motherboard for the DIPmodule F40 and your PC.
For the display of the outputs of the CANopen software, the tool HyperTerminal can be used, which can be found on a Windows operating system under *Programs →Accessories →Communications*.
Connect via HyperTerminal to the used COM-interface with the following settings: 9600 bit/s, 8 data bits, no parity, 1 stop bit and no data flow control.

The signal lines *TxD*, *RxD* and *GND* are required.
Pin configuration on IBM PC (9- pole plug)

| | | | | | |
|---|---|---|---|---|---|
| RxD | Pin 2 | Received data line | TxD | Pin 3 | Output data line |
| GND | Pin 5 | Ground (GND) | | | |

Pin configuration on IMP PC (25- pole plug)

| | | |
|---|---|---|
| TxD | Pin 2 | Output data line |
| RxD | Pin 3 | Received data line |
| GND | Pin 7 | Ground (GND) |

Pin configuration on the motherboard for the DIPmodule (9- pole plug)

| | | |
|---|---|---|
| TxD | Pin 2 | Output data line |
| RxD | Pin 3 | Received data line |
| GND | Pin 5 | Ground (GND) |

Therefore the following configuration for the connection cable between PC and motherboard is incidental.

| **D-SUB-9 (PC-Side)** | **Connection Jack** | **D-SUB-9 (MC-Side)** | **Connector** |
|---|---|---|---|
| Description | Pin-Nr. | Description | Pin-Nr. |
| RxD | 2 | TxD | 2 |
| TxD | 3 | RxD | 3 |
| GND | 5 | GND | 5 |

*Table 2: RS 232 Connection cable*

## 2.5 Wiring of the CAN-Network

The D-SUB-9 plug "P2" on the motherboard with the DIPmodule F40 and the D-SUB-9 plug "CAN" on the CANopen IO-box are used for the connection to the CAN-bus.

Make use of the following configuration:

| **Pin** | **Description** |
|---|---|
| 2 | CAN_L (dominant low) |
| 3 | CAN_GND |
| 6 | CAN_GND (optional) |
| 7 | CAN_H (dominant high) |
| 9 | CAN_VCC (+7..+13VDC) |

*Table 3: Wiring of the CAN-bus*

For the use of the CANopen Starter Kit the signals CAN_L and CAN_H are relevant. *Figure 4* shows the fundamental bus circuit.

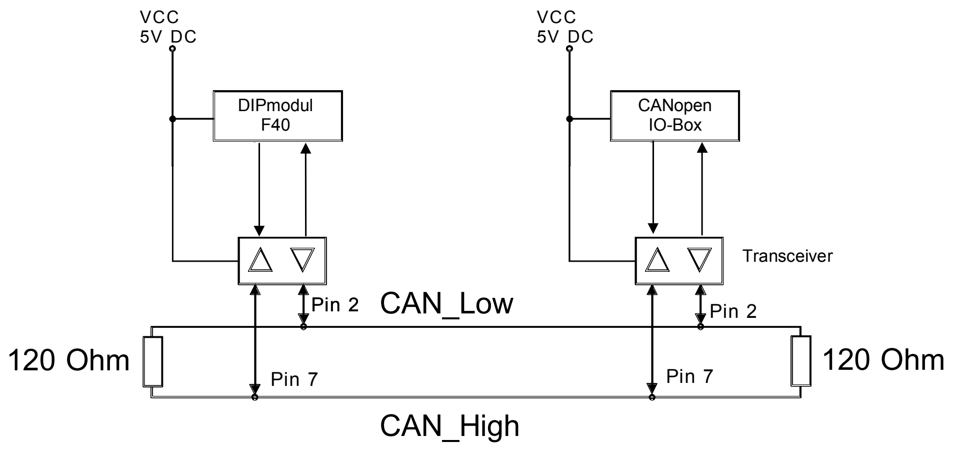


*Figure 4: Fundamental CAN-bus circuit (without galvanic decoupling)*

A twisted two-wire line with a terminating resistance of 120Ω on both ends can be used as the CAN-bus cable. The wave impedance of the cable should amount to 120Ω. The cross section is determined by the length of the line *(see **Fehler! Verweisquelle konnte nicht gefunden werden.**)*.

| max. Length of Line [m] | max. Bitrate [kBit/s] | Resistivity [mΩ/m] | Cross Section of Line [mm²] |
|---|---|---|---|
| 30 | 1000 | 70 | 0,25..0,34 |
| 100 | 500 | <60 | 0,34..0,60 |
| 500 | 100 | <40 | 0,50..0,60 |
| 1000 | 20 | <26 | 0,75..0,80 |

*Table 4: Recommended line parameters for CAN-cable*

## 2.5.1 Connection of the CAN- Cable Set

Included in the delivery of the CANopen Starter Kit is the cable set WK054. This consists of a 3m long 9-pole flat cable with 7 D-SUB connection jacks, 1 D-SUB plug (to extend the cable) and 2 plugs in the casing with the 120Ω terminating resistances. The cable also provides the power supply of the CANopen IO-box.

The cable set is appropriate for laboratory use, the quick start up and for testing purposes. For long term usage please use a cable like described above.

The plugs with the terminating resistances are to be connected to the ends of the CAN- bus. The motherboard with the DIPmodule F40, the CANopen IO-box and the USB-CANmodule can be connected to the free connection jacks of the CAN-bus afterwards. However, do not connect the USB-CANmodule to your PC yet.

At one of the both plugs with the terminating resistances end two cables. These usually function as the supplier of CAN_VCC (red, +7 to +13 VDC) and CAN_GND (black) in case of galvanic decoupling. A galvanic decoupling is not available for the CANopen Starter Kit. Here the two cables have to be used to provide the supply voltage for the CANopen IO-box *(see chapter 2.3)*.

When all boards are adjusted properly and the CAN-bus is connected, the power supply can be activated.

**Caution:**
All modifications have to take place in a not energized condition!

# 3 Installation of the Tool Software

## 3.1 Installation of the Development Environment Softune

The delivery content includes the FUJITSU Microcontrollers DVD version 5.2 with the corresponding Softune Workbench for the F2MC-16LX family. Insert the DVD, start the installation and follow the instructions during installation. Before you may use the software, you must register your Software Development Environment at FUJITSU. European customers can proceed with the registration directly via the following online form.

https://mcu.emea.fujitsu.com/cusreg/htm/cusreg_form.htm

If your location is outside Europe (EMEA), please contact your local sales office.

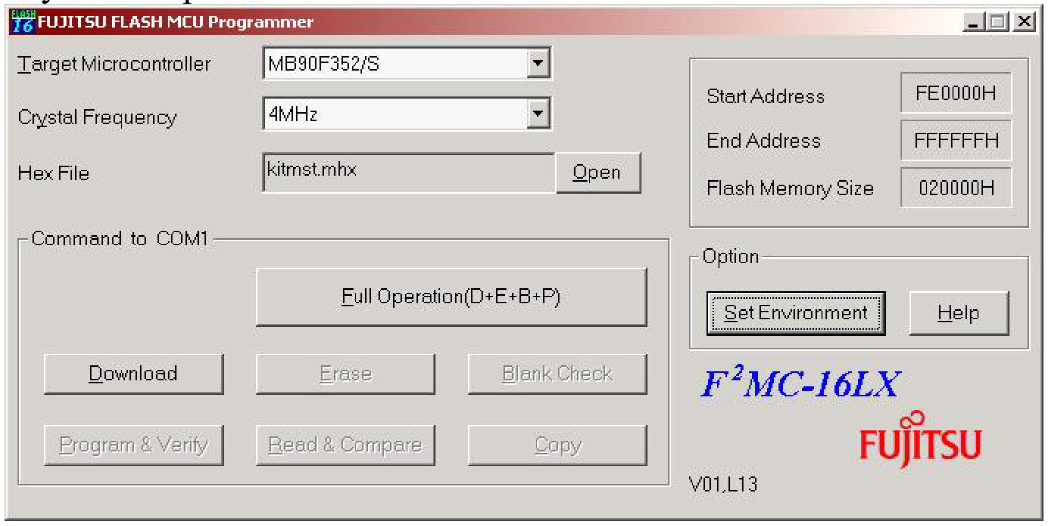
### 3.1.1 Installation of the Fujitsu Flash MCU Programmer

You need the Flash MCU Programmer to program a check program into the internal flash of the microcontroller before loading the *Accemic Debugger (see chapter 3.2)* into the DIPmodule F40.

The Flash Programmer is freeware and can be found on the SYS TEC product-CD. Therefor switch to the "*CDROM:\Tools\FUJITSU\Flasher_for_F2MC-16LX<Version>*" directory and run the setup "*Pcw16setup.exe*".

Follow the instructions during the installation.

After the installation, start the Flash Programmer from the program group "*FUJITSU FLASH MCU Programmer*". Then set up the *Target Microcontroller* MB90F352/S. Please set up the COM interface used on your computer Under "*Set Environment*".



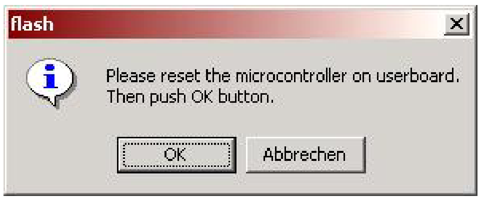
FUJITSU FLASH MCU Programmer

Target Microcontroller: MB90F352/S
Crystal Frequency: 4MHz
Hex File: kitmst.mhx

Start Address: FE0000H
End Address: FFFFFFH
Flash Memory Size: 020000H

V01,L13

Now connect the motherboard with the DIPmodule F40 to your computer using a RS 232 cable and activate the supply voltage for the motherboard. Then press the button "*Download*" in the dialog box. Next you will be asked to reset the microcontroller.


flash

Please reset the microcontroller on userboard.
Then push OK button.

Therefor press the two buttons "BOOT" and "RESET" on the motherboard with the DIPmodule F40 in the following temporal sequence.

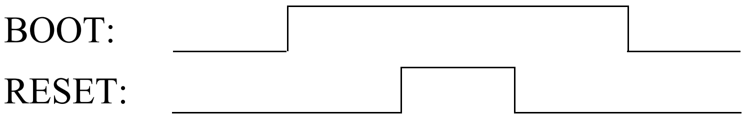


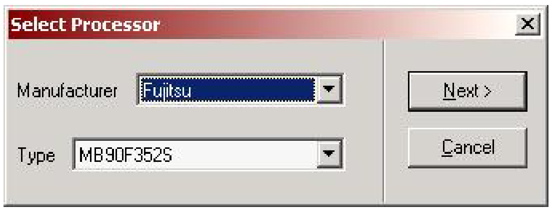
*Figure 5: Initiating the burn-in-mode at DIPmodule F40*

Confirm the dialog box with OK. Now the check program is being burnt into the flash of the microcontroller. After this process, press the reset button on the motherboard of the DIPmodule F40 and close the program. Continue the installation of the *Accemic MDE Debugger* in the next chapter.

## 3.2 Installation of the Accemic MDE Debugger

The *Accemic MDE Debugger* can be downloaded from internet site of company Accemic: http://www.accemic.com. This will require a registration. Run the setup of the debugger and follow the instructions. For the unlimited use of the *Accemic Debugger* you need a license that you need to buy additionally.

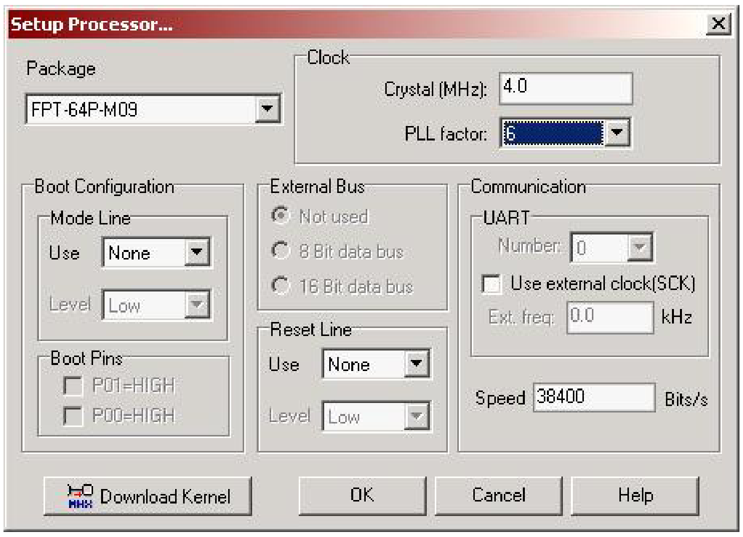
Start the *Accemic MDE* from the identically named program group. At the first start you will be asked to adjust several settings. In the first dialog set up the microcontroller type:


Select Processor

Manufacturer: Fujitsu

Type: MB90F352S

Confirm these setting with "Next >". In the next dialog you adjust the properties of the microcontroller. At that point please set the PLL factor to 6 and enter the requested transfer rate under "Speed":


Setup Processor...

Package: FPT-64P-M09

Clock: Crystal (MHz): 4.0; PLL factor: 6

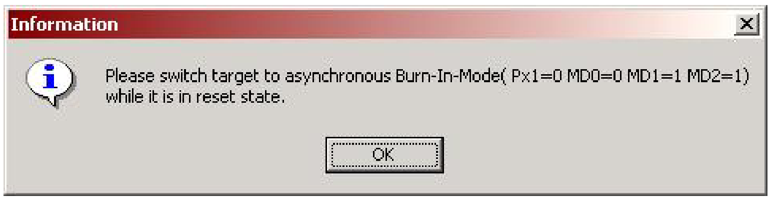
Boot Configuration: Mode Line: Use None; Level Low. Boot Pins: P01=HIGH, P00=HIGH (unchecked)

External Bus: Not used (selected); 8 Bit data bus; 16 Bit data bus

Reset Line: Use None; Level Low

Communication: UART Number: 0; Use external clock(SCK) (unchecked); Ext. freq: 0.0 kHz; Speed 38400 Bits/s

Now connect the motherboard with the DIPmodule F40 to your computer using a RS 232 cable and activate the supply voltage for the motherboard. Then press the button "Download Kernel" in the dialog box. Next a dialog will ask you change in the Burn-In-Mode:


Information

Please switch target to asynchronous Burn-In-Mode( Px1=0 MD0=0 MD1=1 MD2=1) while it is in reset state.

Therefor please proceed as shown in *Figure 5* in *chapter 3.1.1*.

After that please confirm the dialog box with "OK". The communication module for the *Accemic Debugger* is being copied into the flash. This communication module is only started if you press the "RESET" button at the motherboard of the DIPmodule F40 after finishing the download (without BOOT).

Now you can use this debugger for your projects with the DIPmodule F40. Please keep in mind that for those projects the following settings have to be made in the *Softune Workbench*:

- The *monitor16LX.asm* file has to be included in the project.
- The Define ACC_MON has to be set in the settings of the assembler.
- The Define INIT_SERIAL has to be disabled in the settings of the C compiler.

Further information to this can be found in *chapter 4* of this manual.

**Reference:**
When using the *Accemic MDE Debugger* it is not possible to view printf()-outputs via the serial interface (e.g. via HyperTerminal), because the *Accemic MDE Debugger* needs this interface for the communication with the communication module in the flash of the DIPmodule F40.

## 3.3 Installation of the CANopen Starter Kit Software

Insert the SYS TEC product-CD in the CD drive, switch to the "*CDROM:\Products\CANopen Starterkit-152\Software\*" directory and run the *Kit-152.exe* file. This setup automatically installs the CANopen kit library for the DIPmodule F40 for the development environment Softune (SO-1018) and the CANopen demo for Windows (SO-1008).

Please follow the instructions during the installation.

At the end of the installation you get offered to install the CANopen Configuration Suite demo (SO-1047), the ODBuilder demo (SO-1007) and the USB-CANmodule Utility Disk (SO-387) automatically. Permit these additional installations and press "NEXT".

### 3.3.1 Installation of the CANopen Configuration Suite Demo

The CANopen Configuration Suite demo is being installed automatically with the CANopen kit software if it was selected when running the *Kit-152.exe* on the SYS TEC product-CD. Should this installation have not been carried out automatically you also can install it afterwards. Therefor switch on the SYS TEC product-CD to the "*CDROM:\Products\CCS_SO-1047\Software\*" directory and run the *CCS_Setup.exe* file.

Please follow the instructions during the installation.

The demo is limited to a bitrate of 125kBit/s and to the CANopen node addresses 0x20 and 0x40. Through the installation of a license key, the demo can be unlocked to the full version. This license key can be purchased from the SYS TEC electronic GmbH.

### 3.3.2 Installation of the ODBuilder Demo

The demo version of the ODBuilder is also being installed automatically when running the *Kit-152.exe* from the SYS TEC product-CD. However, it can also be installed afterwards by running the *SetupDemo.exe* in the "*CDROM:\Products\OD_Builder_SO-1007\Software*" directory on the product-CD.

Please follow the instructions during the installation.

The demo is a full executable version. However, the created C-files for the CANopen object directory cannot be used with the CANopen kit library for the DIPmodule F40.

### 3.3.3 Installation of the USB-CANmodule Utility Disk

The USB-CANmodule Utility Disk contains the hardware drivers, the documentation and a CAN view program for the Windows OS and a demo project for Microsoft Visual Studio C/C++ version 5.0 or more recent.

Please do not plug the USB-CANmodule in the USB port of your computer before having installed the driver. The installation is also offered by the *Kit-152.exe*. Should you have disabled the installation there, switch in the "*CDROM:\Products\USB-CANmodul_Series\Software\SO-387\*" directory on the SYS TEC product-CD and run the *Setup.exe* file.

Please follow the instructions during the installation. Further references to the installation can be found in the manual *L-487*.

# 4 Implementing of the CANopen Software

After configuring the hardware like described in *chapter 2 and* connecting the boards via the CAN-bus-cable and installing the necessary software (see *chapter 3*), the CANopen Starter Kit software is being loaded in the DIPmodule F40 and executed.

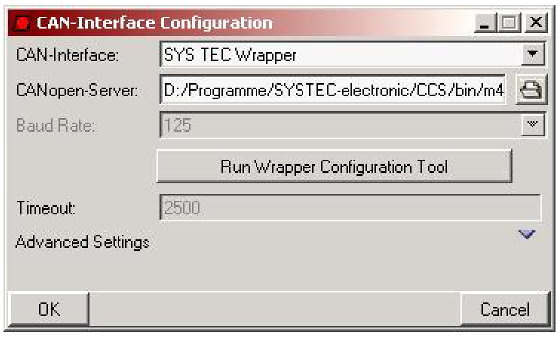
Please proceed as follows during the first start up:

1. Connect the motherboard of the DIPmodule F40 via the provided serial connection cable to the COM interface at you PC.

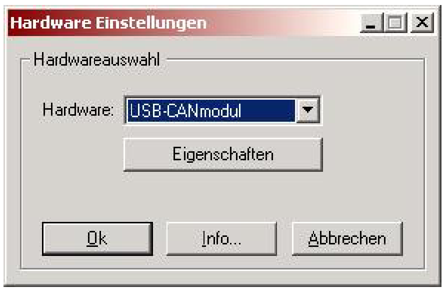
2. Connect all components with the CAN-cable like described in *chapter 2.5.1*.

3. Activate the power supply of the components and connect the USB-CANmodule to your computer.

4. Initiate the  CDM ("*CANopen Device Monitor*") from the CANopen configuration suite . At first start you are asked to configure the CAN-interface. Set the CAN-interface to "*SYS TEC Wrapper*". Press the "*Run Wrapper Configuration Tool*" button and select the "*USB-CANmodule*" hardware. Under "*Properties*" you can set the device number of the USB-CANmodule. Should you run only one USB-CANmodule on your computer, you can leave the device number at a value of 255, whereas the hardware driver uses the module on the computer which it found first. Confirm all settings with "OK".



CAN-Interface: SYS TEC Wrapper
CANopen-Server: D:/Programme/SYSTEC-electronic/CCS/bin/m4
Baud Rate: 125
Timeout: 2500



Hardware: USB-CANmodul

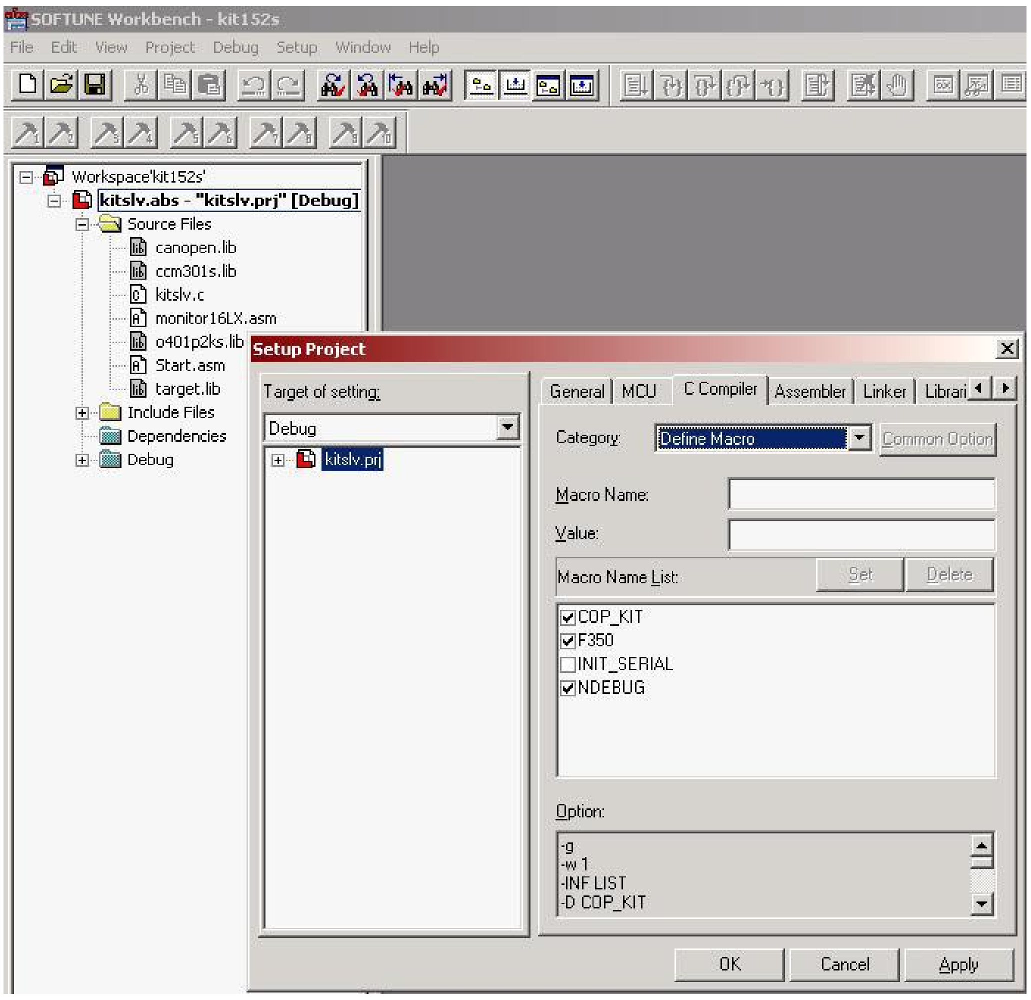
5. In the CDM select in the menu *Connection* → *Connect*. With this, the red LED at the USB-CANmodule should go out.

6. Open the Softune development environment and run the working set *kit152s.wsp* from the "*C:\systec\cop\target\dipmodul-f40\no_os\softune\kitslv*" *directory*.

   The projects *kit152s.wsp* (CANopen Slave) and *kit152m.wsp* (CANopen Master), which are included in the kit, each contain a target environment called *Debug* and one called *Release*. *Debug* is prepared to work with the *Accemic-Debugger* whereas *Release* generates a code which is being executed directly in the flash of the CPU (without debugger). The differences of the projects are annotated in the following.

7. If you use the *Accemic MDE Debugger* (i.e. without display of the printf()-instructions, it is necessary that the *monitor16LX.asm* file is included in the project. This is a part of the Accemic installation. The file is already listed in the project. Nevertheless, you have to copy it from the Accemic installation directory to the path of the project. Therefor copy all files from the "*include*" directory of the Accemic installation to the "*systec\cop\target\dipmodul-f40\no_os\softune\source"* directory. In contrast to the target environment *Release*, the Define INIT_SERIAL is disabled in the project settings *(Menu Project* → *Setup Project...* → *C Compiler* → *Category: Define Macro*), and the Define ACC_MON is enabled. Should that not be the case, please enable it accordingly. Also the Define ACC_MON has to be enabled in the project settings for the assembler (*Menu*

*Project* → *Setup Project...* → *Assembler* → *Category: Define Macro*). Retranslate the project and proceed with *step 17*.



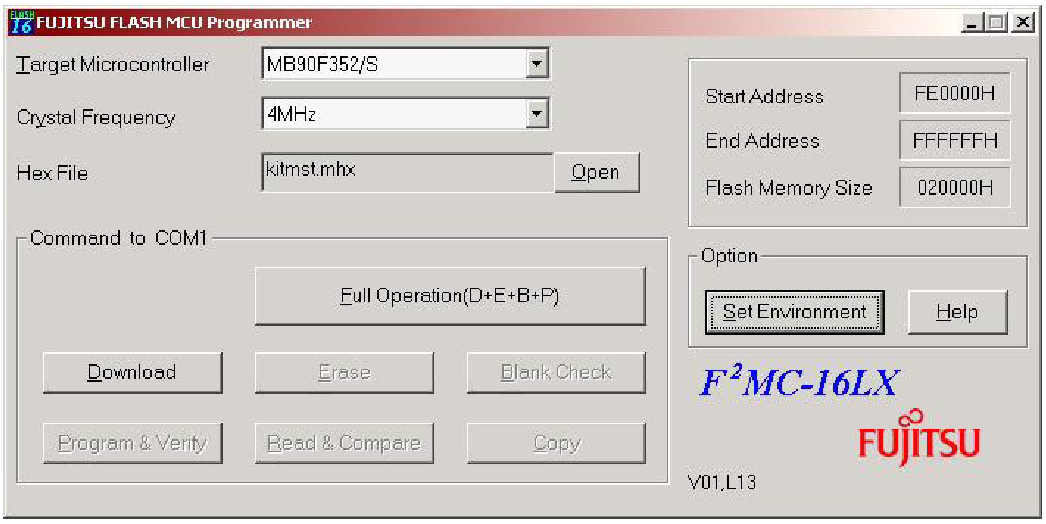
Category: Define Macro

Macro Name List:
- [✓] COP_KIT
- [✓] F350
- [ ] INIT_SERIAL
- [✓] NDEBUG

Option:
```
-g
-w 1
-INF LIST
-D COP_KIT
```

8. If you do not use an *Accemic MDE Debugger*, remove the *monitor16LX.asm* from the project if necessary and set the Define INIT_SERIAL (C Compiler) and disable the Define ACC_MON (C Compiler and Assembler) in the project settings. Retranslate the project.

9. Run the FUJITSU FLASH MCU Programmer FMC16LX and set up the COM interface used by you under "*Set Environment*".



FUJITSU FLASH MCU Programmer

Target Microcontroller: MB90F352/S
Crystal Frequency: 4MHz
Hex File: kitmst.mhx

Start Address: FE0000H
End Address: FFFFFFH
Flash Memory Size: 020000H

10. Press "*Download*". At that point you are asked to set the burn-in-mode. Therefor proceed as described in *chapter 3.1.1, Figure 5*. If you confirm the request with "OK", the flash controller program is being transmitted into the hardware. Afterwards confirm the status signal at the end of the transmit with "OK".

11. Press "*Open*" and set up the *kitslv.mhx* file from the "*C:\systec\cop\target\dipmodul-f40\no_os\softune\kitslv\Debug\ABS\*" or the "*C:\systec\cop\target\dipmodul-f40\no_os\softune\kitslv\Release\ABS\*" directory.

12. Afterwards press the "*Erase*" button to delete the flash of the DIPmodule F40.

13. Press "*Program & Verify*" to program the compiled project into the flash of the DIPmodule F40. After that, close the FMC16LX tool.

14. Run HyperTerminal and configure it for the used COM-interface (9600 Bit/s, 8 data bits, no parity, 1 stop bit and not data flow control). If the status signal "COM… could not be opened…" appears, please make sure you closed the FMC16LX tool.

15. Reset the DIPmodule F40 by pressing the "RESET" button on the motherboard. Thereby the project is being run on the hardware.

    You can see the following output in HyperTerminal:

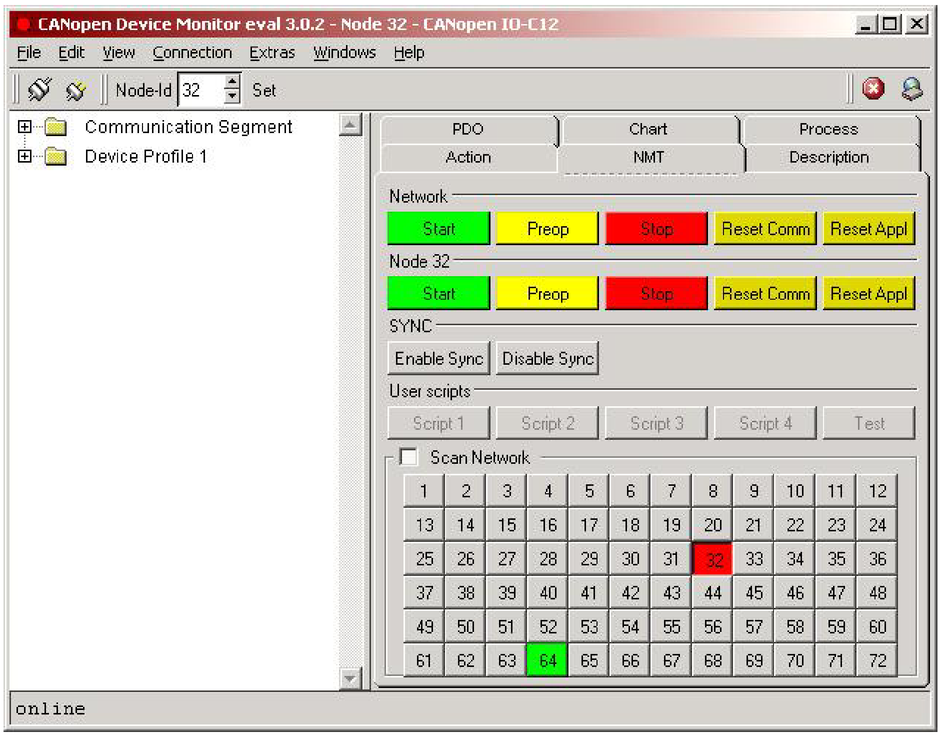
    ```
    ******************************************
    CANopen Slave Kit demo
    (c) SYS TEC electronic GmbH
    CANopen V5.29.0
    ******************************************

    init CANopen! Node ID = 0x20
    connect this node to network...
    ...
    ```

    Furthermore you can see the constantly recurring output "*TX-PDO 0 event*". This PDO is being sent in a 500ms phase via the CAN-message with the CAN-ID 0x2C0. This leads to flashing LEDs on the CANopen IO-box.

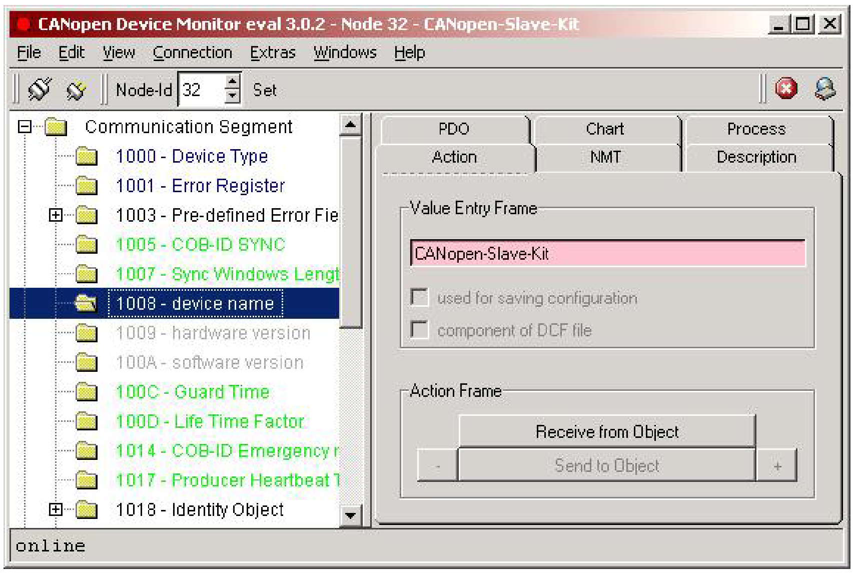
16. Press one of the 8 buttons at the CANopen IO-box. This leads to the transmission of a PDO with the CAN-ID 0x1C0 and to the appearance of the output "*RX-PDO 0 event*" in HyperTerminal.

17. Now switch to the CDM, press the "*NMT*" sheet in the right part of the window and select the node number 32. Answer the question "Do you want to load ... " with "No".



18. Press *File* → *Load EDS* → *Select another file* in the CDM menu and select the *o401p2ks.eds* file in the "*C:\systec\cop\objdicts*\o401p2ks\" directory.

19. Switch to the "*Action*" sheet in the right part of the window, select the object 0x1008 (device name) in the left window by expanding the tree structure "*Communication Segment*" and press the "*Receive from Object*" button. Now the device name of the CANopen Starter Kit is being read and displayed on the right side of the window: "*CANopen-Slave-Kit*".



You can customize the CANopen demo application *kitslv.c* according to your needs. Nevertheless, please consider the restrictions of the CANopen Starter Kit software according *chapter 4.2*.

## 4.1 Further References

Besides the CANopen slave kit library, the "*C:\systec\cop\target\dipmodul-f40\no_os\softune\kitmst*" directory also contains a project for a CANopen master demo. This has the same restrictions as the slave, however it supports the NMT master services according to DS-301. In the demo application the SDO client and the emergency consumer are also being used there.

Two demo projects (slave and master) with the same restrictions are included for the windows OS in the "*C:\systec\cop\target\x86\windows\vc7*" directory. These can be used with the provided USB-CANmodule. However, please consider that the tools of the CANopen configuration suite are not able to access the USB-CANmodule simultaneously with these Windows demos.

Please take the description of the CANopen functions from the CANopen software manual (L-1020). The same applies for operating the CANopen device monitor CDM (L-1055).

## 4.2 Restrictions of the CANopen Kit Software

The CANopen slave kit library has the following restrictions:

- Suitable CAN bitrates: only 125kBit/s
- Adjustable node number: only 0x20 and 0x40.
- Limited object directory with 8 x digital input 8 bit, 8 x digital output 8 bit, 4 x analog input 16 bit and 4 x analog output 16 bit.
- Heartbeat producer but no heartbeat consumer and no emergency consumer.
- 2 TPDOs and 2 RPDOs, static PDO mapping (defined according to *DS-401 V2.1 from chapter 6.2.4*), fixed inhibit time of 500ms, no event timer (sub index 5 of the communication parameter of the TPDOs)

The CANopen master kit library offers the possibility to monitor 2 slave nodes via life guarding. Furthermore the emergency consumer can be configured for 2 nodes, and a SDO client (object 0x1280) can be used. These additional services at the master are limited to the node numbers 0x20 and 0x40.

## 4.3 References for the Demo Applications

The demo applications *kitslv.c* for the slave and *kitmst.c* for the master are adjusted according to the corresponding object directories and the appositive Gegenknoten (CANopen IO-box).

### 4.3.1 The Initialization Structure

Even though some initialization parameters of the CANopen kit are fixed, the initialization structure has to be specified. If invalid values are delivered to this structure the CcmInitCANopen() function emits the corresponding error code. A list of all possible error codes can be found in the manual of the *CANopen Stacks L-1020*.

```
CONST tCcmInitParam ROM CcmInitParamInst1_g =
{
    NODE_ID,                       // NodeId
    BAUDRATE,                      // baudrate
    NULL,                          // address of BDI table
    0,                             // size of BDI table
    0xFFFFFFFFL,
    0x00000000L,
    {{0}},                         // hardware parameter

    #if (DEV_SYSTEM != _DEV_WIN32_)
    TgtEnableCanInterrupt1,        // for 1st CAN-Controller
    #else
    NULL,
    #endif

    AppCbNmtEvent,                 // NMT callback function
    ObdInitRam                     // OD init function
};
```

### 4.3.2 The Variable Table

The variable table links the C-variables in the application to the particular objects in the object directory. Through this it can be ensured that the applications can access quickly on the current data in the object directory.

```
CONST tVarParam ROM aVarTab_g[] =
{
    // digital input 8 bit
    {kVarValidAll, 0x6000, 0x01, sizeof (BYTE),
        &abDigitalIn8Bit_g[0],    NULL, NULL},
    ...
};
```

The content of this table must not be changed since it is depending on the used object directory, which is fixed at the CANopen kit.

### 4.3.3 The PDO Table

The PDO table contains the communication parameters for the PDO linking, the pointer to the data that has to be used and the pointer to the callback-functions of the PDO. These parameters can be modified at the CANopen kit. Nevertheless, please consider the restrictions concerning the transmission type, inhibit time and the event timer. An input in the PDO table has the following structure:

```
typedef struct
{
    WORD                m_wPdoCommuIndex;
    DWORD               m_dwCanId;
    BYTE                m_bTxTyp;
    WORD                m_wInhibitTime;
    WORD                m_wEventTime;
    BYTE                m_bSize;
    void MEM*           m_pVarData;
    tPdoCallback        m_fpCallback;

} tPdoStaticParam;

CONST tPdoStaticParam ROM aPdoTab_g[] =
{
    // 1st RPDO
    {0x1400,
        (0x000001C0L), 255,   0, 0,
        1, &abDigitalOut8Bit_g[0],
        AppCbRxPdo},
    ...
};
```

| Parameter | Meaning |
|---|---|
| m_wPdoCommuIndex | Communication index of the PDO in the OD (0x1400-0x15FF for RPDOs, 0x1800-0x19FF for TPDOs) |
| m_dwCanId | COB-ID of the PDO according to sub index 1 of the communication parameters |
| m_bTxTyp | Transmission type des PDO according to sub index 2 of the communication parameters (parameter is being ignored) |
| m_wInhibitTime | Inhibit time of the PDO according to sub index 3 of the communication parameters (parameter is being ignored) |

| Parameter | Meaning |
|---|---|
| m_wEventTime | Event time of the PDO according to sub index 5 of the communication parameters (parameter is being ignored) |
| m_bSize | Number of the bytes to be transmitted in the CAN-message of the PDO (between 0 and 8) |
| m_pVarData | Pointer on up to 8 coherent data bytes that are transmitted in the CAN-message of the PDO. |
| m_fpCallback | Pointer on the  callback-function that is accessed after the transmission or receiving of the PDO. This pointer can be ZERO. |

Please consider that the assignment of the PDO data in the parameter m_pVarData matches with the fixed mapping *(see Table 5)* and the variable table *(see chapter 4.3.2)*. Otherwise it may occur that the data, that is possibly transmitted via SDO for the corresponding object, does not match with the data in the PDO.

| Mapping Objects | Mapping Data |
|---|---|
| **RPDO1:** | |
| 0x1600/0 | 8 mapped objects |
| 0x1600/1 | Objekt 0x6200/1 |
| 0x1600/2 | Objekt 0x6200/2 |
| 0x1600/3 | Objekt 0x6200/3 |
| 0x1600/4 | Objekt 0x6200/4 |
| 0x1600/5 | Objekt 0x6200/5 |
| 0x1600/6 | Objekt 0x6200/6 |
| 0x1600/7 | Objekt 0x6200/7 |
| 0x1600/8 | Objekt 0x6200/8 |
| **RPDO2:** | |
| 0x1601/0 | 4 mapped objects |
| 0x1601/1 | Objekt 0x6411/1 |
| 0x1601/2 | Objekt 0x6411/2 |
| 0x1601/3 | Objekt 0x6411/3 |
| 0x1601/4 | Objekt 0x6411/4 |
| **TPDO1:** | |
| 0x1800/0 | 8 mapped objects |

| **Mapping Objects** | **Mapping Data** |
|---|---|
| 0x1800/1 | Objekt 0x6000/1 |
| 0x1800/2 | Objekt 0x6000/2 |
| 0x1800/3 | Objekt 0x6000/3 |
| 0x1800/4 | Objekt 0x6000/4 |
| 0x1800/5 | Objekt 0x6000/5 |
| 0x1800/6 | Objekt 0x6000/6 |
| 0x1800/7 | Objekt 0x6000/7 |
| 0x1800/8 | Objekt 0x6000/8 |
| **TPDO2:** | |
| 0x1801/0 | 4 mapped objects |
| 0x1801/1 | Objekt 0x6401/1 |
| 0x1801/2 | Objekt 0x6401/2 |
| 0x1801/3 | Objekt 0x6401/3 |
| 0x1801/4 | Objekt 0x6401/4 |

*Table 5: Fixed PDO Mapping according to DS-401 V2.1*

### 4.3.4 Use of the PDO Variables

The C-variables of a PDO are transmitted by the PDO table *(see chapter 4.3.3)* to the CANopen. The transmit variables have to be changed by the application (e.g. due to a digital entrance) to transmit the variable to the CAN-bus. Is a PDO received by the CANopen stack then the values contained are copied into the reception variables in the application.

After transmitting or receiving a PDO, the CANopen stack calls up a PDO callback function that is defined by the PDO table. Through this function you can implement a specific reaction to this event. Since only static PDO mapping is supported in the CANopen kit, the PDO variables, corresponding to the called up PDO callback function, are easy to find. The PDO-variables have already been defined by you in the PDO table.

There are two different methods to modify the PDO-variables respectively to read them.

1. A process loop in which you can read and write the variables constantly is located in the main program. Has e.g. the value of a digital entrance changed, you write the new value in the PDO-variable and call up the CcmSignalStaticPdo() function with the corresponding PDO communication parameter. In the next run of the process the CANopen Stack transmits this PDO (provided the inhibit time is expired). The reception variables can be used immediately to operate a digital exit.

   Example:
```
// Prozessschleife
while (1)
{
    // hat sich der digitale Eingang geändert, dann senden
    if (abDigitalIn8Bit_g[0] != READ_IO ())
    {
        abDigitalIn8Bit_g[0] = READ_IO ();
        CcmSignalStaticPdo (0x1800);
    }

    // Ausgänge direkt schreiben
    WRITE_IO (abDigitalOut8Bit_g[0]);

    // CANopen Prozess ausführen
    CcmProcess ();
}
```

2. You can read or write the PDO variables as a reaction within the PDO callback-function. This has e.g. the advantage that you only set an exit, if the PDO really was received. Or you transmit the values after a given calculation algorithm.

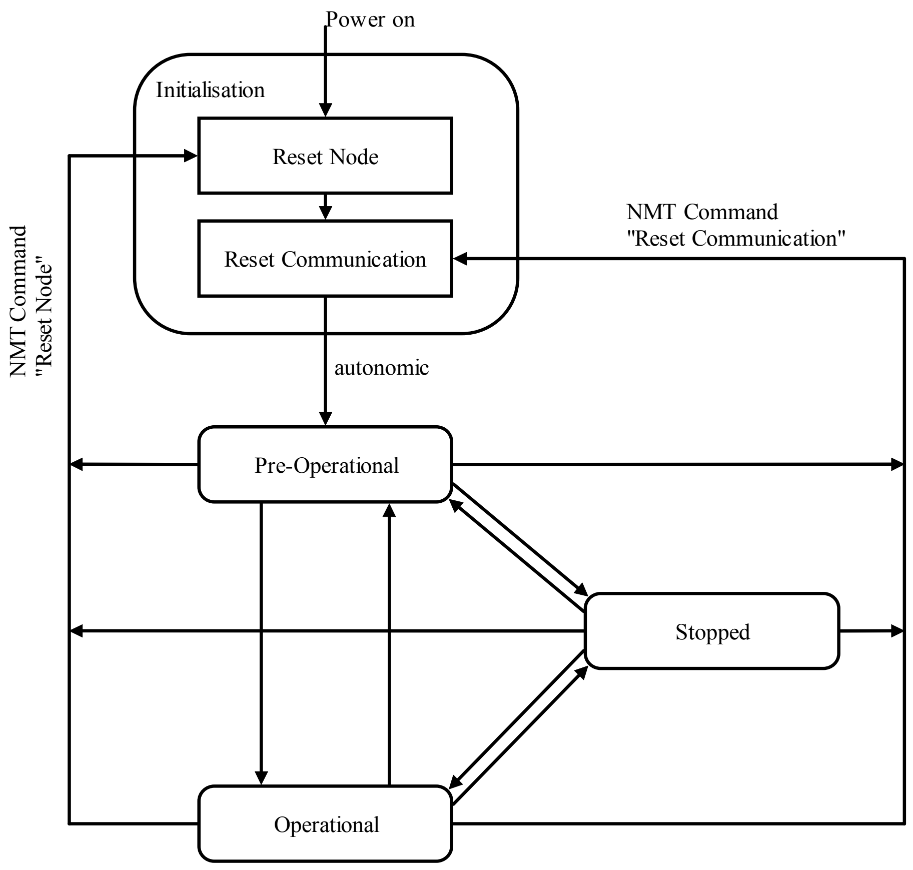
   Example:
```
// Callback-Funktion für Sende-PDO
tCopKernel PUBLIC AppCbTxPdo (WORD wPdoCommuIndex_p)
{
    // als nächstes den nachfolgenden Wert senden
    abDigitalIn8Bit_g[0] += 1;
    CcmSignalStaticPdo (0x1800);
}

// Callback-Funktion für Empfangs-PDO
tCopKernel PUBLIC AppCbRxPdo (WORD wPdoCommuIndex_p)
{
    // Ausgänge schreiben
    WRITE_IO (abDigitalOut8Bit_g[0]);
```

```
        return kCopSuccessful;
    }
```

### 4.3.5 The NMT Callback-Function

The NMT callback function informs you about an event at the NMT state machine. Such an event can be the shift into another state or a NMT command such as "Reset Communication" or "Reset Node".



The application has to perform further initializations in this callback function. Furthermore the application can react to changes of the NMT state machine, for instance to trigger a status LED.

```
tCopKernel PUBLIC AppCbNmtEvent (tNmtEvent NmtEvent_p)
{
    // welchens Ereignis ist aufgetreten?
    switch (NmtEvent_p)
    {
        case kNmtEvEnterInitialising:
            SWITCH_STATUS_LED (OFF);
            ...
            break;

        case kNmtEvResetNode:
            SWITCH_STATUS_LED (OFF);
            break;

        case kNmtEvResetCommunication:
            SWITCH_STATUS_LED (OFF);
            break;

        case kNmtEvEnterPreOperational:
            SWITCH_STATUS_LED (OFF);
            break;

        case kNmtEvEnterOperational:
            SWITCH_STATUS_LED (ON);
            break;

        case kNmtEvEnterStopped:
            SWITCH_STATUS_LED (OFF);
            break;
    }

    return kCopSuccessful;
}
```

# Index

**Document:** **KIT-152**
**Number of Document: L-1078d_5, Edition May 2011**

**How would you improve this manual?**

**Did you spot any mistakes in this manual?** Page

**Sent in by:**

| | |
|---|---|
| Customer ID: | |
| Name: | |
| Firm: | |
| Address: | |
| | |

**Send to:**

SYS TEC electronic GmbH
August-Bebel-Str. 29
D-07973 Greiz, Germany
Fax : +49 (0) 3661 62 79 99

Published by

© SYS TEC electronic GmbH May 2011

Ordering No. L-1078d_5
Printed in Germany